Όραση Υπολογιστών

4η Εργασία

Περιεχόμενα

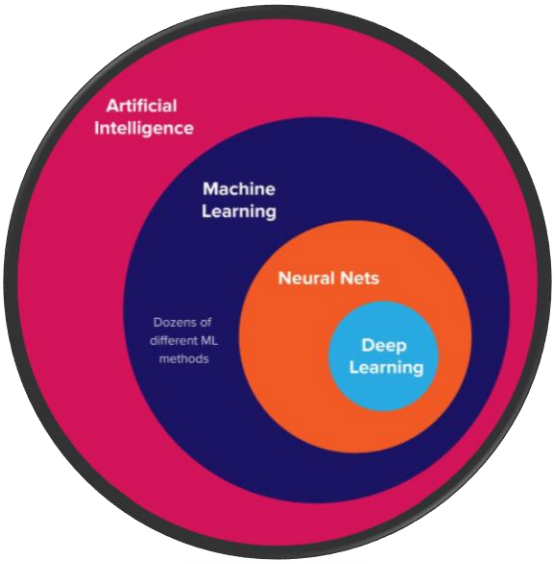
1. Εισαγωγή

Στην παρούσα εργασία επιλύεται το πρόβλημα της ταξινόμησης πολλαπλών κλάσεων (multi-class classification), με τη χρήση συνελικτικών νευρωνικών δικτύων (convolutional neural networks). Πιο συγκεκριμένα, μελετώνται δύο διαφορετικές μέθοδοι για την ταξινόμηση. Αρχικά, υλοποιείται ένα μη προ-εκπαιδευμένο δικτύου αποκλειστικά για το τρέχον πρόβλημα ταξινόμησης. Στη συνέχεια, διερευνώνται τα προ-εκπαιδευμένα νευρωνικά δίκτυα που είναι διαθέσιμα στη βιβλιοθήκη Keras της Tensorflow.

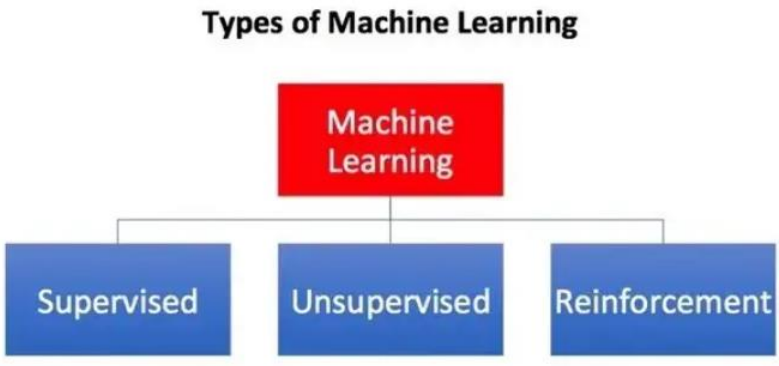
2. Νευρωνικά Δίκτυα (Neural Networks)

Τα τεχνητά νευρωνικά δίκτυα (artificial neural networks) είναι αλγόριθμοι εμπνευσμένοι από τους βιολογικούς νευρώνες μέσα στο ανθρώπινο σώμα. Αποτελούν υποκατηγορία της μηχανικής μάθησης (machine learning) και εμπεριέχουν τα βαθιά νευρωνικά δίκτυα (deep neural networks) ή βαθιά μάθηση

(deep learning). Όλα τα παραπάνω είναι υποκατηγορίες ενός εκτεταμένου πεδίου που ονομάζουμε τεχνητή νοημοσύνη (artificial intelligence).

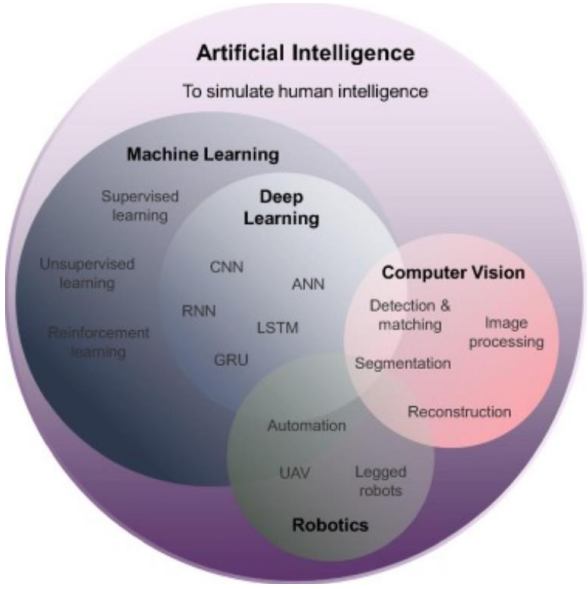


Το πρόβλημα της ταξινόμησης ανήκει στην κατηγορία επιβλεπόμενης μηχανικής μάθησης (supervised machine learning), επειδή οι κλάσεις των εικόνων είναι γνωστές, έχουμε δηλαδή τις ετικέτες (labels).

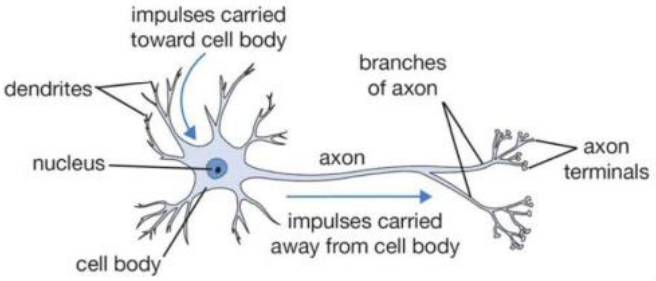


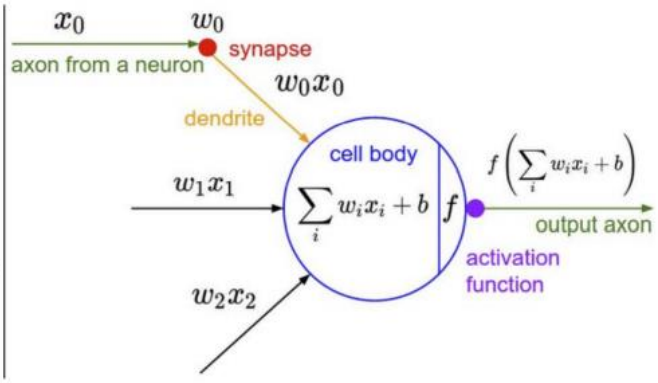
Τα νευρωνικά δίκτυα που χρησιμοποιούνται για την ταξινόμηση εικόνων, ονομάζονται συνελικτικά νευρωνικά δίκτυα (convolutional neural networks). Είναι στην ουσία υποκατηγορία των νευρωνικών δικτύων που χρησιμοποιούνται κατά

κόρον στην όραση υπολογιστών, σε μια πληθώρα εφαρμογών όπως φαίνεται και στην παρακάτω εικόνα.



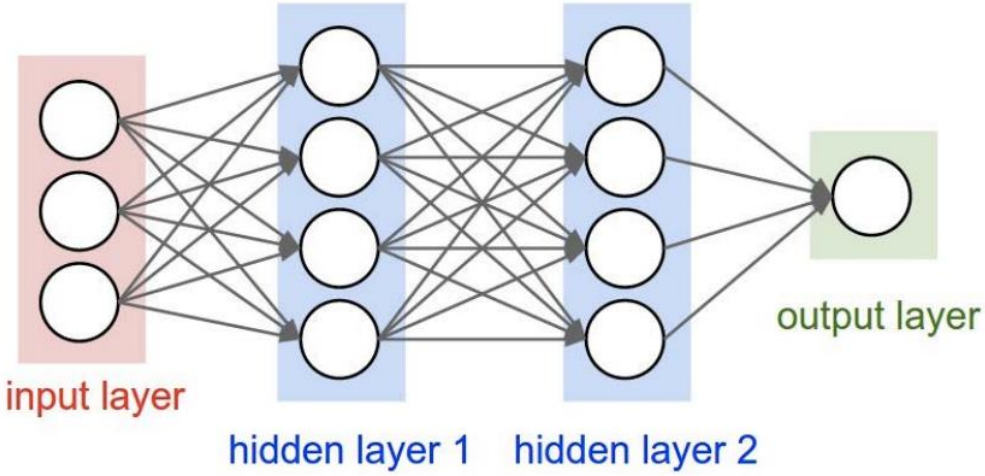
Ο τεχνητός νευρώνας είναι εμπνευσμένος από το βιολογικό νευρώνα του ανθρώπινου εγκεφάλου. Ο συσχετισμός αποτυπώνεται στην παρακάτω εικόνα και στον παρακάτω πίνακα.





Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από ένα επίπεδο εισόδου (input layer), ένα επίπεδο εξόδου (output layer) και κάποια κρυμμένα επίπεδα (hidden layers).



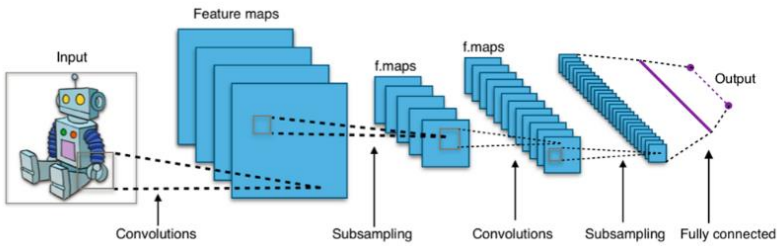
Στα προβλήματα ταξινόμησης, το επίπεδο εξόδου έχει τόσους νευρώνες, όσες και οι κλάσεις που θέλουμε να ταξινομήσουμε.

3. Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)

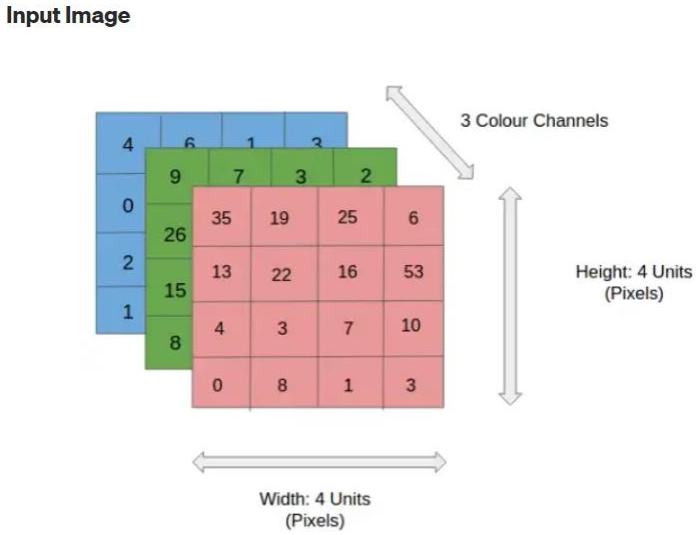
Τα Συνελικτικά Νευρωνικά Δίκτυα (CNN) είναι μια κατηγορία τεχνητών νευρωνικών δικτύων (ANN), που χρησιμοποιείται πιο συχνά για την ανάλυση οπτικών εικόνων. Ένα CNN μπορεί να λάβει μια εικόνα στην είσοδο, να αποδώσει σημασία σε διάφορα αντικείμενα της εικόνας και να διαφοροποιήσει το ένα από το άλλο, ώστε να κατανοήσει το περιεχόμενό της. Είναι ιδιαίτερα χρήσιμα στην εύρεση μοτίβων σε οπτικές εικόνες για την αναγνώριση αντικειμένων και κλάσεων.

Τα CNN χρησιμοποιούν σχετικά λίγη προ-επεξεργασία σε σύγκριση με άλλους αλγόριθμους ταξινόμησης εικόνων. Το δίκτυο μαθαίνει να βελτιστοποιεί τα φίλτρα (kernels) μέσω της αυτοματοποιημένης εκμάθησης, ενώ στους παραδοσιακούς αλγόριθμους αυτά τα φίλτρα είναι κατασκευασμένα με το χέρι.

Σε ένα CNN τα κρυμμένα επίπεδα (layers) εκτελούν λειτουργίες με σκοπό την εκμάθηση χαρακτηριστικών. Τα πιο κοινά επίπεδα είναι η συνέλιξη (convolution) και η υποδειγματοληψία (pooling/subsampling).

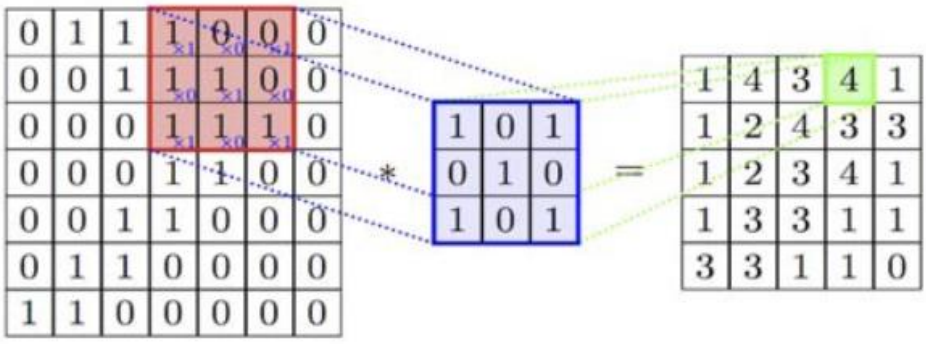


Ένα συνελικτικό επίπεδο (conv2D), κυρίως χρησιμοποιείται για την εξαγωγή χαρακτηριστικών από την εικόνα εισόδου. Τα πρώτα στρώματα είναι υπεύθυνα για την καταγραφή χαρακτηριστικών χαμηλού επιπέδου. Με πρόσθετα επίπεδα συνέλιξης, η αρχιτεκτονική προσπαθεί να εξάγει χαρακτηριστικά υψηλού επιπέδου.

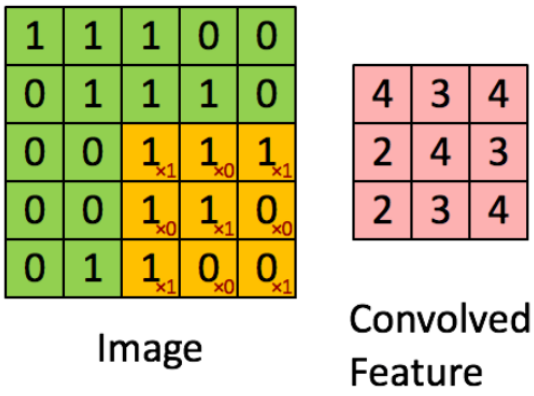


Τα φίλτρα Conv2D εκτείνονται μέσω των τριών καναλιών της εικόνας (κόκκινο-πράσινο-μπλε, RGB). Τα φίλτρα είναι διαφορετικά για κάθε κανάλι.

