Εφαρμογές Μηχανικής Όρασης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

Σωτηργιαννίδης Νικόλαος
ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ | ΓΡΗΓΟΡΗΣ ΝΙΚΟΛΑΟΥ

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Σωτηργιαννίδης Νικόλαος** του **Ανδρέα**, φοιτητής του **Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής** του Πανεπιστημίου Δυτικής Αττικής, πριν αναλάβω την εκπόνηση της πτυχιακής εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε, ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα, σε περίπτωση που το Ίδρυμα έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου εκπόνηση (Π.Ε) με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε πρέπει να ολοκληρωθεί τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία της ανάθεσης τη. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18. Παρ.5 του ισχύοντος Εσωτερικού Κανονισμού»

Ο Δηλών

Ημερομηνία

12/6/2020

Περιεχόμενα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου και επιβλέποντα της παρούσας πτυχιακής, Γρηγόρη Νικολάου. Η εμπιστοσύνη που μου έδειξε από την πρώτη μέρα της γνωριμίας μας μαζί με τις εποικοδομητικές συνομιλίες μας με ώθησαν ακόμα πιο ψηλά.

Πίνακας Εικόνων

Ακρωνύμια

ΤΝΔ…… Τεχνητό Νευρωνικό Δίκτυο
ΣΝΔ…… Συνελικτικό Νευρωνικό Δίκτυο
ΒΝΔ…… Βιολογικό Νευρωνικό Δίκτυο
SGD…… Stochastic Gradient Descent
CNN……... Convolutional Neural Network
FCN…… Fully Convolutional Network
ML……..…. Machine Learning
ANN…… Artificial Neural Network
Adam……... Adaptive Moment Estimation
ADAS……..... Adaptive Driver Assistance Systems
CC ……... Cruise Control
EEG……. Electroencephalography
ECG……. Electrocardiogram
EOG……. Electrooculography
SSS……... Stanford Sleepiness Scale
KSS……... Karolinska Sleepiness Scale
DMS……. Driver Monitoring System
LDWS……... Lane Departure Warning System
DDS…….. Drowsiness Detection System
HSL……... Hue Saturation Lightness

Σημειογραφία

Με σκοπό τόσο την καλύτερη κατανόηση του περιεχομένου της παρούσας πτυχιακής όσο και τη σωστή παρουσίαση αυτού είναι απαραίτητο να ορίσουμε μια σημειογραφία για την αναπαράσταση των μεγεθών με τα οποία ασχοληθούμε.

L : αριθμός επιπέδων

$\ell^{[k]}$: επίπεδο k, k=0,1,2,...,k-1

$n^{[k]}$: αριθμός νευρώνων στο επίπεδο ℓ^k

$\ell_i^{[k]}$: νευρώνας i στο επίπεδο ℓ^k

$a^{[\ell]}$: αριθμός ενεργοποιήσεων στο επίπεδο ℓ^k

$w_{i,j}^{[k]}$: συναπτικό βάρος μεταξύ των νευρώνων $\ell_i^{[k-1]}$ και $\ell_j^{[k]}$

$W^{[k]}$: πίνακας συναπτικών βαρών στο $\ell^{[k]}$

X : όλα τα δεδομένα εισόδου του dataset

Y: επιθυμητές έξοδοι (labels)

$\hat{Y}$: προσεγγίσεις των εξόδων

σ : συνάρτηση ενεργοποίησης

$b^{[k]}$: εξωτερικά εφαρμοζόμενος παράγοντας στο επίπεδο $\ell^{[k]}$

Εικόνα 1 Σημειογραφία (βιβλιογραφία [1])

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο τη μελέτη και υλοποίηση των τεχνητών νευρωνικών δικτύων και συγκεκριμένα των συνελικτικών νευρωνικών καθώς και το πως μπορούν να εφαρμοστούν στη μηχανική όραση και συγκεκριμένα στο πρόβλημα της ανίχνευσης λωρίδων. Θα δοθεί το απαραίτητο υλικό για τη μελέτη τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο και θα αναλυθούν όλες οι απαραίτητες έννοιες για την πλήρη κατανόηση του αντικειμένου. Στο πρώτο κεφάλαιο θα αναφερθούμε σε κάποια από τα πιο δημοφιλή ADAS, συστήματα που βρίσκονται σε πληθώρα οχημάτων. Στη συνέχεια θα γνωρίσουμε τη μηχανική μάθηση, ένα ευρύτερο κλάδο μέρος του οποίου είναι τα νευρωνικά δίκτυα και στη συνέχεια θα μιλήσουμε για τη δομή ενός φυσικού καθώς και ενός τεχνητού νευρώνα. Στο τρίτο κεφάλαιο θα αναλύσουμε σε βάθος τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks), μία ειδική κατηγορία νευρωνικών δικτύων, τα οποία αποτελούν την αιχμή του δόρατος της μηχανικής όρασης. Στη συνέχεια, στο τέταρτο και τελευταίο κεφάλαιο αναλύεται η εφαρμογή και τα αποτελέσματα του νευρωνικού δικτύου ο κώδικας του οποίου είναι γραμμένος σε γλώσσα python.

Η αυτόνομη οδήγηση μπαίνει όλο και πιο πολύ στις ζωές μας με τα πρώτα αυτόνομα οχήματα επιπέδου 3 να βρίσκονται ήδη στους δρόμους και τις εταιρείες να καταναλώνουν τεράστιους πόρους στην έρευνα και ανάπτυξη νέων μεθόδων ενεργητικής και παθητικής ασφάλειας. Ποιο είναι όμως το βασικό συστατικό της αυτόνομης οδήγησης;

Κατά τη διάρκεια της οδήγησης η διατήρηση μιας σταθερής πορείας μέσα στις διαγραμμίσεις του δρόμου αποτελεί βασικό μέλημα κάθε οδηγού. Η διαδικασία αυτή γίνεται σχεδόν ασυναίσθητα τόσο από έμπειρους όσο και από αρχάριους οδηγούς. Σε αντίθεση όμως με τον ανθρώπινο εγκέφαλο ένας υπολογιστής έχει περιορισμένες δυνατότητες. Ελλείψει του κατάλληλου λογισμικού ένας η/υ δε μπορεί, από τη φύση του, να καταλάβει τη διαφορά μεταξύ των λευκών ή κίτρινων λωρίδων και του υπόλοιπου δρόμου.

Στην παρούσα πτυχιακή εργασία παρουσιάζεται ένα σύστημα ανίχνευσης λωρίδων με χρήση μιας απλής κάμερας τοποθετημένη στο εμπρόσθιο μέρος του αυτοκινήτου. Το βίντεο από την κάμερα αναλύεται καρέ από το νευρωνικό δίκτυο με σκοπό να ανιχνευτούν οι λωρίδες. Τέλος, αφού ανιχνευθούν χρωματίζει με πράσινο χρώμα την περιοχή ανάμεσα στις δύο λωρίδες.

Abstract

The present dissertation focuses on the study and implementation of artificial neural networks and specifically convolutional neural networks as well as how they can be applied to machine vision and specifically to the problem of lane detection. The necessary material for the study will be given both in theory and in practice and all the necessary concepts for the full understanding of the subject will be analyzed. In the first chapter we will refer to some of the most popular ADAS, systems that are found in a variety of vehicles. Then we will get to know machine learning, a broader branch of which the neural networks are part of, and then we will discuss the structure of a physical as well as an artificial neuron. In the third chapter we will analyze in depth the Convolutional Neural Networks, a special category of neural networks, which are the bleeding edge of machine vision. Then, in the fourth and final chapter, the application and the results of the neural network are analyzed, the code of which is written in python language.

Autonomous driving is increasingly entering our lives with the first autonomous level 3 vehicles already on the roads and companies consuming huge resources in research and development of new methods of active and passive safety. But what is the key component of autonomous driving?

While driving, maintaining a steady course between the road markings is a key concern for every driver. This process is done almost unconsciously by both experienced and novice drivers. On the other hand, compared to the human brain computers possess limited capabilities. Without the proper software a computer cannot understand the difference between white or yellow stripes and the rest of the road.

In this thesis we present a lane detection system using a simple camera mounted on the front of the car. The video from the camera is analyzed frame by frame by the neural network in order to detect the lane markings. Finally, after being detected, it paints green the road between the two lanes.

Εισαγωγή

Η αυτόνομη οδήγηση έχει προσελκύσει μεγάλο ερευνητικό ενδιαφέρον τις τελευταίες δεκαετίες καθώς όλο και περισσότερες εταιρείες ενσωματώνουν τεχνολογίες όπως lane keeping και emergency braking, collision avoidance systems στα αμάξια που παράγουν. Το κλειδί γι' αυτές τις τεχνολογίες είναι η περιβαλλοντική αντίληψη (environmental perception) κομμάτι της οποίας είναι και η ανίχνευση λωρίδων. Αυτή η αύξηση του ερευνητικού ενδιαφέροντος είχε ως συνέπεια να δημοσιευτούν πολλές έρευνες που προτείνουν λύσεις στο πρόβλημα της ανίχνευσης λωρίδων όπως και σε πολλά άλλα προβλήματα.

Ένα μεγάλο μέρος των ερευνών χρησιμοποιεί τεχνικές του κλασσικού computer vision (σε συνδυασμό με μηχανική μάθηση) το οποίο αν και άκρως αποτελεσματικό απαιτεί εκτενές preprocessing και χρησιμοποιεί χειροποίητα (handcrafted) χαρακτηριστικά (ακμές, γωνίες, χρώμα) που στην προκειμένη περίπτωση είναι application specific. Αυτή η προσέγγιση έχει ως βασικό μειονέκτημα την έλλειψη επεκτασιμότητας λόγω των διακυμάνσεων της οδικής σκηνής. Επιπλέον απαιτείται μεγάλη εμπειρία από τη μεριά του προγραμματιστή για τη σωστή επιλογή των κατάλληλων χαρακτηριστικών (features) που θα χρησιμοποιηθούν κάτι που καθιστά ακόμη πιο δύσκολη την υλοποίηση ενός τέτοιου συστήματος. [2]

Στο παρόν πόνημα προτείνεται μία λύση στο παραπάνω πρόβλημα με τη χρήση συνελικτικών νευρωνικών δικτύων σε περιβάλλον αυτοκινητόδρομου όπου οι λωρίδες είναι είτε κίτρινες είτε άσπρες. Τα δεδομένα που χρησιμοποιήθηκαν φωτογραφίες από αυτοκινητόδρομους τραβηγμένες από το εμπρόσθιο τμήμα του αμαξιού. Για την καλύτερη εκπαίδευση του νευρωνικού δικτύου χρησιμοποιήθηκαν φωτογραφίες με ποικίλες καταστάσεις φωτισμού. Επιπλέον χρησιμοποιήθηκαν κάποιες τεχνικές της μηχανικής όρασης για την περαιτέρω ενίσχυση των δεδομένων εκπαίδευσης.

1. Adaptive Driver Assistance Systems (ADAS)

Είναι γεγονός πως τα ατυχήματα στους δρόμους αποτελούν μία σύγχρονη μάστιγα που διεκδικεί τις ζωές εκατομμυρίων ανθρώπων. Μάλιστα σύμφωνα με τον παγκόσμιο οργανισμό υγείας αφήνουν την τελευταία τους πνοή στην άσφαλτο 1.2 εκατομμύρια κάθε χρόνο [3]. Επιπλέον, ~50 εκατομμύρια άνθρωποι τραυματίζονται κάθε χρόνο σε αυτοκινητιστικά δυστυχήματα, ένας αριθμός που όχι μόνο δεν προβλέπεται να μειωθεί αλλά να αυξηθεί δραματικά στα επόμενα χρόνια [3]. Ο μόνος τρόπος να μειωθούν οι θάνατοι στο δρόμο είναι να παρθούν επαρκή μέτρα αλλά πως;

Δεδομένου πως η οδήγηση αποτελεί σημείο τομής πολλών παραγόντων όπως ο ανθρώπινος παράγοντας, το ίδιο το αμάξι και τα υποσυστήματα του αλλά και η κατάσταση του δρόμου θα πρέπει να ληφθούν μέτρα για όλα τα ανωτέρω. Όπως προδίδει και το όνομα του κεφαλαίου, θα επικεντρωθούμε στα μέτρα που αφορούν το ανθρώπινο λάθος και δε θα μας απασχολήσουν μέτρα που αφορούν τους υπόλοιπους προαναφερθέντες παράγοντες. Ο συνδυασμός λοιπόν της ανάγκης μείωσης των τροχαίων δυστυχημάτων με την τεράστια πρόοδο της τεχνολογίας γέννησε τα ADAS. Στο παρόν κεφάλαιο θα περιγράψουμε μερικά δημοφιλή **Adaptive Driver Assistance Systems** (ADAS), τα οποία βρίσκονται σε μια μεγάλη μερίδα των οχημάτων που κυκλοφορούν στους δρόμους.

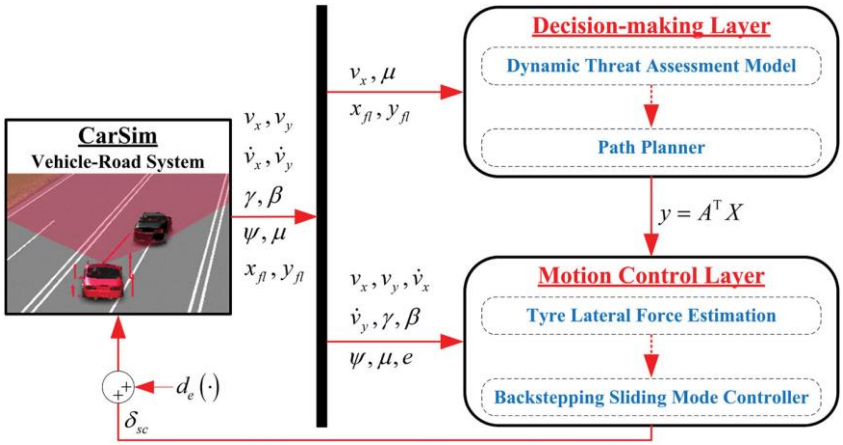
Με τον όρο **Adaptive Driver Assistance Systems** αναφερόμαστε σε όλες τις τεχνολογίες που έχουν σκοπό να κρατήσουν τον οδηγό ασφαλή όσο αυτός οδηγεί το αμάξι του και όχι μόνο. Υπάρχουν πολλά είδη ADAS και κάθε ένα από αυτά διαθέτει διαφορετικά χαρακτηριστικά και διαφορετικό σκοπό. Κάποια αυτά προσφέρουν κρίσιμες δικλείδες ασφαλείας ενώ άλλα προσφέρουν επιπρόσθετες ανέσεις για την εύκολη αποφυγή μικρών ατυχημάτων. Στις παρακάτω παραγράφους θα εστιάσουμε στα συστήματα που αφορούν την ασφάλεια και θα γνωρίσουμε μερικά από αυτά.

Driver error category	Specific influence	Examples of conditions experienced
(A) Objective lack of information	Sight obstruction	Buildings, plants, other vehicles
	Masking stimuli	Darkness, being dazzled, rain, snow, fog
(B) Failure to use information	Failure to spot danger through inattention	Distraction by objects inside/outside of the vehicle, reduced activity as a result of drowsiness/medication, attention focused incorrectly
	Omissions	Omissions of using the turn signal, shifting gears or checking the blind spot because of absent-mindedness
	Disregard/infringements	Deliberate speed violations or disregarding right of way
	Reduction of information processing	Errors caused by false expectations and stereotyping, i.e. assuming that a wide road is a major road
	Processing deficits	Perception of acceleration
(C) Misuse of information	Orientation and expectation errors	Errors in expectations as a result of habituation, e.g. not looking on quiet roads for vehicles approaching from the right
	Misjudgements	Miscalculation of speeds and distances
	Objective slips, incorrect responses	Swerving instead of braking vs. confusion between accelerator and brakes

Εικόνα 2 Αιτίες Ατυχημάτων (βιβλιογραφία [40])

1.1 Σύστημα Αποφυγής Πρόσκρουσης (Collision Avoidance System)

Τη σήμερον ημέρα τα ΙΧ έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητας μας και ένα μεγάλο ποσοστό των ανθρώπων τα επιλέγει για την καθημερινή του μετακίνηση. Η ευελιξία και η ευκολία που παρέχουν στην μετακίνηση έχει ένα μακάβριο τίμημα. Μόνο για το έτος 2015 καταγράφηκαν 1.3 εκατομμύρια θάνατοι [4] από δυστυχήματα στους δρόμους. Καταλυτικό ρόλο στα ατυχήματα αυτά, όπως μπορεί να υποθέσει κανείς, παίζει το ανθρώπινο λάθος. Καθώς λοιπόν η μηχανική μάθηση μπαίνει ολοένα και περισσότερο στις ζωές μας είναι επόμενο πως θα χρησιμοποιηθεί για να διευκολύνει και να προστατεύσει τον οδηγό από επικίνδυνες καταστάσεις.



Εικόνα 3 Αρχιτεκτονική συστήματος αποφυγής σύγκρουσης (βιβλιογραφία [4])

Με το 97.8% όλων των δυστυχημάτων να προκαλούνται από βίαιες προσκρούσεις [4] τα πράγματα δείχνουν αρκετά άσχημα. Το υψηλό ποσοστό θα μπορούσε να αποδοθεί σε αρκετούς λόγους κυριότεροι των οποίων να είναι οι εξής:

- Λάθος εκτίμηση του οδηγού
- Καθυστερημένη αντίδραση ή λάθος χειρισμός

Επομένως προκειμένου να αποφευχθούν επικείμενες συγκρούσεις θα πρέπει το σύστημα να μπορεί να παρακολουθεί συνεχώς τα αμάξια που βρίσκονται κοντά του και να αξιολογεί αν υπάρχει κίνδυνος σύγκρουσης.

Για τη υλοποίηση ενός τέτοιου συστήματος χρειάζεται αρχικά ένας τρόπος να εντοπίζονται εμπόδια. Αισθητήρες όπως laser, πομποδέκτες μικροκυμάτων ή υπερήχων μπορούν να εντοπίσουν εν δυνάμει εμπόδια. Αυτό επιτυγχάνεται με τη μέτρηση του χρόνου που κάνει η αντανάκλαση του πομπού να επιστρέψει στο δέκτη. Μετά με τη βοήθεια ενός υπολογιστικού συστήματος υπολογίζεται τόσο η σχετική απόσταση του εμποδίου από το όχημα όσο και το μέγεθος του εμποδίου. Αν το λογισμικό κρίνει πως το όχημα πλησιάζει το εμπόδιο με επικίνδυνα μεγάλη ταχύτητα τότε είτε αναλαμβάνει τον έλεγχο του τιμονιού και των φρένων προκειμένου να αποφευχθεί η σύγκρουση είτε ειδοποιείται ο οδηγός. Αξίζει να αναφερθεί πως υπάρχουν συστήματα όπως αυτό που περιγράφεται στην βιβλιογραφία [5] όπου χρησιμοποιούν κάμερες και μηχανική όραση προκειμένου να εντοπίσουν και να παρακολουθήσουν τα οχήματα που βρίσκονται κοντά στο αμάξι.

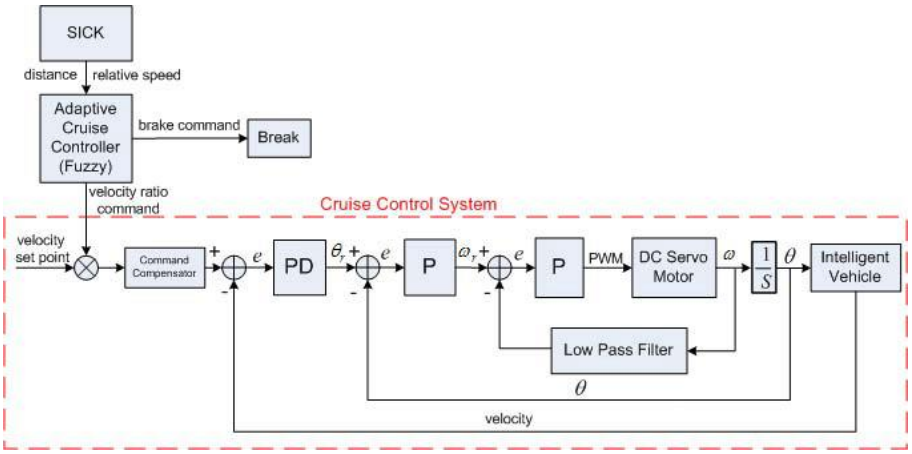
Παρά τα τεχνολογικά άλματα που έχουν λάβει χώρα τα τελευταία χρόνια το αντικείμενο της αποφυγής συγκρούσεων παραμένει πολύ δύσκολο όσον αφορά την σωστή υλοποίηση. Ένα αμάξι που είναι εξοπλισμένο με CAS (Collision Avoidance System) θα πρέπει, βγαίνοντας στους δρόμους, να μπορεί να ανταποκριθεί αποτελεσματικά σε μια πληθώρα καταστάσεων. Σε πολλές από αυτές ίσως χρειαστεί κάποιος απότομος ελιγμός σε σύντομο

χρονικό διάστημα. Αυτό συνεπάγεται με κίνδυνο εκτροπής από το δρόμο καθώς τα λάστιχα ενδέχεται να αρχίσουν να ολισθαίνουν λόγω κορεσμού [4]. Σα να μην έφταναν όλα αυτά, τα ελαστικά όταν βρίσκονται κοντά ή έχουν ξεπεράσει τα όρια ονομαστικής τριβής που αντέχουν αλλάζουν εντελώς συμπεριφορά. Τέλος σύμφωνα με τη βιβλιογραφία [6] φαίνεται πως οι αισθητήρες υπερήχων και μικροκυμάτων παρουσιάζουν διάφορα προβλήματα. Για παράδειγμα αναφέρεται πως λόγω της δομής των αισθητήρων η ακρίβεια τους επηρεάζεται όταν υπάρχουν άλλα αμάξια αριστερά και δεξιά του οχήματος.

Η αναγκαιότητα ενός τέτοιου συστήματος σε συνδυασμό με τις δυσκολίες που αναφέρθηκαν παραπάνω έχουν κινήσει το ενδιαφέρον πολλών ερευνητών και αποτελεί ένα καυτό θέμα για έρευνα. Είναι σίγουρο πως θα συνεχίσει να αποτελεί πεδίο για έρευνα για χρόνια ακόμα.

1.2 Αυτορρυθμιζόμενο σύστημα ελέγχου γκαζιού (Adaptive Cruise Control)

Το συμβατικό σύστημα ελέγχου γκαζιού (Cruise Control ή CC) είναι μία πολύ διαδεδομένη μορφή ADAS μέσω του οποίου ένα αμάξι μπορεί να διατηρεί μία συγκεκριμένη ταχύτητα χωρίς την επέμβαση του οδηγού. Αυτό ήταν και παραμένει πολύ χρήσιμο όταν η διαδρομή αποτελείται από μεγάλες ευθείες χωρίς στροφές (λεωφόρος, εθνική οδός) και δεν υπάρχει κυκλοφοριακή συμφόρηση. Όταν όμως υπάρχει κυκλοφοριακή συμφόρηση το σύστημα χάνει ένα μεγάλο μέρος της λειτουργικότητας του. Προκειμένου να αντιμετωπιστεί το προαναφερθέν πρόβλημα αναπτύχθηκε μία εξέλιξη του συστήματος που αναφέρθηκε ανωτέρω και ονομάζεται **Adaptive Cruise Control** (ACC).



Εικόνα 4 Μπλοκ διάγραμμα ACC (βιβλιογραφία [39])

Το ACC, σε αντίθεση με τον προκάτοχο του, προσφέρει δύο επίπεδα ελέγχου. Σε πλήρη αντιστοιχία με το συμβατικό CC, προσφέρει έλεγχο ταχύτητας αλλά αυτό που διαφοροποιεί δραματικά είναι ο έλεγχος απόστασης. Αυτό σημαίνει πως ενώ μπορεί να λειτουργήσει ακριβώς όπως και το συμβατικό CC, το ACC έχει την επιπλέον δυνατότητα να ρυθμίσει κατάλληλα την ταχύτητα προκειμένου να διατηρηθεί η απόσταση με το προπορευόμενο όχημα. Αυτό επιτυγχάνεται χρησιμοποιώντας πληθώρα αισθητήρων όπως **Laser**, **Lidar** για τον υπολογισμό της σχετικής απόστασης από το όχημα που βρίσκεται μπροστά.

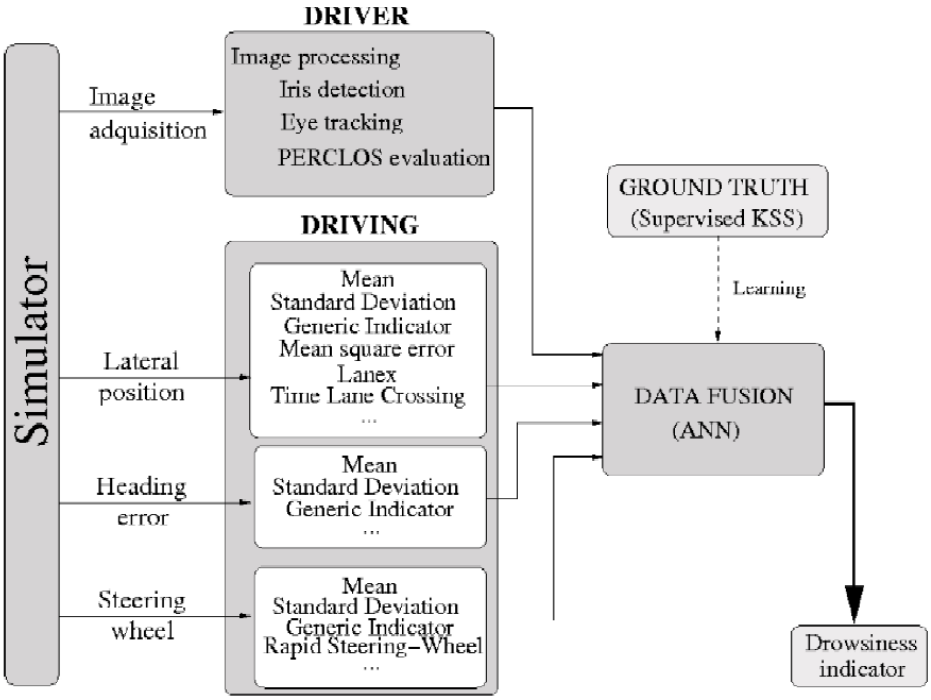
Υπάρχουν δύο τύποι ACC οι οποίοι είναι το Low-speed ACC και το High-speed ACC. Στην περίπτωση του πρώτου πρόκειται για ένα σύστημα που λειτουργεί σε περιπτώσεις έντονης συμφόρησης προκειμένου να διατηρήσει μία σταθερή απόσταση με το προπορευόμενο όχημα. Όσον αφορά το πρώτο, προσφέρει έλεγχο ταχύτητας όπως ακριβώς και συμβατικός προκάτοχος του (CC) με τη σημαντική διαφορά πως αν ένα αμάξι βρεθεί μπροστά και

αποφασίσει να μειώσει ταχύτητα θα συμβεί το εξής. Κατ ’αρχάς το σύστημα θα αναλάβει να φρενάρει το όχημα ώστε πρώτον να μην υπάρξει κάποια σύγκρουση και θα διατηρήσει τέτοια ταχύτητα ώστε η απόσταση των δύο αμαξιών να παραμένει σταθερή. Επιπλέον αν σταματήσει να υπάρχει προπορευόμενο όχημα το αμάξι θα επιταχύνει μέχρι την ταχύτητα που όρισε ο οδηγός και θα τη διατηρήσει.

