|  |
| --- |
| Ατομική Διπλωματική Εργασία  **ΑΝΙΧΝΕΥΣΗ ΦΩΤΕΙΝΗΣ ΠΗΓΗΣ ΓΙΑ ΦΩΤΙΣΜΟ ΑΝΤΙΚΕΙΜΕΝΟΥ ΣΕ AUGMENTED REALITY**  **Στέφανος Κυριακίδης**  **ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**  UCY-new-logo  **ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**  **Μάιος 2018** |

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Ανίχνευση Φωτεινής Πηγής για Φωτισμό Αντικειμένου σε Augmented Reality**

**Στέφανος Κυριακίδης**

Επιβλέπων Καθηγητής

Γιώργος Χρυσάνθου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2018

**Ευχαριστίες**

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου, κ. Γιώργο Χρυσάνθου που με επέλεξε για αυτή την διπλωματική, και για τις κατευθυντήριες γραμμές που μου έδωσε για την συγγραφή αυτής της διπλωματικής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Μελίνο Αβερκίου για την πολύτιμη βοήθειά του που μου παρείχε καθ' όλη την διάρκεια της έρευνάς μου στα νευρωνικά δίκτυα.

Τέλος, θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Χρίστο Χριστοδούλου, ο οποίος με βοήθησε να κατανοήσω τα νευρωνικά δίκτυα εις βάθος στο μάθημά του που ήταν διαθέσιμο στο χειμερινό εξάμηνο.**Περίληψη**

Με τον όρο Augmented Reality (από τώρα AR) εννοούμε τον τρόπο με τον οποίο μπορούμε να συνδέσουμε το πραγματικό τρισδιάστατο περιβάλλον με ένα άλλο, τεχνητό περιβάλλον, του οποίου τα στοιχεία του παράγονται από κάποιον υπολογιστή, και παρουσιάζονται στον χρήστη σαν οπτικές, ακουστικές ή κιναισθητικές πληροφορίες.

Με την δημιουργία τεχνολογιών AR, μπορούν να επιτευχθούν εφαρμογές πιο διαδραστικές προς τον χρήστη, που να του προσφέρουν μια πιο επιβλητική εμπειρία, όπως για παράδειγμα στην παιδεία, σε στρατιωτικές επιχειρήσεις και σε εκπαιδεύσεις αστυνομικών.

Τα τελευταία χρόνια, λόγω του ότι η τεχνολογία έχει δεχθεί ραγδαία ανάπτυξη, η δημιουργία AR περιβάλλων γίνεται όλο και πιο επιτεύξιμη, και η διαθέσιμες τεχνολογίες για την δημιουργία τους όλο και περισσότερες. Η δημιουργία, όμως, ενός αληθοφανούς τεχνητού στοιχείου για το εικονικό μέρος του AR, έγινε πιο απαιτητικό και ανταγωνιστικό. Ένα από τα προβλήματα είναι η σωστή σκίαση του δημιουργημένου αντικειμένου. Ο σκοπός της παρούσας διπλωματικής εργασίας, είναι να βρίσκουμε επιτυχώς την φωτεινή πηγή στον πραγματικό κόσμο, για να δημιουργούμε την κατάλληλη σκίαση για το αντικείμενο μας.