Στην παρακάτω εικόνα η κόκκινη περιοχή εμφανίζει την περιοχή του τρέχοντος φίλτρου που υπολογίζεται και ονομάζεται πεδίο υποδοχής (receptive field). Οι αριθμοί στο φίλτρο ονομάζονται βάρη ή παράμετροι. Το φίλτρο ολισθαίνει σε ολόκληρη την εικόνα και υπολογίζει τους πολλαπλασιασμούς στοιχείο προς στοιχείο και γεμίζει τον πίνακα εξόδου, ο οποίος ονομάζεται χάρτης ενεργοποίησης ή χάρτης χαρακτηριστικών (feature map).



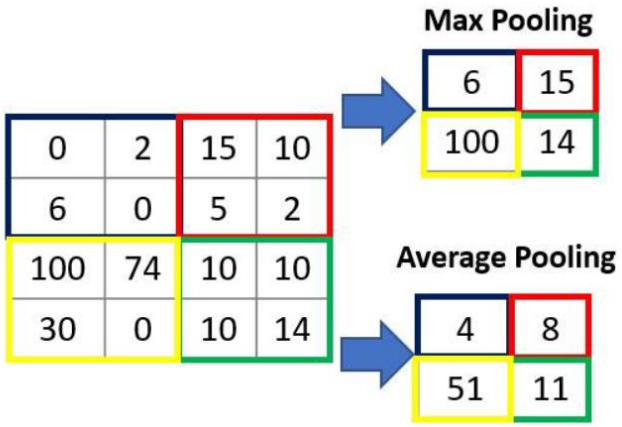
Στην παρακάτω εικόνα μπορούμε να δούμε άλλο ένα παράδειγμα συνέλιξης.



Ανάμεσα στα επίπεδα συνέλιξης συνήθως υπάρχουν επίπεδα συγκέντρωσης ή αλλιώς υποδειγματοληψία (pooling layers).

Οι χάρτες χαρακτηριστικών τροφοδοτούνται στο επίπεδο υποδειγματοληψίας. Αυτό το επίπεδο μειώνει προοδευτικά το χωρικό μέγεθος της αναπαράστασης. Μειώνει τον αριθμό των παραμέτρων και την ποσότητα υπολογισμού στο δίκτυο. Το στρώμα συγκέντρωσης ελέγχει επίσης την υπερπροσαρμογή (overfitting).

Υπάρχουν δύο κύριοι τύποι επιπέδων υποδειγματοληψίας: η υποδειγματοληψία μέγιστης τιμής (max pooling) επιστρέφει τη μέγιστη τιμή από το συγκεκριμένο τμήμα και η υποδειγματοληψία μέσης τιμής (average pooling) επιστρέφει τον μέσο όρο όλων των τιμών από το συγκεκριμένο τμήμα.

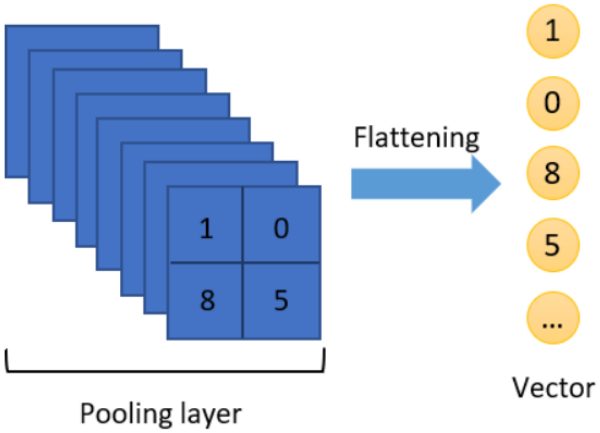


Η υποδειγματοληψία μέγιστης τιμής αποδίδει πολύ καλύτερα από την υποδειγματοληψία μέσης τιμής και είναι το πιο κοινό επίπεδο υποδειγματοληψίας που εφαρμόζεται.

Έπειτα από αλλεπάλληλα επίπεδα συνέλιξης και υποδειγματοληψίας και εφόσον έχουν εξαχθεί τα χαρακτηριστικά υψηλού επιπέδου από την εικόνα εισόδου,

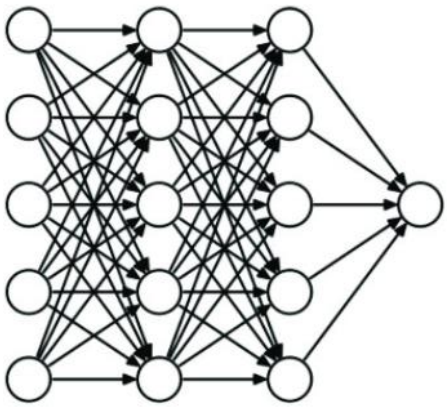
μπορούμε να χρησιμοποιήσουμε ένα πλήρες συνδεδεμένο επίπεδο (fully connected layer/dense layer, FCL) για να συσχετίσουμε τον τελικό χάρτη χαρακτηριστικών με την κατηγορία στην οποία αντιστοιχεί η εικόνα.

Ωστόσο, πριν τροφοδοτήσουμε τον τελικό χάρτη χαρακτηριστικών στο πλήρες συνδεδεμένο επίπεδο, χρειαζόμαστε ένα ενδιάμεσο επίπεδο για να μετατρέψουμε τη διάσταση των δεδομένων σε 1xn, κατάλληλη δηλαδή για ταξινόμηση με το πλήρες συνδεδεμένο επίπεδο. Αυτό το ενδιάμεσο επίπεδο ονομάζεται επίπεδο ισοπέδωσης (Flatten) και μετατρέπει τους πολυδιάστατους πίνακες σε ένα ενιαίο μεγάλο συνεχές γραμμικό διάνυσμα.



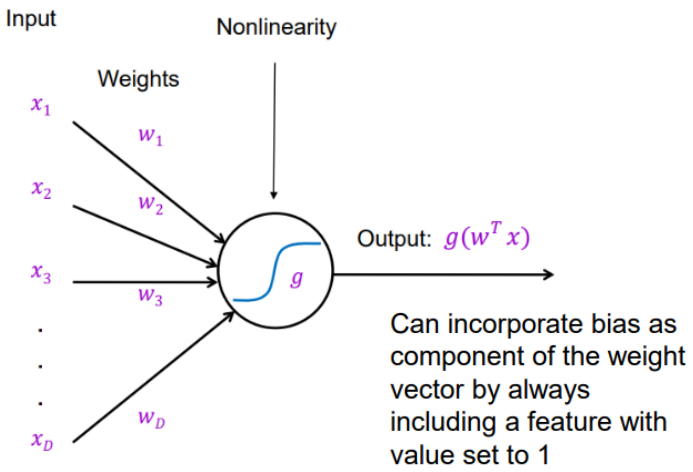
Σε ένα πλήρες συνδεδεμένο επίπεδο κάθε νευρώνας λαμβάνει είσοδο από όλους τους νευρώνες του προηγούμενου επιπέδου. Το επίπεδο έχει έναν πίνακα βάρους W (Weight matrix), ένα διάνυσμα μεροληψίας b (bias vector) και την ενεργοποίηση του προηγούμενου στρώματος a (activation function).

Παρακάτω φαίνεται ένα πλήρες συνδεδεμένο νευρωνικό δίκτυο (Fully Connected/Dense Neural Network).

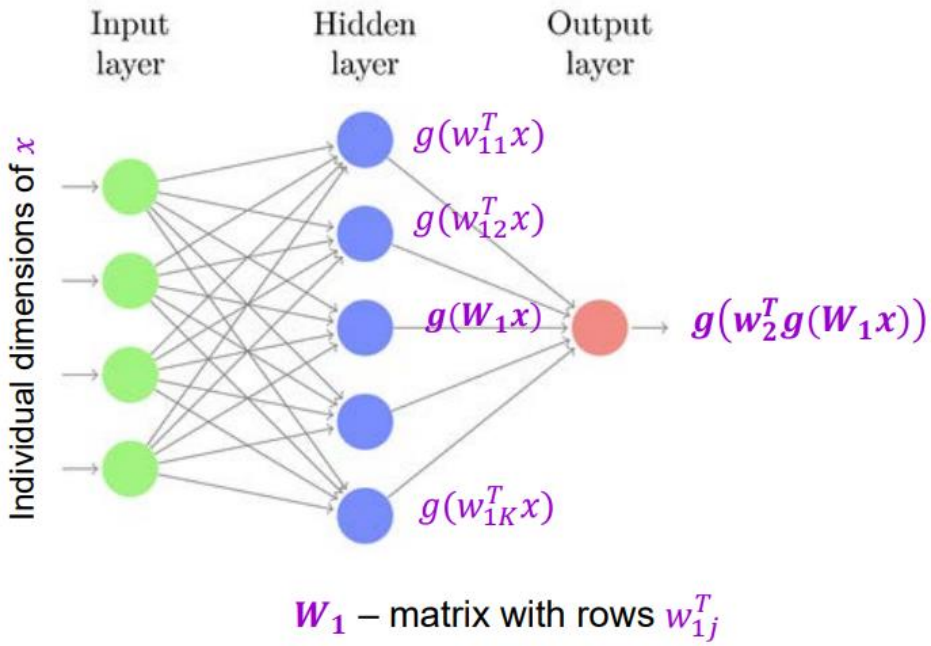


Το πλήρες συνδεδεμένο επίπεδο υλοποιεί τη λειτουργία:

output = activation(dot(input, weight) + bias)



Στην παρακάτω εικόνα φαίνεται ο υπολογισμός για ένα ολόκληρο επίπεδο του δικτύου.

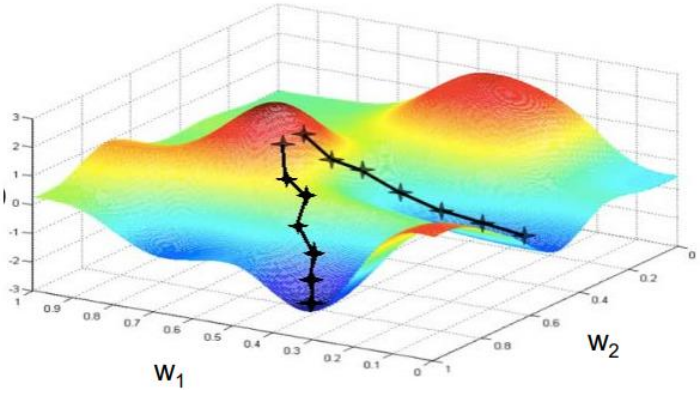


Τα βάρη και το σταθερός όρος b, δηλαδή τις παραμέτρους του δικτύου τις μαθαίνει το δίκτυο με τη μέθοδο της εκπαίδευσης (training). Αρχικά υπολογίζεται η έξοδος του δικτύου (προς τα εμπρός διάδοση, forward propagation) με τυχαίες τιμές W και b. Στη συνέχεια, εφόσον οι ετικέτες (labels) είναι γνωστές (supervised learning), συγκρίνουμε την έξοδο που υπολογίσαμε με την πραγματική έξοδο του δικτύου και υπολογίζουμε το σφάλμα (error), με τη χρήση μιας συνάρτησης κόστους (cost function). Τέλος, γίνεται διόρθωση των W και b με την τεχνική της οπισθοδιάδοσης (backpropagation) με στόχο την ελαχιστοποίηση του κόστους, δηλαδή του σφάλματος. Η επικαιροποίηση των W και b γίνεται με τον αλγόριθμο κατάβαση/κάθοδος κλίσης (gradient descent).

Συνάρτηση κόστους: $$E(\mathbf{w}) = \sum_i l(\mathbf{x}_i, y_i; \mathbf{w})$$

Επικαιροποίηση βαρών: $$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$

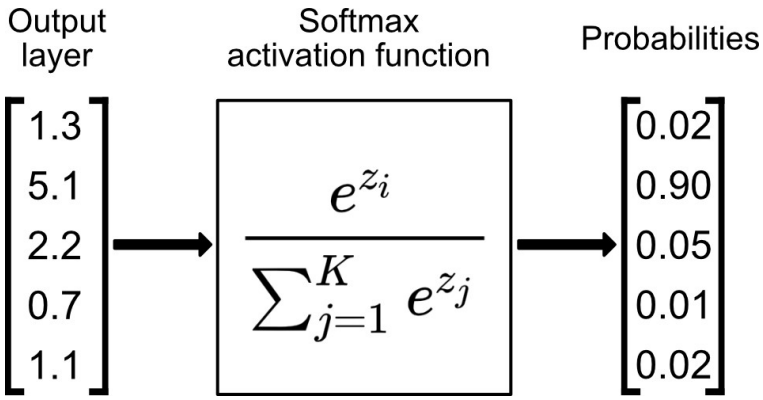
Στην παρακάτω εικόνα φαίνεται ένα παράδειγμα του αλγορίθμου gradient descent για την επικαιροποίηση δύο παραμέτρων W1, W2.