Συστήματα όπως αυτό που περιγράφουμε παραπάνω πέρα από ένα έξτρα επίπεδο ασφάλειας παρέχουν μία άνεση στον οδηγό. Στο υποθετικό σενάριο όπου το προπορευόμενο όχημα κόβει απότομα ταχύτητα ο οδηγός ενδεχομένως να μην καταφέρει αντιδράσει εγκαίρως είτε λόγω απροσεξίας είτε λόγω κούρασης. Τέτοιες καταστάσεις αποφεύγονται με τη χρήση του ACC μειώνοντας την θνησιμότητα στους δρόμους.

1.3 Σύστημα ανίχνευσης κόπωσης (Drowsiness Detection System)

Ανάμεσα στους κυρίαρχους λόγους πρόκλησης θανατηφόρων αυτοκινητιστικών ατυχημάτων βρίσκεται η οδήγηση υπό την επήρεια κούρασης. Μάλιστα σύμφωνα με το NHTSA (**National Highway Traffic Safety Administration**) ~25% των ατυχημάτων που δηλώνονται στην αστυνομία έχουν προκληθεί από κάποιας μορφής απροσεξία από τη μεριά του οδηγού [7]. Θα μπορούσε κανείς να διακρίνει πολλές κατηγορίες απροσεξίας αλλά όλες τους ανήκουν σε μία από δύο μεγάλες κατηγορίες: 1) **Κούραση** 2)**Απόσπαση προσοχής.** Να σημειωθεί πως με τον όρο κούραση αναφερόμαστε σε ένα συνδυασμό συμπτωμάτων όπως



Εικόνα 5 Αρχιτεκτονική Συστήματος Ανίχνευσης Κόπωσης (βιβλιογραφία [7])

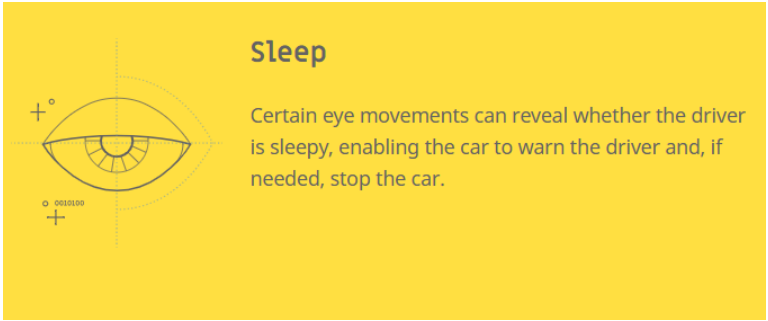
μειωμένα αντανακλαστικά και συνεχές αίσθημα υπνηλίας. Οδηγώντας με τα παραπάνω συμπτώματα έχει ως αποτέλεσμα να αυξηθεί κατά πολύ η πιθανότητα εμπλοκής σε κάποιο δυστύχημα. Συνεπώς είναι αναγκαίο να αναπτυχθούν συστήματα που έχουν τη δυνατότητα

παρακολούθησης και προειδοποίησης του οδηγού προκειμένου να μην τον πάρει ο ύπνος μειώνοντας σημαντικά τις πιθανότητες κάποιου ατυχήματος.

Μέχρι σήμερα έχουν προταθεί πολλά συστήματα που μπορούν επιτυχώς να ανιχνεύσουν τα σημάδια κόπωσης και υπνηλίας. Τα συστήματα αυτά μπορούν να κατηγοριοποιηθούν βάσει του είδους μετρήσεων που παίρνουν όπως για παράδειγμα: βιολογικές μετρήσεις (EEG, ECG), μετρήσεις απόδοσης του οδηγού, μετρήσεις χρησιμοποιώντας αντικειμενικές κλίμακες (KSS, SSS) και άλλα.

Κάθε σύστημα έχει πλεονεκτήματα και μειονεκτήματα και βάσει της αναλογίας αυτών των δύο κρίνεται άμα μπορεί να υλοποιηθεί στην πράξη. Για παράδειγμα στην περίπτωση των βιολογικών μετρήσεων πέρα από το γεγονός πως οι βιολογικές μετρήσεις διαφέρουν από άνθρωπο σε άνθρωπο τίθεται το ερώτημα, πόσο εύκολα μπορούν να πραγματοποιηθούν; Οι συμβατικές μέθοδοι λήψης τέτοιων μετρήσεων (EEG, ECG, EOG) χρησιμοποιούν όργανα που ενδέχεται να κάνουν τον οδηγό να νιώσει άβολα ειδικά από τη στιγμή που μετρήσεις συμβαίνουν κατά τη διάρκεια της οδήγησης. Υπάρχουν ωστόσο τρόποι να παρθούν οι ανωτέρω μετρήσεις με άλλους τρόπους οι οποίοι με τη σειρά τους εισάγουν περισσότερο θόρυβο. Αυτός ο θόρυβος είναι δυνατό να περιοριστεί με περίπλοκες τεχνικές ψηφιακής επεξεργασίας σήματος Κατά συνέπεια μειώνεται σημαντικά η απόδοση του συστήματος και καθίσταται ακατάλληλο για τις περισσότερες περιπτώσεις.

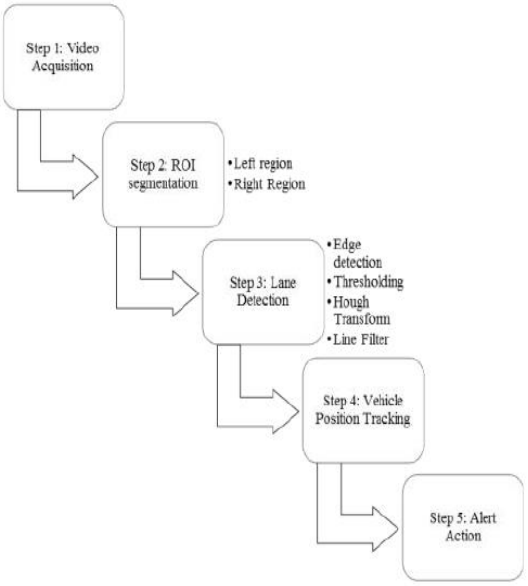
Τα πιο διαδεδομένα συστήματα φαίνεται να αποτελούν αυτά που βασίζονται στη μηχανική όραση μέσω της οποίας παρακολουθείται το πρόσωπο του οδηγού και καταγράφονται κάποιες ενδείξεις. Σε αυτές συμπεριλαμβάνονται η διάρκεια του βλεφαρίσματος, το ποσοστό βλεφαρίσματος, το κατά πόσο το βλέμμα του οδηγού παραμένει σταθερό κ.α. Όλα αυτά πάσχουν από μειωμένη ακρίβεια σε πραγματικές συνθήκες καθώς ο φωτισμός δεν είναι πάντα ιδανικός ή ο οδηγός μπορεί να φοράει γυαλιά. Γι' αυτό το λόγο έχουν αναπτυχθεί διάφορα συστήματα που χρησιμοποιούν κάμερες υπεριώδους φωτός, δύο ή και περισσότερες κάμερες για stereo vision. Πολλά από αυτά βρίσκονται στο εμπόριο όπως το Smart Eye [8] (Εικόνα 6), DMS [9] και πολλά ακόμα. Το μεγάλο αρνητικό των υλοποιήσεων που βρίσκονται στο εμπόριο είναι πως είναι κλειστού κώδικα και χρειάζονται αρκετά καλό καλιμπράρισμα για κάθε χρήστη [7]



Εικόνα 6 Ένα από τα χαρακτηριστικά του SmartEye (βιβλιογραφία [8])

1.4 Σύστημα προειδοποίησης απόκλισης από τη λωρίδα (LDWS)

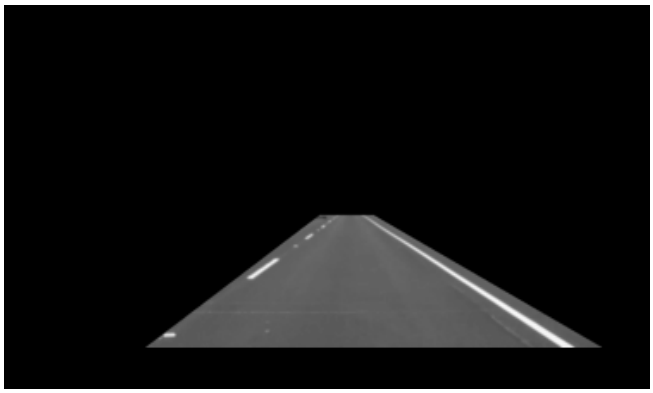
Καίριο ρόλο σε ένα ADAS παίζει το LDWS το οποίο έχει ως σκοπό να προειδοποιήσει τον οδηγό όταν αυτός αποκλίνει από τη λωρίδα είτε λόγω απροσεξίας είτε λόγω κούρασης. Είναι ένα σύστημα που βασίζεται εξ 'ολοκλήρου στη μηχανική όραση με όποιες αδυναμίες συνεπάγεται αυτό για τις οποίες θα μιλήσουμε παρακάτω. Το LDWS αποτελεί την ολοκληρωμένη έκδοση του συστήματος που παρουσιάζεται στο παρόν πόνημα και γι' αυτό το λόγο θα αναφερθούμε σε αυτό ελαφρώς εκτενέστερα. Παράλληλα θα δοθούν κομμάτια κώδικα σε γλώσσα python για την υλοποίηση ενός LDWS.



Εικόνα 7 Σκελετός ενός LDWS (βιβλιογραφία [41])

Το σύστημα αυτό μπορεί να υλοποιηθεί χρησιμοποιώντας τεχνικές της κλασσικής μηχανικής όρασης όπως και με τη χρήση μηχανικής μάθησης και νευρωνικών. Στο κεφάλαιο αυτό θα επικεντρωθούμε στις τεχνικές του κλασσικού machine vision. Ένα τέτοιο σύστημα έχει το πλεονέκτημα, σε σχέση με όσα συστήματα αναφέραμε μέχρι στιγμής, πως μπορεί να υλοποιηθεί χωρίς τη χρήση εξειδικευμένων αισθητήρων. Μία κάμερα στο μπροστινό μέρος του αμαξιού συνοδευόμενη με το κατάλληλο λογισμικό μπορεί να αποφέρει πολύ καλά αποτελέσματα ξοδεύοντας ελάχιστα χρήματα. Να σημειωθεί πως στο παρόν κεφάλαιο θα περιγράψουμε μία απλουστευμένη μορφή του συστήματος αυτού καθώς δεν είναι μέσα στους στόχους της πτυχιακής η εμβάθυνση στο παρόν θέμα. Θα αναφερθούμε ωστόσο εν συντομία σε κάποιες πιο περίπλοκες τεχνικές.Ο αλγόριθμος διακρίνεται σε μία σειρά από βήματα. Πρώτα η κάμερα καταγράφει βίντεο στο οποίο φαίνεται ο δρόμος μπροστά στο όχημα. Κάθε καρέ που καταγράφεται μετατρέπεται από RBG σε grayscale. Ένα αναγκαίο βήμα προκειμένου να εφαρμοστούν οι αλγόριθμοι για τους οποίους θα γίνει λόγος παρακάτω. Είναι σημαντικό η θέση της να

Εικόνα 9 Εικόνα από την εμπρόσθια κάμερα



Εικόνα 8 Περιοχή ενδιαφέροντος

παραμένει σταθερή καθώς το λογισμικό θεωρεί δεδομένη τη θέση της κάμερας. Το βίντεο επεξεργάζεται σε πραγματικό χρόνο και εξάγεται η περιοχή ενδιαφέροντος ή αλλιώς ROI (**Region Of Interest**). Ο λόγος που γίνεται αυτό είναι επειδή ο στόχος ενός τέτοιου συστήματος είναι γνωρίζει τη θέση του οχήματος σε σχέση με το κέντρο της λωρίδας. Επομένως μας ενδιαφέρει το κομμάτι του δρόμου που βρίσκεται μπροστά στο αμάξι και όχι η υπόλοιπη εικόνα. Αυτό εξοικονομεί υπολογιστική ισχύ καθώς δεν ξοδεύονται πόροι του συστήματος για την εφαρμογή των αλγόριθμων σε ολόκληρη την εικόνα αλλά σε ένα μικρό κομμάτι της. Ένα δείγμα κώδικα φαίνεται παρακάτω.

```
def ROI_select(img, vertices):

    #Δημιουργούμε μία κενή μάσκα
    mask = np.zeros_like(img)

    if len(img.shape) > 2:
        channel_count = img.shape[2]
        ignore_mask_color = (255,) * channel_count
    else:
        ignore_mask_color = 255

    # χρωματίζουμε όλα τα εικονοστοιχεία μέσα στο πολύγωνο που ορίζεται
από την μεταβλητή vertices
    cv2.fillPoly(mask, vertices, ignore_mask_color)

    #επιστρέφεται η εικόνα όπου μόνο τα εικονοστοιχεία μέσα στο πολύγωνο
έχουν μη μηδενική τιμή
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image
```

Έχοντας απομονώσει την περιοχή ενδιαφέροντος μπορούμε να εφαρμόσουμε κάποιους αλγόριθμους μηχανικής όρασης. Πρώτο βήμα είναι η ανίχνευση ακμών με τον αλγόριθμο **Canny Edge**. Ο αλγόριθμος **Canny Edge** πήρε το όνομα του από τον δημιουργό του John Canny και έχει τη δυνατότητα να ανιχνεύει τις ακμές σε μία εικόνα. Ως ακμή ορίζεται ένα σημείο της εικόνας όπου η φωτεινότητα αλλάζει απότομα με το μέτρο και τη γωνία να δίνονται από τους τύπους:

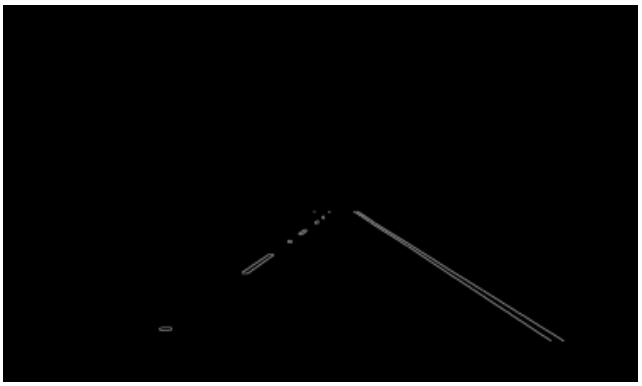
$$Edge_{gradient} = \sqrt{\left(G_x^2 + G_y^2\right)}$$

$$Angle(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Σε γλώσσα python είναι ως εξής:

```
cv2.Canny(image, low_threshold, high_threshold)
```

όπου $image$ είναι η εικόνα εισόδου, $low\_threshold$ και $high\_threshold$ είναι οι σταθερές οι οποίες ορίζουν ποια εικονοστοιχεία ανήκουν σε κάποια ακμή. Πιο συγκεκριμένα ο αλγόριθμος θεωρεί ακμή οποιοδήποτε $pixel > high\_threshold$ και απορρίπτει τα $pixels < low\_threshold$. Όσα πέφτουν ανάμεσα σε αυτές τις τιμές απορρίπτονται εκτός κι αν συνδέονται με κάποια «ισχυρή» ακμή.



Εικόνα 10 Canny Edge

Έχοντας ανιχνεύσει επιτυχώς τις ακμές στην περιοχή ενδιαφέροντος εφαρμόζεται ο αλγόριθμος **Hough Transform** ο οποίος βρίσκει τις γραμμές που υπάρχουν σε μία εικόνα. Ο αλγόριθμος αυτός δέχεται στην είσοδο του μία εικόνα όπως την Εικόνα 10 και επιστρέφει μία λίστα που περιέχει ζευγάρια συντεταγμένων (endpoints) $(x_1, y_1), (x_2, y_2)$ των γραμμών που ανιχνεύθηκαν. Στη συνέχεια αυτές φιλτράρονται ανάλογα με την κλίση του ώστε να προσδιοριστεί ποιες ανήκουν στην δεξιά και ποιες στην αριστερή λωρίδα. Στη συνέχεια βρίσκουμε τους συντελεστές ενός πολυωνύμου που ικανοποιούν τις σχέσεις:

Εικόνα 11 Hough Transform

$$f(y) = Ay^2 + By + C \text{ και } f(x) = Ax^2 + Bx + C.$$

Θεωρητικά μπορούμε να υπολογίσουμε τους συντελεστές μόνο του ενός πολυωνύμου καθώς οι γραμμές είναι σχεδόν παράλληλες και ενδέχεται αυτά τα δύο πολυώνυμα να έχουν ίδιες παραμέτρους.

Ο αλγόριθμος δε θα πρέπει να ανιχνεύει μόνο ευθείες αλλά και στροφές παντός τύπου. Για να μπορέσει να το κάνει αυτό θα πρέπει να μπορεί να υπολογιστεί η καμπυλότητα της στροφής. Η καμπυλότητα οπουδήποτε σημείου x της συνάρτησης $x = f(y)$ μπορεί να υπολογιστεί από τον τύπο:

$$R_{curve} = \frac{\left[1 + \left(\frac{dx}{dy}\right)^2\right]^{\frac{3}{2}}}{\left|\frac{d^2x}{dy^2}\right|}$$

Παίρνοντας την πρώτη και δεύτερη παράγωγο ενός από τα δύο πολυώνυμα παραπάνω έχουμε τελικά τη σχέση:

$$R_{curve} = \frac{(1 + (2Ay + b)^2)^{3/2}}{|2A|}$$

Στη συνέχεια είναι πολύ εύκολο να υπολογιστεί μαθηματικά η θέση του αμαξιού στη λωρίδα. Η θέση αυτή παρακολουθείται συνεχώς από το πρόγραμμα και μόλις ανιχνεύσει απόκλιση από τη λωρίδα χωρίς να έχει προηγηθεί ενεργοποίηση του φλας αυτό μπορεί είτε να διορθώσει αυτόματα την πορεία είτε να προειδοποιήσει τον οδηγό.

Δυστυχώς συστήματα σαν και αυτό, παρά τα άλματα της τεχνολογίας, υποφέρει από τις απότομες αλλαγές του φωτισμού και από το έντονο σκοτάδι. Επίσης υπάρχουν κάποιες στιγμές μέσα στη μέρα όπου οι ακτίνες του ήλιου πέφτουν με τέτοιο τρόπο πάνω στην κάμερα και την τυφλώνουν. Αυτό έχει ως αποτέλεσμα να μειώνεται αισθητά η απόδοση του αλγόριθμου. Δυστυχώς είναι πολύ δύσκολο για ένα σύστημα να ανταπεξέλθει σε όλες τις πιθανές συνθήκες φωτισμού λόγω αδυναμιών της μηχανικής όρασης. Μπορεί όμως να φτάσει σε ένα πολύ ικανοποιητικό επίπεδο.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε πως υπάρχουν πολύ καλύτερες μέθοδοι για την υλοποίηση όσων περιεγράφηκαν παραπάνω. Μερικές από αυτές καταφέρνουν και λύνουν κάποια από τα προβλήματα της προηγούμενης παραγράφου. Για παράδειγμα η μετατροπή όλων καρέ που τραβάει η κάμερα προκαλεί ένα σημαντικό πρόβλημα και γι' αυτό στην πράξη χρησιμοποιείται σπάνια. Όπως φαίνεται στην εικόνα, η αριστερή λωρίδα έχει

Εικόνα 12 Grayscale vs HLS thresholding

εξαφανιστεί καθιστώντας την ανίχνευση της δύσκολη αν όχι αδύνατη. Στην πράξη χρησιμοποιείται ένα διαφορετικό χρωματικό χώρο (color space) που ονομάζεται HLS. Χρησιμοποιώντας το και εφαρμόζοντας κάποιου είδους color thresholding διατηρείται όσο το δυνατόν περισσότερη πληροφορία και βελτιώνεται αισθητά η ανίχνευση λωρίδων. Είναι προφανές στην **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** πως στην π ερίπτωση του HLS thresholding διατηρείται περισσότερη πληροφορία και η αριστερή λωρίδα είναι ορατή. Υπάρχουν πολλές ακόμα τεχνικές που χρησιμοποιούνται όπως sliding window, gradient thresholding κ.ά. με τις οποίες δε θα ασχοληθούμε αλλά καλό είναι να γνωρίζουμε την ύπαρξη τους.

Όπως μπορεί να διαπιστώσει κανείς προκειμένου να υλοποιηθεί ένα LDWS με τεχνικές της μηχανικής όρασης απαιτείται εκτενής προ επεξεργασία προκειμένου να ανιχνευθούν επιτυχώς οι λωρίδες. Υπάρχουν πάρα πολλές παράμετροι τις οποίες ο προγραμματιστής θα πρέπει να ρυθμίσει κάτι που είναι άκρως χρονοβόρο και χωρίς εγγυημένα αποτελέσματα σε όλες τις συνθήκες. Αυτό καλεί για μία πιο εξεζητημένη προσέγγιση, μία προσέγγιση που θα απαλλάξει τον σχεδιαστή από το βάρος της ρύθμισης τόσων διαφορετικών παραμέτρων. Στα παρακάτω κεφάλαια θα γνωρίσουμε τους τρόπους που μπορούμε να πετύχουμε ίδια και

καλύτερα αποτελέσματα όσον αφορά την ανίχνευση λωρίδων δίνοντας σε ένα υπολογιστικό σύστημα τη δυνατότητα να ρυθμίζει μόνο του της παραμέτρους και να μαθαίνει χωρίς τη δική μας παρέμβαση.

2. Μηχανική Μάθηση

Για την επίλυση πολύπλοκων προβλημάτων απαιτείται η χρήση ευφυών προσεγγίσεων που βασίζονται στην ικανότητα ενός συστήματος να μαθαίνει. Έτσι δημιουργήθηκε η μηχανική μάθηση (machine learning ή ML) ένα υποπεδίο της τεχνητής νοημοσύνης που έχει ως σκοπό να δώσει, μέσω κατάλληλων αλγορίθμων, τη δυνατότητα μάθησης στους υπολογιστές.

Πιο συγκεκριμένα η μηχανική μάθηση αποτελεί το γνωστικό αντικείμενο που ευθύνεται για τη μελέτη και ανάπτυξη αλγορίθμων και στατιστικών μοντέλων που θα επιτρέψουν σε ένα υπολογιστικό σύστημα να εκτελεί μία εργασία χωρίς σαφείς οδηγίες. Ένας τέτοιος αλγόριθμος είναι ικανός να αναγνωρίσει μοτίβα και ενδιαφέρουσες δομές στα δεδομένα που του παρέχονται και να δημιουργήσει ένα στατιστικό μοντέλο ικανό να κάνει προβλέψεις και να παίρνει αποφάσεις. Αυτό έδωσε τη δυνατότητα στους επιστήμονες να μοντελοποιούν προβλήματα που μέχρι πρότινος ήταν δύσκολο να διατυπωθούν και αντικατέστησε ένα μέρος της ανθρώπινης συμβολής. Για παράδειγμα ένας αλγόριθμος εκμάθησης μπορεί και αναγνωρίζει μόνος του τα χρήσιμα χαρακτηριστικά (features) σε μία εικόνα (στην περίπτωση της μηχανικής όρασης). Αυτό παλαιότερα γινόταν χειροκίνητα και απαιτούσε μεγάλη εμπειρία από τη μεριά του προγραμματιστή καθώς είναι δύσκολο να γνωρίζουμε εκ των προτέρων ποια χαρακτηριστικά είναι σημαντικά.

Ο όρος μηχανική μάθηση επινοήθηκε το 1959 από τον **Arthur Samuel**, έναν πρωτοπόρο της τεχνητής νοημοσύνης, ο οποίος διατύπωσε και τον πρώτο ορισμό της μηχανικής μάθησης «**η μηχανική μάθηση είναι το πεδίο έρευνας το οποίο δίνει τη δυνατότητα στους υπολογιστές να μάθουν χωρίς να προγραμματίζονται επακριβώς**». Έκτοτε έχουν προταθεί αρκετοί ορισμοί εκ των οποίων ο πιο περιεκτικός και πρακτικός είναι αυτός που δόθηκε από τον:

- **Mitchell (1997)**, «Ένα πρόγραμμα υπολογιστή θεωρείται ότι μαθαίνει από την εμπειρία E σε σχέση με μία κατηγορία εργασιών T και μία μετρική απόδοσης P, αν η απόδοση του σε εργασίες της T, όπως μετριούνται από την P, βελτιώνονται με την εμπειρία E»

Όπως είναι αναμενόμενο οι αλγόριθμοι εκμάθησης χρησιμοποιούνται κατά κόρων τόσο στη μηχανική όραση, που θα μας απασχολήσει παρακάτω, όσο και σε αντικείμενα όπως η ιατρική και μετεωρολογία.. Η ευρεία διαθεσιμότητα τόσο της υπολογιστικής ισχύς όσο και των δεδομένων έχει καταστήσει τη μηχανική μάθηση ένα αναπόσπαστο εργαλείο για τους επιστήμονες του κλάδου αυτού.

2.1 Τύποι μηχανικής μάθησης

Κατά τη διάρκεια της τελευταίας δεκαετίας η μηχανική μάθηση έχει σημειώσει πρωτόγνωρη πρόοδο σε ποικίλους τομείς όπως αναγνώριση εικόνας (image recognition), αυτόνομη οδήγηση, ανάλυση ιατρικής εικόνας ακόμα και στο παίξιμο περίπλοκων παιχνιδιών όπως το **Go**. Ανάλογα με τη φύση τους προς επίλυση προβλήματος μπορούμε να χρησιμοποιήσουμε μία από τις τρεις τεχνικές μηχανικής μάθησης που περιγράφονται παρακάτω.

2.1.1 Εποπτευόμενη μάθηση (supervised learning)

Σε αυτή την κατηγορία μηχανικής μάθησης παρέχουμε στο σύστημα μεταβλητές εισόδου (x) και μεταβλητές εξόδου (Y) και ο αλγόριθμος μάθησης προσπαθεί να μάθει το γενικό κανόνα ώστε να μπορεί να αντιστοιχίσει τις εισόδους με τις εξόδους.

$$Y = f(x)$$

Σύμφωνα με το Zurada [10] η εποπτευόμενη μάθηση μπορεί να παρομοιαστεί με μία τάξη όπου οι μαθητές μαθαίνουν δίνοντας απαντήσεις στα ερωτήματα που θέτει ο δάσκαλος και αυτός με τη σειρά επιβραβεύει τις σωστές και διορθώνει τις λανθασμένες.

Τα δεδομένα που εισάγονται στον αλγόριθμο αποτελούνται από επισημασμένα παραδείγματα (labeled examples) όπου στην περίπτωση του προβλήματος ανίχνευσης λωρίδων είναι εικόνες στις οποίες έχουμε επισημάνει τα σημεία που βρίσκονται οι λωρίδες.

Κλασικά παραδείγματα εποπτευόμενης μάθησης αποτελούν οι:

- Παλινδρόμηση (regression)
- Κατηγοριοποίηση (classification)

2.1.2 Μη εποπτευόμενη μάθηση (unsupervised learning)

Στη μη εποπτευόμενη μάθηση, σε αντίθεση με την εποπτευόμενη, δε παρέχουμε στο σύστημα καμία πληροφορία για το ποια είναι η σωστή έξοδος. Το σύστημα προσπαθεί να μοντελοποιήσει την υποκείμενη δομή ή κατανομή των δεδομένων με σκοπό να μάθει περισσότερα γι' αυτά τα δεδομένα.

Ο λόγος που ονομάζεται "μη εποπτευόμενη μάθηση" δεν είναι επειδή δεν υπάρχει δάσκαλος διότι μάθηση χωρίς δάσκαλο είναι αδύνατη αλλά επειδή, σύμφωνα πάλι με το Zurada [10], στην προκειμένη περίπτωση είναι σα να βλέπουν οι μαθητές μία βιντεοσκοπημένη διάλεξη. Αυτό σημαίνει πως ο δάσκαλος δεν είναι σε θέση να απαντήσει σε ερωτήσεις ή να δώσει διευκρινίσεις.

Τα δεδομένα που εισάγουμε στο σύστημα ονομάζονται μη επισημασμένα παραδείγματα (unlabeled examples)και στο πρόβλημα της ανίχνευσης λωρίδων είναι εικόνες

από δρόμους χωρίς καμία πληροφορία για το που βρίσκονται οι λωρίδες. Σε αυτή την περίπτωση ο αλγόριθμος θα πρέπει, όπως αναφέρθηκε και παραπάνω, να ανακαλύψει ενδιαφέρουσες δομές στα δεδομένα που του δίνονται ώστε να αποκτήσει την ικανότητα να ανιχνεύει λωρίδες.

Παραδείγματα εποπτευόμενης μάθησης αποτελούν:

- Ομαδοποίηση (Clustering)
- Συσχέτιση (Association)

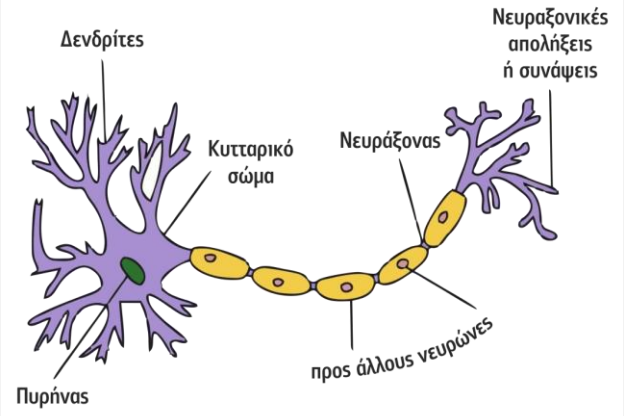
2.1.3 Ημι-εποπτευόμενη μάθηση (semi supervised learning)

Για λόγους πληρότητας αξίζει να αναφέρουμε πως υπάρχει μία ακόμα κατηγορία μηχανικής μάθησης, την ήμι-εποπτευόμενη. Στην ημι-εποπτευόμενη μάθηση παρέχουμε στον αλγόριθμο μία μίξη επισημασμένων και μη επισημασμένων παραδειγμάτων. Το μοντέλο με τη σειρά του θα πρέπει να ανακαλύψει τις ενδιαφέρουσες δομές στα δεδομένα και να τα οργανώσει προκειμένου να κάνει χρήσιμες προβλέψεις.

2.2 Νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks ή εν συντομία ANN) αποτελούν μία σχετικά νέα περιοχή στην επιστήμη των υπολογιστών η οποία έχει γνωρίσει τεράστια άνθηση τα τελευταία χρόνια. Με τον όρο αυτό αναφερόμαστε σε ένα σύνολο διαφορετικών μαθηματικών μοντέλων εμπνευσμένων από τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου που προσπαθούν να μιμηθούν τη συμπεριφορά του [11].

Είναι γνωστό στην επιστημονική κοινότητα πως ο εγκέφαλος αποτελείται από μία πληθώρα νευρωνικών δικτύων στα οποία θα αναφερόμαστε με τον όρο Βιολογικά Νευρωνικά Δίκτυα ή σε σύντμηση ΒΝΔ. Η συλλογική συμπεριφορά των δικτύων του εγκεφάλου προσδίδει στον εγκέφαλο την ικανότητα μάθησης, ανάκλησης και γενίκευσης από δεδομένα ή πρότυπα [12]. Κάθε δίκτυο αποτελείται από ένα μεγάλο αριθμό διακριτών στοιχείων με υψηλό βαθμό διασύνδεσης, τους νευρώνες. Ο νευρώνας αποτελεί τη στοιχειώδη ανεξάρτητη μονάδα επεξεργασίας του δικτύου και ο ρόλος του περιλαμβάνει τη λήψη, επεξεργασία και διαμοιρασμό πληροφορίας, με μορφή ηλεκτρικού ρεύματος, στους υπόλοιπους νευρώνες.



Εικόνα 13 Βιολογικός Νευρώνας (βιβλιογραφία

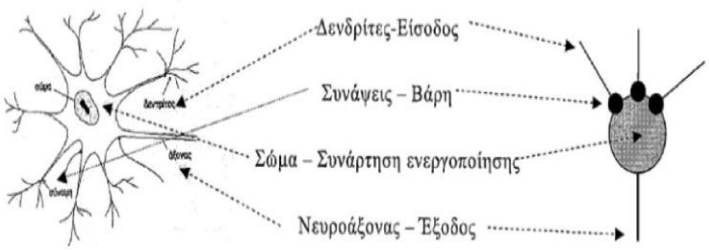
Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ) προσπαθούν να μιμηθούν τον τρόπο λειτουργίας των αντίστοιχων ΒΝΔ με σκοπό να αποκτήσουν οι υπολογιστές τις ίδιες λειτουργικές ικανότητες που παρουσιάζει ο εγκέφαλος (μνήμη, ανάκληση, γενίκευση). Όπως είναι επόμενο λοιπόν η δομή τους είναι παρόμοια καθώς αποτελούνται από ένα σύνολο στοιχείων που

ονομάζονται τεχνητοί νευρώνες. Οι επικοινωνία μεταξύ των νευρώνων επιτυγχάνεται μέσω συνδέσεων που ονομάζονται συνάψεις. Αξίζει να σημειωθεί πως ο βαθμός αλληλεπίδρασης μεταξύ τον νευρώνων δεν είναι πάντα ο ίδιος και μάλιστα μεταβάλλεται συνεχώς κατά την εκπαίδευση. Ο βαθμός αλληλεπίδρασης καθορίζεται από το συναπτικό βάρος για το οποίο θα μιλήσουμε παρακάτω. Έτσι ένα ΤΝΔ δίκτυο μπορεί να εξελιχθεί και να προσαρμοστεί καθώς αλληλοεπιδρά με το περιβάλλον του.

Παρά το γεγονός ότι τα ΒΝΔ και τα ΤΝΔ μοιράζονται αρκετά χαρακτηριστικά και η ορολογία που χρησιμοποιείται για να τα περιγράψει είναι η ίδια, θα πρέπει να τονιστεί πως απέχουν πάρα πολύ τόσο στον τρόπο λειτουργίας (ένας βιολογικός νευρώνας λειτουργεί με αρκετά περίπλοκο τρόπο) όσο και σε έκταση. Επομένως τα ΤΝΔ σε καμία περίπτωση δε πλησιάζουν την πολυπλοκότητα των αντίστοιχων βιολογικών χωρίς αυτό να σημαίνει ότι δεν είναι ικανά να επιλύσουν αρκετά πολύπλοκα προβλήματα.

2.3 Τεχνητός Νευρώνας

Σε πλήρη αντιστοιχία με τα ΒΝΔ, τα ΤΝΔ απαρτίζονται από ένα σύνολο υπολογιστικών μονάδων που ονομάζονται τεχνητοί νευρώνες. Οι τεχνητοί νευρώνες, όπως μπορεί να συμπεράνει κανείς, αποτελούν μία αρκετά απλοποιημένη εκδοχή των νευρώνων που συναντάμε στον ανθρώπινο εγκέφαλο. Σκοπός τους είναι η λήψη, επεξεργασία και διάδοση ηλεκτρικών σημάτων στους υπόλοιπους νευρώνες. Τα μοντέλα αυτά εμπνεύστηκαν από την έρευνα που έκαναν οι Hodgkin & Huxley πάνω στο πως ένας νευρώνας παράγει και διαδίδει ηλεκτρικούς παλμούς.



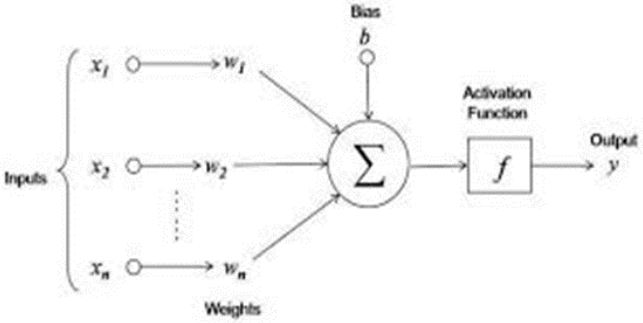
Εικόνα 14 Αντιστοιχία βιολογικού με τεχνητού νευρώνα (βιβλιογραφία [5])

Κάθε νευρώνας μπορεί να διαθέτει μία ή και παραπάνω εισόδους. Οι είσοδοι αυτοί μπορεί να είναι οι έξοδοι κάποιων άλλων νευρώνων ή εξωτερικά ερεθίσματα. Σε αντίθεση με τις εισόδους η έξοδος μπορεί να είναι μόνο μία η οποία όμως μπορεί να οδηγηθεί στην είσοδο ενός ή πολλών επιπλέον νευρώνων. Κάθε είσοδος χαρακτηρίζεται από ένα συναπτικό βάρος το οποίο ορίζει πόσο πολύ ή λίγο συνεισφέρει η κάθε είσοδος στην έξοδο.

Οι τεχνητοί νευρώνες ποικίλουν σε είδη και κάθε είδος εξειδικεύεται σε ένα συγκεκριμένο τύπο εργασίας π.χ. επεξεργασία εικόνας. Όπως είναι λογικό η επιλογή του κατάλληλου τεχνητού νευρώνα γίνεται βάσει της φύσης του προς επίλυση προβλήματος ενώ δεν είναι σπάνιο να υπάρχουν πολλά διαφορετικά είδη νευρώνων στο ίδιο δίκτυο.

2.3.1 Αρχή λειτουργίας τεχνητού νευρώνα

Κάθε νευρώνας όπως είπαμε μπορεί να διαθέτει μία ή και παραπάνω εισόδους τις οποίες θα συμβολίζουμε ως $\{\tilde{x}_n\}$ $με$ $n = 1,2 \dots, N$ αλλά μόνο μία έξοδο $\{\tilde{y}_n\}$. Κάθε είσοδος αντιστοιχείται με ένα συναπτικό βάρος $\{w_n\}$ $με$ $n = 1,2, \dots, N$ το οποίο υποδηλώνει πόσο συνεισφέρει κάθε είσοδος στην έξοδο του νευρώνα. Γνωρίζοντας τα παραπάνω και μελετώντας την *εικόνα 4* μπορούμε να χωρίσουμε τη λειτουργία του νευρώνα σε τρία μέρη.



Εικόνα 15 Τεχνητός Νευρώνας (βιβλιογραφία [5])

Πρώτα όλες οι είσοδοι πολλαπλασιάζονται με τα αντίστοιχα βάρη τους. Στη συνέχεια οι σταθμισμένες είσοδοι προστίθενται με ένα εξωτερικά εφαρμοζόμενο παράγοντα που τον ονομάζουμε bias.

$$z = \sum_{i=1}^{n}(x_i w_i) + b$$

Η λειτουργία του νευρώνα μέχρι στιγμής είναι γραμμική, δηλαδή η έξοδος του είναι ένας γραμμικός συνδυασμός των εισόδων. Λόγω του γεγονότος ότι ένας γραμμικός νευρώνας έχει περιορισμένες δυνατότητες και δε μπορεί να ανταπεξέλθει σε πιο περίπλοκες διαδικασίες εφαρμόζουμε μία συνάρτηση ενεργοποίησης όπως φαίνεται παρακάτω:

$$a = \sigma(b + \sum_{i=1}^{n} X_i W_i)$$

2.4 Συναρτήσεις Ενεργοποίησης

Γιατί όμως χρειαζόμαστε συναρτήσεις ενεργοποίησης; Όπως αναφέρθηκε παραπάνω, ένας γραμμικός νευρώνας έχει πολύ περιορισμένες δυνατότητες και είναι ανίκανος να υπολογίσει περίπλοκες συναρτήσεις όπως επίσης να ανταπεξέλθει σε δεδομένα που παρουσιάζουν χαοτική συμπεριφορά. Για να αντιμετωπίσουμε το πρόβλημα αυτό εφαρμόζουμε μία μη γραμμική συνάρτηση ενεργοποίησης που δίνει στο μοντέλο την ικανότητα να ανταπεξέλθει σε περίπλοκα δεδομένα όπως εικόνες, ήχος κτλ. Επιπλέον η φύση της συνάρτησης αυτής επηρεάζει το πόσο γρήγορα ή αργά θα μαθαίνει το μοντέλο κάτι που θα δούμε παρακάτω.

Όπως και με τον τύπο των νευρώνων έτσι και εδώ έχουμε μια πληθώρα συναρτήσεων ενεργοποίησης. Κάθε μία από αυτές προτιμάται σε διαφορετικούς τύπους προβλημάτων ενώ είναι πολύ συνηθισμένο ένα νευρωνικό δίκτυο να διαθέτει πολλά είδη συναρτήσεων ενεργοποίησης.

Για λόγους πληρότητας θα αναφέρουμε πως στο πρώτο μοντέλο νευρώνα χρησιμοποιήθηκε μία βηματική συνάρτηση (Heaveside step function) όμως γρήγορα παρατηρήθηκε πως οι μη γραμμικές συναρτήσεις και οι παράγωγοι τους προσφέρουν χρήσιμες πληροφορίες που μπορούν να χρησιμοποιηθούν για την περεταίρω βελτίωση του συστήματος.

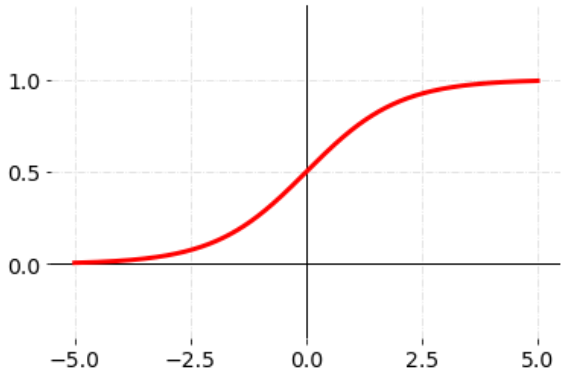
2.4.1 Σιγμοειδής συνάρτηση (sigmoid function)

Η σιγμοειδής συνάρτηση είναι μία μονοτονική, φραγμένη και παραγωγίσιμη συνάρτηση με μαθηματικό τύπο:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

που πήρε το όνομα της από το χαρακτηριστικό της σχήμα που θυμίζει λατινικό “S”. Γνώρισε μεγάλη άνθηση λόγω των χαρακτηριστικών της όπως για παράδειγμα το γεγονός πως η ομαλή κλίση (smooth gradient) της συνάρτησης αυτής αποτρέπει την εμφάνιση «αναπηδήσεων» στις τιμές της εξόδου (exploding gradient). Επιπλέον φράζει τις εξόδους των νευρώνων ανάμεσα στις τιμές 0 και 1 κάτι που την καθιστά ιδανική για προβλήματα δυαδικής κατηγοριοποίησης (binary classification) και λογιστικής παλινδρόμησης (logistic regression).

Γρήγορα όμως ξεπεράστηκε διότι παρατηρήθηκε πως για πολύ μεγάλες ή πολύ μικρές τιμές του Z η κλίση της συνάρτησης είναι κοντά στο μηδέν με αποτέλεσμα το μοντέλο να μη μπορεί να εκπαιδευτεί περεταίρω. Τέλος μπορεί εύκολα να παρατηρήσει κανείς πως η συνάρτηση δεν είναι zero- centered κάτι που, όπως αποδείχτηκε, προκαλεί προβλήματα στην οπισθοδιάδοση.

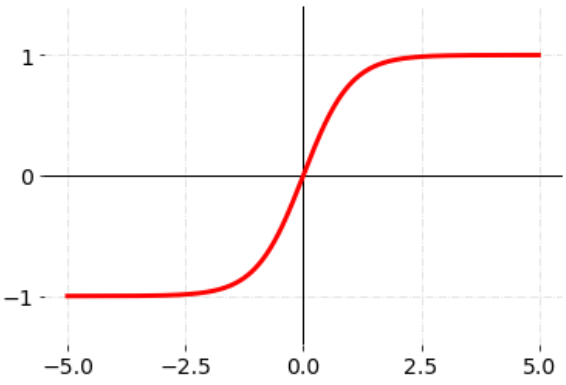


Εικόνα 16 Σιγμοειδής Συνάρτηση

2.4.2 Υπερβολική εφαπτομένη (hyperbolic tangent-tanh)

Μια παραλλαγή της σιγμοειδούς συνάρτησης που γνωρίσαμε νωρίτερα είναι η υπερβολική εφαπτομένη (tanh). Η tanh είναι zero centered και επομένως δε συναντάμε το πρόβλημα που είχε η σιγμοειδής. Αυτό την καθιστά προτιμότερη σχεδόν σε κάθε περίπτωση.

Μολαταύτα η tanh χρησιμοποιείται σπάνια έως καθόλου πλέον καθώς παρουσιάζει τα ίδια μειονεκτήματα με τη σιγμοειδή συνάρτηση. Ο μαθηματικός τύπος της είναι ο εξής:



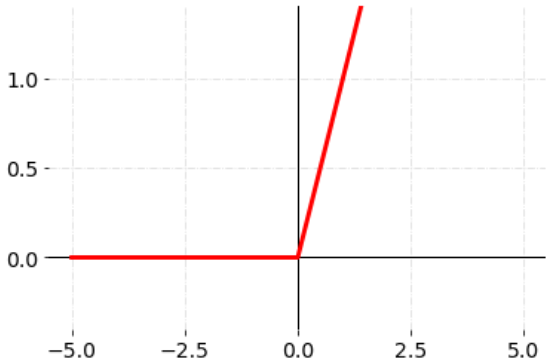
Εικόνα 17 Tanh

$$\sigma(z) = \frac{1 + e^{-2z}}{1 - e^{-2z}}$$

2.4.3 Ανορθωμένη Γραμμική Μονάδα - ReLU (Rectified Linear Unit)

Μια συνάρτηση που χρησιμοποιείται πάρα πολύ στα νευρωνικά ιδίως στα συνελικτικά είναι η ReLU η οποία όντας υπολογιστικά αποδοτική, σε αντίθεση με τις προηγούμενες δύο, επιταχύνει σημαντικά την εκπαίδευση του δικτύου. Ο μαθηματικός της τύπος φαίνεται παρακάτω:

$$\sigma(z) = \max(0, z) = \begin{cases} 0, & if\ z < 0 \\ 1, & if\ z \geq 0 \end{cases}$$



Εικόνα 18 RELU

Ωστόσο, κοιτώντας τη γραφική παράσταση μπορεί κανείς να διακρίνει πως για μηδενικές ή αρνητικές τιμές εισόδων η κλίση της συνάρτησης γίνεται μηδέν. Αυτό έχει ως αποτέλεσμα οι νευρώνες των οποίων οι είσοδοι είναι στα αρνητικά να «νεκρώνουν» και η έξοδος τους να είναι συνεχώς μηδέν καθιστώντας τους άχρηστους. Αυτό το φαινόμενο είναι γνωστό ως dying ReLU problem και αποτελεί ένα από τα σημαντικότερα μειονεκτήματα της εν λόγω συνάρτησης καθώς, λόγω της φύσης της ReLU, ένας «νεκρός» νευρώνας δύσκολα επανέρχεται. Αυτό ενέχει τον κίνδυνο ένα μεγάλο μέρος του δικτύου, εφόσον υπάρξουν οι κατάλληλες συνθήκες, να μείνει ανενεργό μειώνοντας σημαντικά την αποτελεσματικότητα του.

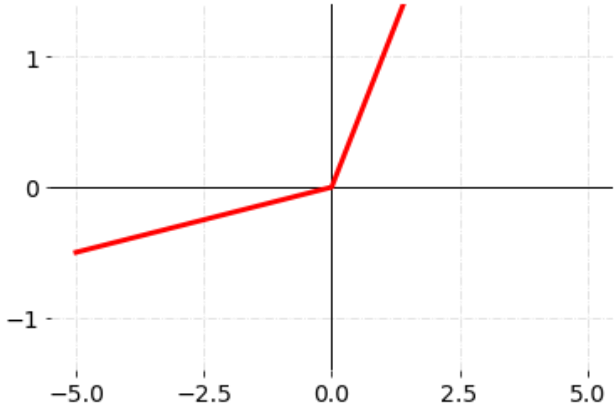
2.4.4 Leaky ReLU & Parametric ReLU (PReLU)

Για την αποφυγή του προαναφερθέντος προβλήματος προστέθηκε στη ReLU μία μικρή θετική κλίση στα αρνητικά και πήρε το όνομα διαρρέουσα ανορθωμένη γραμμική μονάδα (leaky ReLU). Με αυτό τον τρόπο η κλίση της συνάρτησης δε γίνεται ποτέ μηδέν αποτρέποντας έτσι την απενεργοποίηση των νευρώνων. Ο μαθηματικός τύπος είναι ο εξής:

$$\sigma(z) = \max(0.01z, z) = \begin{cases} 0.01, & if\ z < 0 \\ 1, & if\ z \geq 0 \end{cases}$$

Υπάρχει και μία παραλλαγή της Leaky ReLU , η Parametric ReLU όπου η κλίση στα αρνητικά δεν είναι πλέον προκαθορισμένη αλλά είναι μια ακόμη παράμετρος που θα πρέπει να μάθει το δίκτυο.

$$\sigma(z) = \max(az, z) = \begin{cases} a, & if\ x < 0 \\ 1, & if\ x \geq 0 \end{cases}$$

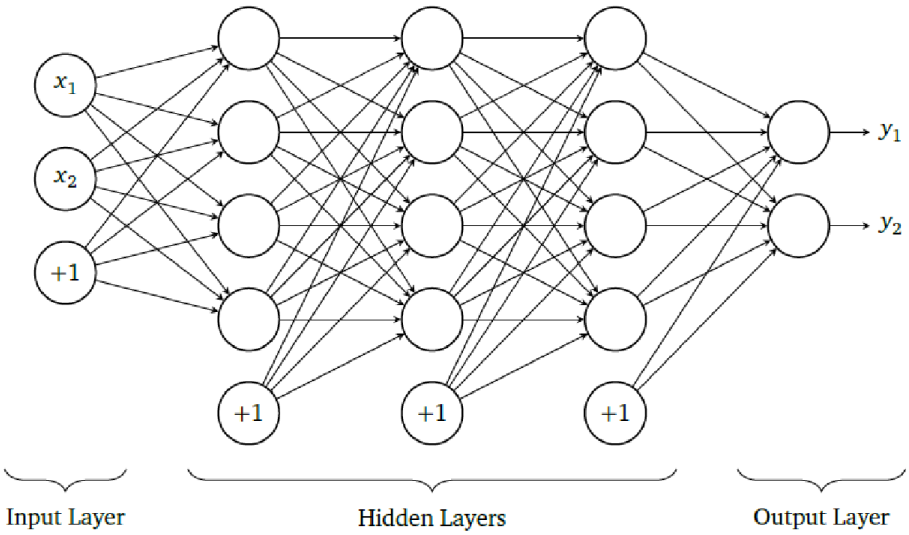


Εικόνα 19 Leaky Relu & Prelu

2.5 Μοντέλο Πολυεπίπεδου Νευρωνικού Δικτύου

Ένα νευρωνικό δίκτυο αποτελείται, όπως έχει αναφερθεί πολλάκις, από ένα σύνολο νευρώνων συνδεδεμένων με τέτοιο τρόπο ώστε η έξοδος του κάθε νευρώνα να αποτελεί είσοδο σε άλλους νευρώνες. Πιο συγκεκριμένα η έξοδος ενός προγενέστερου νευρώνα συνδέεται με την είσοδο ενός μεταγενέστερου νευρώνα κ.ο.κ.

Τα νευρωνικά δίκτυα διακρίνονται σε επίπεδα (**Layers**). Η είσοδος του δικτύου αποτελεί το επίπεδο εισόδου (**Input Layer**) και αντίστοιχα η έξοδος το επίπεδο εξόδου (**Output Layer**). Όλα τα επίπεδα που παρεμβάλλονται μεταξύ αυτών των δύο ονομάζονται κρυφά επίπεδα (**Hidden Layers**) καθώς τα αποτελέσματα των υπολογισμών τους δεν είναι άμεσα διαθέσιμα σε εμάς. Ένα δίκτυο ενδέχεται να έχει κανένα, ένα ή και παραπάνω από ένα κρυφά επίπεδα. Η αρίθμηση των επιπέδων ξεκινάει από το μηδέν όπου επίπεδο 0 είναι η είσοδος, επίπεδο 1 είναι το πρώτο κρυφό επίπεδο κ.ο.κ.



Εικόνα 20 Μοντέλο νευρωνικού δικτύου [13]

Στην παραπάνω εικόνα φαίνεται ένα νευρωνικό δίκτυο με τρία κρυφά επίπεδα. Ο εξωτερικά εφαρμοζόμενος παράγοντας bias απεικονίζεται ως ένας νευρώνας του οποίου η

έξοδος είναι πάντα 1. Θα πρέπει να αναφερθεί πως ο bias δεν προσμετράτε στον αριθμό νευρώνων του κάθε επιπέδου επομένως για το παραπάνω νευρωνικό δίκτυο έχουμε:

- 5 επίπεδα επομένως L = 4
- $n^{[0]} = 3,\ n^{[1]} = 4,\ n^{[2]} = 4,\ n^{[3]} = 4,\ n^{[4]} = 2$

Χρησιμοποιώντας τη σημειογραφία μπορούμε να εξάγουμε τις παρακάτω εξισώσεις:

$$Z^{[1]} = W^{[1]T}X + b^{[1]}, \quad \alpha^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]T}\alpha^{[1]} + b^{[2]}, \quad \alpha^{[2]} = \sigma(Z^{[2]})Z^{[3]} = W^{[3]T}a^{[2]} + b^{[3]}, \quad a^{[3]} = \sigma(Z^{[3]})$$
$$Z^{[4]} = W^{[4]T}a^{[3]} + b^{[4]}, \quad \hat{y} = a^{[4]} = \sigma(Z^{[4]})$$

3. Μηχανική Όραση

Η όραση αποτελεί μία από τις πιο συναρπαστικές και ταυτόχρονα περίπλοκες λειτουργίες του ανθρώπινου οργανισμού μέσω της οποίας αντιλαμβανόμαστε το χώρο γύρω μας. Για την ακρίβεια είναι τόσο περίπλοκη που ένα μεγάλο μέρος του εγκεφάλου είναι αφιερωμένο σε αυτή. Ένα ιδιαίτερο χαρακτηριστικό της όρασης είναι πως χρησιμοποιώντας απλά το φως που αντανακλάται στα μάτια μας από διάφορα αντικείμενα μπορεί να συμπεράνει χαρακτηριστικά του τρισδιάστατου κόσμου. Με άλλα λόγια το κομμάτι του εγκεφάλου που ευθύνεται για την αίσθηση της όρασης μπορεί να μέσω μίας δισδιάστατης εικόνας που φτάνει στα μάτια μας να αντιληφθεί την τρισδιάστατη φύση του αντικειμένου.

Αυτή ακριβώς τη δυνατότητα, και πολλές άλλες, προσπαθεί να δώσει η επιστήμη που ονομάζουμε μηχανική όραση στους υπολογιστές. Πιο συγκεκριμένα, μηχανική όραση ονομάζουμε την επιστήμη που μέσω ειδικών αλγορίθμων επιχειρεί να δώσει σε ένα υπολογιστή την ικανότητα να αντιλαμβάνεται τον κόσμο γύρω του και να πραγματοποιεί εργασίες όπως *κατηγοριοποίηση* (classification), *εντοπισμό* (localization), *ανίχνευση αντικειμένων* (object detection) και οποιοδήποτε συνδυασμό των παραπάνω. Η υψηλή αποτελεσματικότητα της μηχανικής όρασης σε συνδυασμό με την πρόοδο των αισθητήρων και καμερών είχε ως αποτέλεσμα να χρησιμοποιηθεί σε μία πληθώρα εφαρμογών και να αλλάξει ριζικά πολλούς τομείς (ρομποτική ιατρική, αυτόνομη οδήγηση κ.α.).