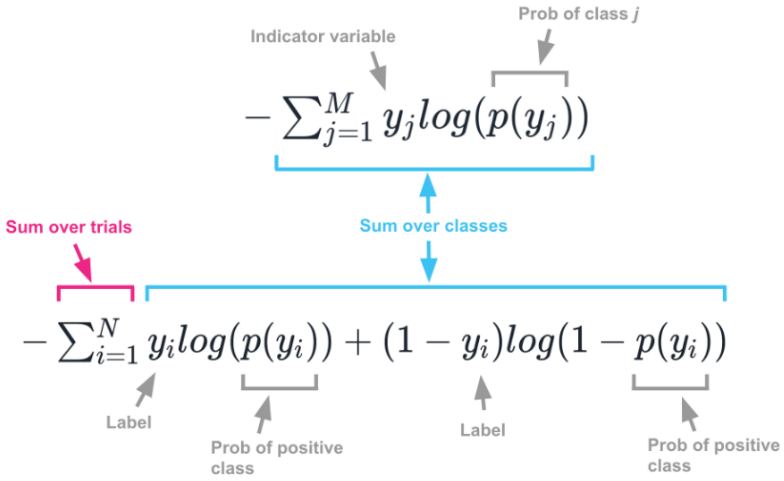
Όπως αναφέραμε παραπάνω η χρήση πλήρως συνδεδεμένου επιπέδου γίνεται για τον υπολογισμό της κατηγορίας στην οποία ανήκει η εικόνα εισόδου, δηλαδή για την ταξινόμηση της εικόνας. Για να επιτύχουμε την ταξινόμηση, όταν έχουμε πολλές κλάσεις, χρησιμοποιούμε σαν συνάρτηση ενεργοποίησης του πλήρους συνδεδεμένου επιπέδου, τη softmax. Με τη συνάρτηση αυτή για n κλάσεις, λαμβάνουμε n πιθανότητες. Η κλάση με την υψηλότερη πιθανότητα, αντιστοιχεί στην κλάση που έχουμε προβλέψει ότι ανήκει η εικόνα εισόδου. Αν ο ταξινομητής μας είναι ο βέλτιστος, τότε η πιθανότητα μιας κλάσης θα είναι κοντά στο 1 και οι υπόλοιπες πιθανότητες θα είναι κοντά στο 0.

$$\text{softmax}(f_1, \ldots, f_c) = \left(\frac{\exp(f_1)}{\sum_j \exp(f_j)}, \ldots, \frac{\exp(f_c)}{\sum_j \exp(f_j)} \right)$$

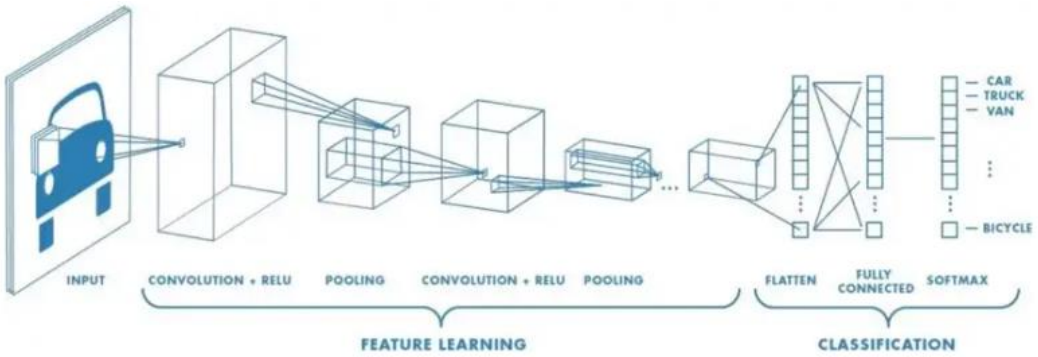
Παρακάτω φαίνεται η μετατροπή των τιμών εξόδου του δικτύου σε πιθανότητες, μέσω της συνάρτησης ενεργοοποίησης. Έχουμε 5 κλάσεις και η εικόνα εισόδου με βάση την μεγαλύτερη πιθανότητα, ανήκει νστην κλάση 2 (πιθανότητα 0.90).



Σαν συνάρτηση κόστους χρησιμοποιείται η cross-entropy.



Παρακάτω φαίνεται η αρχιτεκτονική ενός συνελικτικού δικτύου που περιγράψαμε προηγουμένως.



4. Αρχιτεκτονική Συνελικτικού Δικτύου

Η αρχιτεκτονική συνελικτικού δικτύου που χρησιμοποιείται για το τρέχον πρόβλημα ταξινόμησης είναι η εξής:

Εφαρμόζουμε τρία επίπεδα συνέλιξης (convolution) και στη συνέχεια ένα επίπεδο υποδειγματοληψίας μέγιστης τιμής (max pooling). Αυτό γίνεται τέσσερεις φορές και κάθε φορά, διπλασιάζουμε τον αριθμό νευρώνων που χρησιμοποιούνται στα τριπλά επίπεδα convolution, ξεκινώντας από 32 (στη συνέχεια 64, 128, 256).

Συνέλιξη (Convolution)

Παρατηρούμε ότι με τη χρήση πολλαπλών επιπέδων συνέλιξης βελτιώνεται πάρα πολύ η απόδοση του δικτύου.

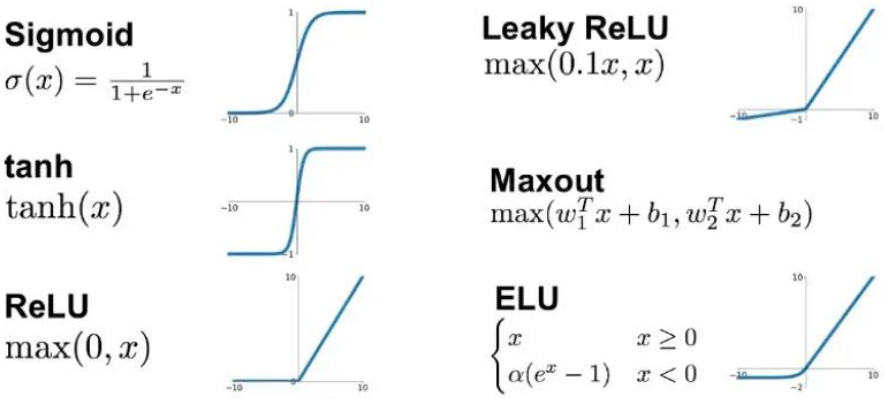
Υποδειγματοληψία (Pooling)

Στα επίπεδα υποδειγματοληψίας, χρησιμοποιείται υποδειγματοληψία μέγιστης τιμής (max pooling). Θα μπορούσε να χρησιμοποιηθεί και υποδειγματοληψία μέσης τιμής (average pooling), όμως το max pooling στο συγκεκριμένο πρόβλημα φαίνεται ότι αποδίδει καλύτερα.

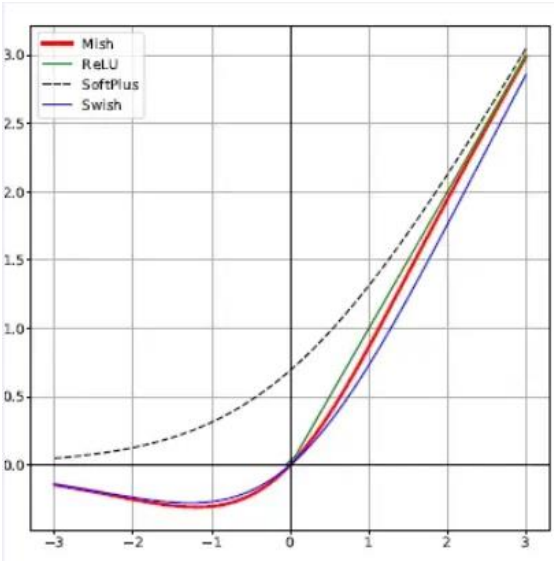
Συνάρτηση ενεργοποίησης (Activation function)

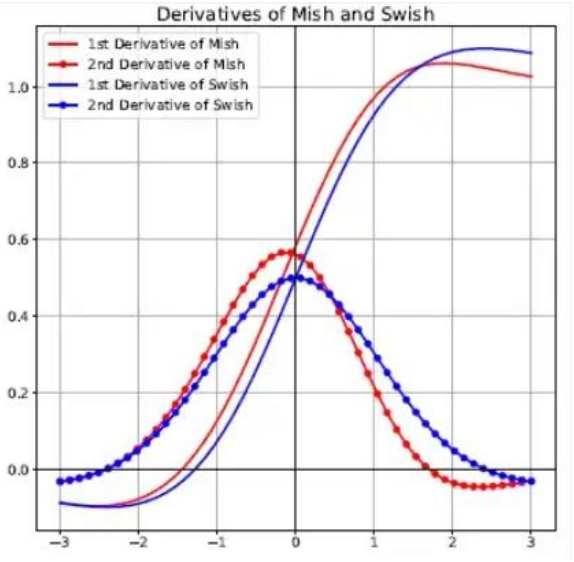
Η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η Rectified Linear Unit (ReLU).

Άλλες συναρτήσεις που θα μπορούσαν να χρησιμοποιηθούν ή και δοκιμάστηκαν, χωρίς κάποια ιδιαίτερη επιτυχία είναι οι παρακάτω:



Κάποιες συναρτήσεις ενεργοποίησης πιο πρόσφατες που συνήθως βελτιώνουν αρκετά την απόδοση του δικτύου είναι οι Softplus, Swish, Mish.



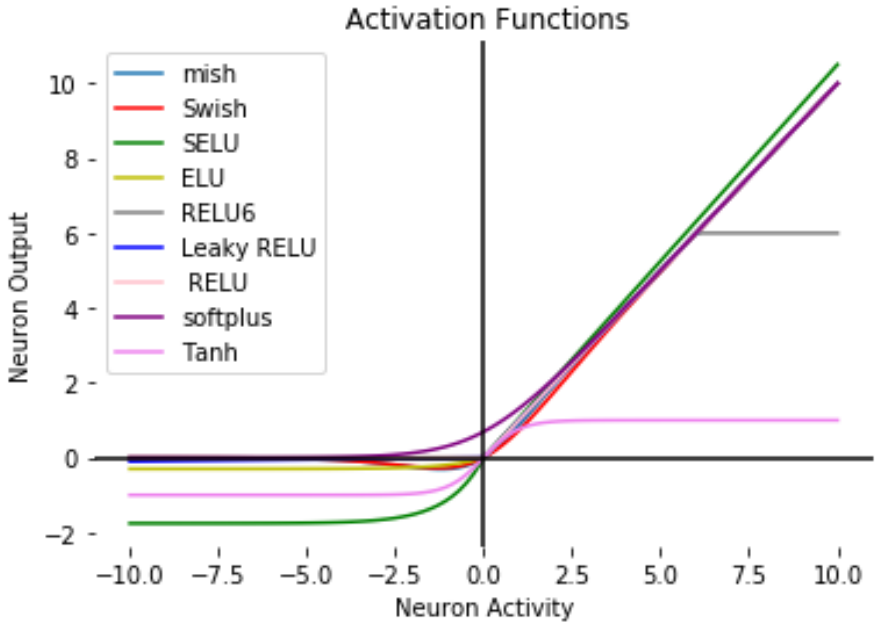


Softplus $f(x) = \ln(1 + e^x)$

Swish $f(x) = x \cdot \sigma(x)$

Mish $f(x) = x \tanh(\text{softplus}(x))$

Οι διαφορετικές συναρτήσεις μεταφοράς φαίνονται στην παρακάτω γραφική παράσταση, στην οποία μπορούμε να τις συγκρίνουμε μεταξύ τους.



Ισοπέδωση (Flatten)

Μετά τα πρώτα 16 επίπεδα convolution και max pooling ακολουθεί ένα επίπεδο ισοπέδωσης (Flatten), το οποίο μετατρέπει τους πολυδιάστατους πίνακες σε ένα ενιαίο μεγάλο συνεχές γραμμικό διάνυσμα, το οποίο στη συνέχεια θα τροφοδοτήσουμε σε ένα πλήρες συνδεδεμένο επίπεδο.

Πλήρες συνδεδεμένο επίπεδο (Dense layer/Fully connected layer)

Χρησιμοποιούμε για την ταξινόμηση, ένα πλήρες συνδεδεμένο επίπεδο, με τόσους νευρώνες όσες και οι κλάσεις που έχουμε και συνάρτηση ενεργοποίησης τη softmax, που χρησιμοποιείται σε προβλήματα ταξινόμησης πολλαπλών κλάσεων, όπως αναφέραμε παραπάνω.

Θα μπορούσαμε να χρησιμοποιήσουμε πολλαπλά dense layers πριν το τελικό επίπεδο. Με αυτόν τον τρόπο τα dense layers επεξεργάζονται περισσότερο τα feature maps που έχουν εξαχθεί από τα επίπεδα συνέλιξης και σε πολλές περιπτώσεις μπορεί να βελτιώσουν την απόδοση του δικτύου.

Βελτιστοποιητές (Optimizers)

Οι βελτιστοποιητές είναι αλγόριθμοι που χρησιμοποιούνται για την ελαχιστοποίηση μιας συνάρτησης απώλειας (loss function). Κάθε βελτστοποιητής μεταβάλλει με διαφορετικό τρόπο τα βάρη και τον ρυθμό εκμάθησης του νευρωνικού δικτύου για την ελαχιστοποίηση των απωλειών (σφάλματος).

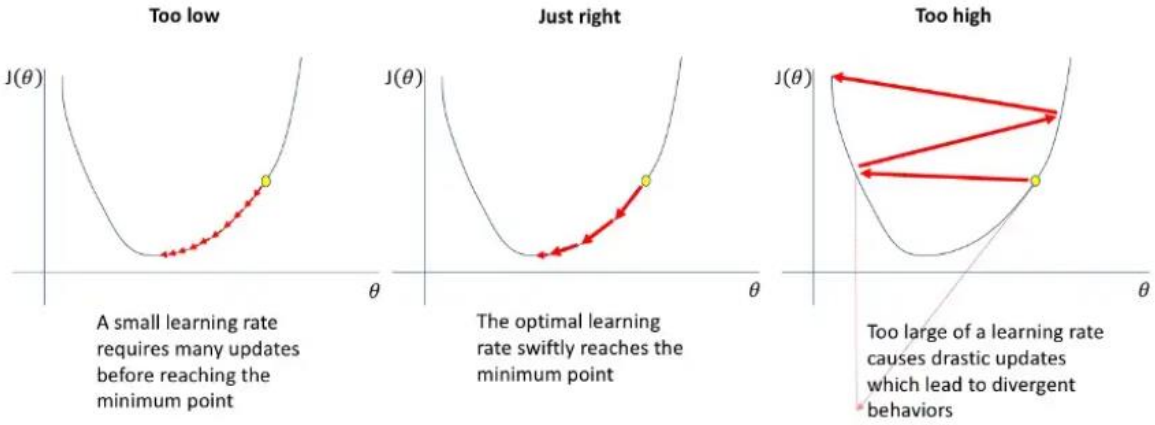
Ο αλγόριθμος κατάβασης/καθόδου κλίσης (Gradient Descent ή Batch Gradient Descent) είναι ένας βελτιστοποιητής, στον οποίο οι παράμετροι του μοντέλου ενημερώνονται στο τέλος κάθε εποχής (το σφάλμα υπολογίζεται για το σύνολο των δεδομένων εκπαίδευσης). Μία παραλλαγή του gradient descent είναι ο στοχαστικός αλγόριθμος καθόδου κλίσης (Stochastic Gradient Descent), στον οποίο οι παράμετροι του μοντέλου ενημερώνονται μία προς μία (το σφάλμα υπολογίζεται για κάθε ένα δείγμα του συνόλου των δεδομένων εκπαίδευσης). Επίσης, μία άλλη παραλλαγή είναι ο Mini-Batch Gradient Descent, στον οποίο οι παράμετροι του μοντέλου ενημερώνονται, με βάση το σφάλμα που υπολογίζεται για ένα μικρό υποσύνολο (mini-batch) των δεδομένων.

Άλλοι βελτιστοποιητές είναι οι SGD with Momentum, AdaGrad (Adaptive Gradient Descent), RMS-Prop (Root Mean Square Propagation), Adam(Adaptive Moment Estimation), κ.λπ.