Αρχικά η μηχανική όραση στηριζόταν εξ ’ολοκλήρου στην προ επεξεργασία των εικόνων και την εξαγωγή διαφόρων χαρακτηριστικών από αυτές. Αυτό απαιτούσε τεράστια ποσά χρόνου από τη μεριά του προγραμματιστή για τη σωστή ρύθμιση διαφόρων παραμέτρων. Επιπλέον η εξαγωγή των χαρακτηριστικών βασιζόταν εξ’ ολοκλήρου στον ανθρώπινο παράγοντα, πράγμα που σημαίνει δύο πράγματα. Κατ’ αρχάς ο προγραμματιστής θα πρέπει, βασιζόμενος στην εμπειρία του, να αποφασίσει ποια χαρακτηριστικά είναι σημαντικά και ποια όχι. Στην πράξη όμως τις περισσότερες φορές όμως είναι δύσκολο να γνωρίζουμε εκ των προτέρων ποια χαρακτηριστικά είναι χρήσιμα και ποια όχι. Το δεύτερο και ίσως σημαντικότερο μειονέκτημα είναι η έλλειψη εκπαίδευσης. Αυτό σημαίνει πως ο αλγόριθμος δε μαθαίνει και επομένως δε βελτιώνει τις προβλέψεις του. Όλα εξαρτώνται από το πόσο καλά έχει ρυθμίσει ο προγραμματιστής τις διάφορες παραμέτρους του προγράμματος. Οι ρυθμίσεις αυτές ωστόσο όσο καλές και να είναι όταν, στο παράδειγμα της αυτόνομης οδήγησης, αλλάζει ριζικά η οδική

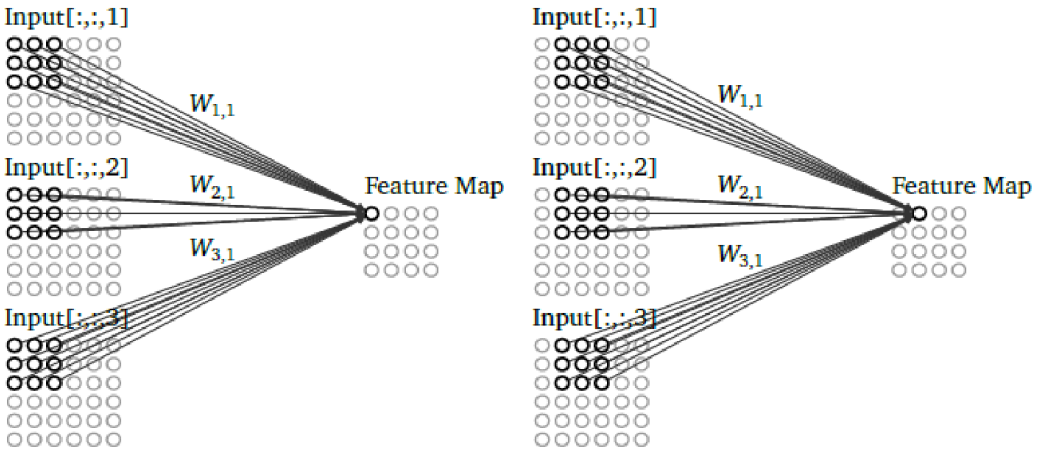
σκηνή ενδέχεται να μην αποφέρουν το επιθυμητό αποτέλεσμα. Με την έλευση των νευρωνικών δικτύων και πιο συγκεκριμένα των συνελικτικών νευρωνικών δικτύων το βάρος της ρύθμισης των παραμέτρων περνάει από τα χέρια του προγραμματιστή στο ίδιο το δίκτυο λύνοντας τα παραπάνω προβλήματα.

3.1 Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (ή σε σύντμηση ΣΝΔ) αποτελούν μία ειδική κατηγορία τεχνητών νευρωνικών δικτύων που ειδικεύονται στην επεξεργασία δεδομένων που διαθέτουν πλεγματοειδή τοπολογία π.χ. μία εικόνα μπορεί να αναπαρασταθεί με ένα δισδιάστατο πίνακα. Η ονομασία τους προκύπτει από το γεγονός ότι χρησιμοποιούν μία μαθηματική πράξη που ονομάζεται συνέλιξη (*convolution*) η οποία αντικαθιστά τον πολλαπλασιασμό πινάκων που είδαμε στα νευρωνικά δίκτυα του προηγούμενου κεφαλαίου.

Δομικά «τουβλάκια» των ΣΝΔ αποτελούν τα επίπεδα συνέλιξης και χωρικής υποδειγματοληψίας τα οποία είναι ειδικά σχεδιασμένα ώστε να δέχονται και να επεξεργάζονται δεδομένα με δύο ή περισσότερες διαστάσεις. Εν συντομία σε ένα επίπεδο συνέλιξης κάθε φίλτρο εξάγει χαρακτηριστικά (features) από τα δεδομένα εισόδου και τα τοποθετεί σε ένα χάρτη χαρακτηριστικών (feature map) ο οποίος αποτελεί και την έξοδο του επιπέδου. Το επίπεδο χωρικής υποδειγματοληψίας μειώνει το χωρικό μέγεθος των χαρτών αυτών. Όλα αυτά θα αναλυθούν σε παρακάτω παραγράφους όπου θα μιλήσουμε για τα κρυφά επίπεδα ενός ΣΝΔ.

Η τεράστια επιτυχία που έχουν γνωρίσει τα συνελικτικά νευρωνικά δίκτυα έγκειται σε δύο βασικές τους ιδιότητες: την **αραιή συνδεσιμότητα** (sparse connectivity) και την **κοινή χρήση παραμέτρων** (parameter sharing) (Εικόνα 21). Με τον όρο αραιή συνδεσιμότητα εννοούμε πως δεν αλληλοεπιδρούν όλες οι είσοδοι με όλες τις εξόδους σε αντίθεση με τα

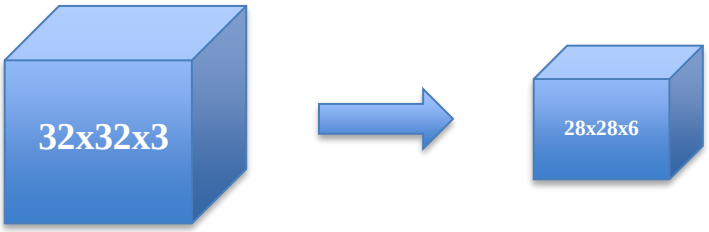


Εικόνα 21Sparse connectivity & Parameter sharing (βιβλιογραφία [13])

παραδοσιακά νευρωνικά δίκτυα όπου λόγω της φύσης του πολλαπλασιασμού πινάκων κάθε είσοδος αλληλοεπιδρά με κάθε έξοδο. Με άλλα λόγια η έξοδος κάθε επιπέδου εξαρτάται από ένα μικρό αριθμό εισόδων. Η ιδιότητα αυτή πηγάζει από το γεγονός ότι τα φίλτρα που χρησιμοποιούμε είναι κατά πολύ μικρότερα από τα δεδομένα εισόδου. Αυτό έχει ως αποτέλεσμα να έχουμε πολύ λιγότερες παραμέτρους που σημαίνει πως οι απαιτήσεις του μοντέλου όσον αφορά τη μνήμη θα είναι αισθητά μειωμένες όπως επίσης ο υπολογισμός της εξόδου θα γίνεται ταχύτερα. Εν συνεχεία κοινή χρήση παραμέτρων σημαίνει πως οι παράμετροι

που μαθαίνονται για ένα σημείο της εικόνας θα χρησιμοποιηθούν και στην υπόλοιπη πριν ενημερωθούν τα βάρη. Για παράδειγμα όταν το δίκτυο μάθει να ανιχνεύει ακμές σε ένα σημείο της εικόνας θα μπορεί να το κάνει και στην υπόλοιπη εικόνα. Αυτό μειώνει ακόμα περισσότερο της παραμέτρους του δικτύου δίχως να βελτιώνει δυστυχώς περαιτέρω την ταχύτητα του.

Για να γίνει πιο κατανοητό το πόσο μειώνονται οι παράμετροι θα δώσουμε ένα αριθμητικό παράδειγμα. Έστω ότι έχουμε μία εικόνα $\mathbf{32 \times 32 \times 3}$ στην οποία εφαρμόζονται 6 φίλτρα $\boldsymbol{f = 5 \times 5}$. Η εικόνα που θα προκύψει θα έχει διαστάσεις $\mathbf{28 \times 28 \times 6}$



Για να πετύχουμε το παραπάνω με πλήρως συνδεμένα δίκτυα θα πρέπει να συνδέσουμε $\mathbf{32 * 32 * 3 = 3072}$ νευρώνες με τους $\mathbf{28 * 28 * 6 = 4704}$ νευρώνες του επόμενου κρυφού επιπέδου. Επομένως θα έχουμε συνολικά $\mathbf{3072 * 4704 \approx 14}$ *εκ* παραμέτρους. Από την άλλη μεριά ένα συνελικτικό δίκτυο θα χρειαζόταν συνολικά **156** παραμέτρους.

3.2 Επίπεδα ΣΝΔ

Όπως συμβαίνει και στα κλασσικά νευρωνικά δίκτυα έτσι και εδώ, τα ΣΝΔ αποτελούνται από διάφορα δομικά «τουβλάκια» τα οποία ονομάζονται επίπεδα (**layers**). Κάθε «τουβλάκι» εκτελεί μία βασική λειτουργία όπως συνέλιξη αλλά υπάρχουν επίπεδα που εκτελούν αρκετά περίπλοκες διεργασίες τα οποία όμως δε θα μας απασχολήσουν στην παρούσα πτυχιακή. Στις παρακάτω παραγράφους θα γνωρίσουμε τα επίπεδα αυτά καθώς και τη λειτουργία τους.

3.2.1 Επίπεδο συνέλιξης

Σε αυτό το σημείο θα εξερευνήσουμε το πιο σημαντικό επίπεδο στα ΣΝΔ, το επίπεδο της συνέλιξης. Αποτελείται από **M** φίλτρα τα οποία συνελίσονται με τα δεδομένα εισόδου και παράγονται **M** χάρτες χαρακτηριστικών. Για την καλύτερη κατανόηση όσων θα περιγράψουμε παρακάτω θα πρέπει καταρχάς να απαντήσουμε στις ερωτήσεις:

- Τι είναι φίλτρο;
- Τι είναι συνέλιξη;

Στα ΣΝΔ δίκτυα αντί για πολλαπλασιασμό πινάκων συναντάμε μία πράξη που ονομάζεται συνέλιξη (convolution) η οποία εφαρμόζεται μεταξύ των δεδομένων εισόδου και του πίνακα βαρών και περιγράφεται μαθηματικά από τον τύπο:

$$(K * I)_{i,j} = \sum_{m} \sum_{n} I_{i-m,j-n} K_{m,n}$$

Με I να συμβολίζει μία εικόνα και K να συμβολίζει το φίλτρο
Επομένως η ενεργοποίηση του κάθε νευρώνα στα ΣΝΔ θα δίνεται από:

$$a = \sigma\left(\sum_{i=0} x_{i^*} w_i + b\right)$$

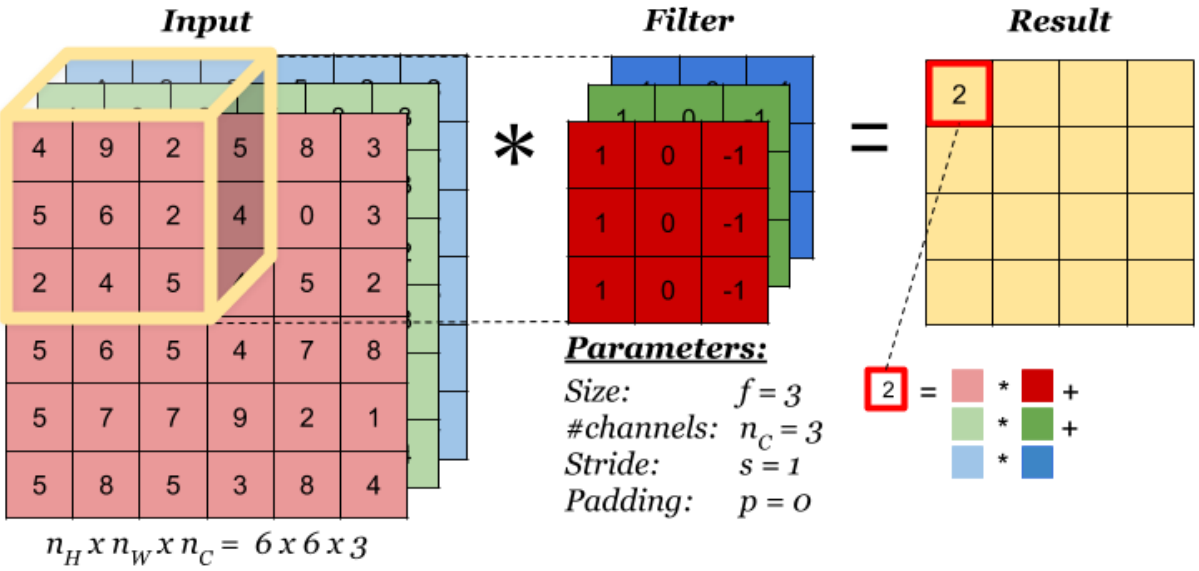
Φίλτρο ή, όπως το συναντάμε στην αγγλική βιβλιογραφία, **kernel** στην πιο απλή του μορφή είναι ένας δισδιάστατος πίνακας με διακριτούς αριθμούς που ονομάζουμε βάρη. Τέτοια φίλτρα συναντάμε συχνά στην επεξεργασία εικόνας με τη διαφορά όμως ότι στην προκειμένη περίπτωση τα βάρη μαθαίνονται από το δίκτυο. Τα βάρη αρχικοποιούνται με τυχαίους αριθμούς οι οποίοι όμως αλλάζουν κατά τη διάρκεια της εκπαίδευσης καθώς το δίκτυο προσπαθεί να πετύχει την επιθυμητή έξοδο. Θα μπορούσε να πει κανείς πως τα βάρη αποτελούν την προηγούμενη γνώση του δικτύου.

-1	**0**	**1**
-2	**0**	**2**
-1	**0**	**1**

Εικόνα 22 Παράδειγμα φίλτρου (Sobel Operator)

Σε αυτό το σημείο θα πρέπει να τονιστεί πως τα φίλτρα που θα συναντήσουμε σε αυτή την πτυχιακή είναι τρισδιάστατα ($f \times f \times n_c$), δηλαδή υπάρχει και μία τρίτη διάσταση που υποδηλώνει το πόσα κανάλια έχει το φίλτρο. Η διάσταση αυτή αναφέρεται συχνά ως *βάθος* (**depth**).

Έστω λοιπόν ότι έχουμε 64 φίλτρα διαστάσεων $3 \times 3 \times 3$ ($f \times f \times n_c$) και μία εικόνα $6 \times 6 \times 3$ ($h \times w \times$ d**).** Κατά τη συνέλιξη το φίλτρο σαρώνει τα δεδομένα εισόδου κατά μήκος και κατά πλάτος και πραγματοποιεί πράξεις εσωτερικού γινομένου μεταξύ του φίλτρου και της εξεταζόμενης περιοχής και το αποτέλεσμα τοποθετείται στο χάρτη χαρακτηριστικών. Η διαδικασία συνεχίζεται έως ότου συμπληρωθεί ο χάρτης χαρακτηριστικών ο οποίος θα έχει διαστάσεις $h' \times w' \times 64$. Προκειμένου να υπολογίσουμε τις ακριβείς διαστάσεις της εξόδου θα πρέπει να μιλήσουμε για δύο ακόμα παραμέτρους που καλείται ο σχεδιαστής του δικτύου να ρυθμίσει.



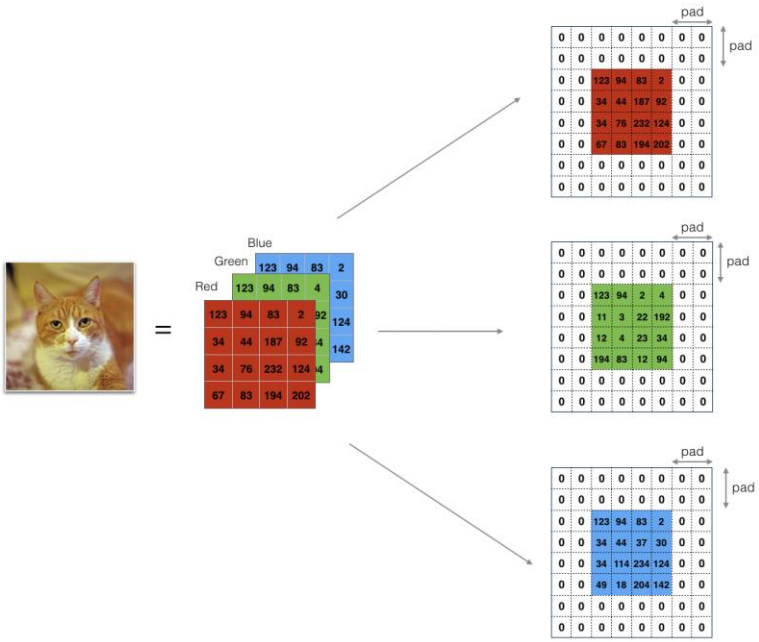
Εικόνα 23 Συνέλιξη (βιβλιογραφία [14])

Οι παράμετροι ή σωστότερα υπερπαράμετροι είναι οι ακόλουθες:

- Βηματισμός (stride): Ένα φίλτρο καθώς σαρώνει τα δεδομένα κάνει βήματα είτε στον οριζόντιο είτε στον κατακόρυφο άξονα, αυτό το βήμα ονομάζεται **stride** και υποδηλώνει πόσο πυκνή θα είναι η δειγματοληψία της εισόδου. Όταν η παράμετρος αυτή παίρνει μικρές τιμές δίνει καλύτερες αναπαραστάσεις της εισόδου στην έξοδο οι οποίες έχουν μεγάλο μέγεθος. Από την άλλη όσο μεγαλώνει το **stride** οι αναπαραστάσεις μικραίνουν σε μέγεθος αλλά χάνεται πληροφορία με αποτέλεσμα να είναι πιο τραχιές. Θεωρώντας ότι έχουμε ένα φίλτρο διαστάσεων $f \times f$ που συνελίσσεται με μία εικόνα $h \times w$ και με $stride = 1$ οι διαστάσεις του πίνακα χαρακτηριστικών υπολογίζεται ως εξής:

$$h' = \left[\frac{h - f + s}{s}\right] \qquad w' = \left[\frac{w - f + s}{s}\right]$$

- Γέμισμα (padding): Κατά τη διάρκεια της συνέλιξης συμβαίνει το εξής. Οι διαστάσεις της εξόδου είναι μικρότερες από αυτές της εισόδου (πράγμα επιθυμητό σε κάποιες εφαρμογές και ανεπιθύμητο σε άλλες) κάτι που εν δυνάμει μας περιορίζει όσον αφορά το πόσο βαθύ μπορεί να είναι το δίκτυο μας. Σε εφαρμογές που η διατήρηση του μεγέθους της εικόνας είναι καίριας σημασίας χρησιμοποιούμε μία τεχνική που ονομάζεται zero-padding. Η ιδέα πίσω από αυτή την τεχνική είναι αρκετά απλή. Αυξάνουμε τις διαστάσεις της εισόδου με μηδενικά (Εικόνα 24) με σκοπό η έξοδος να έχει τις επιθυμητές διαστάσεις.



Εικόνα 24 Παράδειγμα zero-padding (βιβλιογραφία [14])

Συμβολίζοντας την παράμετρο αυτή με το γράμμα p μπορούμε να τροποποιήσουμε τις παραπάνω εξισώσεις ως εξής:

$$\boldsymbol{h' = \left[\frac{h - f + s + 2p}{s}\right]} \qquad \boldsymbol{w' = \left[\frac{w - f + s + 2p}{s}\right]}$$

Τελειώνοντας το παράδειγμα μας οι διαστάσεις του πίνακα χαρακτηριστικών βάσει των παραπάνω τύπων θα είναι $4 \times 4 \times 64$

Προσοχή! Στη βιβλιογραφία της επεξεργασίας εικόνων η διαδικασία που περιγράψαμε παραπάνω αναφέρεται ως cross-correlation και όχι ως συνέλιξη (convolution). Παρόλο που υπάρχει μία μικρή διαφορά στις δύο αυτές πράξεις όσον αφορά τη μηχανική μάθηση και τα νευρωνικά είναι πανομοιότυπες.

Συνοψίζοντας, το επίπεδο συνέλιξης:

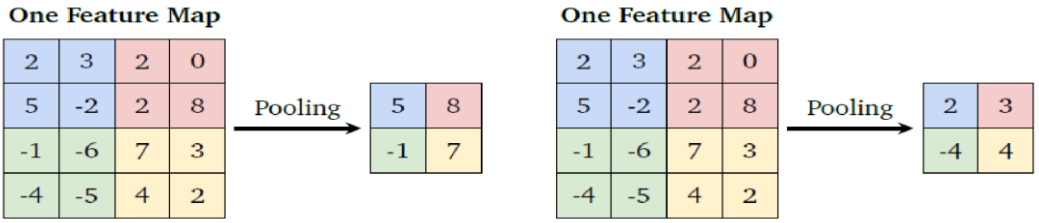
- δέχεται ένα τόμο (volume) διαστάσεων $h_1 \times w_1 \times d_1$
- Χρειάζεται να προσδιοριστούν οι παρακάτω υπερπαράμετροι
  - Αριθμός των φίλτρων $\boldsymbol{n_c}$
  - Μέγεθος των φίλτρων $\boldsymbol{f}$
  - Βηματισμός $\boldsymbol{s}$
  - Γέμισμα $\boldsymbol{p}$
- Παράγει έναν χάρτη χαρακτηριστικών διαστάσεων $h' \times w' \times d'$
  - Όπου $\boldsymbol{h' = \left[\frac{h-f+s+2p}{s}\right]}$ και $\boldsymbol{w' = \left[\frac{w-f+s+2p}{s}\right]}$ και $\boldsymbol{d' = n_c}$

3.2.2 Επίπεδο χωρικής υποδειγματοληψίας

Ένα πρόβλημα που παρουσιάζεται στο συνελικτικό επίπεδο ενός δικτύου είναι πως οι εξαγόμενοι χάρτες παρουσιάζουν μεγάλη ευαισθησία στους μετασχηματισμούς των δεδομένων εισόδου. Αυτό συμβαίνει διότι καταγράφεται η ακριβής θέση των χαρακτηριστικών με αποτέλεσμα έστω και μία μικρή αλλαγή στα δεδομένα εισόδου να συντελεί στην παραγωγή ενός τελείως διαφορετικού χάρτη χαρακτηριστικών. Το πρόβλημα αυτό αντιμετωπίζεται μέσω της υποδειγματοληψίας και ενώ πρακτικά μπορούμε να πετύχουμε υποδειγματοληψία επιλέγοντας τον κατάλληλο βηματισμό (*stride*) στο επίπεδο συνέλιξης, μία αποδοτικότερη μέθοδος είναι η χωρική υποδειγματοληψία.

Βασικό πλεονέκτημα της μεθόδου αυτής είναι η ανοχή στους μετασχηματισμούς στα δεδομένα εισόδου. Αυτή η ιδιότητα είναι πάρα πολύ σημαντική ειδικά σε περιπτώσεις που μας ενδιαφέρει περισσότερο άμα υπάρχει ένα χαρακτηριστικό παρά η ακριβής θέση του. Επιπλέον με τρόπο που θα περιγράψουμε παρακάτω μειώνονται αισθητά χωρικές διαστάσεις των δεδομένων χωρίς όμως να χάνεται σημαντική πληροφορία. Επομένως δεν βλάπτεται η ακρίβεια του δικτύου και ταυτόχρονα μειώνονται σημαντικά οι παράμετροι που πρέπει να υπολογιστούν. Αυτό με τη σειρά του αυξάνει την ταχύτητα εκπαίδευσης του δικτύου.

Τα επικρατέστερα είδη χωρικής υποδειγματοληψίας είναι η συγκέντρωση τοπικού μεγίστου (*max pooling*) και η συγκέντρωση τοπικού μέσου όρου (*average pooling*). Στην πρώτη περίπτωση και σε πλήρη αντιστοιχία με όσα είδαμε στο επίπεδο συνέλιξης έχουμε ένα φίλτρο διαστάσεων $\boldsymbol{f} \times \boldsymbol{f}$ που σαρώνει την είσοδο κατά μήκος και κατά πλάτος με βήμα s και εξάγει τη μέγιστη τιμή από κάθε εξεταζόμενη περιοχή. Στη δεύτερη περίπτωση ισχύουν ότι και παραπάνω με τη μόνη διαφορά πως τώρα υπολογίζεται ο μέσος όρος της εξεταζόμενης περιοχής και τοποθετείται στον παραγόμενο χάρτη. Το μέγεθος του φίλτρου, το βήμα καθώς και το άμα θα υπάρχει ή όχι γέμισα καθορίζονται από το σχεδιαστή του δικτύου ανάλογα με τις ανάγκες του προβλήματος.



Εικόνα 25 a) Max pooling b)Average Pooling (βιβλιογραφία [13])

Σε πλήρη αντιστοιχία με το επίπεδο συνέλιξης, μπορούμε να συνοψίσουμε το επίπεδο χωρικής υποδειγματοληψίας ως εξής:

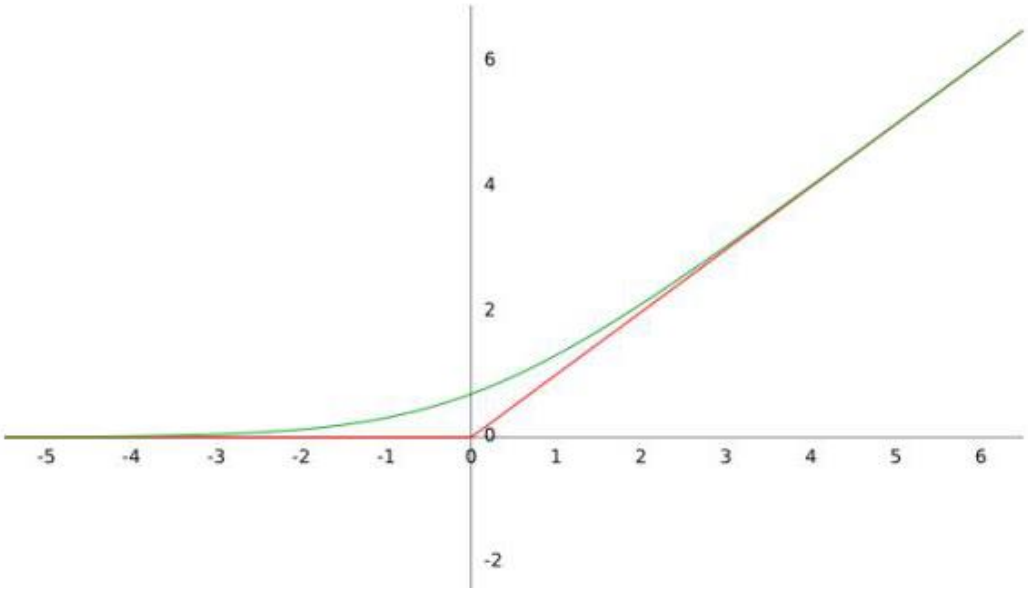
- Δέχεται έναν τόμο (volume) διαστάσεων $\boldsymbol{h_1} \times \boldsymbol{w_1} \times \boldsymbol{d_1}$
- Προσδιορίζοντας τις παρακάτω υπερπαραμέτρους:
  - Μέγεθος του φίλτρου $\boldsymbol{f}$
  - Βηματισμός $\boldsymbol{s}$
- Παράγει ένα χάρτη διαστάσεων $\boldsymbol{h'} \times \boldsymbol{w'} \times \boldsymbol{d'}$

3.2.3 Μη γραμμικότητα

Σε πλήρη αντιστοιχία με τα κλασσικά νευρωνικά δίκτυα είναι αναγκαίο και εδώ να απαλείψουμε τη γραμμικότητα του δικτύου. Εάν δε γίνεται αυτό, το δίκτυο όπως μπορεί πολύ εύκολα να συμπεράνει κανείς, θα αδυνατεί να ανταπεξέλθει σε περίπλοκα δεδομένα και ιδίως σε δεδομένα με χαοτική συμπεριφορά. Έτσι λοιπόν χρησιμοποιούμε μη γραμμικές συναρτήσεις ενεργοποίησης

Οι πιο διαδεδομένες συναρτήσεις ενεργοποίησης όσον αφορά τα ΣΝΔ είναι οι: ReLU, PreLU όπως και η Softplus της οποίας η απόκριση είναι πολύ κοντά με αυτή της ReLU. Οι συναρτήσεις αυτές είναι εύκολες στον υπολογισμό (computational efficient) αλλά τα προβλήματα που αναφέρθηκαν σε προηγούμενη παράγραφο συνεχίζουν να υφίστανται.