Ο βελτιστοποιητής Adam είναι ένας από τους πιο δημοφιλείς και διάσημους αλγόριθμους βελτιστοποίησης. Δίνει πολύ καλά αποτελέσματα σε πάρα πολλά προβλήματα νευρωνικών δικτύων. Είναι μια μέθοδος στην οποία ο ρυθμός εκμάθησης (learning rate) είναι μεταβλητός. Για παράδειγμα, στην αρχή θέλουμε ο ρυθμός εκμάθησης να είναι μεγάλος για να οδηγηθούμε γρήγορα σε

ελαχιστοποίηση της συνάρτησης απωλειών. Όμως στη συνέχεια, θέλουμε ο ρυθμός εκμάθησης να μειωθεί, ώστε να μην ξεπεράσουμε κάποιο τοπικό ελάχιστο και οδηγηθούμε σε ταλαντώσεις. Οπότε, ο ρυθμός εκμάθησης προσαρμόζεται, κατά τη διάρκεια της εκπαίδευσης.

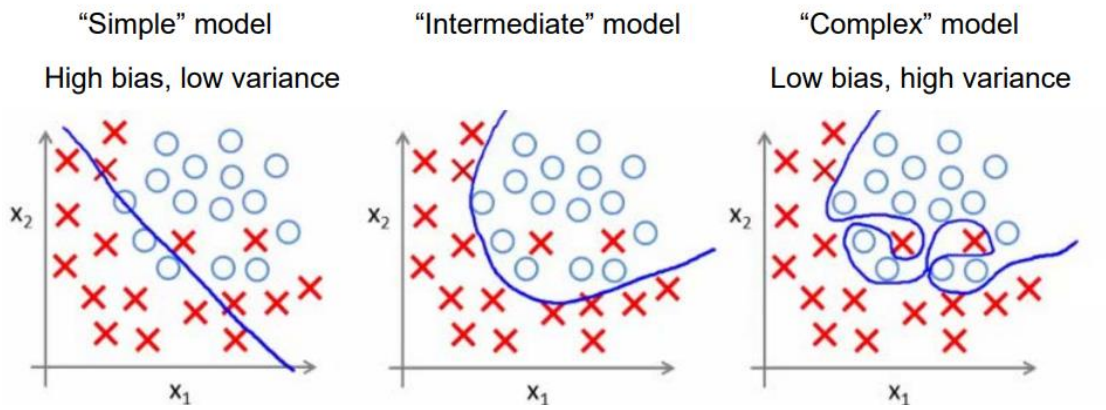
Στο συγκεκριμένο πρόβλημα ταξινόμησης, επιλέξαμε να χρησιμοποιήσουμε τον Adam.



Συνάρτησης απώλειας (loss function)

Μια συνάρτηση απώλειας είναι μια συνάρτηση που συγκρίνει τον στόχο (πραγματική τιμή) και τις προβλεπόμενες τιμές εξόδου και μετρά την απόδοση του νευρωνικού δικτύου. Κατά την εκπαίδευση, στοχεύουμε να ελαχιστοποιήσουμε την απώλεια μεταξύ των προβλεπόμενων και των πραγματικών αποτελεσμάτων.

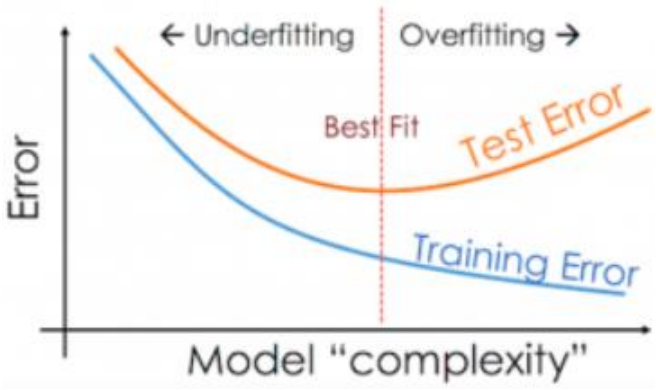
Κατά το validation και το testing, μπορούμε να δούμε πως μεταβάλλεται η απώλεια και να μεταβάλλουμε τις υπερπαραμέτρους του δικτύου, ώστε να μειωθεί περαιτέρω το σφάλμα, δηλαδή η απώλεια.



Για παράδειγμα, μεγάλη απώλεια και στην εκπαίδευση και στην δοκιμή μας δείχνει ότι το νευρωνικό δίκτυο δεν μπορεί να συλλάβει τα σημαντικά χαρακτηριστικά των δεδομένων εκπαίδευσης (underfitting), διάγραμμα στα αριστερά. Για την αντιμετώπιση αυτού του προβλήματος αρκεί να κάνουμε πιο πολύπλοκο το δίκτυο,

δηλαδή να χρησιμοποιήσουμε περισσότερα επίπεδα (layers) και περισσότερους νευρώνες ανά επίπεδο.

Αν η απώλεια είναι πολύ μικρή στην εκπαίδευση και στην επικύρωση, αλλά πολύ μεγαλύτερη στη δοκιμή, τότε το μοντέλο μαθαίνει τόσο καλά τα δεδομένα, που καταλήγει να μαθαίνει θόρυβο (overfitting). Το μοντέλο δεν μπορεί να γενικεύσει), διάγραμμα στα δεξιά. Για την αντιμετώπιση αυτού του προβλήματος αρκεί να κάνουμε το δίκτυο πιο απλό ή να χρησιμοποιήσουμε κανονικοποίηση (regularization) των δεδομένων.



Στην εποπτευόμενη μάθηση, υπάρχουν δύο κύριοι τύποι συναρτήσεων απώλειας που συσχετίζονται με τους δύο κύριους τύπους νευρωνικών δικτύων: παλινδρόμηση (regression) και ταξινόμηση (classification).

Στην παλινδρόμηση συνήθως χρησιμοποιούνται το μέσο τετραγωνικό σφάλμα και το μέσο απόλυτο σφάλμα. Στην ταξινόμηση χρησιμοποιούνται η Δυαδική Διασταυρούμενη Εντροπία (Binary Cross-Entropy) για δύο κλάσεις και η Κατηγορική Διασταυρούμενη Εντροπία (Categorical Cross-Entropy) για πολλές κλάσεις.

Εφόσον, το πρόβλημα που έχουμε περιλαμβάνει πολλαπλές κλάσεις, χρησιμοποιούμε σαν συνάρτηση απωλειών την Categorical Cross-Entropy.

Τέλος, η συνάρτηση κόστους (cost function) μετρά το σφάλμα του μοντέλου σε μια ομάδα δεδομένων, ενώ η συνάρτηση απώλειας (loss function) ασχολείται με ένα μόνο δείγμα των δεδομένων. Έτσι, εάν L είναι η συνάρτηση απώλειας, τότε υπολογίζουμε τη συνάρτηση κόστους αθροίζοντας την απώλεια L στα δεδομένα εκπαίδευσης, επικύρωσης ή δοκιμής. Για παράδειγμα, μπορούμε να υπολογίσουμε το κόστος ως τη μέση απώλεια:

$$Cost(f, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L\left(\hat{y}_i, y_i\right) \qquad (\hat{y}_i = f(\mathbf{x}_i))$$

5. Τεχνικές για βελτίωση της απόδοσης

Παρακάτω αναφέρονται ορισμένες τεχνικές που μπορούν να αυξήσουν την απόδοση του δικτύου.

Επαύξηση δεδομένων (Data augmentation)

Η επαύξηση δεδομένων είναι ένα σύνολο τεχνικών για την τεχνητή αύξηση της ποσότητας δεδομένων με τη δημιουργία νέων δεδομένων από τα υπάρχοντα. Αυτό περιλαμβάνει την πραγματοποίηση μικρών αλλαγών στα δεδομένα εκπαίδευσης ή τη χρήση μοντέλων βαθιάς μάθησης για τη δημιουργία νέων σημείων δεδομένων.

Εμείς θα εστιάσουμε στην πραγματοποίηση κάποιων μετασχηματισμών για την επαύξηση του συνόλου δεδομένων μας. Με την εφαρμογή κατάλληλου είδους και ποσότητας μετασχηματισμών στις εικόνες είναι δυνατόν να αυξηθεί η απόδοση του δικτύου. Παρακάτω αναφέρουμε ορισμένους μετασχηματισμούς που μπορούν να πραγματοποιηθούν με τη χρήση της κλάσης Image Data Generators.

Μετατόπιση (Shift)

Μπορεί το αντικείμενο να μην βρίσκεται πάντα στο κέντρο της εικόνας. Για να ξεπεράσουμε αυτό το πρόβλημα μπορούμε να μετατοπίσουμε τα εικονοστοιχεία της εικόνας είτε οριζόντια είτε κάθετα. Αυτό γίνεται προσθέτοντας μια ορισμένη σταθερή τιμή σε όλα τα εικονοστοιχεία.

Η κλάση ImageDataGenerator έχει το όρισμα height_shift_range για κατακόρυφη μετατόπιση εικόνας και width_shift_range για οριζόντια μετατόπιση εικόνας. Εάν η τιμή είναι δεκαδικός αριθμός (float), αυτό θα υποδείκνυε το ποσοστό πλάτους ή ύψους της εικόνας προς μετατόπιση.

Εμείς επιλέξαμε width_shift_range=0.1 και height_shift_range=0.1 για να ήμαστε σίγουροι ότι κατά τη μετατόπιση το αντικείμενο δε θα φύγει εκτός της εικόνας.

Περιστροφή (Rotation)

Η περιστροφή εικόνας είναι μία από τις ευρέως χρησιμοποιούμενες τεχνικές επαύξησης και επιτρέπει στο μοντέλο να γίνει αμετάβλητο ως προς τον προσανατολισμό του αντικειμένου.

Η κλάση ImageDataGenerator έχει το όρισμα rotation_range για την περιστροφή της εικόνας σε οποιονδήποτε βαθμό μεταξύ 0 και 360 μοιρών, ο οποίος καθορίζεται από έναν ακέραιο αριθμό.

Εμείς επιλέξαμε rotation_range=15 για να ήμαστε σίγουροι ότι κατά την περιστροφή, ορισμένες εικόνες δε θα αλλάξουν κλάση. Εφόσον το σύνολο δεδομένων περιλαμβάνει εικόνες σημάτων κυκλοφορίας, αν περιστρέψουμε πάρα πολύ κάποια εικόνα είναι δυνατόν το σήμα να αναγνωριστεί ότι ανήκει σε

λανθασμένη κλάση. Για παράδειγμα, τα σήματα υποχρεωτικής κατεύθυνσης εξαρτάται η σημασία τους από τον προσανατολισμό τους.

Όταν η εικόνα περιστρέφεται, μερικά εικονοστοιχεία θα μετακινηθούν εκτός της εικόνας και θα αφήσουν μια κενή περιοχή που πρέπει να συμπληρωθεί. Ο κενός χώρος μπορεί να συμπληρωθεί με διαφορετικούς τρόπους, όπως μια σταθερή τιμή ή οι πλησιέστερες τιμές εικονοστοιχείων, κ.λπ. Αυτό καθορίζεται με το όρισμα fill_mode και η προεπιλεγμένη τιμή είναι "nearest" που απλώς συμπληρώνει την κενή περιοχή με τις πλησιέστερες τιμές εικονοστοιχείων.

Αναστροφή (Flip)

Η αναστροφή εικόνων είναι επίσης μια τεχνική επαύξησης.

Η κλάση ImageDataGenerator έχει τις παραμέτρους horizontal_flip και vertical_flip για αναστροφή κατά μήκος του οριζόντιου ή του κατακόρυφου άξονα.

Αυτή η τεχνική δε χρησιμοποιείται, γιατί όπως είπαμε παραπάνω, εφόσον το σύνολο δεδομένων περιλαμβάνει εικόνες σημάτων κυκλοφορίας, αν αναστρέψουμε ένα σήμα είναι δυνατόν να αναγνωριστεί ότι ανήκει σε λανθασμένη κλάση, όπως τα δύο παρακάτω σήματα τα οποία ανήκουν σε διαφορετικές κλάσεις.

Επίσης, πολλά σήματα είναι συμμετρικά κατά τον οριζόντιο άξονα, οπότε μια οριζόντια αναστροφή δε θα επέφερε κάποια βελτίωση.

Φωτεινότητατα (Brightness)

Αλλάζει τυχαία τη φωτεινότητα της εικόνας. Είναι επίσης μια πολύ χρήσιμη τεχνική επαύξησης γιατί τις περισσότερες φορές το αντικείμενό μας δεν θα βρίσκεται σε

τέλεια κατάσταση φωτισμού. Έτσι, καθίσταται επιτακτική η ανάγκη να εκπαιδεύσουμε το μοντέλο μας σε εικόνες υπό διαφορετικές συνθήκες φωτισμού.

Η φωτεινότητα μπορεί να ελεγχθεί στην κλάση ImageDataGenrator μέσω του ορίσματος brightness_range, το οποίο αποδέχεται μια λίστα με δύο δεκαδικές τιμές (float) και επιλέγει μια τιμή μετατόπισης φωτεινότητας από αυτό το εύρος. Τιμές μικρότερες από 1.0 κάνουν την εικόνα πιο σκοτεινή, ενώ τιμές πάνω από 1.0 φωτίζουν την εικόνα.

Βέβαια, με τη χρήση αυτής της τεχνικής στο σύνολο δεδομένων μας δεν παρατηρήθηκε ιδιαίτερη βελτίωση, οπότε αποφασίσαμε να μην τη χρησιμοποιήσουμε.

Μεγέθυνση (Zoom)

Η κλάση ImageDataGenerator παίρνει μια τιμή float για μεγέθυνση στο όρισμα zoom_range. Αποδέχεται επίσης μια λίστα με δύο τιμές που καθορίζουν το κατώτερο και το ανώτερο όριο. Διαφορετικά, εάν καθορίσουμε τιμή float, τότε η μεγέθυνση θα γίνει στο εύρος [1-εύρος_zoom,1+εύρος_ζουμ]. Οποιαδήποτε τιμή μικρότερη από 1 θα μεγεθύνει την εικόνα. Ενώ οποιαδήποτε τιμή μεγαλύτερη από 1 θα σμικρύνει την εικόνα.

Εμείς επιλέξαμε zoom_range=0.1. Επιλέξαμε μία μικρή τιμή για να ήμαστε σίγουροι ότι το αντικείμενο αν ήδη είναι πολύ μεγάλο δεν θα περικοπεί.

Όλες τις παραπάνω τιμές τις επιλέξαμε μετά από δοκιμές. Παρατηρήσαμε ότι όσο μεγαλύτερες τιμές χρησιμοποιούσαμε στις τεχνικές επαύξησης, τόσο έπεφτε η απόδοση του δικτύου. Με μικρές τιμές υπήρχε μικρή βελτίωση στην απόδοση.

Σύνολο δεδομένων επικύρωσης (Validation data set)

Το σύνολο δεδομένων μας αποτελείται από το σύνολο δεδομένων εκπαίδευσης και δοκιμής. Θα ήταν χρήσιμο να χωρίσουμε το σύνολο εκπαίδευσης και να πάρουμε ένα μικρό κομμάτι του για να δημιουργήσουμε το σύνολο δεδομένων επικύρωσης.