Στην παρούσα πτυχιακή η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η ReLU καθώς για το συγκεκριμένο πρόβλημα (ανίχνευση λωρίδων) παρουσίασε τα καλύτερα αποτελέσματα.

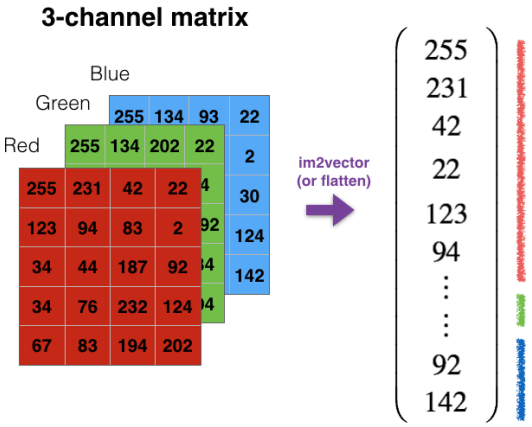


Εικόνα 26 Συσχέτιση ReLU με Softplus (βιβλιογραφία [5])

3.2.4 Πλήρως Συνδεδεμένο Επίπεδο (Fully Connected Layer)

Στην πλειοψηφία τους τα συνελικτικά νευρωνικά δίκτυα πέρα από τα επίπεδα που περιγράψαμε παραπάνω διαθέτουν ένα ή παραπάνω πλήρως συνδεδεμένα επίπεδα. Τα επίπεδα αυτά βρίσκονται πάντα στο τέλος του δικτύου και αποτελούν την έξοδο του. Ωστόσο υπάρχουν συνελικτικά δίκτυα που δε διαθέτουν τέτοιου είδους επίπεδα στα οποία θα αναφερθούμε παρακάτω.

Για να μπορέσουν να λειτουργήσουν τα επίπεδα αυτά θα πρέπει να διαμορφωθούν κατάλληλα οι χάρτες που εξάχθηκαν από το τελευταίο συνελικτικό ή υποδειγματοληπτικό επίπεδο. Η δισδιάστατη φύση τους καθιστά ανίκανους να εισαχθούν σε τέτοιου είδους επίπεδα. Αναγκαίο είναι λοιπόν οι δισδιάστατοι πίνακες να μετατραπούν σε διανύσματα (Εικόνα 27). Κάθε τιμή στο διάνυσμα αυτό θα αποτελεί είσοδο σε ένα νευρώνα.



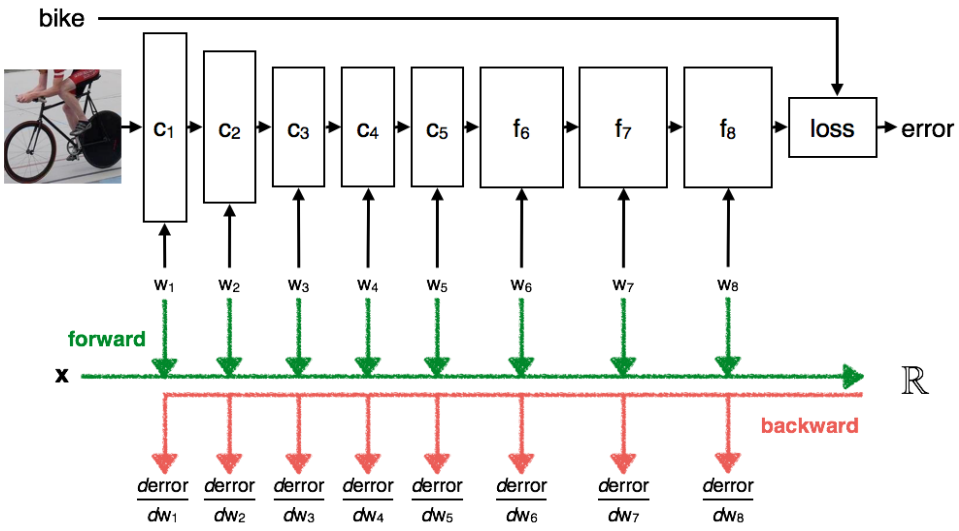
Εικόνα 27 Μετατροπή πίνακα σε διάνυσμα (βιβλιογραφία [14])

Ανάλογα με το πρόβλημα μπορεί να υπάρχουν ένα ή περισσότερα επίπεδα τα οποία δε διαφέρουν καθόλου στη λειτουργία από τα νευρωνικά δίκτυα που γνωρίσαμε στο προηγούμενο κεφάλαιο. Η προσαρμογή των βαρών γίνεται με τον αλγόριθμο οπισθοδιάδοσης σφάλματος που θα μελετήσουμε παρακάτω.

3.3 Εκπαίδευση του δικτύου

Με τον όρο εκπαίδευση του δικτύου αναφερόμαστε στην προσπάθεια του δικτύου να πετύχει την επιθυμητή έξοδο προσαρμόζοντας κατάλληλα τις παραμέτρους του. Ωστόσο δεν έχει γίνει αναφορά για το πως επιτυγχάνεται αυτό τόσο στα κλασσικά όσο και στα συνελικτικά νευρωνικά δίκτυα. Σε αυτό το κεφάλαιο θα προσπαθήσουμε να σκιαγραφήσουμε τον τρόπο που εκπαιδεύεται ένα συνελικτικό νευρωνικό δίκτυο. Αποκτώντας στην πορεία μία ιδέα πρώτα θεωρητικό και ύστερα σε μαθηματικό επίπεδο για το τι συμβαίνει κατά τη διάρκεια αυτής της διαδικασίας.

Η εκπαίδευση ενός δικτύου διακρίνεται σε δύο φάσεις: την **εμπροσθοδιάδοση** (forward pass) και την **οπισθοδιάδοση** (backward pass). Κατά τη διάρκεια της εμπροσθοδιάδοσης η πληροφορία κινείται από το επίπεδο εισόδου προς το επίπεδο εξόδου. Κατά τη διάρκεια αυτού του περάσματος αρχικοποιούνται τυχαία οι παράμετροι του δικτύου και υπολογίζονται οι ενεργοποιήσεις $\boldsymbol{a^l}$ των κρυφών επιπέδων και εν κατακλείδι η έξοδος. Στη συνέχεια υπολογίζεται το σφάλμα το οποίο αντικατοπτρίζει πόσο κοντά είναι η πραγματική είσοδος με την επιθυμητή.



Εικόνα 28 Γραφική απεικόνιση της εμπροσθοδιάδοσης και οπισθοδιάδοσης (βιβλιογραφία [15])

Για να γίνουμε πιο συγκεκριμένοι, όταν αναφερόμαστε στην εκπαίδευση ενός δικτύου εννοούμε την προσπάθεια του να πετύχει την επιθυμητή έξοδο τροποποιώντας τις παραμέτρους (W, b). Γι' αυτό το σκοπό θα πρέπει να οριστεί ένα μέγεθος μέσω του οποίου θα μπορέσουμε να αξιολογήσουμε την επίδοση του δικτύου. Ένα τέτοιο μέγεθος είναι η συνάρτηση κόστους την οποία προσπαθεί να ελαχιστοποιήσει το δίκτυο προσαρμόζοντας κατάλληλα τις παραμέτρους του. Προκειμένου να το πετύχει αυτό πρέπει να υπολογίσει την κλίση (παράγωγο) της συνάρτησης κόστους ως προς όλα τα βάρη και biases. Κάθε παράγωγος υποδεικνύει πόσο ευαίσθητη είναι η συνάρτηση κόστους στην αλλαγή του κάθε βάρους και bias. Με άλλα λόγια μας δείχνει πόσο πολύ (ή λίγο) θα μεταβληθεί η συνάρτηση κόστους αν μεταβάλλουμε την εν λόγω παράμετρο.

$$\nabla J(\boldsymbol{W}, \boldsymbol{b}) = \begin{bmatrix} 3.20 \\ \vdots \\ 0.10 \end{bmatrix}$$

Για παράδειγμα έστω ότι έχουμε υπολογίσει την κλίση της συνάρτησης κόστους ως προς όλα τα βάρη & biases για ένα δείγμα και έχουμε πάρει το παραπάνω διάνυσμα. Αυτό που μπορούμε να συμπεράνουμε είναι πως η επίδραση του πρώτου στοιχείου είναι 32 φορές μεγαλύτερη από αυτή του τελευταίου. Συνεπώς αλλάζοντας το συγκεκριμένο βάρος θα δούμε μεγαλύτερη επίπτωση στη συνάρτηση κόστους. Έτσι λοιπόν αφού το δίκτυο ολοκληρώσει το forward pass και υπολογιστεί η έξοδος υπολογίζεται και το σφάλμα

3.3.1 Αλγόριθμος οπισθοδιάδοσης σφάλματος – back-propagation algorithm

Τώρα που είδαμε με απλά λόγια τι συμβαίνει κατά τη διάρκεια της οπισθοδιάδοσης ήρθε η ώρα να «λερώσουμε» τα χέρια μας με λίγα μαθηματικά. Η σημειογραφία που θα χρησιμοποιηθεί παρακάτω είναι ελαφρώς διαφορετική με αυτή που έχει χρησιμοποιηθεί μέχρι στιγμής.

Για την πραγματοποίηση του αλγορίθμου οπισθοδιάδοσης θα πρέπει να γίνουν τα ακόλουθα βήματα σύμφωνα με τη βιβλιογραφία [16]:

1. Υπολογίζουμε κατά τη διάρκεια του forward pass τα: $\boldsymbol{z^l_{x,y} = w^l * \sigma\left(z^{l-1}_{x,y}\right) + b^l_{x,y}}$ και $\boldsymbol{a^l_{x,y} = \sigma\left(z^l_{x,y}\right)}$
2. Υπολογίζουμε το διάνυσμα σφάλματος $\boldsymbol{\delta^L = \nabla_a J \odot \sigma(z^L)}$
3. Στέλνουμε προς τα «πίσω» το σφάλμα και για κάθε επίπεδο l = L-1, L-2,…,2 υπολογίζουμε $\boldsymbol{\delta^l_{x,y} = \delta^{l+1} * ROT180\left(w^{l+1}_{x,y}\right)\sigma\left(z^l_{x,y}\right)}$
4. Τέλος η κλίση της συνάρτησης κόστους υπολογίζεται: $\boldsymbol{\frac{\partial J}{\partial W^l_{a,b}} = \delta^l_{a,b} * \sigma\left(ROT180\left(z^{l-1}_{a,b}\right)\right)}$

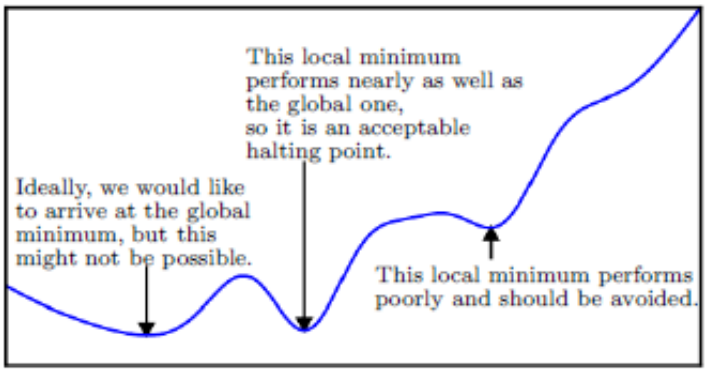
Γνωρίζοντας τις κλίσεις $\frac{\partial J}{\partial W^l_{a,b}}$ μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο Gradient Descent για την ενημέρωση των παραμέτρων.

$$\boldsymbol{\Theta \leftarrow \Theta - a \cdot \frac{\partial J(\Theta)}{\partial \Theta}}$$

3.3.2 Αλγόριθμος Gradient Descent

Στην πλειοψηφία τους τα νευρωνικά δίκτυα περιλαμβάνουν κάποιου είδους βελτιστοποίηση. Ο όρος βελτιστοποίηση αναφέρεται στην ελαχιστοποίηση ή (μερικές φορές) στη μεγιστοποίηση κάποιας συνάρτησης $g(x)$ μεταβάλλοντας τη μεταβλητή x. Στην πράξη βέβαια οι περισσότεροι αλγόριθμοι βελτιστοποίησης προσπαθούν να ελαχιστοποιήσουν τη συνάρτηση $g(x)$.

Η συνάρτηση που θέλουμε να ελαχιστοποιήσουμε στην προκειμένη περίπτωση ονομάζεται συνάρτηση κόστους (cost function) αλλά στην ξένη βιβλιογραφία μπορεί να συναντήσει κανείς διάφορα ονόματα

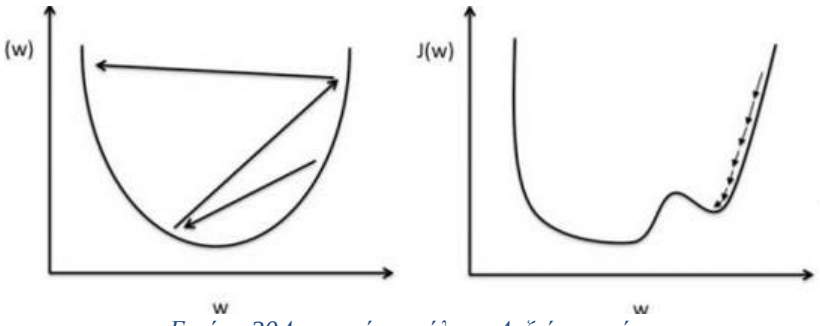


Εικόνα 29 Αναπαράσταση καθολικών και τοπικών ελαχίστων (βιβλιογραφία [17])

για τις προς ελαχιστοποίηση συναρτήσεις όπως **objective function**, **criterion**. Πως επιτυγχάνεται λοιπόν η ελαχιστοποίηση της συνάρτησης κόστους; Πριν μπορέσουμε να απαντήσουμε σε αυτή την ερώτηση θα πρέπει να θυμηθούμε κάποια πράγματα από τα μαθηματικά.

Έστω λοιπόν ότι έχουμε μία συνάρτηση $\boldsymbol{y = g(x)}$ $\boldsymbol{με}$ $\boldsymbol{y, x \in \mathbb{R}}$. Από τα μαθηματικά γνωρίζουμε πως η παράγωγος $\boldsymbol{g'(x)}$ της συνάρτησης μας δίνει την κλίση της στο σημείο x ή με άλλα λόγια μας υποδεικνύει πόσο πρέπει να μεταβάλλουμε την είσοδο για να δούμε αντίστοιχη μεταβολή στην έξοδο.

Επομένως η παράγωγος μίας συνάρτησης μπορεί εν γένει να μας προσφέρει χρήσιμες πληροφορίες όσον αφορά την ελαχιστοποίηση της διότι μας υποδεικνύει πως πρέπει να μεταβάλουμε το x(είσοδος) για να δούμε μία αλλαγή, προς τη σωστή κατεύθυνση, στο y(έξοδος). Γνωρίζοντας τα παραπάνω μπορούμε να κάνουμε μικρά βήματα προς την κατεύθυνση που υποδεικνύει η αρνητική παράγωγος της συνάρτησης με σκοπό να βρούμε κάποιο τοπικό ή καθολικό ελάχιστο.



Εικόνα 30Αριστερά: μεγάλο α Δεξιά: μικρό α (βιβλιογραφία [17])

Τοπικό ελάχιστο (*local minimum*) είναι ένα σημείο στο οποίο η $\boldsymbol{g(x)}$ παίρνει τη μικρότερη τιμή σε σχέση με τα γειτονικά σημεία ενώ καθολικό μέγιστο (*global minimum*) είναι ένα σημείο στο οποίο η συνάρτηση $\boldsymbol{g(x)}$ παίρνει τη μικρότερη δυνατή τιμή. Υπάρχουν επίσης μερικά σημεία για τα οποία ισχύει $\boldsymbol{g'(x) = 0}$ με αποτέλεσμα να μην έχουμε καμία πληροφορία όσον αφορά σε ποια κατεύθυνση πρέπει να κινηθούμε. Τα σημεία αυτά ονομάζονται **κρίσιμα σημεία**.

Στην πράξη οι περισσότερες, αν όχι όλες, οι συναρτήσεις που θα κληθούμε να ελαχιστοποιήσουμε θα έχουν πάνω από μία εισόδους $\boldsymbol{g: \mathbb{R}^n \rightarrow \mathbb{R}}$. Προκειμένου να ελαχιστοποιηθεί μία τέτοια συνάρτηση θα χρησιμοποιήσουμε τις μερικές παραγώγους της συνάρτησης. Υπολογίζοντας τη μερική παράγωγο μιας συνάρτησης ως προς κάθε μεταβλητή βρίσκουμε πόσο αλλάζει η g όταν μεταβάλλεται μόνο μία μεταβλητή ενώ οι υπόλοιπες παραμένουν σταθερές. Υπολογίζοντας λοιπόν όλες τις μερικές παράγωγους (βλέπε προηγούμενο κεφάλαιο) μπορούμε να βρούμε την κατεύθυνση μέσω της οποίας η συνάρτηση ελαχιστοποιείται πιο γρήγορα. Με άλλα λόγια βρίσκουμε το πιο απότομο μονοπάτι (*steepest path)* Μαθηματικά εκφράζεται ως εξής:

$$\boldsymbol{w_{ij}^{[k]} = w_{ij}^{[k]} - a\frac{\partial J(w,b)}{\partial w_{i,j}^{[k]}}}$$

$$\boldsymbol{b_i^{[k]} = b_i^{[k]} - \alpha\frac{\partial J(w,b)}{\partial b_i^{[k]}}}$$

Οι παραπάνω σχέσεις υπολογίζονται συνεχώς κατά τη διάρκεια της εκπαίδευσης μέχρι να βρεθεί κάποιο τοπικό ή στην καλύτερη περίπτωση καθολικό ελάχιστο. Να τονιστεί ότι

κανείς δεν μας εγγυάται πως θα βρεθεί έστω και ένα τοπικό ελάχιστο (σε ένα εύλογο χρονικό διάστημα). Παρόλα αυτά πάντα επιτυγχάνεται μία αρκετά μικρή τιμή της συνάρτησης κόστους. [17]

Στις παραπάνω εξισώσεις με α συμβολίζουμε το ρυθμό μάθησης (*learning rate*) ο οποίος καθορίζει το πόσο μικρά ή μεγάλα βήματα θα κάνει ο αλγόριθμος. Η σωστή επιλογή αυτής της παραμέτρου επηρεάζει σημαντικά την ταχύτητα της εκπαίδευσης (Εικόνα 30) καθώς εάν $\alpha = πολύ\ μικρό$ → το δίκτυο μαθαίνει πολύ αργά και εάν $\alpha = πολύ\ μεγάλο$ → ο αλγόριθμος θα κάνει τεράστια βήματα και ενδεχομένως θα προσπερνάει τα ελάχιστα.

3.3.3 Αλγόριθμος Stochastic Gradient Descent

Ο αλγόριθμος **Gradient Descent** που περιγράψαμε προηγουμένως εφαρμόζεται σε όλο το σετ δεδομένων εκπαίδευσης και αποτελούσε τον πιο διαδεδομένο αλγόριθμο μέχρι κάποια στιγμή. Καθώς όμως, για την επίλυση περίπλοκων προβλημάτων, ο όγκος των δεδομένων εκπαίδευσης ανεβαίνει εκθετικά ο αλγόριθμος **Gradient Descent** έγινε σιγά σιγά απαγορευτικός λόγω του χρόνου που απαιτείται για την εκπαίδευση του δικτύου.

Λύση σε αυτό το πρόβλημα δίνει μία παραλλαγή του αλγόριθμου **Gradient Descent** που ονομάζεται **Stochastic Gradient Descent** που σε αντίθεση με τον προκάτοχο του εφαρμόζεται σε μικρά πακέτα (mini batches) δεδομένων εκπαίδευσης. Σε κάθε βήμα του αλγόριθμου παίρνουμε ένα μικρό πακέτο $\mathbb{B} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, .., \boldsymbol{x}^{(\acute{N})}\}$ από το σετ εκπαίδευσης με $\acute{N}$ να είναι συνήθως ένας μικρός αριθμός παραδειγμάτων από 1 μέχρι μερικές εκατοντάδες παραδείγματα και εφαρμόζουμε τον αλγόριθμο. Επομένως μπορούμε να υπολογίσουμε προσεγγιστικά την κλίση της συνάρτησης κόστους χρησιμοποιώντας κάθε φορά ένα μικρό κομμάτι από το σετ εκπαίδευσης κάτι που ελαττώνει πολύ το χρόνο εκπαίδευσης του δικτύου.

3.3.4 Αλγόριθμος Adam (Adaptive Moment Estimation)

Adam είναι ένας αλγόριθμος βελτιστοποίησης που παρουσιάστηκε από τους **Diederik P. Kingma** και **Jimmy Lei Ba** στο ICRL το 2015 [18]. Ειδικά σχεδιασμένος για τη βελτιστοποίηση νευρωνικών δικτύων και ειδικά όσων διαθέτουν πολλά κρυφά επίπεδα υποσχόταν υψηλότερες επιδόσεις σε σύγκριση με τους υπόλοιπους αλγόριθμους.

Το όνομα Adaptive Moment Estimation (Adam) πηγάζει από γεγονός πως υπολογίζει την πρώτη και δεύτερη στιγμή (first and second moment) της κλίσης ώστε να προσαρμόσει κατάλληλα το ρυθμό μάθησης. Με τον όρο στιγμή, όσον αφορά τις τυχαίες μεταβλητές, αναφερόμαστε στην αναμενόμενη τιμή αυτής της μεταβλητής υψωμένη στην δύναμη n τη στιγμή N

$$\boldsymbol{m_n = E[X^n]}$$

Ο αλγόριθμος περιγράφεται λεπτομερώς στη βιβλιογραφία [18] και θα μπορούσε να συμπτυχθεί στα εξής βήματα:
Ο αλγόριθμος ADAM:

1. **Χρειάζεται:** βήμα $\boldsymbol{\epsilon}$ (προτεινόμενη τιμή: 0.001)

2. **Χρειάζεται:** Exponential decay rates ρ_1 *και* $\rho_2 \in [0,1)$ (προτεινόμενες τιμές: 0.0 και 0.999 αντίστοιχα)
3. **Χρειάζεται:** Μικρή σταθερά δ για αριθμητική σταθεροποίηση. (Προτεινόμενη τιμή 10^{-8})
4. **Χρειάζεται:** Αρχικές παραμέτρους $\boldsymbol{\theta}$**,** αρχικοποιούμε τις μεταβλητές $\boldsymbol{s} = \boldsymbol{0}, \boldsymbol{r} = \boldsymbol{0}$. Αρχικοποιούμε το χρονικό βήμα $\boldsymbol{t} = \boldsymbol{0}$
5. Όσο το κριτήριο παύσης δεν ικανοποιείται:
    a. Γίνεται δειγματοληψία ενός μικρού πακέτου δεδομένων (minibatch) $\mathbb{B} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, .., \boldsymbol{x}^{(\acute{N})}\}$ N παραδειγμάτων μαζί με τις αντίστοιχες επισημάνσεις $\boldsymbol{y}^{(i)}$
    b. Υπολογίζεται η κλίση: $\boldsymbol{g} \leftarrow \frac{\boldsymbol{1}}{\boldsymbol{m}} \boldsymbol{\nabla_\theta \Sigma_i L}(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \quad \boldsymbol{t} \leftarrow \boldsymbol{t} + \boldsymbol{1}$
    c. Γίνεται η ενημέρωση της πρώτης εκτιμώμενης στιγμής $\boldsymbol{s} \leftarrow \boldsymbol{\rho_1 s} + (\boldsymbol{1} - \boldsymbol{\rho_1})\boldsymbol{g}$
    d. Γίνεται η ενημέρωση της δεύτερης εκτιμώμενης στιγμής $\boldsymbol{s} \leftarrow \boldsymbol{\rho_2 r} + (\boldsymbol{1} - \boldsymbol{\rho_2})\boldsymbol{g} \odot \boldsymbol{g}$
    e. Γίνεται η διόρθωση: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{\boldsymbol{1} - \boldsymbol{\rho_1^t}}$
    f. Γίνεται η διόρθωση: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{\boldsymbol{1} - \boldsymbol{r_2^t}}$
    g. Υπολογίζεται η ενημέρωση των παραμέτρων $\triangle \boldsymbol{\theta} = -\boldsymbol{\epsilon} \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \boldsymbol{\delta}}$
    h. Ενημερώνονται οι παράμετροι: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \triangle \boldsymbol{\theta}$

3.3.5 Πως επιλέγουμε τον κατάλληλο αλγόριθμο βελτιστοποίησης;

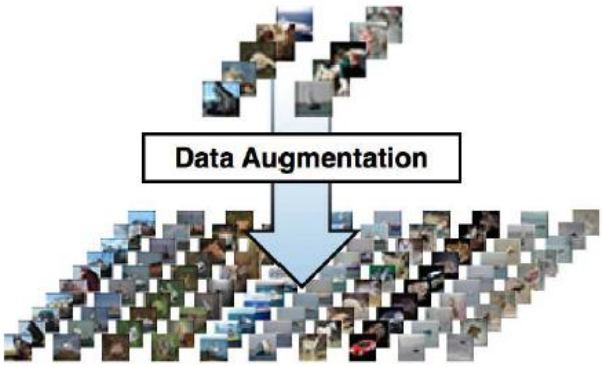
Μέχρι στιγμής έχουμε περιγράψει τρεις αλγόριθμους βελτιστοποίησης και υπάρχουν πολλοί ακόμα στους οποίους δεν έχουμε αναφερθεί (RMSprop, SGD with momentum, AdaGrad κ.α.). Με τέτοια πληθώρα διαθέσιμων αλγόριθμων, υπάρχει κάποιος που να μας παρέχει τα καλύτερα αποτελέσματα ανεξάρτητα με τη φύση του προς επίλυση προβλήματος;

Δυστυχώς, δεν υπάρχει κάποιος αλγόριθμος που να επιδεικνύει ικανοποιητικά αποτελέσματα σε διαφορετικού τύπου προβλήματα [19]. Στη βιβλιογραφία [19] ελέγχθηκε πληθώρα αλγόριθμων τα αποτελέσματα των οποίων έδειξαν πως οι τεχνικές με προσαρμοζόμενο ρυθμό μάθησης έχουν ελαφρώς καλύτερη επίδοση. Ωστόσο αξίζει να σημειωθεί πως δεν υπήρξε ξεκάθαρος νικητής.