Το ποσοστό του συνόλου εκπαίδευσης που θα καταλαμβάνει το σύνολο επικύρωσης καθορίζεται από το όρισμα validation_split της κλάσης ImageDataGenerator.

Η τιμή που επιλέχθηκε είναι validation_split=0.2, δηλαδή το σύνολο επικύρωσης καταλαμβάνει το 20% του συνόλου εκπαίδευσης.

Το σύνολο επικύρωσης χρησιμοποιείται για την παροχή μιας αμερόληπτης αξιολόγησης ενός μοντέλου κατά την ρύθμιση των υπερπαραμέτρων του μοντέλου. Το μοντέλο βλέπει περιστασιακά αυτά τα δεδομένα, αλλά δεν μαθαίνει από αυτά. Χρησιμοποιούμε τα αποτελέσματα του συνόλου επικύρωσης για να ενημερώσουμε

τις υπερπαραμέτρους του δικτύου. Το σύνολο επικύρωσης είναι επίσης γνωστό ως σύνολο ανάπτυξης (Dev set ή Development set). Αυτό είναι λογικό αφού βοηθά στο στάδιο ανάπτυξης του μοντέλου.

Πρώιμη διακοπή (Early stopping)

Κατά την εκπαίδευση ενός δικτύου, υπάρχει ένα σημείο κατά τη διάρκεια της εκπαίδευσης που το μοντέλο θα σταματήσει να γενικεύει και θα αρχίσει να μαθαίνει τον στατιστικό θόρυβο στο σύνολο δεδομένων εκπαίδευσης. Αυτή η υπερπροσαρμογή (overfitting) του συνόλου δεδομένων εκπαίδευσης θα οδηγήσει σε σφάλματα κατά τη γενίκευση, καθιστώντας το μοντέλο λιγότερο χρήσιμο για την πραγματοποίηση προβλέψεων σε νέα δεδομένα.

Η πρόκληση είναι να εκπαιδεύσουμε το δίκτυο για αρκετές εποχές, αλλά όχι τόσο πολύ, ώστε να υπερπροσαρμοστεί στα δεδομένα εκπαίδευσης. Μία εποχή είναι ένα πλήρες πέρασμα των δεδομένων από το δίκτυο.

Μια προσέγγιση για την επίλυση αυτού του προβλήματος είναι να αντιμετωπίζεται ο αριθμός των εποχών εκπαίδευσης ως υπερπαράμετρος, να εκπαιδεύεται το μοντέλο πολλές φορές με διαφορετικές τιμές και στη συνέχεια να επιλέγεται ο αριθμός των εποχών που έχουν ως αποτέλεσμα την καλύτερη απόδοση σε ένα σύνολο δοκιμής. Το μειονέκτημα αυτής της προσέγγισης είναι ότι απαιτεί πολλαπλά μοντέλα να εκπαιδευτούν και να απορριφθούν. Αυτό μπορεί να είναι υπολογιστικά αναποτελεσματικό και χρονοβόρο.

Μια εναλλακτική προσέγγιση είναι να εκπαιδεύσουμε το μοντέλο μία φορά για μεγάλο αριθμό εποχών. Κατά τη διάρκεια της εκπαίδευσης, το μοντέλο αξιολογείται σε ένα σύνολο δεδομένων επικύρωσης, μετά από κάθε εποχή. Εάν η απόδοση του μοντέλου στο σύνολο δεδομένων επικύρωσης αρχίσει να υποβαθμίζεται (π.χ. η απώλεια αρχίζει να αυξάνεται ή η ακρίβεια αρχίζει να μειώνεται), τότε η διαδικασία εκπαίδευσης διακόπτεται. Στη συνέχεια χρησιμοποιείται το μοντέλο από τη στιγμή που διακόπτεται η εκπαίδευση και είναι γνωστό ότι επιτυγχάνει τη γενίκευση.

Αυτή η διαδικασία ονομάζεται πρώιμη διακοπή (early stopping).

Παρακάτω βλέπουμε το σημείο στο οποίο ορίσαμε την πρώιμη διακοπή.

```
# Early stopping
early_stopping = EarlyStopping(monitor='val_loss', mode='min', re
store_best_weights=True, patience=25, verbose=1)
```

Με το όρισμα monitor, παρακολουθούμε την τιμή του σφάλματος επικύρωσης (validation loss). Όσο αυτή η τιμή μειώνεται το μοντέλο μας εκπαιδεύεται ομαλά. Επειδή μας ενδιαφέρει η ελάχιστη τιμή, στο όρισμα mode βάζουμε min. Το όρισμα patience=25, δηλώνει ότι αν για 25 εποχές δεν μειωθεί το σφάλμα επικύρωσης, τότε το δίκτυο δεν μπορεί να βελτιωθεί με περισσότερη εκπαίδευση, οπότε η εκπαίδευση διακόπτεται, πριν φτάσουμε τον αριθμό των εποχών που ορίσαμε αρχικά. Συνήθως ο αριθμός εποχών επιλέγεται να είναι μεγάλος, εμείς για

παράδειγμα ορίσαμε 250 εποχές. Τέλος, με το restore_best_weights=True, όταν φτάσουμε σε πρώιμη διακοπή, τα βάρη γίνονται ίσα με την τιμή που είχαν στην εποχή με το χαμηλότερο σφάλμα επικύρωσης (δηλαδή εδώ τα βάρη γίνονται ίσα με την τιμή που είχαν 25 εποχές πριν).

Την τιμή patience την επιλέξαμε μετά από δοκιμές, ώστε να μην διακόπτεται πολύ νωρίς η εκπαίδευση, πριν επιτύχουμε το χαμηλότερο δυνατό σφάλμα επικύρωσης.

Checkpoints

Ένας τρόπος να αποθηκεύσουμε τα βάρη του μοντέλου μας, όταν έχουμε ρυθμίσει κατάλληλα τις υπερπαραμέτρους είναι με τη χρήση σημείων ελέγχου (checkpoints).

```
# Checkpoints
checkpoint_filepath = '/content/gdrive/MyDrive/Colab Notebooks/Belgian Traffic Sign Best Model/my_belgian_traffic_signs.hdf5'
model_checkpoint = ModelCheckpoint(filepath=checkpoint_filepath, monitor='val_loss', mode='min', save_best_only=True)
```

Στο τέλος κάθε εποχής, αν το σφάλμα επικύρωσης είναι μικρότερο από πριν, αποθηκεύεται ένα checkpoint στο φάκελο checkpoint_filepath, αυτό μας το δείχνει το όρισμα save_best_only=True. Μόλις ολοκληρωθεί η εκπαίδευση, έχουμε κρατήσει το μοντέλο με την καλύτερη απόδοση. Οπότε, μπορούμε χωρίς να επαναλάβουμε ολόκληρη τη διαδικασία της εκπαίδευσης να φορτώσουμε το μοντέλο από το φάκελο και να το χρησιμοποιήσουμε για προβλέψεις.

```
# Checkpoints
checkpoint_filepath = '/content/gdrive/MyDrive/Colab Notebooks/Belgian Traffic Sign Best Model/my_belgian_traffic_signs.hdf5'
best_model = load_model(checkpoint_filepath)
```

Χρονοδιαγράμματα ρυθμού εκμάθησης (Learning rate schedulers)

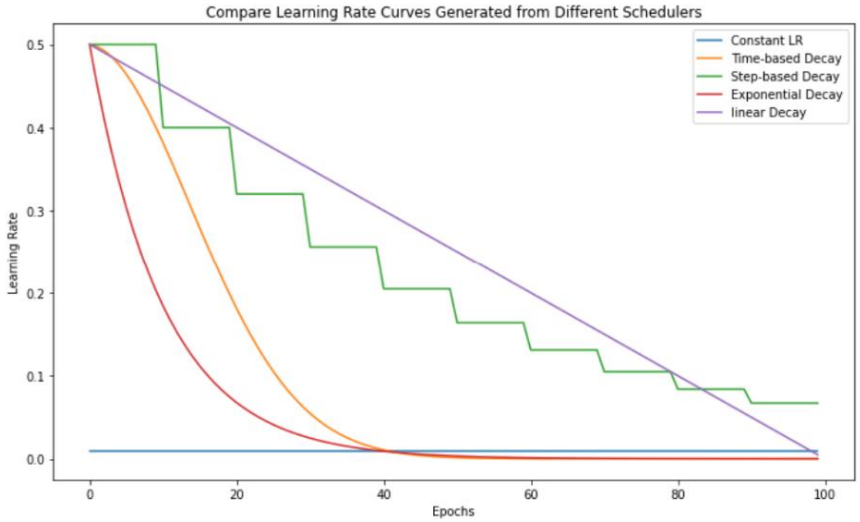
Κατά την εκπαίδευση, είναι συχνά χρήσιμο να μειώνεται ο ρυθμός εκμάθησης καθώς προχωρά η εκπαίδευση. Αυτό μπορεί να γίνει χρησιμοποιώντας προκαθορισμένα χρονοδιαγράμματα ρυθμού εκμάθησης (learning rate schedules) ή προσαρμοστικές μεθόδους ρυθμού εκμάθησης (adaptive learning rate methods).

Για τις προσαρμοστικές μεθόδους ρυθμού εκμάθησης έχουμε αναφερθεί προηγουμένως και είπαμε ότι σαν βελτιστοποιητής χρησιμοποιείται ο Adam.

Τα χρονοδιαγράμματα ρυθμού εκμάθησης επιδιώκουν να προσαρμόσουν τον ρυθμό εκμάθησης (learning rate) κατά τη διάρκεια της εκπαίδευσης, μειώνοντάς τον σύμφωνα με ένα προκαθορισμένο πρόγραμμα. Κάποια χρονοδιαγράμματα ρυθμών εκμάθησης (learning rate schedules) είναι τα time-based decay, step decay and exponential decay.

Στην αρχή της εκπαίδευσης, ο ρυθμός εκμάθησης έχει οριστεί να είναι μεγάλος, προκειμένου να φτάσει σε ένα σύνολο βαρών που δίνουν σχετικά καλή απόδοση. Με την πάροδο του χρόνου, αυτά τα βάρη προσαρμόζονται ώστε να επιτυγχάνουν υψηλότερη ακρίβεια αξιοποιώντας ένα μικρό ποσοστό εκμάθησης.

Παρακάτω φαίνονται ορισμένοι ρυθμοί εκμάθησης με χρονοδιαγράμματα, κατά τη διάρκεια της εκπαίδευσης.



Μια πολύ γνωστή κλάση που χρησιμοποιείται για να επιβάλλει στο ρυθμό εκμάθησης ένα χρονοδιάγραμμα είναι η ReduceLROnPlateau. Ανήκει στην κατηγορία step-based decay. Αυτό σημαίνει ότι ο ρυθμός εκμάθησης είναι σταθερός για ένα συγκεκριμένο αριθμό εποχών και στη συνέχεια μειώνεται σε κάποια άλλη τιμή για έναν ακόμα αριθμό εποχών. Στο διάγραμμα φαίνεται σαν μία σκάλα, δηλαδή κάθε φορά που μειώνεται ο ρυθμός εκμάθησης δημιουργείται ένα σκαλοπάτι.

```
# Scheduler
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_los
s', factor=0.5, patience=5, min_lr=1e-6)
```

Κάθε 5 εποχές αν δεν μειωθεί το σφάλμα επικύρωσης, ο ρυθμός εκμάθησης γίνεται learning_rate = learning_rate*factor. Εδώ στην ουσία ο ρυθμός εκμάθησης μειώνεται στο μισό. Τα ορίσματα patience και factor αποτελούν υπερπαραμέτρους, τις οποίες καθορίζουμε μετά από δοκιμές.

Batch size (Batch/Stochastic/Mini-Batch Gradient Descent)

Το μέγεθος παρτίδας (batch size) είναι ο αριθμός των δειγμάτων εκπαίδευσης που πρέπει να επεξεργαστούν πριν από την ενημέρωση των εσωτερικών παραμέτρων του μοντέλου.

Με δοκιμές ανάμεσα στις τιμές 32, 64, 128, 256, επιλέξαμε σαν βέλτιστη τιμή το 64.

Μέγεθος εισόδου (Input size)

Στο πρώτο επίπεδο συνέλιξης ορίζεται το μέγεθος των εικόνων που θα δέχεται το δίκτυό μας. Έχουμε input_shape=(IMG_HEIGHT, IMG_WIDTH, IMG_COLOR). Το IMG_COLOR = 3, επειδή οι εικόνες μας έχουν 3 κανάλια (RGB). Τα IMG_HEIGHT, IMG_WIDTH τα ορίζουμε 256 (χρησιμοποιήσαμε τη default τιμή). Οι εικόνες του συνόλου δεδομένων μας έχουν διαφορετικό μέγεθος, οπότε μέσω της εντολής flow_from_directory, ορίζουμε target_size=(IMG_HEIGHT, IMG_WIDTH), αλλάζουμε το μέγεθος των εικόνων (resize) και τις μετατρέπουμε όλες σε μέγεθος IMG_HEIGHT X IMG_WIDTH. Παρατηρήσαμε επίσης ότι αν μειώσουμε το input size σε 128 X 128, το μη προεκπαιδευμένο δίκτυο πετυχαίνει λίγο υψηλότερη ακρίβεια, όμως στο προεκπαιδευμένο η ακρίβεια μειώνεται αρκετά. Οπότε για να μπορούμε να συγκρίνουμε τα δύο δίκτυα επιλέξαμε το 256 X 256.

Κανονικοποίηση (Regularization/Normalization)

Όλες οι μέθοδοι κανονικοποίησης (Regularization) συμβάλλουν στην αποφυγή της υπερπροσαρμογής (overfitting).

Επανακλιμάκωση (Rescale)

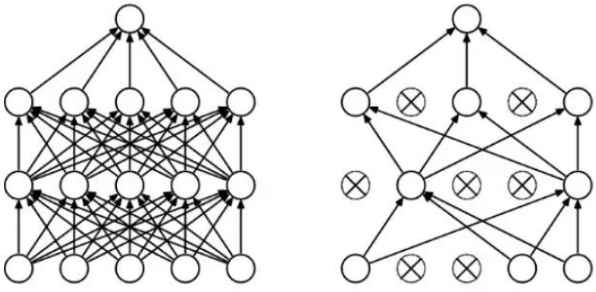
Η μέγιστη τιμή που μπορεί να πάρει ένα εικονοστοιχείο μιας εικόνας είναι η τιμή 255. Οπότε μπορούμε να διαιρέσουμε κάθε εικονοστοιχείο, κάθε εικόνας εισόδου με το 255, ώστε κάθε τιμή εισόδου να κυμαίνεται στο εύρος 0 έως 1. Αν οι τιμές εισόδου κυμαίνονται στο ίδιο εύρος, έχει αποδειχθεί ότι βελτιώνεται η απόδοση του δικτύου.