Αυτή τη στιγμή οι πιο διαδομένες τεχνικές βελτιστοποίησης είναι οι: SGD, SGD with momentum, RMSprop, RMSprop with momentum, AdaGrad και Adam. Η χρήση των οποίων από ότι φαίνεται εξαρτάται από την οικειότητα που έχει ο εκάστοτε χρήσης με τον αλγόριθμο [17]

3.3.6 Επαύξηση Δεδομένων (Data Augmentation)

Καθώς τα νευρωνικά δίκτυα εξελίσσονται και οι αρχιτεκτονικές τους γίνονται όλο και πιο περίπλοκες, οι ανάγκη τους δεδομένα αυξάνεται. Παρά το γεγονός όμως πως τα σετ εκπαίδευσης έχουν αυξηθεί σε αριθμό τα τελευταία χρόνια είναι ουκ ολίγες οι περιπτώσεις όπου δεν υπάρχουν επαρκή δεδομένα για τη σωστή εκπαίδευση του δικτύου. Ο λόγος είναι πως ένα μεγάλο ποσοστό των μεγάλων dataset δεν είναι διαθέσιμο στο ευρύ κοινό. Με το εκπαιδεύσουμε ένα νευρωνικό και ιδίως ένα συνελικτικό νευρωνικό δίκτυο με μη επαρκή δεδομένα στερούμε τη δυνατότητα του δικτύου να ανταπεξέρχεται σε δεδομένα που δεν έχει ξαναδεί. Αυτό είναι ένα φαινόμενο γνωστό και ως υπερεκπαίδευση (**overfit**) και στις παρακάτω παραγράφους θα περιγράψουμε κάποιους από τους τρόπους αντιμετώπισης του.



Εικόνα 31 Επαύξηση δεδομένων (βιβλιογραφία [20])

Ένας από τους πιο διαδεδομένους τρόπους είναι ο γεωμετρικός μετασχηματισμός των δεδομένων. Οι διαθέσιμοι μετασχηματισμοί περιλαμβάνουν την περιστροφή, περικοπή όπως και την αναστροφή των εικόνων εισόδου (Εικόνα 32). Οι μετασχηματισμοί αυτοί αλλοιώνουν τη γεωμετρία της εικόνας χαρτογραφώντας την τιμή κάθε εικονοστοιχείου σε μία

Εικόνα 32 Τρόποι επαύξησης των δεδομένων (βιβλιογραφία [20])

διαφορετική θέση. Με αυτό τον τρόπο διατηρείται το υποκείμενο σχήμα της κλάσης που αντιπροσωπεύει η εικόνα και μεταβάλλεται μόνο η θέση ή ο προσανατολισμός του αντικειμένου.

Ο ίσως πιο δημοφιλής γεωμετρικός μετασχηματισμός είναι η αναστροφή της εικόνας κατά μήκος του κάθετου άξονα της. Η ευκολία να πραγματοποιηθεί με κώδικα σε συνδυασμό με το πόσο υπολογιστικά αποδοτικός είναι αυτός ο μετασχηματισμός τον καθιστούν μία σίγουρη επιλογή.

Στην περίπτωση της περιστροφής χαρτογραφούνται τα εικονοστοιχεία (x, y) της εικόνας σε (x', y') χρησιμοποιώντας τον παρακάτω μαθηματικό τύπο:

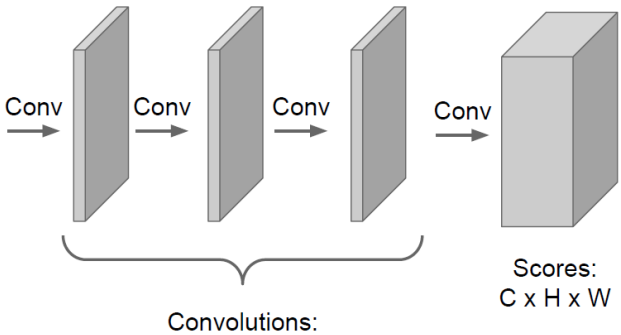
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Δοκιμές έχουν δείξει πως οι τιμές -30^o και $+30^o$ για τη γωνία θ είναι αρκετές για να δημιουργήσουν νέα αναλλοίωτα δείγματα. (**Luke Taylor et al 2017** [20])

Τέλος θα αναφερθούμε στην περικοπή για την οποία δεν υπάρχουν πολλά να πούμε. Είναι η τεχνική μέσω της οποίας ουσιαστικά μειώνουμε την ανάλυση της εικόνας δημιουργώντας νέα δείγματα για το σε δεδομένων μας.

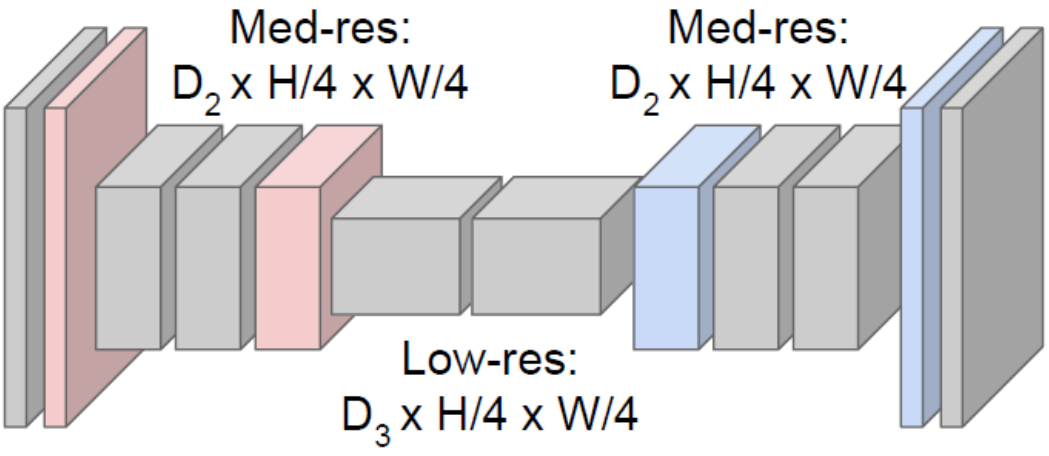
3.4 Πλήρως Συνελικτικά Νευρωνικά Δίκτυα

Σε μία μεγάλη μερίδα συνελικτικών δικτύων πέρα από τα επίπεδα συνέλιξης και χωρικής υποδειγματοληψίας συναντάμε και πλήρως συνδεδεμένα επίπεδα (FC layers) στην έξοδο των εν λόγω δικτύων. Υπάρχει όμως μια ειδική κατηγορία ΣΝΔ από τα οποία απουσιάζουν τα πλήρως συνδεδεμένα επίπεδα. Τα δίκτυα αυτά έχουν αποδειχθεί πολύ αποτελεσματικά σε διάφορους τομείς της μηχανικής όρασης όπως η θεματική κατάτμηση (**semantic segmentation**). Η απουσία των πλήρως συνδεδεμένων επιπέδων τους δίνει ένα μεγάλο πλεονέκτημα όσον αφορά το χρόνο που απαιτείται για την εκπαίδευσης τους. Αυτό συμβαίνει διότι λόγω των ιδιοτήτων που περιγράψαμε παραπάνω (**sparse connectivity** & **parameter sharing**) ο αριθμός των παραμέτρων του δικτύου είναι αισθητά μικρότερος. Επιπλέον το, ίσως, μεγάλο πλεονέκτημα των δικτύων αυτών είναι ότι μπορούν να επεξεργαστούν εικόνες οποιουδήποτε μεγέθους.



Εικόνα 33 Αρχιτεκτονική Πλήρους Συνελικτικού δικτύου (βιβλιογραφία [35])

Μια αρχιτεκτονική ενός τέτοιου δικτύου μπορεί να είναι όπως στην Εικόνα 33. Όπως μπορεί να παρατηρήσει κανείς σε όλα τα επίπεδα η συνέλιξη γίνεται στη φουλ ανάλυση της εικόνας. Αυτό πέρα από μη πρακτικό είναι και απίστευτα υπολογιστικά δαπανηρό. Γι' αυτό το λόγο στην πράξη τα εν λόγω δίκτυα συνήθως έχουν την παρακάτω μορφή.



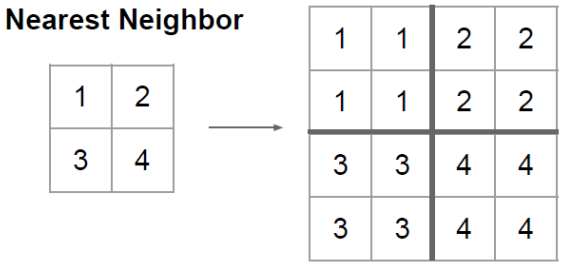
Εικόνα 34 Πλήρως Συνελικτικό Νευρωνικό Δίκτυο (βιβλιογραφία [35])

Όπου τώρα στη θέση των πλήρως συνδεδεμένων επίπεδων έχουμε μία σειρά αναδειγματοληπτικών (Up sampling) επιπέδων που σκοπό έχουν να κάνουν τον χάρτη στην έξοδο να έχει τις ίδιες διαστάσεις με την εικόνα εισόδου.

Μέχρι στιγμής έχουμε δει τρόπους με τους οποίους μπορούμε να κάνουμε υποδειγματοληψία (pooling, strided convolutions κ.α.) αλλά δε γνωρίζουμε πως πραγματοποιείται η αντίστροφη διαδικασία. Στις επόμενες παραγράφους θα σκιαγραφήσουμε μερικούς από τους τρόπους με τους οποίους μπορεί να επιτευχθεί το Up Sampling δηλαδή πως επιτυγχάνει το δίκτυο την αύξηση των διαστάσεων του πίνακα χαρακτηριστικών.

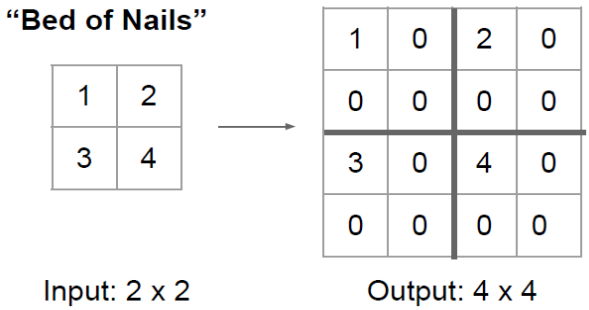
3.4.1 Un-pooling

Ο πιο απλός τρόπος να πραγματοποιήσουμε αναδειγματοληψία (Up Sampling) είναι μέσω της αντίστροφης διαδικασίας του pooling. Υπάρχουν διάφορες εκδοχές αυτής της διαδικασίας με τις επικρατέστερες να είναι οι εξής: nearest neighbor, «Bed of nails» & max unpooling τις οποίες θα γνωρίσουμε παρακάτω. Θυμίζουμε εν τάχει πως η χωρική υποδειγματοληψία επιτυγχάνεται ως εξής: ένα φίλτρο $f \times f$ σαρώνει την είσοδο κατά μήκος και κατά πλάτος με βήμα s και εξάγει την μέγιστη τιμή (στην περίπτωση του max pooling) ή τον μέσο όρο (στην περίπτωση του average pooling) της εξεταζόμενης περιοχής.



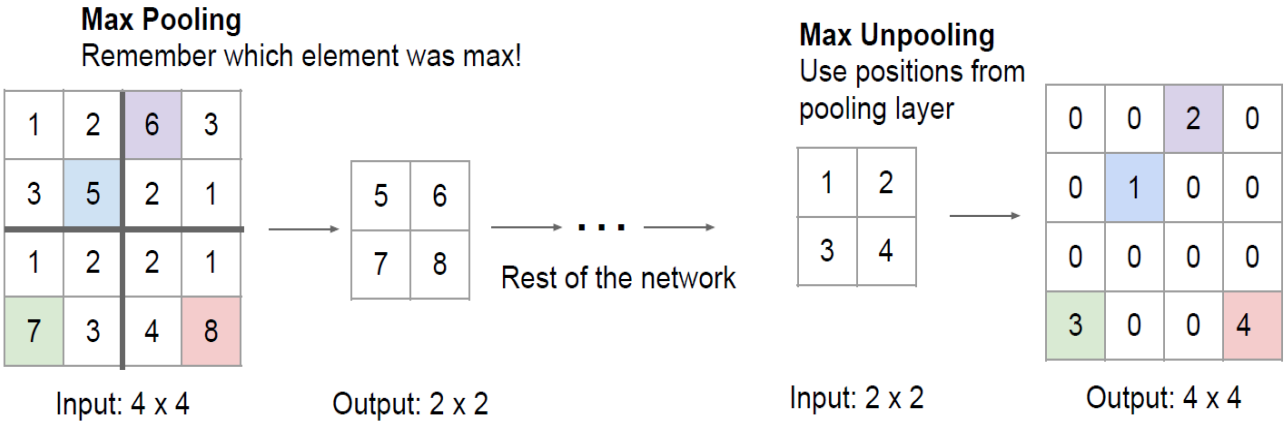
Εικόνα 35 Nearest Neighbor Unpooling (βιβλιογραφία [35])

Στην περίπτωση του nearest neighbor εισάγεται ένας πίνακας χαρακτηριστικών 2×2 από τον οποίο οι εξάγονται οι ενεργοποιήσεις και τοποθετούνται στον παραγόμενο χάρτη όπως φαίνεται στην Εικόνα 35. Στην περίπτωση του "bed of nails" τοποθετούμε κάθε ενεργοποίηση του εισαγόμενου πίνακα στον παραγόμενο χάρτη και μηδενίζουμε όλα τα υπόλοιπα στοιχεία όπως φαίνεται στην Εικόνα 36.



Εικόνα 36 Bed of Nails Unpooling (βιβλιογραφία [35])

Υπάρχει και μία τρίτη και ίσως αποδοτικότερη μέθοδος για την επίτευξη αναδειγματοληψίας στην οποία το δίκτυο θυμάται την ακριβή θέση των μέγιστων ενεργοποιήσεων. Έτσι μπορεί και τοποθετεί τις ενεργοποιήσεις του εισαγόμενου χάρτη στον εξαγόμενο όπως φαίνεται στην Εικόνα 37



Εικόνα 37 Max Unpooling (βιβλιογραφία [35])

Όλες οι προαναφερθείσες μέθοδοι δεν έχουν παραμέτρους (πράγμα απολύτως λογικό) και επομένως το δίκτυο δε μαθαίνει τίποτα από αυτή τη διαδικασία.

3.4.2 Learnable Up sampling : Transposed Convolution

Πέρα από όσα μάθαμε παραπάνω υπάρχει και μία μέθοδος αναδειγματοληψίας (Up sampling) επιτυγχάνεται μέσω μίας παραλλαγής της συνέλιξης που γνωρίσαμε σε προηγούμενη παράγραφο. Στη ξένη βιβλιογραφία μπορεί να τη συναντήσει κανείς με διάφορα ονόματα όπως *fractionally strided convolution, deconvolution, backward strided convolution*. Παρόλο που ο όρος deconvolution συναντάται σε πληθώρα πηγών, αποτελεί λάθος ονομασία καθώς αφήνει να εννοηθεί πως είναι αντίστροφη πράξη της συνέλιξης (κάτι που δεν ισχύει) [21].

Όπως και προηγουμένως έτσι και τώρα στόχος μας είναι να αυξήσουμε τις διαστάσεις του πίνακα χαρακτηριστικών. Ένας εύκολος τρόπος να εξηγήσουμε πως επιτυγχάνεται αυτό είναι αναπαριστώντας τη συνέλιξη με πολλαπλασιασμό πινάκων ως εξής:

Έστω ότι έχουμε δύο πίνακες $\boldsymbol{A}$ (φίλτρο) και $\boldsymbol{B}$ (είσοδος) όπου για τώρα θεωρούμε πως για τον $\boldsymbol{A}$ ισχύει $\boldsymbol{A^{-1} = A^T}$. Πολλαπλασιάζοντας τους δύο πίνακες παίρνουμε την έξοδο ως εξής:

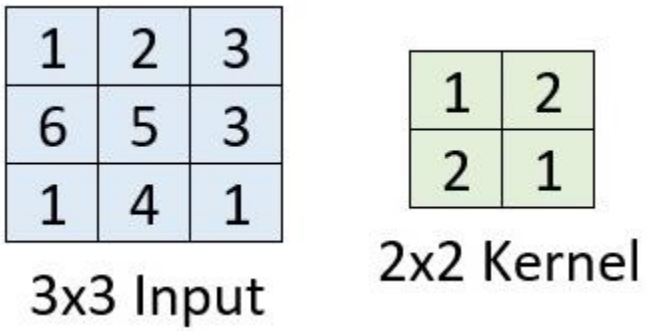
$$\boldsymbol{C = A \times B}.$$

Άμα πολλαπλασιάσουμε και τα δύο μέλη με A^T και έχουμε:

$$\boldsymbol{A^T C = (A^T A) \times B => B = A^T C}$$

Παρατηρούμε λοιπόν πως μπορούμε να πάρουμε τον αρχικό πίνακα B πολλαπλασιάζοντας την έξοδο C με τον A^T.

Επειδή η παραπάνω εξήγηση ενδέχεται να μην είναι απόλυτα κατανοητή θα δώσουμε ένα πιο εύληπτο παράδειγμα. Έστω λοιπόν ότι έχουμε μία εικόνα 3×3 και ένα φίλτρο 2×2



Εικόνα 38 Βιβλιογραφία [37]

Γνωρίζοντας πως εφαρμόζεται η συνέλιξη μπορούμε εύκολα να υπολογίσουμε το αποτέλεσμα το οποίο θα είναι ένας πίνακας 2×2 όπως φαίνεται παρακάτω.

22	21
22	20

Εικόνα 39 Αποτέλεσμα Συνέλιξης (Βιβλιογραφία [37])

Την ίδια πράξη μπορούμε να την αναπαραστήσουμε χρησιμοποιώντας τον πίνακα συνέλιξης (convolution matrix) ο οποίος μας δείχνει όλες τις θέσεις του φίλτρου στην εικόνα εισόδου.

1	2	0	2	1	0	0	0	0
0	1	2	0	2	1	0	0	0
0	0	0	1	2	0	2	1	0
0	0	0	0	1	2	0	2	1

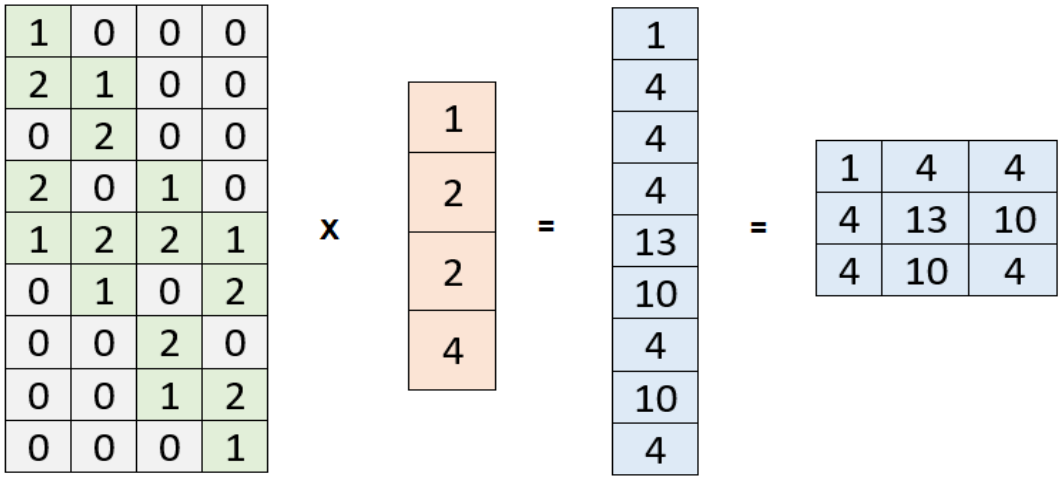
Εικόνα 40 Πίνακας Συνέλιξης (βιβλιογραφία [37])

Πολλαπλασιάζοντας τον παραπάνω πίνακα με την εικόνα εισόδου (αφού την κάνουμε διάνυσμα) παίρνουμε ακριβώς το ίδιο αποτέλεσμα.

Τι γίνεται όμως στην περίπτωση που η είσοδος μας είναι ένας πίνακας 2×2 και θέλουμε η έξοδος μας να έχει διαστάσεις 3×3; Τι γίνεται όταν πρέπει να αυξήσουμε τις διαστάσεις του εξαγόμενου χάρτη; Στο σημείο αυτό θα δούμε γιατί λέγεται transposed convolution. Πιο πάνω αναφέραμε πως ισχύει:

$$\boldsymbol{B = A^T C}$$

Επομένως άμα πολλαπλασιάσουμε τον ανάστροφο του πίνακα (transposed matrix) συνέλιξης με την είσοδο μας (η οποία προήλθε από την έξοδο κάποιου απλού συνελικτικού επιπέδου) θα μας δώσει ένα χάρτη με τις επιθυμητές διαστάσεις (Εικόνα 41)



Εικόνα 41 Transposed Convolution
(βιβλιογραφία [37])

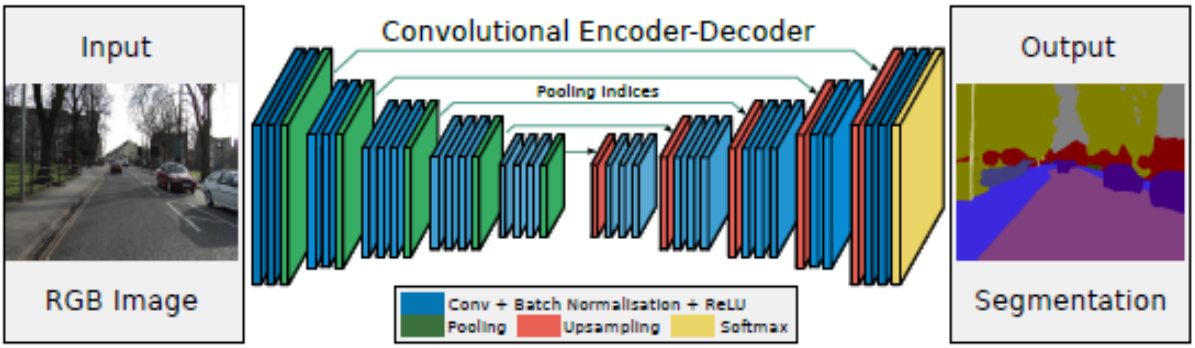
Με αυτό τον τρόπο το δίκτυο καταφέρνει να αυξήσει τις διαστάσεις του εξαγόμενου πίνακα και ταυτόχρονα μαθαίνει να το κάνει με βέλτιστο τρόπο. Αυτό συμβαίνει καθώς οι μη μηδενικές τιμές στον πίνακα συνέλιξης είναι βάρη τα οποία μαθαίνει και προσαρμόζει κατάλληλα το δίκτυο

3.4.3 Αρχιτεκτονική Κωδικοποιητή – Αποκωδικοποιητή (Encoder – Decoder)

Ο βασικός λόγος για τον οποίο έγινε, στις προηγούμενες παραγράφους αναφορά στα Πλήρως Συνελικτικά Δίκτυα ήταν για να μας δοθεί η ευκαιρία να μιλήσουμε για μία πολύ ενδιαφέρουσα αρχιτεκτονική. Η αρχιτεκτονική ονομάζεται Encoder – Decoder, όνομα που πηγάζει από τον τρόπο που λειτουργεί το εν λόγω δίκτυο. Πριν όμως δούμε πως λειτουργεί ένα δίκτυο Encoder – Decoder καλό είναι να δούμε τι είναι αυτό που οδήγησε στη δημιουργία αυτής αρχιτεκτονικής.

Τα CNNs έχουν, χωρίς καμία αμφιβολία, συμβάλει στην τεράστια άνθηση που γνωρίζει η μηχανική όραση τα τελευταία χρόνια. Αυτό δε σημαίνει όμως πως έχουν απεριόριστες δυνατότητες, το αντίθετο μάλιστα. Τα τελευταία επίπεδα των CNNs όντως fully connected περιορίζουν σημαντικά τόσο τις δυνατότητες όσο και τα πεδία εφαρμογής των δικτύων. Εδώ έρχονται τα Fully Convolutional Networks και δίνουν τη λύση. Αντικαθιστώντας τα τελευταία επίπεδα, που συνήθως ήταν fully connected, με συνελικτικά δίνουν στο δίκτυο δυνατότητες που προηγουμένως ήταν αδύνατες. Η ευελιξία και η ακρίβεια τους τα έχει καταστήσει αναντικατάστατα σε εφαρμογές όπως η θεματική κατάτμηση (semantic segmentation) και κατ' επέκταση κατανόηση σκηνής (scene understanding), στις εφαρμογές αφαίρεσης θορύβου από τις εικόνες κ.α. Μάλιστα οι πρώτες δύο από τις προαναφερθείσες εφαρμογές αποτελούν δύο από τους πυλώνες των πλήρως αυτόνομων οχημάτων.

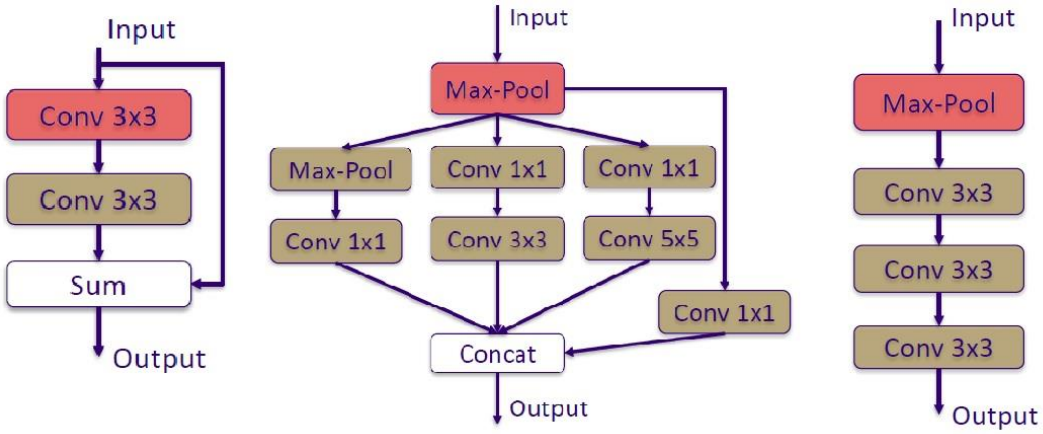
Ένα δίκτυο Encoder – Decoder μπορεί να χωριστεί σε δύο μικρότερα δίκτυα. Πρώτο είναι το δίκτυο του κωδικοποιητή (Encoder) το οποίο ακολουθείται από το δίκτυο του αποκωδικοποιητή (Decoder). Το κομμάτι του κωδικοποιητή στη βασική του μορφή αποτελείται από διαδοχικά επίπεδα συνέλιξης και pooling. Δέχεται λοιπόν τις εικόνες εισόδου



Εικόνα 42 Encoder - Decoder (SegNet) (βιβλιογραφία [22])

και εξάγει κατάλληλα χαρακτηριστικά που περιέχουν τις αναγκαίες πληροφορίες ώστε να πραγματοποιηθεί σωστά η θεματική κατάτμηση. Επειδή τα δεδομένα εισόδου περνούν από διαδοχικά επίπεδα συνέλιξης και max pooling οι διαστάσεις των παραγόμενων χαρτών έχουν αισθητά μειωμένες χωρικές διαστάσεις. Αυτό βοηθάει ιδιαίτερα καθώς μειώνει την απαιτούμενη υπολογιστή και δίνεται η δυνατότητα να υλοποιηθεί βαθύτερο δίκτυο. Στη συνέχεια δέχεται το επίπεδο του αποκωδικοποιητή τον εν λόγω χάρτη χαρακτηριστικών και χρησιμοποιώντας τις μεθόδους που αναλύσαμε σε προηγούμενες παραγράφους επαναφέρει τον χάρτη στις αρχικές του διαστάσεις. Στην πιο απλή μορφή του αποτελείται από διαδοχικά επίπεδα Transposed Convolution και Unpooling. Η αρχιτεκτονική αυτή δέχεται εικόνες οποιουδήποτε μεγέθους κάτι που τη καθιστά ευέλικτη διότι κάθε σετ δεδομένων περιέχει εικόνες με διαφορετικές διαστάσεις.