Με το όρισμα rescale της κλάσης ImageDataGenerator πολλαπλασιάζουμε τα δεδομένα με την τιμή που παρέχεται, δηλαδή με 1/255, άρα στην ουσία διαιρούμε με την τιμή 255.

Dropout

Το Dropout είναι μια μέθοδος, η οποία συμβάλει στην αποφυγή της υπερπροσαρμογής (overfitting).

Κατά τη διάρκεια της εκπαίδευσης, ορισμένοι νευρώνες απορρίπτονται τυχαία με κάποια πιθανότητα, σε κάθε εποχή. Όλες οι συνδέσεις προς τα εμπρός και προς τα πίσω με έναν κόμβο που έχει απορριφθεί αφαιρούνται προσωρινά, δημιουργώντας έτσι μια νέα αρχιτεκτονική δικτύου, διαφορετική από του αρχικού. Οι κόμβοι απορρίπτονται με πιθανότητα απόρριψης p.



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

Στο πρόβλημα υπερπροσαρμογής, το μοντέλο μαθαίνει τον στατιστικό θόρυβο. Για την ακρίβεια, το κύριο κίνητρο της εκπαίδευσης είναι η μείωση της συνάρτησης απώλειας, δεδομένων όλων των μονάδων (νευρώνες). Όμως αν το μοντέλο προσαρμοστεί πάρα πολύ καλά στα δεδομένα εκπαίδευσης, μπορεί να αποτυγχάνει να γενικευτεί στο σύνολο δεδομένων δοκιμής.

Εάν χρησιμοποιήσουμε το dropout, διασφαλίζεται ότι το μοντέλο γενικεύεται και συνεπώς μειώνει το πρόβλημα υπερπροσαρμογής.

Επειδή το δίκτυό μας είναι σχετικά μικρό για τα δεδομένα που έχουμε, δεν υπάρχει μεγάλο πρόβλημα υπερπροσαρμογής. Οπότε, με το dropout δε φάνηκε να υπάρχει κάποια βελτίωση και γι' αυτό δε χρησιμοποιήθηκε στο τελικό δίκτυο.

BatchNormalization

Το batchnormalization είναι μια μέθοδος κανονικοποίησης που συμβάλει στην αποφυγή της υπερπροσαρμογής και στο πρόβλημα των exploding gradients (όταν οι παράγωγοι γίνονται πολύ μεγάλες).

Οι παράμετροι εκπαίδευσης είναι η μέση τιμή (mean a, c) και η τυπική απόκλιση (std b, d), των κατανομών εισόδου και προβολής.

Αρχικά από τις τιμές εισόδου, αφαιρείται το mean και διαιρείται το αποτέλεσμα με το std, για να φέρουμε τα δεδομένα στο ίδιο εύρος τιμών.

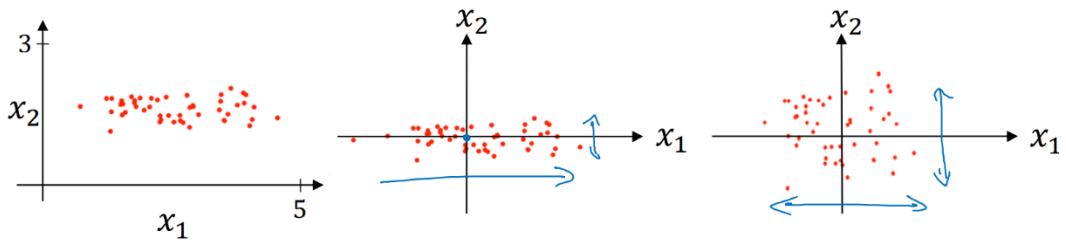
$$\textbf{Normalization Operation: } y_1 = \frac{x-a}{b}$$

Στη συνέχεια, για να προβάλουμε τα δεδομένα, πολλαπλασιάζουμε με το std και προσθέτουμε το mean.

$$\textbf{Projection Operation } y = y1 * d + c$$

Αυτή η νέα προβολή έχει αποδειχτεί ότι συμβάλει στην αύξηση της απόδοσης του δικτύου.

Ένα παράδειγμα φαίνεται στην παρακάτω εικόνα.



Με το BatchNormalization δε φάνηκε να υπάρχει κάποια βελτίωση και γι' αυτό δε χρησιμοποιήθηκε στο τελικό δίκτυο.

Κανονικοποίηση Βαρών (Weight regularization)

Ένας τρόπος για να τιμωρήσουμε την πολυπλοκότητα που μπορεί να οδηγήσει σε υπερπροσαρμογή, θα ήταν να προσθέσουμε όλες τις παραμέτρους (βάρη) στη συνάρτηση απώλειας. Αυτό δεν θα λειτουργήσει, επειδή ορισμένες παράμετροι είναι θετικές και ορισμένες είναι αρνητικές. Μπορούμε να προσθέσουμε τα τετράγωνα όλων των παραμέτρων στη συνάρτηση απώλειας, όμως η απώλειά μας μπορεί γίνει τόσο μεγάλη που το καλύτερο μοντέλο θα ήταν να ορίσουμε όλες τις παραμέτρους στο 0. Για να μην συμβεί αυτό, πολλαπλασιάζουμε το άθροισμα των τετραγώνων με έναν άλλο μικρότερο αριθμό. Αυτός ο αριθμός ονομάζεται weight decay και συχνά συμβολίζεται με λ.

Στον παρακάτω πίνακα μπορούμε να δούμε τις μεθόδους κανονικοποίησης L1 και L2.

L1 Regularization	L2 Regularization
1. L1 penalizes sum of absolute values of weights.	1. L2 penalizes sum of square values of weights.
2. L1 generates model that is simple and interpretable.	2. L2 regularization is able to learn complex data patterns.
3. L1 is robust to outliers.	3. L2 is not robust to outliers.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i| \qquad Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Το λ το αντιμετωπίζουμε σαν υπερπαράμετρο που θέλουμε να βελτιστοποιήσουμε για το πρόβλημά μας, μέσω δοκιμών. Όσο μεγαλύτερη η τιμή του, τόσο μεγαλύτερη κανονικοποίηση πραγματοποιείται. Αν το παρακάνουμε και αυξήσουμε το λ πάρα

πολύ, υπάρχει κίνδυνος να έχουμε το αντίθετο πρόβλημα της υπερπροσαρμογής, δηλαδή το underfitting.

Με την τεχνική Weight regularization δε φάνηκε να υπάρχει κάποια ιδιαίτερη βελτίωση και γι' αυτό δε χρησιμοποιήθηκε στο τελικό δίκτυο.

6. Διαδικασία καθορισμού των υπερπαραμέτρων και της αρχιτεκτονικής

Για τη βελτιστοποίηση των υπερπαραμέτρων και την τελική αρχιτεκτονική που επιλέξαμε, εκτελέσαμε μια σειρά δοκιμών, στις οποίες όποτε υπήρχε βελτίωση κρατούσαμε τα στοιχεία που δοκιμάσαμε και έτσι διαμορφώσαμε το τελικό μας δίκτυο.

Αρχικά, ξεκινήσαμε από εναλλαγή επιπέδων συνέλιξης και υποδειγματοληψίας και στη συνέχεια προχωρήσαμε σε ακόμη πιο περίπλοκα δίκτυα, όπως για παράδειγμα τα τρία επίπεδα συνέλιξης στη σειρά στα οποία καταλήξαμε κιόλας. Δοκιμάσαμε και πιο περίπλοκα δίκτυα, όμως σε αυτά παρουσιαζόταν πολύ έντονα το πρόβλημα της υπερπροσαρμογής, παρ' όλο που εφαρμόστηκαν μέθοδοι κανονικοποίησης.

Επίσης, δοκιμάστηκε ο αριθμός των νευρώνων των επιπέδων. Η προσθήκη περισσότερων πλήρων διασυνδεδεμένων επιπέδων μετά το flatten, δε προσέφερε κάποια ιδιαίτερη βελτίωση.

Τέλος, δοκιμάστηκαν διαφορετικές συναρτήσεις ενεργοποίησης και βελτιστοποιητές.

Η χρήση συναρτήσεων callbacks μας βοήθησε να ελέγξουμε πιο αποτελεσματικά την εκπαίδευση του δικτύου και να αποθηκεύσουμε τις παραμέτρους του. Καθοριστική ήταν η χρήση του scheduler που με κατάλληλη ρύθμιση των υπερπαραμέτρων του, βελτίωσε αρκετά την απόδοση του μοντέλου.

Οτιδήποτε πιο σύνθετο δεν οδηγούσε σε κάποια αξιόλογη βελτίωση της απόδοσης του δικτύου, επιλέξαμε να μην το χρησιμοποιήσουμε.

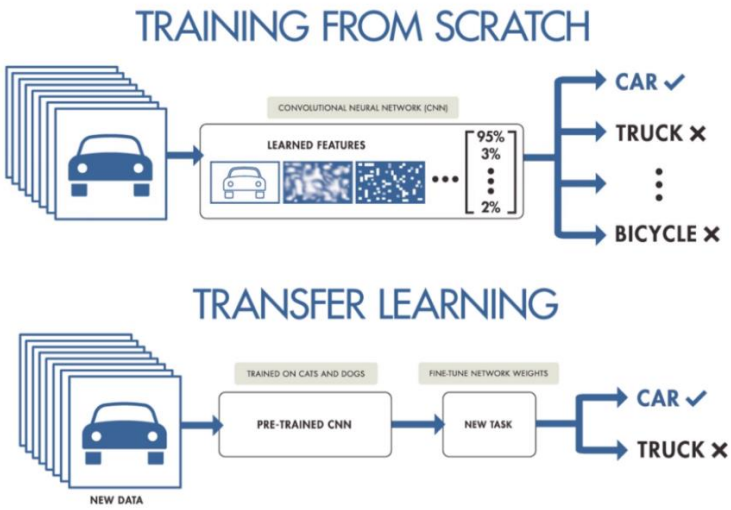
Επιπλέον, δοκιμάστηκαν και διάφοροι μέθοδοι κανονικοποίησης, οι οποίες δε βοήθησαν πολύ. Σε γενικές γραμμές το δίκτυό μας είναι σχετικά μικρό και απλό για τα δεδομένα μας, οπότε δεν είναι τόσο φανερό το πρόβλημα της υπερπροσαρμογής. Αυτός είναι και ο λόγος που η κανονικοποίηση δεν βοηθάει πολύ. Σε περίπτωση που δοκιμάσουμε ένα πιο πολύπλοκο δίκτυο (περισσότερα επίπεδα συνέλιξης, νευρώνες, πλήρως συνδεδεμένα επίπεδα, κ.λπ), πιθανώς με τη χρήση μεθόδων κανονικοποίησης να έχουμε πολύ μεγάλη βελτίωση.

7. Χρήση Προ-εκπαιδευμένων Νευρωνικών Δικτύων

Το Keras περιέχει προεκπαιδευμένα μοντέλα ταξινόμησης εικόνων τα οποία εκπαιδεύονται σε δεδομένα του Imagenet. Το Imagenet είναι μια μεγάλη συλλογή εικόνων που περιέχει 1000 κατηγορίες. Αυτά τα προεκπαιδευμένα μοντέλα είναι σε θέση να ταξινομήσουν οποιαδήποτε εικόνα εμπίπτει σε αυτές τις 1000 κατηγορίες.

Το προεκπαιδευμένο δίκτυο περιέχει συνελικτικά επίπεδα που ακολουθούνται από πλήρως συνδεδεμένα επίπεδα. Τα επίπεδα συνέλιξης εξάγουν χαρακτηριστικά από την εικόνα και τα πλήρως συνδεδεμένα επίπεδα ταξινομούν την εικόνα χρησιμοποιώντας τα χαρακτηριστικά που έχουν εξαχθεί.

Όταν εκπαιδεύουμε ένα CNN σε δεδομένα εικόνας, φαίνεται ότι τα ανώτερα στρώματα του δικτύου μαθαίνουν να εξάγουν γενικά χαρακτηριστικά από εικόνες όπως άκρες, κατανομή χρωμάτων κ.λπ. Καθώς συνεχίζουμε να πηγαίνουμε βαθιά στο δίκτυο, τα επίπεδα τείνουν να εξάγουν περισσότερα εξειδικευμένα χαρακτηριστικά. Οπότε, μπορούμε να χρησιμοποιήσουμε αυτά τα προεκπαιδευμένα μοντέλα που ήδη γνωρίζουν πώς να εξάγουν χαρακτηριστικά και να αποφύγουμε την εκπαίδευση από την αρχή. Αυτή η έννοια είναι γνωστή ως μεταφερόμενη μάθηση (transfer learning).



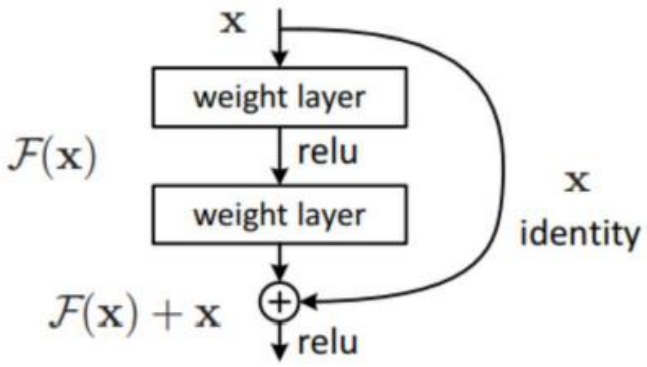
Πριν εκπαιδεύσουμε το δίκτυο, μπορούμε να παγώσουμε (freeze) μερικά από τα επίπεδά του. Με τον όρο freeze εννοούμε ότι τα βάρη του δεν ενημερώνονται κατά τη διαδικασία της εκπαίδευσης. Συχνά είναι χρήσιμο να παγώνουμε τα αρχικά επίπεδα που συμβάλουν στην εξαγωγή γενικών χαρακτηριστικών και να εκπαιδεύουμε μόνο τα τελευταία επίπεδα που συμβάλουν στην εξαγωγή ειδικών χαρακτηριστικών και στη διαδικασία ταξινόμησης.