Μεταγενέστερες υλοποιήσεις (UNet κ.α.) εισήγαγαν στην αρχιτεκτονική επίπεδα dropout και batch normalization σε μία προσπάθεια να αντιμετωπιστεί η υπερεκπαίδευση. Επίσης πρέπει να τονιστεί πως τόσο στο κομμάτι της κωδικοποίησης όσο και της



Εικόνα 43Encoding Blocks: ResNet, Inception, VGG (βιβλιογραφία [42])

αποκωδικοποίησης υπάρχουν δομικά τουβλάκια των οποίων τα ονόματα πηγάζουν από τα αντίστοιχα δίκτυα στα οποία πρώτο εμφανίστηκαν. Στην Εικόνα 43 φαίνονται τα τουβλάκια της

κωδικοποίησης ενώ τα αντίστοιχα της αποκωδικοποίησης είναι ίδια απλά στη θέση των συνελίξεων και max-pool υπάρχουν τα Transposed Convolution & max unpooling.

3.5 SegNet

Η ίσως πιο διαδεδομένη υλοποίηση της αρχιτεκτονικής που περιγράφεται παραπάνω είναι το δίκτυο **SegNet**. Η μεγάλη ευελιξία του σε συνδυασμό με τα καλά αποτελέσματα πυροδότησε μια σειρά παρόμοιων βελτιώσεων του ανωτέρω δικτύου. Παρακάτω να σκιαγραφήσουμε μερικά βασικά χαρακτηριστικά του ώστε να γίνει αντιληπτό τι το έκανε τόσο σημαντικό.

Το SegNet είναι ένα πλήρως συνελικτικό δίκτυο (**FCN**) το οποίο εκτελεί θεματική κατάτμηση με αρκετά μεγάλη ακρίβεια. Το κίνητρο πίσω από την υλοποίηση του ήταν οι εφαρμογές κατανόησης σκηνής, συνεπώς υπήρξε πρόνοια κατά το σχεδιασμό του ώστε να είναι μην είναι δαπανηρό τόσο όσον αφορά τη μνήμη όσο και τους υπολογιστικούς πόρους. Σε συνδυασμό με το γεγονός με το πολύ μικρότερο αριθμό παραμέτρων το δίκτυο αυτό επιδεικνύει ιδιαίτερα καλά αποτελέσματα σε σχέση με άλλες αρχιτεκτονικές. Πιο συγκεκριμένα σύμφωνα με τους σχεδιαστές του, το **SegNet** διαχειρίζεται τη μνήμη πολύ καλύτερα από άλλες αρχιτεκτονικές ενώ παρουσιάζει ανταγωνιστικά αποτελέσματα .

Η δομή του δικτύου, όπως μπορεί να υποθέσει κανείς, χωρίζεται σε δύο κομμάτια. Στην Εικόνα 42 διακρίνονται τα δύο μέρη του δικτύου που είναι ο **Encoder** και ο **Decoder**. Αυτό που έχει ιδιαίτερο ενδιαφέρον είναι πως το κομμάτι του **Decoder** είναι καθρέπτης του **Encoder**. Με άλλα λόγια, συναντάμε και στα δύο μέρη του δικτύου τον ίδιο αριθμό κρυφών επιπέδων έχοντας φυσικά αντικαταστήσει τα επίπεδα συνέλιξης και max pooling με επίπεδα Transposed Convolution και unpooling. Στην έξοδο του **Decoder** υπάρχει ένα ακόμα επίπεδο που πραγματοποιεί κατηγοριοποίηση πολλών κλάσεων (multi class classification) κατηγοριοποιώντας έτσι κάθε εικονοστοιχείο της εικόνας ξεχωριστά.

Στο κομμάτι του κωδικοποιητή σε κάθε κρυφό επίπεδο εφαρμόζεται συνέλιξη προκειμένου να παραχθούν οι αντίστοιχοι feature maps. Στη συνέχεια εφαρμόζεται **Batch Normalization** το οποίο ακολουθείται από την εφαρμογή μίας μη γραμμικής συνάρτησης ενεργοποίησης που στην προκείμενη περίπτωση είναι η **ReLU**. Εν συνεχεία πραγματοποιείται χωρική υποδειγματοληψία έχοντας μέγεθος φίλτρου 2×2 και βηματισμό 2. Ένα αξιοσημείωτο χαρακτηριστικό είναι πως το δίκτυο κατά τη διάρκεια της χωρικής υποδειγματοληψίας αποθηκεύει τις τοποθεσίες που είχαν τη μέγιστη τιμή κάτι που μειώνει αισθητά την κατανάλωση μνήμης σε σύγκριση με όταν αποθηκεύονται ολόκληροι οι χάρτες.

Ο αποκωδικοποιητής από τη μεριά του χρησιμοποιεί τις αποθηκευμένες τοποθεσίες για να επαναφέρει τους χάρτες στις αρχικές τους διαστάσεις. Στη συνέχεια οι νέοι χάρτες περνάνε μέσα από επίπεδα **Transposed Convolution** και ύστερα πραγματοποιείται **Batch Normalization.** Η τελική έξοδος τροφοδοτεί έναν **SoftMax** classifier ο οποίος κατηγοριοποιεί όλα τα εικονοστοιχεία της εικόνας.

4. Εφαρμογή και κώδικας

Η παρούσα πτυχιακή αναπτύχθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού python και τις βιβλιοθήκες TensorFlow και Keras που προσφέρουν εύκολη υλοποίηση περίπλοκων νευρωνικών. Η αρχιτεκτονική που επιλέχτηκε είναι αυτή των Fully Convolutional Networks εν μέρει εμπνευσμένη από το paper [22] η οποία αποδείχθηκε αρκετά αποτελεσματική. Λόγω έλλειψης υπολογιστή με δυνατή διακριτή κάρτα γραφικών όλος ο κώδικας γράφτηκε και δοκιμάστηκε στην αρχή στην πλατφόρμα Floydhub όπου μπορεί κάποιος να «νοικιάσει» μέχρι πρότινος απλησίαστες σε τιμή κάρτες γραφικών. Στη συνέχεια, λόγω της ευκολίας που παρέχει, χρησιμοποιήθηκε το **Google Collaboratory**. Το θετικό αυτών των υπηρεσιών είναι πως σου παρέχουν ένα περιβάλλον προγραμματισμού που είναι φορτωμένο πάντα με την τελευταία έκδοση της **Python** και όλων των διαδεδομένων **modules** (numpy, matplotlib, scikit-learn κ.α.).

4.1 Δεδομένα εκπαίδευσης

Η εύρεση δεδομένων εκπαίδευσης για το παρόν πρόβλημα αποδείχτηκε δυσκολότερη από το αναμενόμενο καθώς τα καλά datasets (tuSimple , DeepDrive & CULane) είχαν μέγεθος μεγαλύτερο από 10Gb. Αυτό και μόνο τα κατέστησε απαγορευτικά γιατί καμία από τις πλατφόρμες που ανέφερα παραπάνω δε σου επιτρέπει να φορτώσεις τέτοιο όγκο δεδομένων χωρίς να το πληρώσεις αδρά. Μετά από αρκετό ψάξιμο βρήκα στο Github ένα dataset κομμένο και ραμμένο στις ανάγκες μου. Το οποίο όχι μόνο δεν ήταν τεράστιο αλλά ο δημιουργός του το είχε φτιάξει έτσι που ήταν έτοιμο για χρήση χωρίς καμία προεργασία. Παραθέτω παρακάτω μερικά στοιχεία του dataset όπως τα έδωσε ο δημιουργός [23] του (βλέπε

Παράρτημα):

- Οι εικόνες προέρχονται από βίντεο ανάλυσης 720p που τραβήχτηκαν με τη βοήθεια κινητού τηλεφώνου.
- Συλλέχθηκαν συνολικά 21.054 εικόνες από 12 βίντεο.
- Από τα 21.054 frames τα 14.235 κρίθηκαν κατάλληλα για χρήση.
- Χρησιμοποιήθηκαν εικόνες (frames) από το Udacity's Advanced Lane Lines project
- Όλες οι εικόνες που συλλέχθηκαν από τα βίντεο σμικρύνθηκαν σε ανάλυση 160×80 pixels με σκοπό την ελαχιστοποίηση του χρόνου εκπαίδευσης.
- Οι εικόνες περιέχουν και δύσκολες περιοχές όπως διασταυρώσεις και σημεία που διεξάγονται έργα όπως επίσης στροφές.
- Χρησιμοποιήθηκαν διάφορες τεχνικές της μηχανικής όρασης για τον εμπλουτισμό του dataset με 4.404 επιπλέον frames.
- Μετά από όλη την προεργασία το dataset περιέχει 12.764 εικόνες κατάλληλες για την εκπαίδευση του δικτύου.

Τα βίντεο που χρησιμοποιήθηκαν για το τεστάρισμα του κώδικα προήλθαν από διάφορα public repositories στο GitHub.

4.2 Αρχιτεκτονική Δικτύου

Η παρούσα πτυχιακή πέρασε από πολλά στάδια κατά τη διάρκεια της υλοποίησης του δικτύου. Αρχικά χρησιμοποιήθηκαν τεχνικές του κλασικού machine vision προκειμένου να ανιχνευθούν επιτυχώς οι λωρίδες. Αυτή η προσέγγιση είχε πολύ καλά αποτελέσματα αλλά το εκτενές fine tuning που χρειαζόταν την έκανε λιγότερο ιδανική. Επιπλέον ενώ το κλασσικό machine vision χρησιμοποιείται εκτενώς, δεν αποτελεί πλέον το state of the art οπότε άρχισα να αναζητώ άλλες λύσεις. Στη συνέχεια χρησιμοποιήθηκαν συνελικτικά νευρωνικά δίκτυα. Τα αποτελέσματα ήταν πολύ καλύτερα (αναμενόμενο)καθώς το δίκτυο μάθαινε από μόνο του ποια χαρακτηριστικά ήταν σημαντικά. Μετά την επαφή μου με το state of the art δίκτυο **SegNet** [22] και παρά τα ικανοποιητικά αποτελέσματα της προηγούμενης υλοποίησης αποφάσισα να δοκιμάσω κάτι παρόμοιο. Η αρχιτεκτονική που επέλεξα τελικά ήταν αυτή του **Encoder – Decoder** που ανήκει στα Fully Convolutional Networks.

4.2.1 Γιατί Encoder – Decoder;

Αυτή τη στιγμή υπάρχουν πάρα πολλές αρχιτεκτονικές νευρωνικών δικτύων που μπορούν να ανιχνεύσουν με ακρίβεια τις λωρίδες στο δρόμο. Μία από τις πιο ενδιαφέρουσες αρχιτεκτονικές περιγράφεται στη βιβλιογραφία [24] όπου χρησιμοποιήθηκε ένας συνδυασμός δικτύων CNN και RNN επιδεικνύοντας πολύ καλά αποτελέσματα. Η επιλογή της αρχιτεκτονικής Encoder – Decoder έγινε βάσει δύο λόγων. Ο πρώτος από αυτούς ήταν το γεγονός πως μου κίνησε το ενδιαφέρον το **SegNet** [22] και τα αποτελέσματα που επιδεικνύει**.** Ο δεύτερος και ίσως πιο σημαντικός ήταν πως η φύση του dataset που χρησιμοποιήθηκε εν τέλει, επιτάσσει τη χρήση της ανωτέρω αρχιτεκτονικής. Τυπικά λοιπόν το δίκτυο φτιάχτηκε γύρω από το σετ δεδομένων που ήταν πιο εύκολα προσβάσιμο.

Η δομή του δικτύου που παρουσιάζεται είναι εμφανώς επηρεασμένη από αυτή του **SegNet**. Υπάρχουν προφανώς αρκετές διαφορές με τη σημαντικότερη να είναι ο αριθμός των

φίλτρων που εφαρμόζονται σε κάθε επίπεδο. Η επιλογή του αριθμού των φίλτρων έγινε με τέτοιο τρόπο ώστε να θυμίζει το **SegNet** χωρίς να χρησιμοποιήσω τον ίδιο αριθμό. Ο λόγος ήταν πως λόγω του τρόπου που φορτώνονται τα δεδομένα εκπαίδευσης τελείωνε η μνήμη της κάρτας γραφικών και πετούσε **OOM error**. Πέρα από το σκελετό του δικτύου ο οποίος είναι μία μικρότερη έκδοση του SegNet (με κάποιες διαφορές) όλα τα υπόλοιπα κομμάτια του μπήκαν μετά από trial & error. Για παράδειγμα αφού υλοποιήθηκε η βασική μορφή του δικτύου, δοκιμάστηκε και παρατηρήθηκε εκτενές overfitting. Στην αρχή προστέθηκαν επίπεδα batch normalization και dropout μετά από κάθε επίπεδο συνέλιξης αλλά γρήγορα συνειδητοποίησα πως ήταν άκρως υπολογιστικά δαπανηρό και έτσι κράτησα μόνο τα επίπεδα dropout. Μετά από αρκετές δοκιμές βρήκα το κατάλληλο βάθος του νευρωνικού ώστε και να έχει καλά αποτελέσματα αλλά και να εκπαιδεύεται γρήγορα.

4.3. Κώδικας

4.3.1 Το δίκτυο

Πρώτο βήμα στον κώδικα είναι η εισαγωγή των κατάλληλων βιβλιοθηκών της Python (**numpy**, **TensorFlow** κλπ.). Τρέχοντας το παρακάτω κομμάτι κώδικα φορτώνονται επιτυχώς όλα τα εργαλεία που θα μας χρειαστούν για την υλοποίηση του δικτύου. Αξίζει να σημειωθεί πως με την πρώτη γραμμή βεβαιωνόμαστε πως φορτώνεται το **TensorFlow 2.x**. Είναι σημαντικό αυτό καθώς άμα κυκλοφορήσει κάποια επόμενη έκδοση του **TensorFlow** πλατφόρμες όπως το **Google Collaboratory** την ενσωματώνουν αυτόματα και υπάρχει μεγάλη πιθανότητα να παρουσιαστεί κάποιο σφάλμα κατά την εκτέλεση του προγράμματος.

```
%tensorflow_version 2.x
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import pickle
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
tf.keras.backend.clear_session()
```

Το επόμενο βήμα είναι να φορτώσουμε τα δεδομένα εκπαίδευσης τα οποία είναι σε μορφή **`pickle`**. Για να τα φορτώσουμε λοιπόν χρησιμοποιούμε τη μέθοδο **`pickle.load()`** όπως φαίνεται παρακάτω. Στη συνέχεια με τη χρήση της βιβλιοθήκης **`Matplotlib`** τυπώνουμε στην οθόνη στην τύχη μία εικόνα από τα δεδομένα για να βεβαιωθούμε πως φορτώθηκαν επιτυχώς.

```
train_images = pickle.load(open("/content/drive/My Drive/Dataset/full_CN
N_train.p", 'rb'))
train_labels = pickle.load(open("/content/drive/My Drive/Dataset/full_CN
N_labels.p", "rb"))
```

```
plt.imshow(train_images[20]);
```

Να σημειωθεί πως ο παραπάνω τρόπος είναι πολύ δαπανηρός και συνιστάται. Ο λόγος είναι πως σώζοντας ένα ολόκληρο **dataset** σε ένα αρχείο **pickle** και φορτώνοντας το με τον τρόπο που φαίνεται παραπάνω φορτώνεται όλο στη μνήμη. Στην προκειμένη περίπτωση δε δημιούργησε κάποιο πρόβλημα αλλά σε μεγαλύτερα dataset αυτό θα σήμαινε πως θα τελείωνε η μνήμη μας αμέσως.

Με τις δύο παρακάτω γραμμές μετατρέπουμε τόσο τις εικόνες εκπαίδευσης όσο και τις επισημάνσεις (labels) σε πίνακες τύπου **NumPy** (βιβλιοθήκη επιστημονικών υπολογισμών)

```
train_images = np.array(train_images)
#Κανονικοποιούμε τις τιμές ώστε οι τιμές των πίξελ να έχουν τις τιμές 0 ή 1
labels = np.array(train_labels)/255
```

Ανακατεύουμε εικόνες και τα αντίστοιχα **labels** και δημιουργούμε δύο μικρότερα dataset.

```
train_images, labels = shuffle(train_images, labels)
X_train, x_val, Y_train, y_val = train_test_split(train_images, labels, test_size=0.1)
```

Το πρώτο σετ δεδομένων που αποτελείται από τα (X_{train}, Y_{train}) το ονομάζουμε σετ εκπαίδευσης (***training set***) καθώς με αυτό θα εκπαιδεύσουμε το δίκτυο. Το δεύτερο σετ δεδομένων που δημιουργούμε αποτελείται από τα (X_{val}, Y_{val}) και ονομάζεται σετ επαλήθευσης (***validation set)***. Ένα τέτοιο σετ χρησιμεύει στο να δούμε πόσο καλά λειτουργεί το δίκτυο σε δεδομένα που δεν έχει ξαναδεί. Η απόδοση του δικτύου στο σετ επαλήθευσης μας δίνει μία καλή εικόνα για το πόσο καλά μαθαίνει το δίκτυο.

Προκειμένου να απεικονίσουμε γραφικά τόσο το συνάρτηση κόστους όσο και την ακρίβεια στα σετ επαλήθευσης (***validation accuracy***) και σετ εκπαίδευσης (***training accuracy***) φορτώνουμε το **TensorBoard**

```
%load_ext tensorboard
```

Και διαγράφουμε προηγούμενες καταχωρήσεις προκειμένου να μη μπλεχτούν με αυτές που θα γίνουν.

```
!rm -rf ./logs/
```

Οι υπερπαράμετροι του δικτύου αποτελούν ένα πολύ σημαντικό κομμάτι του κώδικα καθώς η σωστή ρύθμιση τους ήταν ίσως το δυσκολότερο μέρος στην υλοποίηση του δικτύου. Η σωστή επιλογή για παράδειγμα του ρυθμού μάθησης (***learning rate***) έχει σημαντικές επιπτώσεις (τόσο θετικές όσο και αρνητικές) στην εκπαίδευση του δικτύου. Δυστυχώς δεν υπάρχει κάποιος χρυσός κανόνας στη ρύθμιση των παραμέτρων αυτών παρά μόνο **trial & error**.

```
#@markdown  **Μέγεθος πακέτου δεδομένων**
batch_size = 16 #@param {type: "number"}

#@markdown  **Βηματισμός (stride)**
stride = 1 #@param {type: "number"}
```

```
#@markdown  **Τύπος padding**
padding_type = 'VALID' #@param [ 'SAME' , 'VALID' ]

#@markdown **Αριθμός επαναλήψεων**
epochs = 50 #@param {type: 'number'}

#@markdown **Μέγεθος φίλτρου pooling**
pool_size = 2 #@param {type: "number"}

#@markdown **Dropout Rate**
dropout_rate = 0.2 #@param {type: "number"}

#@markdown **Ρυθμός μάθησης**
learning_rate = 0.001 #@param {type: "number"}
```

Βασικότατο ρόλο στην εκπαίδευση του δικτύου παίζει η ρυθμός μάθησης. Έχει παρατηρηθεί πως το να μειώνουμε σταδιακά το ρυθμό μάθησης κατά τη διάρκεια της εκπαίδευσης έχει θετικές επιπτώσεις στην επίδοση.

```
lr_decay_schedule = keras.optimizers.schedules.InverseTimeDecay(learning
_rate, decay_steps=(len(X_train)/batch_size)*10,decay_rate = 1,
staircase = False)
```

Το πρώτο κομμάτι του δικτύου αποτελεί τον κωδικοποιητή (**encoder**) και απαρτίζεται από μία σειρά συνελικτικών (**layers.Conv2D**), υποδειγματοληπτικών (**layers.MaxPool2D**) επιπέδων όπως και μερικά επίπεδα απόσυρσης (**layers.Dropout**).

```
input_shape = X_train.shape[1:]
encoder_input = tf.keras.Input(shape = input_shape, name = 'Encoder_Inpu
t') #Στην προκειμένη περίπτωση πρέπει όλες οι εικόνες να έχουν διαστάσει
ς (80,160,3)

b1 = layers.BatchNormalization()(encoder_input)
c1 = layers.Conv2D(8, (3,3), strides= stride, padding = padding_type, ac
tivation = 'relu', name = 'Conv_layer1')(b1)
c2 = layers.Conv2D(16, (3,3), strides = stride, padding = padding_type,
activation ='relu', name = 'Conv_layer2')(c1)
p1 = layers.MaxPool2D(pool_size=pool_size)(c2)
c3 = layers.Conv2D(16, (3,3), strides=stride, padding = padding_type, ac
tivation = 'relu', name = 'Conv_layer3')(p1)
d1 = layers.Dropout(dropout_rate)(c3)
c4 = layers.Conv2D(32, (3,3), strides = stride, padding = padding_type,
activation = 'relu', name = 'Conv_layer4')(d1)
d2 = layers.Dropout(dropout_rate)(c4)
c5 = layers.Conv2D(64, (3,3), strides = stride, padding = padding_type,
activation = 'relu', name = 'Conv_layer5')(d2)
d3 = layers.Dropout(dropout_rate)(c5)
p2 = layers.MaxPool2D(pool_size= pool_size)(d3)
c6 = layers.Conv2D(64, (3,3), strides = stride, padding = padding_type,
activation = 'relu', name = 'Conv_layer6')(p2)
```

```
d4 = layers.Dropout(dropout_rate)(c6)
c7 = layers.Conv2D(64, (3,3), strides = stride, padding = padding_type,
activation = 'relu', name = 'Conv_layer7')(d4)
c8 = layers.Conv2D(128, (3,3),strides = stride, padding = padding_type,
activation = 'relu', name = 'Conv_layer8')(c7)
d5 = layers.Dropout(dropout_rate)(c8)
encoder_output = layers.MaxPool2D(pool_size= pool_size)(d5)
encoder = tf.keras.Model(encoder_input, encoder_output, name = 'Encoder'
)
encoder.summary()
```

Στο τέλος δημιουργούμε το μοντέλο του κωδικοποιητή με την εντολή **`encoder = keras.Model(encoder_input, encoder_output, name = "Encoder)`** και τυπώνουμε μία σύνοψη του δικτύου καλώντας τη μέθοδο **`summary`** ως εξής: **`encoder.summary()`**

Το δεύτερο μέρος του δικτύου αποτελεί τον αποκωδικοποιητή (**decoder**) και είναι καθρέπτης του κωδικοποιητή. Σκοπός του είναι να επαναφέρει τον χάρτη χαρακτηριστικών, που λόγω του πρώτου τμήματος έχει πολύ μικρές διαστάσεις, στις αρχικές του διαστάσεις. Όπως είναι επόμενο λοιπόν αποτελείται από μία σειρά **Transposed Convolution (`Conv2DTranspose`)**, **unpooling layers (`UpSampling2D`)** όπως φαίνεται παρακάτω. Αργότερα προστέθηκαν και επίπεδα **`Dropout`** όπως και μερικά skip connections.

```
up1 = layers.UpSampling2D(size = pool_size)(encoder_output)
skip_connection1 = tf.concat([up1, c7], axis = -1)
dc1 = layers.Conv2DTranspose(64, (3,3), strides = stride, padding= paddi
ng_type, activation ='relu', name='Deconv_layer1')(skip_connection1)
d6 = layers.Dropout(dropout_rate)(dc1)
dc2 = layers.Conv2DTranspose(64, (3,3), strides = stride, padding = padd
ing_type,  activation = 'relu', name = 'Deconv_layer2')(d6)
d7 = layers.Dropout(dropout_rate)(dc2)
up2 = layers.UpSampling2D(size = pool_size)(d7)
dc3 = layers.Conv2DTranspose(32, (3,3), strides = stride, padding = padd
ing_type, activation ='relu', name = 'Deconv_layer3')(up2)
d8 = layers.Dropout(dropout_rate)(dc3)
dc4 = layers.Conv2DTranspose(32, (3,3), padding = padding_type, activati
on ='relu', name ='Deconv_layer4')(d8)
d9 = layers.Dropout(dropout_rate)(dc4)
dc5 = layers.Conv2DTranspose(16, (3,3), padding = padding_type, activati
on ='relu', name ='Deconv_layer5')(d9)
d10 = layers.Dropout(dropout_rate)(dc5)
up3 = layers.UpSampling2D(size = pool_size)(d10)
dc6 = layers.Conv2DTranspose(16, (3,3), padding = padding_type, activati
on ='relu', name ='Deconv_layer6')(up3)
decoder_out = layers.Conv2DTranspose(1, (3,3), padding = padding_type, a
ctivation ='relu', name ='Deconv_layer7')(dc6)
```

Αφού ορίσουμε τα κρυφά επίπεδα του αποκωδικοποιητή δημιουργούμε το μοντέλο του δικτύου ορίζοντας ως είσοδο την είσοδο του κωδικοποιητή και έξοδο την έξοδο του αποκωδικοποιητή.

```
#Δημιουργούμε το μοντέλο του Νευρωνικού Δικτύου.
encoder_decoder = tf.keras.Model(encoder_input, decoder_out, name = 'Enc
oder_Decoder_Network')
encoder_decoder.summary()   #Τυπώνουμε στην οθόνη μία "περίληψη" του δικ
τύου.
```

Πάντα εκτυπώνουμε μία περίληψη του δικτύου με σκοπό να εντοπίσουμε αν έχουμε παραλείψει κάποιο κρυφό επίπεδο ή οποιοδήποτε άλλο λάθος.

Κάνουμε compile το δίκτυο και ορίζουμε ποιον αλγόριθμο βελτιστοποίησης θέλουμε να χρησιμοποιήσουμε (Adam στην προκειμένη περίπτωση) καθώς και τη συνάρτηση σφάλματος.

```
encoder_decoder.compile(optimizer = keras.optimizers.Adam(lr_decay_sched
ule, name = "Adam"), loss = keras.losses.MeanSquaredError(), metrics=['a
ccuracy'])
```

Η τεχνική της επαύξησης των δεδομένων (***data augmentation)*** αποτελεί ένα διαδεδομένο τρόπο αντιμετώπισης τόσο του ***overfitting*** όσο και για την βελτίωση της επίδοσης του δικτύου. Μπορεί κανείς πολύ εύκολα να πραγματοποιήσει μια πληθώρα μετασχηματισμών στις εικόνες εισόδου και, θεωρητικά, να βελτιώσει την απόδοση του δικτύου.

```
datagen = ImageDataGenerator(channel_shift_range=0.3,
                             rotation_range = 0.1
)
datagen.fit(X_train)
```

Στην προκειμένη περίπτωση η βελτίωση ήταν αμελητέα, ενώ σε πολλές περιπτώσεις χειροτέρευε τα πράγματα. Αποφάσισα όμως να μην το αφαιρέσω και μετά από αρκετές προσπάθειες βρήκα ένα σετ τιμών που βελτιώνουν, έστω και ελάχιστα, την έξοδο του δικτύου.

Τέλος ξεκινάμε την εκπαίδευση του δικτύου προσδιορίζοντας του ποια είναι τα δεδομένα εκπαίδευσης, ποια τα δεδομένα επαλήθευσης και πόσες επαναλήψεις να κάνει.

```
history = encoder_decoder.fit(datagen.flow(X_train, Y_train, batch_size
= batch_size), steps_per_epoch = len(X_train)/batch_size,
                              epochs = epochs, verbose = 1,
                              validation_data=(x_val, y_val),
                              callbacks=[tensorboard_callback]
)
```

Μετά το πέρας της εκπαίδευσης αποθηκεύουμε το εκπαιδευμένο πλέον δίκτυο σε ένα αρχείο **.h5** με την παρακάτω εντολή.

```
encoder_decoder.save('Weights.h5')
```

Το αρχείο αυτό θα μας χρησιμεύσει αργότερα για ανίχνευση των λωρίδων σε βίντεο.

4.3.2 Απόδοση δικτύου

Στην παράγραφο 3.3 Εκπαίδευση του δικτύου αναφερθήκαμε εν συντομία στις συναρτήσεις σφάλματος. Η συνάρτηση σφάλματος είναι ουσιαστικά μία αναπαράσταση του

πόσο καλά λειτουργεί το δίκτυο δηλαδή το πόσο καλά μαθαίνει από τα δεδομένα εκπαίδευσης. Σκοπός της εκπαίδευσης είναι η ελαχιστοποίηση της συνάρτησης σφάλματος κοινώς προσπαθούμε να πετύχουμε όσο το δυνατόν μικρότερο σφάλμα.

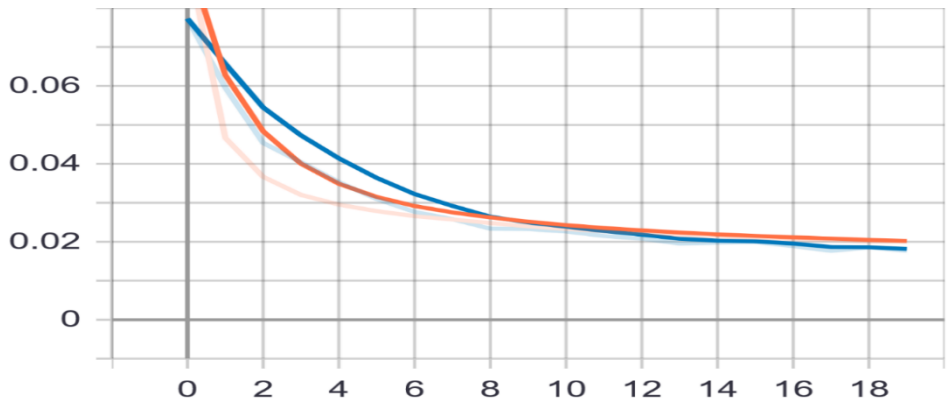
Υπάρχουν πάρα πολλές συναρτήσεις σφάλματος και κάθε μία από αυτές χρησιμοποιείται σε προβλήματα διαφορετικής φύσεως. Για παράδειγμα η cross entropy loss function χρησιμοποιείται σε προβλήματα δυαδικής κατηγοριοποίησης (binary classification). Αντίστοιχα όταν έχουμε πάνω από δύο κλάσεις (κατηγορίες) χρησιμοποιούμε μια παραλλαγή που ονομάζεται categorical cross entropy.

Στο πρόβλημα της ανίχνευσης λωρίδων, και μετά από αρκετές δοκιμές, παρατήρησα πως η Mean Squared Error loss Function (MSE) αποδίδει καλύτερα από τις υπόλοιπες υποψήφιες συναρτήσεις. Ο μαθηματικός της τύπος φαίνεται παρακάτω:

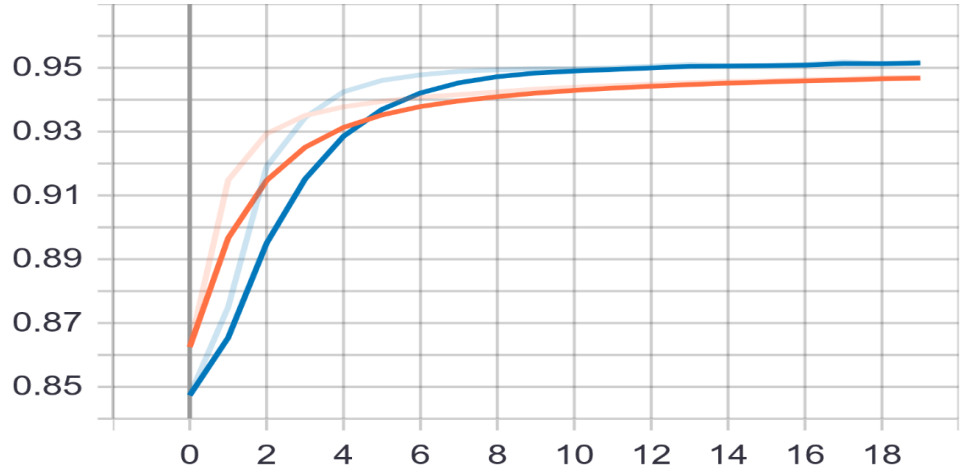
$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \widehat{Y}_i\right)^2$$

Η MSE υπολογίζει για κάθε δείγμα το τετράγωνο της διαφοράς της επιθυμητής εξόδου Y και της πραγματικής εξόδου $\widehat{Y}_i$ και αθροίζει όλα τα αποτελέσματα. Κατά την εκπαίδευση προσπαθούμε να μειώσουμε όσο το δυνατότερο γίνεται τη συνάρτηση σφάλματος. Αξίζει να αναφερθεί πως αυτή δεν πρόκειται να μηδενίσει (τουλάχιστον όχι σε κάποιο εύλογο χρονικό διάστημα) οπότε επιδιώκουμε να βρούμε τις κατάλληλες υπερπαράμετρους ώστε να ελαχιστοποιηθεί. Σύμφωνα με τις δοκιμές που έκανα κατά τη διάρκεια της δημιουργίας του δικτύου ακρίβεια $\geq 95\%$ αποφέρει ικανοποιητικά αποτελέσματα.

Παρακάτω μπορούμε να δούμε μέσω του **TensorBoard** τη γραφική αναπαράσταση τόσο της ακρίβειας του δικτύου όσο και της συνάρτησης σφάλματος. Και οι δύο φαίνονται παρακάτω. Η γραφική με πορτοκαλί χρώμα αφορά το σετ επαλήθευσης ενώ η άλλη αφορά το σετ εκπαίδευσης.



Εικόνα 44 Epoch - Loss



Εικόνα 45 Epoch - Accuracy

Είναι πολύ σημαντικό να καταλαβαίνουμε τι μας δείχνουν οι παραπάνω γραφικές παραστάσεις (πέρα από τα προφανή). Αν για παράδειγμα κατά τη διάρκεια της εκπαίδευσης παρατηρούσαμε πως η ακρίβεια στο σετ εκπαίδευσης αυξανόταν ενώ τη ακρίβεια στο σετ επαλήθευσης μειωνόταν αυτό θα σήμαινε πως το δίκτυο «υποφέρει» από υπερεκπαίδευση (overfit).

4.3.3 Εφαρμογή σε βίντεο

Για την εφαρμογή του κώδικα σε βίντεο απαιτείται για μία ακόμη φορά να φορτώσουμε τις κατάλληλες βιβλιοθήκες.

```
%tensorflow_version 2.x
import tensorflow as tf
import numpy as np
import cv2
from skimage.transform import resize
from moviepy.editor import VideoFileClip
from IPython.display import HTML
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
```

Φορτώνουμε το εκπαιδευμένο δίκτυο το οποίο θα χρησιμοποιήσουμε για να ανιχνεύσουμε τις λωρίδες σε βίντεο.

```
model = load_model('/content/Trained_Model [20].h5')
```

Δημιουργούμε μία κλάση και δύο κενές λίστες τις οποίες θα χρησιμοποιήσουμε παρακάτω.

```
class Lanes():
    def __init__(self):
        self.recent_fit = []
        self.avg_fit = []
```

Ορίζουμε μία συνάρτηση η οποία θα κάνει όλη του δουλειά για εμάς (Βλέπε

Παράρτημα). Η παρακάτω συνάρτηση δέχεται βίντεο ανάλυσης 1280 × 720 αλλά μπορεί να δεχτεί και οποιαδήποτε άλλη ανάλυση αλλάζοντας το όρισμα που δέχεται η συνάρτηση **resize.**

```
def road_lines(image):

    # Ετοιμάζουμε την εικόνα για εισαγωγή στο δίκτυο

    small_img = np.array(resize(image, (80, 160, 3), preserve_range=True
))
    small_img = small_img[None,:,:,:]

    # Κάνουμε "προβλέψεις" με το νευρωνικό δίκτυο
    prediction = model.predict(small_img)[0] * 255



    # Προσθέτουμε την πρόβλεψη στη λίστα recent.fit
    lanes.recent_fit.append(prediction)
    # Χρησιμοποιούμε μόνο τα 5 τελευταία frames για τον υπολογισμού του
μέσου όρου
    if len(lanes.recent_fit) > 5:
        lanes.recent_fit = lanes.recent_fit[1:]

    # Υπολογίζουμε το μέσο τιμή
    lanes.avg_fit = np.mean(np.array([i for i in lanes.recent_fit]), axi
s = 0)

    #Δημιουργούμε κενή εικόνα
    blanks = np.zeros_like(lanes.avg_fit).astype(np.uint8)
    lane_drawn = np.dstack((blanks, lanes.avg_fit, blanks))

    # Επαναφέρουμε την εικόνα στο αρχικό της μέγεθος.
    lane_image = resize(lane_drawn, (720,1280,3)).astype(np.uint8)

    # Ενώνουμε την πρόβλεψη με την αρχική εικόνα
    result = cv2.addWeighted(image, 1, lane_image, 1, 0)
    return result

lanes = Lanes()

# Μέρος που θα αποθηκευτεί το βίντεο
vid_output = 'proj_reg_vid2.mp4'
# Τοποθεσία του εισαγόμενου βίντεο
clip1 = VideoFileClip("/content/drive/My Drive/Test Videos/challenge.mp4
")
vid_clip = clip1.fl_image(road_lines)
%time vid_clip.write_videofile(vid_output, audio=False)
```

Εικόνα 46 Έξοδος του δικτύου (a)

Εικόνα 47 Έξοδος δικτύου (β)

Εικόνα 48 Έξοδος δικτύου (γ)

4.3.4 Χρήση του μοντέλου

Η ανίχνευση λωρίδων μπορεί να θεωρηθεί ως η ραχοκοκαλιά της περιβαλλοντικής αντίληψης, μίας ιδιότητας που όλα τα αυτόνομα οχήματα πρέπει να διαθέτουν. Και αυτό γιατί δε νοείται αυτόνομο όχημα που να μη μπορεί να αναγνωρίσει τα όρια της λωρίδας που κινείται (ego lane). Παρακάτω θα αναφερθούμε πολύ γρήγορα στο που μπορεί να χρησιμοποιηθεί ένα σύστημα σαν και αυτό που παρουσιάστηκε στην παρούσα πτυχιακή.

Το παρόν δίκτυο όντας κομμάτι του συστήματος προειδοποίησης απόκλισης λωρίδας μπορεί να βρει εφαρμογή σε συστήματα LDWS (βλέπε 1.4 Σύστημα προειδοποίησης απόκλισης από τη λωρίδα (LDWS)). Θα μπορούσε κάλλιστα να συνδυαστεί με ένα σύστημα ανίχνευσης αυτοκινήτων και αποφυγής σύγκρουσης όπως αυτό που περιγράφεται στην βιβλιογραφία [5]. Παρέχοντας έτσι στο εν δυνάμει αυτόνομο όχημα την ικανότητα να γνωρίζει τα όρια της λωρίδας που κινείται και να παρακολουθεί τα αυτοκίνητα που κινούνται γύρω του.

Σα σύστημα αποτελεί τη βάση της αυτόνομης οδήγησης και πρακτικά μπορεί να χρησιμοποιηθεί σε συνδυασμό με οποιοδήποτε ADAS προσφέροντας ένα επιπλέον επίπεδο αυτονομίας.

4.3.5 State-of-the-art υλοποιήσεις

Το πρόβλημα της ανίχνευσης λωρίδων μελετάται εκτενώς και όσο τα αυτόνομα οχήματα μπαίνουν στη mainstream αγορά η μελέτη αυτή θα εντατικοποιείται. Μέρα με τη μέρα δημοσιεύονται μελέτες που προτείνουν λύσεις στο πρόβλημα αυτό προτείνοντας καινοτόμες αρχιτεκτονικές. Έτσι λοιπόν θεωρώ πως είναι πρέπον να αναφερθούν μερικές από αυτές τις υλοποιήσεις. Για όποιον λοιπόν ενδιαφέρεται να γνωρίσει μερικές από τις state of the art lane detection υλοποιήσεις παραθέτω μία σύντομη περιγραφή κάποιων από αυτών.

Πρώτα θα αναφερθούμε στο **PINet** [25] ένα δίκτυο που επιδεικνύει ακρίβεια της τάξης του 96% κάτι που αποτελεί μεγάλο κατόρθωμα. Χρησιμοποιεί μία αρχιτεκτονική που οι συγγραφείς του ονομάζουν Hourglass. Ένας από τα χαρακτηριστικά που το κάνουν να ξεχωρίζει είναι πως η έξοδος του έχει πιο μικρότερο μέγεθος σε σύγκριση με υλοποιήσεις που χρησιμοποιούν θεματική κατάτμηση (semantic segmentation) εξοικονομώντας μνήμη. Επιπλέον, σύμφωνα πάντα με τους συγγραφείς, έχει μικρότερο ποσοστό false positive από οποιαδήποτε άλλη αρχιτεκτονική. Όλα αυτά έρχονται όμως με ένα κόστος καθώς απαιτείται αρκετά μεγάλη υπολογιστική ισχύ κάτι που οι δημιουργοί του αναφέρουν πως θα λυθεί.

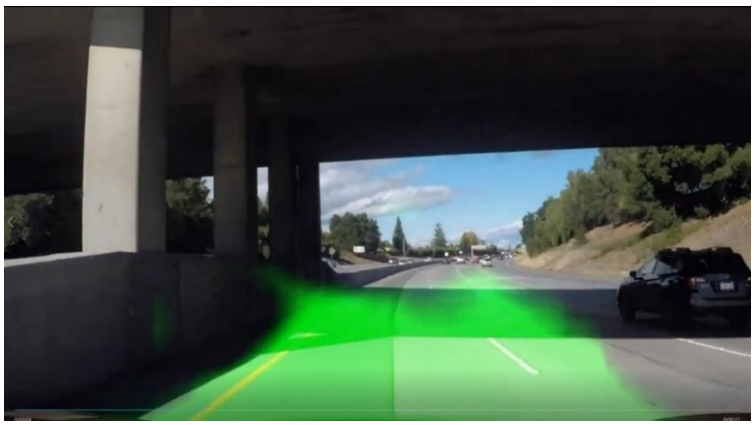
Η δεύτερη υλοποίηση στην οποία θα αναφερθούμε είναι το **ENet-SAD** [26] μία πρόταση που πετυχαίνει εξαιρετική ακρίβεια στο σετ δεδομένων **tuSimple** της τάξης του 95%. Μάλιστα οι δημιουργοί του αναφέρουν πως έχει ~20 φορές λιγότερες παραμέτρους και είναι ~10 φορές πιο γρήγορο από άλλα state-of-the-art δίκτυα ενώ ταυτόχρονα οι επιδόσεις του είναι εξαιρετικές.

Το τρίτο και τελευταίο δίκτυο στο οποίο θα αναφερθούμε είναι το **LaneNet** [27]. Το δίκτυο αυτό έχει τη δυνατότητα να ανιχνεύσει ένα μεταβλητό αριθμό λωρίδων αλλά αυτό που το διαφοροποιεί από άλλα δίκτυα είναι η ικανότητα του να ανταπεξέρχεται στις αλλαγές λωρίδας. Επιπλέον επιδεικνύει αρκετά μεγάλη ταχύτητα πετυχαίνοντας 50fps κάτι που είναι ιδιαίτερα εντυπωσιακό και μας δίνει μία καλή εικόνα για το τι δυνατότητες έχουν τα νευρωνικά δίκτυα.

Κλείνοντας δε γίνεται να μη γίνει αναφορά στο επονομαζόμενο transfer learning μία τεχνική που έχει γνωρίσει μεγάλη άνθηση. Η ιδέα πίσω από το transfer learning είναι απλή, μπορούμε να χρησιμοποιήσουμε ένα προ-εκπαιδευμένο δίκτυο που έχει εκπαιδευτεί για ένα σκοπό και να τα χρησιμοποιήσουμε για να πραγματοποιήσουμε κάποια διαφορετική διεργασία. Για παράδειγμα υπάρχουν δίκτυα με εκατοντάδες κρυφά επίπεδα και εκατομμύρια παραμέτρους που έχουν εκπαιδευτεί με πελώρια σετ δεδομένων (ImageNet), η εκπαίδευση των οποίων πολλές φορές διαρκεί μέρες. Μπορούμε να πάρουμε λοιπόν τέτοια δίκτυα και τροποποιώντας τα κατάλληλα έχουμε τη δυνατότητα να χρησιμοποιήσουμε τη γνώση τους για να ανιχνεύσουμε λωρίδες ή οτιδήποτε άλλο θέλουμε. Με αυτό τον τρόπο χρησιμοποιούμε γνώση που υπό άλλες συνθήκες δεν θα ήταν διαθέσιμη σε εμάς ώστε να καταφέρουμε πράγματα που μέχρι πρότινος ήταν αδύνατα

Συμπεράσματα και μελλοντικές βελτιώσεις

Η υλοποίηση του παρόντος νευρωνικού δικτύου αποδείχτηκε πιο δύσκολη από όσο περίμενα αλλά παρέμεινε αρκετά πιο εύκολη από την αντίστοιχη υλοποίηση με τεχνικές της μηχανικής όρασης. Τα αποτελέσματα του δικτύου είναι ικανοποιητικά αλλά παρουσιάζει μία έντονη ευαισθησία στις σκιές (Εικόνα 49) και στις απότομες αλλαγές του φωτισμού. Αυτό θα μπορούσε ενδεχομένως να διορθωθεί με ένα μεγαλύτερο dataset ή χρησιμοποιώντας διαφορετική αρχιτεκτονική δικτύου όπως για παράδειγμα **Recurrent Neural Networks** (RNN) όπως φαίνεται στη βιβλιογραφία [24]. Δυστυχώς όμως με τα μέσα που διαθέτω μου είναι αδύνατο να χρησιμοποιήσω τα (τεράστια) σετ δεδομένων των μεγάλων πανεπιστημίων.



Εικόνα 49 Ευαισθησία δικτύου στις σκιές

Επιπλέον σε επόμενη φάση θα ήθελα να δημιουργήσω το δικό μου σετ δεδομένων αλλά καθώς αυτό προϋποθέτει πολύ χρόνο και εμπειρία δεν ήταν δυνατόν να το επιχειρήσω στα πλαίσια της παρούσας πτυχιακής. Αυτό θα μου έδινε μεγαλύτερη ευελιξία καθώς και πλήρη έλεγχο των δεδομένων. Επίσης το γεγονός πως το dataset που χρησιμοποίησα είναι σε μορφή `pickle` σημαίνει πως φορτώνεται μονομιάς στη μνήμη κάτι που είναι υπολογιστικά δαπανηρό και με απέτρεψε από να χρησιμοποιήσω βαθύτερο δίκτυο.

Εν κατακλείδι, ο παρών κώδικας μπορεί να βελτιωθεί με ποικίλους τρόπους αλλά είμαι πολύ ευχαριστημένος για την απόδοση που έχει. Ευελπιστώ να μπορέσω στο μέλλον προσθέσω πολλά πράγματα που έχω στο μυαλό μου

Παράρτημα

Στο παρόν πόνημα χρησιμοποιήθηκε **open source** κώδικας καθώς και ένα σετ δεδομένων των οποίων ο δημιουργός επιτρέπει την χρήση από τον καθένα. Ο μόνος όρος που έθεσε είναι αναγραφή του παρακάτω μηνύματος σε περίπτωση που χρησιμοποιηθεί μέρος της δουλειάς του.

```
MIT
License

        Copyright (c) 2017 Michael Virgo

        Permission is hereby granted, free of charge, to any person obtaining a copy
        of this software and associated documentation files (the "Software"), to deal
        in the Software without restriction, including without limitation the rights
        to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
        copies of the Software, and to permit persons to whom the Software is
        furnished to do so, subject to the following conditions:

        The above copyright notice and this permission notice shall be included in all
        copies or substantial portions of the Software.

        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
        IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
        FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
        AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
        OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
        SOFTWARE.
```

Ευρετήριο Όρων

Βιβλιογραφία

E. Calli, Faster Convolutional Neural Networks, Nijmegen, 2017. 1]

Google Inc, «TensorFlow,» Google, [Ηλεκτρονικό]. Available: 2] TensorFlow.org.

M. Lu, K. Wevers και R. Van Der Heijden, «Technical Feasibility of 3] Advanced Driver Assistance Systems (ADAS) for Road Traffic Safety,» *Transportation Planning and Technology*, pp. 3-23, 30 Μάρτιος 2005.

H. Xiangkun, L. Yulong, L. Chen και J. Xuewu, «Emergency steering control 4] of autonomous vehicle for collision avoidance and stabilisation,» *Vehicle System Dynamics*, p. 27, 1 Νοέμβριος 2018.

K. M. Πολυχρονόπουλος, Σύστημα Μηχανικής Όρασης για Αποφυγή της 5] Σύγκρουσης Οχημάτων, Πάτρα, 2018.

D. C. H. Shaw και J. Z. Z. Shaw, «VEHICLE COLLISION AVOIDANCE 6] SYSTEM». Ηνωμένες Πολιτείες Αμερικής Ευρεσιτεχνία 5,529,138, 25 Ιούνιος 1995.

I. G. Daza, L. M. Bergasa, S. Bronte, J. J. Yebes, J. Almazan και R. Arroyo, 7] «Fusion of Optimized Indicators from Advanced Driver Assistance Systems (ADAS) for Driver Drowsiness Detection,» *Sensors*, pp. 1106-1131, 9 Ιανουάριος 2014.

«Smarteye,» Smart Eye AB, 2019. [Ηλεκτρονικό]. Available: 8] https://smarteye.se/. [Πρόσβαση 16 Μάιος 2020].

Seeingmachines, «Seeingmachines.com,» Seeingmachines, 2020. 9] [Ηλεκτρονικό]. Available: https://www.seeingmachines.com/technology/. [Πρόσβαση 16 Μάιος 2020].

J. M. Zurada, Introduction to Artificial Neural Systems, WEST 10] PUBLISHING COMPANY, 1992.

Γ. Αικατερίνη, «Μηχανική Μάθηση,» σε *Τεχνητή νοημοσύνη*, Αθήνα, Αττική: 11] Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, 2015, p. 263.

Γ. Μπουτάλης και Γ. Συρακούλης, Υπολογιστική Νοημοσύνη & Εφαρμογές, 12] Ξάνθη, 2010, p. 262.

C. Munker, Using Convolutional Neural Networks to distinguish vehicle pose 13] and vehicle class, Darmstadt, 2016.

A. Ng, «Coursera,» [Ηλεκτρονικό]. Available: 14] https://www.coursera.org/learn/convolutional-neural-networks/home/welcome.

A. Vedaldi και A. Zisserman, «VGG Convolutional Neural Networks
15] Practical,» 2017. [Ηλεκτρονικό]. Available:
http://www.robots.ox.ac.uk/~vgg/practicals/cnn/index.html#part-21-using-back-propagation-in-practice. [Πρόσβαση 28 Δεκέμβρης 2019].

«Deep Learning,» Standford University, [Ηλεκτρονικό]. Available:
16] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/. [Πρόσβαση 20 Ιανουαρίου 2020].

I. Goodfellow, Y. Bengio και A. Courville, «Deep Learning,» MIT Press,
17] 2016.

J. L. Ba και D. P. Kingma, «Adam: A Method for Stochastic Optimazation,»
18] ICRL, 2015.

T. Schaul, I. Antonoglou και D. Silver , «Cornell University,» 25 Φλεβάρης
19] 2014. [Ηλεκτρονικό]. Available: https://arxiv.org/pdf/1312.6055.pdf. [Πρόσβαση 12 Δεκέμβρης 2019].

L. Taylor και G. Nitsche, «Improving Deep Learning using Generic Data
20] Augmentation,» Cape Town, 2017.

V. Dumoulin και F. Visin, «A guide to convolution arithmetic for deep,»
21] Montreal, 2018.

V. Badrinarayanan,, A. Kendall και R. Cipolla,, SegNet: A Deep
22] Convolutional Encoder - Decoder Architecture for Image Segmentation, 2016.

M. Virgo, «Github,» 4 Μαΐου 2017. [Ηλεκτρονικό]. Available:
23] https://github.com/mvirgo/MLND-Capstone/blob/master/MLND%20Capstone%20Project%20Report.pdf.

Z. Qin, J. Hanwen, D. Qiyu, Y. Yuanhao, C. Long και W. Qian, «Robust
24] Lane Detection from Continuous Driving Scenes Using Deep Neural Networks,» *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, 2019,* τόμ. 69, αρ. 1, pp. 41-54, 25 Οκτώβρης 2019.

K. Yeongmin, J. Jiwon, K. Donghwuy και J. Moongu, «Key Points
25] Estimation and Point Instance Segmentation Approach for Lane Detection,» 2020.

H. Yuenan, M. Zheng, L. Chunxiao και L. C. Chen, «Learning Lightweight
26] Lane Detection CNNs by Self Attention Distillation,» 2019.

D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans και L. Van Gool,
27] «Towards End-to-End Lane Detection: an Instance Segmentation Approach,» Λέβεν, 2018.

K. Salman , H. Rahmani, S. Syed Afaq Ali και M. Bennamoun, A Guide to
28] Convolutional Neural Networks for Computer Vision, Morgan & Claypool, 2018.

S. Khosla, «www.geeksforgeeks.org,» [Ηλεκτρονικό]. Available:
29] https://www.geeksforgeeks.org/cnn-introduction-to-padding/. [Πρόσβαση 20 Ιανουρίου 2020].

«https://www.zybuluo.com,» 15 Δεκεμβρης 2018. [Ηλεκτρονικό]. Available:
30] https://www.zybuluo.com/hongchenzimo/note/1086311#1kernel. [Πρόσβαση 12 Αυγουστος 2019].

S. Saha, «Towards Data Science,» 15 Δεκέμβρης 2018. [Ηλεκτρονικό].
31] Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Πρόσβαση 12 Αύγουστος 2019].

B. Li και R. Venkatesan, Convolutional Neural Networks in Visual
32] Computing, CRC Press, 2018.

Π. Αθανάσιος, Συνελικτικά Νευρωνικά Δίκτυα στην Υπολογιστική Όραση,
33] Πάτρα, 2016.

G. Gwardys, «Grzegorz Gwardys Experience with Machine Learning/ Deep
34] Learning and more...,» 19 Ιούνιος 2016. [Ηλεκτρονικό]. Available: https://grzegorzgwardys.wordpress.com/2016/04/22/8/. [Πρόσβαση 18 Οκτώβρης 2019].

F.-F. Li, J. Johnson και S. Yeung, «Lecture 11: Detection and Segmentation,»
35] Stanford University , 17 Μάιος 2017. [Ηλεκτρονικό]. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf. [Πρόσβαση 18 Δεκεμβρης 2019].

«http://ufldl.stanford.edu/,» [Ηλεκτρονικό]. Available:
36] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/.

Chris, «Understanding transposed convolutions,» 2019.
37]

A. Shaout, D. Colella και S. Awad, «Advanced Driver Assistance Systems -
38] Past, Present and Future,» IEEE, Dearborn, Michigan, 2012.

P. Worrawut, T. Somphong και P. Manukid, «Adaptive Cruise Control for an
39] Intelligent Vehicle,» σε *Proceedings of the 2008 IEEE,* Prathumthani, 2008.

M. Staubach, «Factors correlated with traffic accidents as a basis for
40] evaluating Advanced Driver Assistance Systems,» *Accident Analysis and Prevention 41,* pp. 1025-1033, 13 Ιούνιος 2009.

C. Winserng και Y. L. Phooi, «A Framework for Lane Departure Warning
41] System for Various Lane Markings,» 15 Ιανουάριος 2017. [Ηλεκτρονικό]. Available: https://www.researchgate.net/publication/312345978_A_Framework_for_Lane_Departure_Warning_System_for_Various_Lane_Markings. [Πρόσβαση 10 Μάιος 2020].

S. Mehta, «Paul G. Allen School of Computer Science & Engineering,»
42] [Ηλεκτρονικό]. Available: https://courses.cs.washington.edu/courses/cse576/17sp/notes/Sachin_Talk.pdf. [Πρόσβαση 5 Ιούνιος 2020].