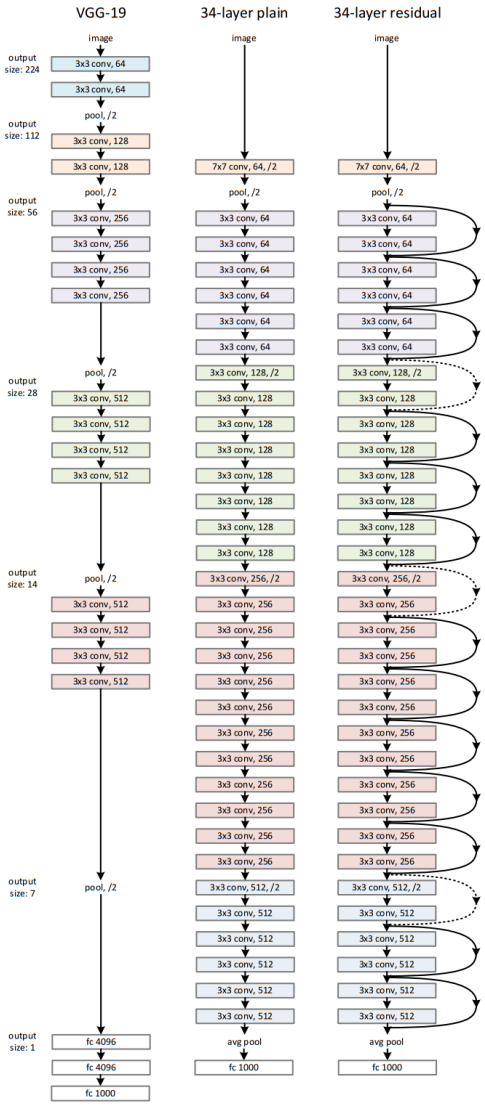
Στον παρακάτω πίνακα μπορούμε να δούμε τα διαθέσιμα προεκπαιδευμένα δίκτυα που υπάρχουν στο Keras.

Module	DL Model Functions
densenet	DenseNet121(), DenseNet169(), DenseNet201()
efficientnet	EfficientNetB0(), EfficientNetB1(), EfficientNetB2(), EfficientNetB3(), EfficientNetB4(), EfficientNetB5(), EfficientNetB6(), EfficientNetB7()
inception_resnet_v2	InceptionResNetV2()
inception_v3	InceptionV3()
mobilenet	MobileNet()
mobilenet_v2	MobileNetV2()
nasnet	NASNetLarge(), NASNetMobile()
resnet	ResNet101(), ResNet152(),
resnet50	ResNet50()
resnet_v2	ResNet101V2(), ResNet152V2(), ResNet50V2()
vgg16	VGG16()
vgg19	VGG19()
xception	Xception()

Το ResNet σημαίνει Residual Network και είναι ένας συγκεκριμένος τύπος συνελικτικού νευρωνικού δικτύου (CNN) που εισήχθη το 2015 στο paper «Deep Residual Learning for Image Recognition» από τους He Kaiming, Zhang Xiangyu, Ren Shaoqing και Sun Jian.

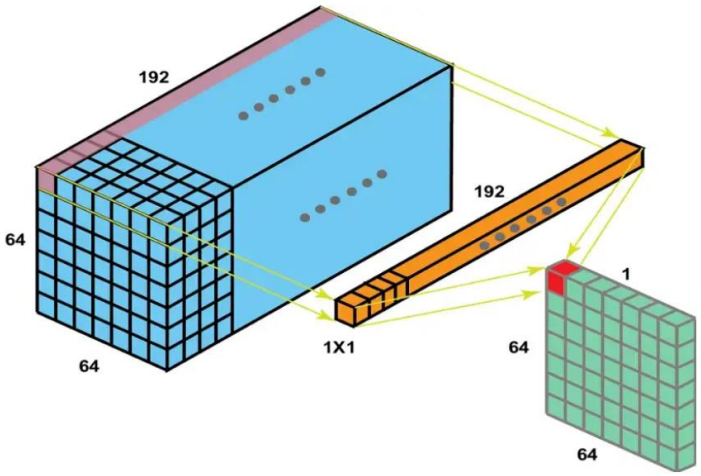
Η αρχική αρχιτεκτονική του ResNet ήταν το ResNet-34, το οποίο περιλάμβανε 34 σταθμισμένα επίπεδα. Παρείχε έναν νέο τρόπο προσθήκης περισσότερων συνελικτικών επιπέδων σε ένα CNN, χωρίς να υπάρχει το πρόβλημα μηδενισμού της παραγώγου (vanishing gradient), χρησιμοποιώντας την έννοια των ταυτοτικών συνδέσεων (shortcut/skip connections). Μια ταυτοτική σύνδεση παρακάμπτει ορισμένα επίπεδα, μετατρέποντας ένα κανονικό δίκτυο σε ένα ResNet.





Στα νευρωνικά δίκτυα VGG (VGG-16 και VGG-19), κάθε συνελικτικό δίκτυο έχει ένα φίλτρο 3×3. Ωστόσο, ένα ResNet έχει λιγότερα φίλτρα και είναι λιγότερο περίπλοκο από ένα VGGNet. Η αρχιτεκτονική του ResNet-50 είναι βασισμένη στο μοντέλο που απεικονίζεται στο σχήμα, αλλά με μια σημαντική διαφορά. Το ResNet 50 επιπέδων χρησιμοποιεί ένα bottleneck design για το μπλοκ. Ένα bottleneck residual μπλοκ χρησιμοποιεί συνελίξεις 1×1, οι οποίες μειώνουν τον αριθμό των παραμέτρων και τους πολλαπλασιασμούς πινάκων. Αυτό επιτρέπει πολύ πιο γρήγορη εκπαίδευση κάθε επιπέδου.

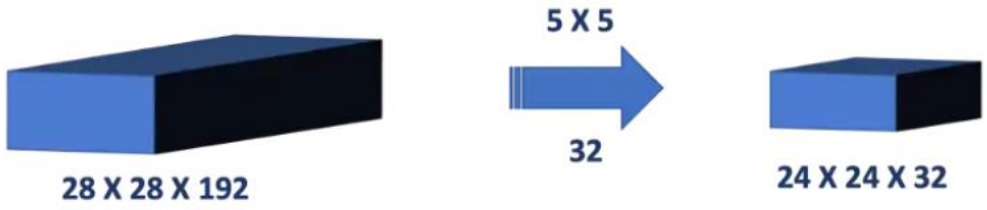
Το 1X1 Convolution σημαίνει απλώς ότι το φίλτρο έχει μέγεθος 1X1Χβάθος προηγούμενου feature map. Αυτό το φίλτρο 1X1 θα σέρνεται σε όλη την εικόνα εισόδου. Στην παρακάτω εικόνα μπορούμε να δούμε ένα παράδειγμα 1X1 Convolution.



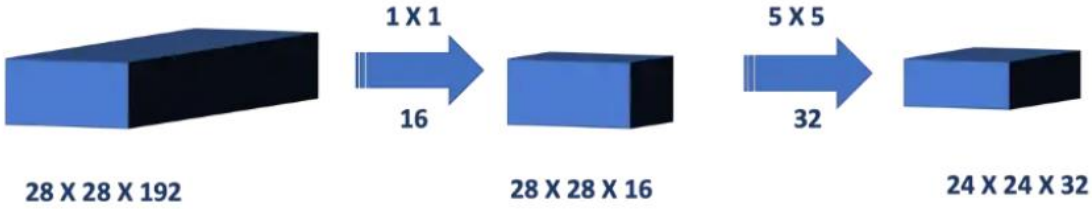
Αν επιλέξουμε ένα φίλτρο 1X1X192, τότε η έξοδος θα έχει το ίδιο ύψος και πλάτος με την είσοδο, αλλά μόνο ένα κανάλι (64X64X1).

Σε μια εικόνα αν θέλουμε να μειώσουμε το βάθος και αλλά να διατηρήσουμε το Ύψος Χ Πλάτος των feature maps ίδιο, τότε μπορούμε να επιλέξουμε φίλτρα 1X1 (Αριθμός φίλτρων = Κανάλια εξόδου) για να επιτύχουμε αυτό το αποτέλεσμα. Αυτή η διαδικασία ονομάζεται μείωση διαστάσεων (Dimensionality reduction).

Με το παρακάτω παράδειγμα μπορούμε να κατανοήσουμε καλύτερα πώς η μείωση των διαστάσεων θα μειώσει το υπολογιστικό φορτίο. Θέλουμε να συνδυάσουμε χάρτες χαρακτηριστικών εισόδου 28X28 X192 με φίλτρα 5X5X32. Αυτό θα έχει ως αποτέλεσμα 120.422 εκατομμύρια πράξεις. Με την προσθήκη του επιπέδου 1X1 Conv πριν από το 5X5 Conv, διατηρώντας παράλληλα το ύψος και το πλάτος του χάρτη χαρακτηριστικών, μειώσαμε τον αριθμό των πράξεων κατά παράγοντα 10 (12.4 εκατομμύρια πράξεις).



Number of Operations : (28X28X32) X (5X5X192) = 120.422 Million Ops



Number of Operations for 1 X 1 Conv Step : (28X28X16) X (1X1X192) = 2.4 Million Ops
Number of Operations for 5 X 5 Conv Step : (28X28X32) X (5X5X16) = 10 Million Ops
Total Number of Operations = 12.4 Million Ops

Το προεκπαιδευμένο δίκτυο που επιλέξαμε να χρησιμοποιήσουμε είναι το ResNet50. Το ResNet-50 είναι ένα συνελικτικό νευρωνικό δίκτυο 50 επιπέδων (49 συνελικτικά επίπεδα και ένα επίπεδο MaxPooling).

Η αρχιτεκτονική ResNet 50 επιπέδων περιλαμβάνει τα ακόλουθα στοιχεία, όπως φαίνεται στον παρακάτω πίνακα:

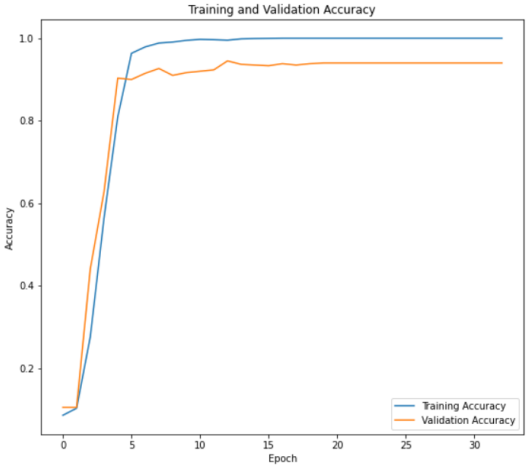
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

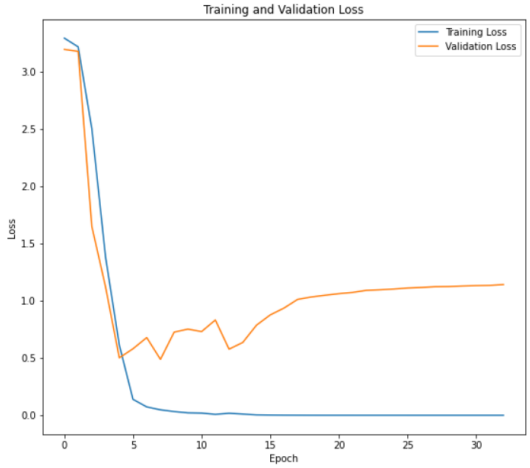
- Ένα 7×7 επίπεδο συνέλιξης με 64 φίλτρα με stride 2.
- Ένα max pooling layer 3X3 με stride 2.
- 9 ακόμη επίπεδα, 3×3 με 64 φίλτρα, ένα άλλο 1×1 με 64 φίλτρα και ένα τρίτο με 1×1 με 256 φίλτρα. Αυτά τα 3 επίπεδα επαναλαμβάνονται 3 φορές.
- 12 ακόμη επίπεδα, 1×1 με 128 φίλτρα, 3×3 με 128 φίλτρα και 1×1 με 512 φίλτρα, που επαναλαμβάνονται 4 φορές.
- 18 ακόμη επίπεδα, 1×1 με 256 φίλτρα, 3×3 με 256 φίλτρα και 1×1 με 1024 φίλτρα, που επαναλαμβάνονται 6 φορές.
- 9 ακόμη επίπεδα, 1×1 με 512 φίλτρα, 3×3 με 512 φίλτρα και 1×1 με 2048 φίλτρα που επαναλαμβάνονται 3 φορές.

Μετά τα πρώτα 50 επίπεδα του δικτύου, χρησιμοποιούμε ένα επίπεδο average pooling, ακολουθούμενο από ένα πλήρως συνδεδεμένο επίπεδο με 1000 νευρώνες, χρησιμοποιώντας τη συνάρτηση ενεργοποίησης softmax για ταξινόμηση.

8. Αποτελέσματα

Μη προεκπαιδευμένο δίκτυο χωρίς επαύξηση δεδομένων



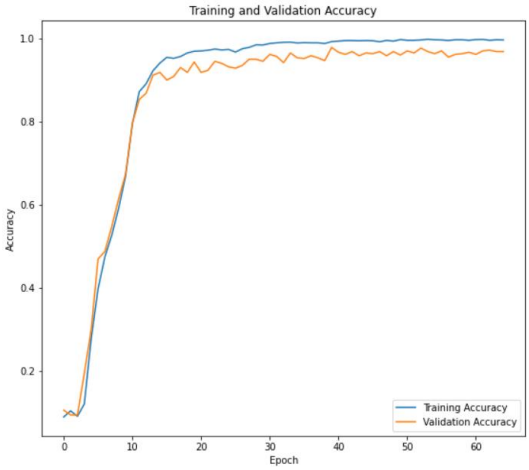


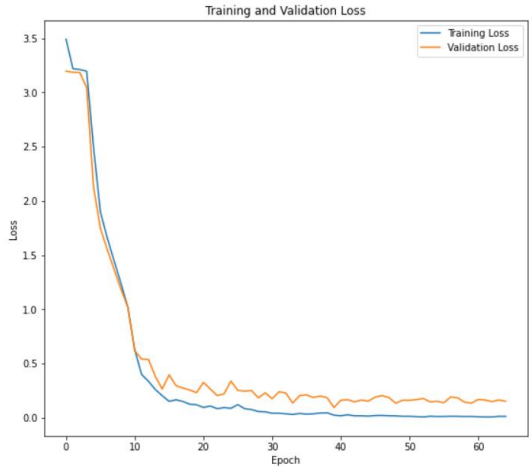
Train accuracy: 99.10%

Test accuracy: 93.39%

Παρατηρούμε ότι η διαφορά ανάμεσα στο training loss και στο validation loss είναι μεγάλη. Αυτό σημαίνει ότι το μοντέλο μας έχει υπερπροσαρμοστεί στα δεδομένα (overfitting, high variance). Ενώ στην εκπαίδευση η ακρίβεια είναι αρκετά μεγάλη, στο στάδιο δοκιμής η ακρίβεια πέφτει.

Μη προεκπαιδευμένο δίκτυο με επαύξηση δεδομένων





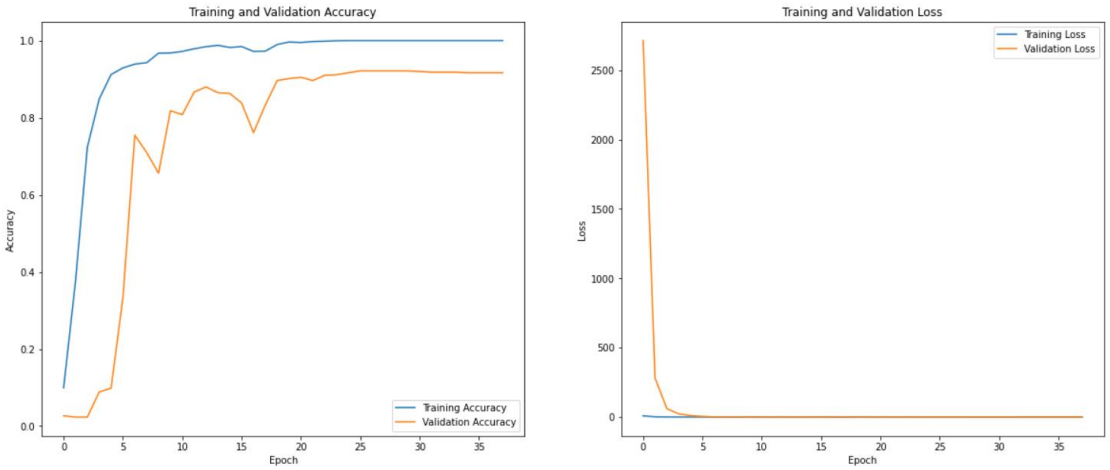
Train accuracy: 99.23%

Test accuracy: 98.19%

Παρατηρούμε ότι η διαφορά ανάμεσα στο training loss και στο validation loss είναι πολύ μικρότερη από πριν. Η ακρίβεια στο στάδιο δοκιμής είναι λίγο μικρότερη από αυτή της εκπαίδευσης. Πλέον δεν υπάρχει το πρόβλημα της υπερπροσαρμογής, στο βαθμό που υπήρχε πριν χρησιμοποιήσουμε την επαύξηση δεδομένων.

Με την επαύξηση δεδομένων, μπορούμε να μειώσουμε ή και να εξαλείψουμε το overfitting χωρίς να χρειάζεται να προσθέσουμε νέες εικόνες στο σύνολο δεδομένων μας, κάτι που είναι πολύ ακριβό αν όχι αδύνατο.

Προεκπαιδευμένο δίκτυο χωρίς επαύξηση δεδομένων

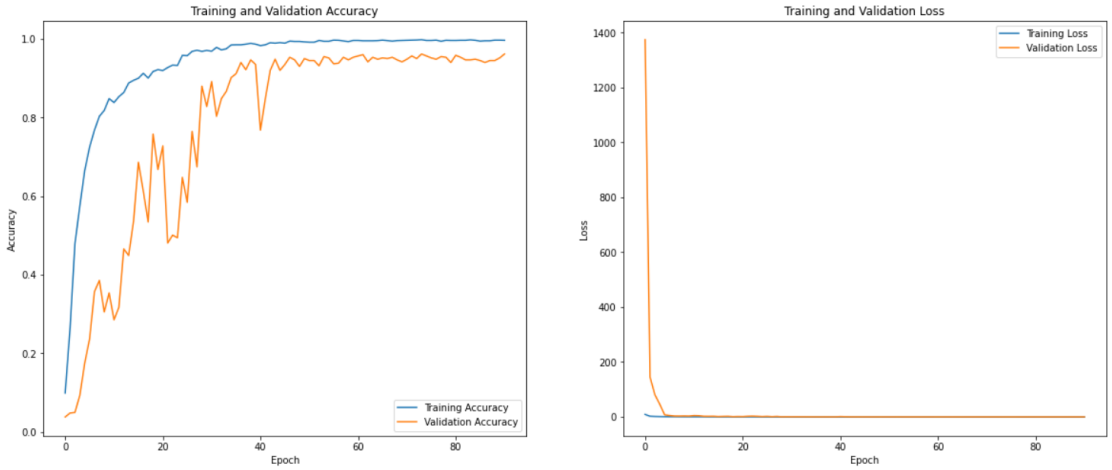


Train accuracy: 95.44%

Test accuracy: 83.57%

Παρατηρούμε ότι η διαφορά ανάμεσα στο training accuracy και στο validation accuracy είναι μεγάλη. Υπάρχει overfitting, εδώ όμως επίσης το training accuracy, δεν είναι τόσο μεγάλο, όσο ήταν με το μη προεκπαιδευμένο δίκτυο. Αυτό σημαίνει ότι εκτός από high variance (overfitting), υπάρχει και high bias (underfitting)στα δεδομένα, το οποίο μπορεί να οφείλεται στο γεγονός ότι το σύνολο δεδομένων που διαθέτουμε είναι σχετικά μικρό.

Προεκπαιδευμένο δίκτυο με επαύξηση δεδομένων



Train accuracy: 99.67%

Test accuracy: 96.74%

Παρατηρούμε ότι η διαφορά ανάμεσα στο training accuracy και στο validation accuracy είναι υπολογίσιμη. Υπάρχει overfitting, εδώ όμως επίσης το training accuracy, είναι αρκετά μεγάλο. Με την επαύξηση δεδομένων καταφέραμε να εξαλείψουμε το underfitting. Βέβαια, ακόμα υπάρχει το πρόβλημα τις υπερπροσαρμογής, το οποίο όμως φαίνεται να έχει βελτιωθεί.

Σύνοψη αποτελεσμάτων

Στον παρακάτω πίνακα μπορούμε να δούμε τα αποτελέσματα των πειραμάτων που έγιναν. Η επαύξηση δεδομένων αύξησε πάρα πολύ την ακρίβεια του σταδίου δοκιμής. Επίσης, στο προεκπαιδευμένο δίκτυο η επαύξηση δεδομένων οδήγησε και στην αύξηση της ακρίβειας του σταδίου εκπαίδευσης.

Το γεγονός ότι το μη προεκπαιδευμένο μοντέλο, έχει μεγαλύτερη ακρίβεια από το προεκπαιδευμένο, μπορεί να οφείλεται στο overfitting, επειδή ακόμα και με την επαύξηση δεδομένων, τα δεδομένα μας εξακολουθούν να είναι λίγα αναλογικά με το μέγεθος του προεκπαιδευμένου δικτύου.

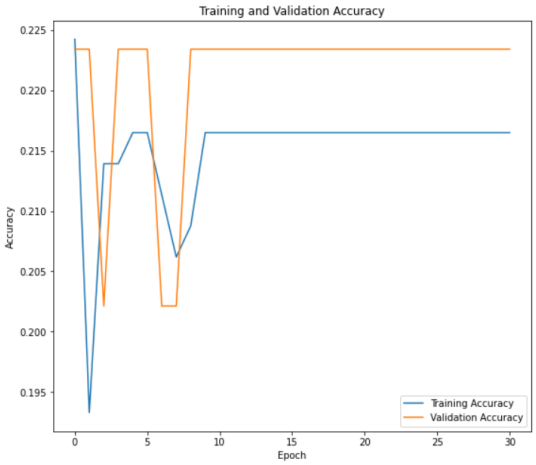
	Without data augmentation		**With data augmentation**	
	From scratch training	Pretrained network	From scratch training	Pretrained network
Train accuracy	99.10%	95.44%	99.23%	99.67%
Test accuracy	93.39%	83.57%	**98.19%**	**96.74%**

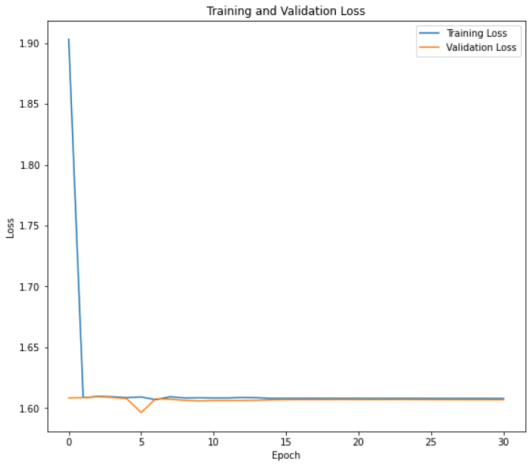
Όλα τα παραπάνω προέκυψαν με input size 256 X 256. Έγινε επίσης μια δοκιμή με input size 128 X 128 και η ακρίβεια στο μη προεκπαιδευμένο δίκτυο ήταν 98.42%, ενώ στο προεκπαιδευμένο ήταν 94.04%. Οπότε, για το **μη προεκπαιδευμένο δίκτυο** επιλέγουμε τελικά **input size 128 X 128** με ακρίβεια **98.42%** και για το **προεκπαιδευμένο δίκτυο** επιλέγουμε τελικά **input size 256 X 256** με ακρίβεια **96.74%**.

9. Δοκιμή δικτύων σε διαφορετικό σύνολο δεδομένων

Τα δύο μοντέλα που χρησιμοποιήσαμε στο BelgiumTS - Belgian Traffic Sign Dataset, δοκιμάσαμε να τα χρησιμοποιήσουμε και σε υποσύνολο της βάσης εικόνων Caltech 256 Image Dataset | Kaggle.

Μη προεκπαιδευμένο δίκτυο



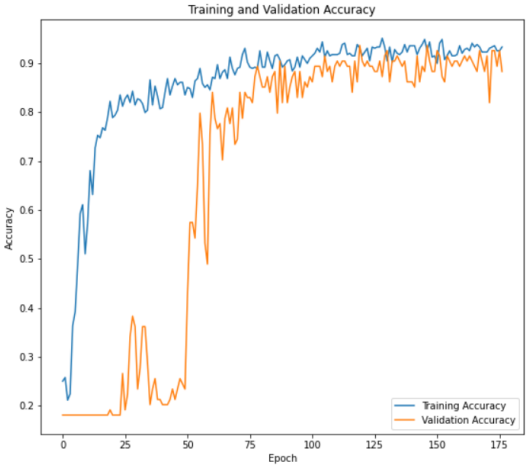


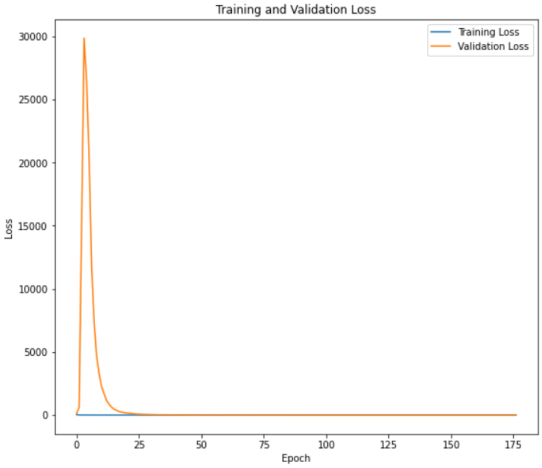
Train accuracy: 21.65%

Test accuracy: 21.15%

Το μη προεκπαιδευμένο δίκτυο δεν κατάφερε να κάνει fit στα δεδομένα, υπάρχει high bias (underfitting). Η αρχιτεκτονική μας δεν είναι κατάλληλη για το συγκεκριμένο σύνολο δεδομένων. Πιθανότατα πρέπει να αναπτυχθεί μια απλούστερη αρχιτεκτονική, γιατί το σύνολο δεδομένων είναι πολύ πιο μικρό από πριν.

Προεκπαιδευμένο δίκτυο





Train accuracy: 93.04%

Test accuracy: 92.31%

Τα αποτελέσματα, όσον αφορά το προεκπαιδευμένο δίκτυο, είναι πολύ πιο ικανοποιητικά. Ακόμα υπάρχει κάποιο underfitting, το οποίο το καταλαβαίνουμε επειδή το training accuracy παραμένει σχετικά σε χαμηλή τιμή, η οποία βέβαια μπορεί να θεωρηθεί ικανοποιητική. Επίσης, το training accuracy και το testing accuracy είναι σχετικά κοντά, άρα το πρόβλημα της υπερπροσαρμογής δεν υπάρχει σε μεγάλο βαθμό.

Μετά τη σύγκριση που έγινε μεταξύ μη προεκπαιδευμένου και προεκπαιδευμένου δικτύου, σε ένα διαφορετικό σύνολο δεδομένων, μπορούμε να καταλήξουμε στο συμπέρασμα ότι **ένα μη προεκπαιδευμένο δίκτυο μπορεί να είναι πολύ καλό για ένα συγκεκριμένο σύνολο δεδομένων**, όμως **δεν μπορεί να γενικευτεί εύκολα και να χρησιμοποιηθεί σε ένα άλλο σύνολο δεδομένων**. Αντίθετα, **ένα προεκπαιδευμένο δίκτυο μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά σύνολα δεδομένων**.

10. Βιβλιογραφία

1. https://blockgeni.com/essentials-of-ai-vs-ml-vs-deep-learning/
2. https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications
3. https://www.researchgate.net/figure/Biological-Neuron-versus-Artificial-Neural-Network_fig4_325870973
4. https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f
5. https://www.sciencedirect.com/science/article/abs/pii/S0886779820306313
6. https://www.javatpoint.com/artificial-neural-network
7. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
8. https://towardsai.net/p/machine-learning/beginner-guides-to-convolutional-neural-network-from-scratch-kuzushiji-mnist-75f42c175b21
9. https://chris-said.io/2020/12/26/two-things-that-confused-me-about-cross-entropy/
10. https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60
11. https://sh-tsang.medium.com/review-mish-a-self-regularized-non-monotonic-activation-function-a7afe19b4af7
12. https://medium.com/geekculture/different-activation-functions-for-deep-neural-networks-you-should-know-ea5e86f51e84
13. https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0
14. https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9

15. https://vitalflux.com/overfitting-underfitting-concepts-interview-questions/
16. https://www.baeldung.com/cs/cost-vs-loss-vs-objective-function
17. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
18. https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/
19. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7
20. https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/
21. https://machinelearningmastery.com/check-point-deep-learning-models-keras/
22. https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler
23. https://keras.io/api/callbacks/reduce_lr_on_plateau/
24. https://androidkt.com/how-to-add-l1-l2-regularization-in-pytorch-loss-function/
25. https://medium.com/unpackai/stay-away-from-overfitting-l2-norm-regularization-weight-decay-and-l1-norm-regularization-795bbc5cf958
26. https://towardsdatascience.com/step-by-step-guide-to-using-pretrained-models-in-keras-c9097b647b29
27. https://medium.datadriveninvestor.com/introducing-transfer-learning-as-your-next-engine-to-drive-future-innovations-5e81a15bb567
28. https://datagen.tech/guides/computer-vision/resnet-50/
29. https://arxiv.org/abs/1512.03385
30. https://medium.com/analytics-vidhya/talented-mr-1x1-comprehensive-look-at-1x1-convolution-in-deep-learning-f6b355825578
31. https://keras.io/getting_started/
32. Διαφάνειες μαθήματος “ Όραση Υπολογιστών”, Δημοκρίτειο Πανεπιστήμιο Θράκης