**Περιεχόμενα**

Κεφάλαιο **1**

**Εισαγωγή**……………………………………………………… **1**

1.1 Τι είναι το AR 1

1.2 Περιγραφή του προβλήματος 2

1.3 Σκοπός ΑΔΕ 2

1.4 Περίγραμμα ΑΔΕ 3

Κεφάλαιο **2**

**Προηγούμενη Εργασία**……………..………………………………….. **4**

2.1 Προηγούμενα Αποτελέσματα 4

2.2 Νέα Προσέγγιση 5

Κεφάλαιο **3**

**Νευρωνικά Δίκτυα**…………………………………………. **7** 3.1 Ιστορική Αναδρομή 7

3.2 Είδη Νευρωνικών Δικτύων 7

3.3 Νευρωνικά Δίκτυα Perceptron 8

3.4 Νευρωνικά Δίκτυα Kohonen 11

3.5 Νευρωνικά Δίκτυα Με Λήψη Αποφάσεων Markov 13

3.6 Convolutional Νευρωνικά Δίκτυα 15

3.7 Επιλογή Νευρωνικού Δικτύου 18

3.8 Επιλογή Εφαρμογής για Επίλυση Προβλήματος 19

Κεφάλαιο **4**

**Θεωρητική Λύση**…………….……………………………………….. **21**

4.1 Θεωρία Λύσης 21

4.2 Τρόπος Υλοποίησης 22

4.3 Αναμενόμενα Αποτελέσματα 25

Κεφάλαιο **5**

**Στήσιμο Λύσης**………………..………………….……………. **27**

5.1 Στήσιμο Unity 27

5.2 Στήσιμο Νευρωνικού Δικτύου 29

Κεφάλαιο **6**

**Αποτελέσματα**………………..………………….……………. **31**

6.1 Δείγματα 31

6.2 Έλεγχος 32

6.3 Σύγκριση Αποτελεσμάτων 36

Κεφάλαιο **7**

**Συμπεράσματα** ……….………………………………………………… **38**

Κεφάλαιο **8**

**Μελλοντική Δουλειά** .………………………………………………… **40**

**Β ι β λ ι ο γ ρ α φ ί α …………………………………………………. 42**

**Π α ρ ά ρ τ η μ α Α………...………………………...…….…………Α-1**

**Π α ρ ά ρ τ η μ α Β……………………………..……………….……Β-1**

**Κεφάλαιο 1**

**Εισαγωγή**

1.1 Τι είναι το AR 1

1.2 Περιγραφή του προβλήματος 2

1.3 Σκοπός ΑΔΕ 3

1.4 Περίγραμμα ΑΔΕ 3

* 1. **Τι είναι το AR**

AR είναι ο τρόπος με τον οποίο μπορούμε να ενώσουμε το πραγματικό περιβάλλον με computer-generated στοιχεία μέσω διαφόρων τεχνολογιών. Μπορούμε, δηλαδή, να εισάγουμε στοιχεία από τον υπολογιστή στο οπτικό ή ακουστικό πεδίο του χρήστη, όπως για παράδειγμα μέσω AR γυαλιών, μέσω της οθόνης του κινητού, φορώντας ακουστικά παραγωγής τρισδιάστατων ήχων.

Η σημαντικότητα της ανάπτυξης τεχνολογιών AR, έχει αποδειχθεί σε πολλούς τομείς βοηθητικός, όπως αρχαιολογία, παιδεία και αρχιτεκτονική, και ζωτικής σημασίας σε άλλους, όπως εκπαίδευση αστυνομικών, real-time παροχή πληροφοριών σε στρατιωτικές επιχειρήσεις και εκπαίδευση πιλότων. Επίσης, η βιομηχανία ηλεκτρονικών παιχνιδιών άρχισε να δείχνει αρκετά μεγάλο ενδιαφέρον στις δυνατότητες και στην εμπειρία που μπορεί να προσφέρει μια τέτοια τεχνολογία.

Ως ιδέα, η παραγωγή τεχνολογιών AR υπήρξε ένας από τους πιο σημαντικούς στόχους του τομέα της Πληροφορικής. Η πρώτη ιδέα κοντά στο AR, έγινε το 1901 από τον συγγραφέα L. Frank Braum, στο μυθιστόρημά του "The Master Key", όπου ο πρωταγωνιστής αποκτά γυαλιά που, όποτε τα φοράει, του παρουσίαζαν στο μέτωπο του ατόμου που παρακολουθούσε στοιχεία για τον χαρακτήρα του. Ωστόσο, η δημιουργία AR τεχνολογιών, είναι πολύ πρόσφατη, και χρονολογείται στις αρχές του 1980.

Στο παρόν στάδιο, λόγω της ραγδαίας ανάπτυξης της τεχνολογίας, και λόγω της αυξανόμενης ζήτησης τεχνολογιών AR από άλλους τομείς και κλάδους, η χρήση και ανάπτυξη των AR τεχνολογιών παρουσιάζουν πολλά προβλήματα λόγω των απαιτήσεων που υπάρχουν στην τελειοποίηση του τομέα αυτού.

* 1. **Περιγραφή του προβλήματος**

Λόγω του ότι το AR είναι η ομαλή και αρμονική συνύπαρξη του πραγματικού κόσμου με τον τεχνητό, η δημιουργία του τεχνητού μέρους πρέπει να είναι όσο πιο αληθοφανή και ακριβή γίνεται, έτσι ώστε ο χρήστης να έχει μία ουσιώδη και αποδοτική εμπειρία. Ένα από τα προβλήματα που έχει το AR, είναι η προβολή των αλλαγών σε πραγματικό χρόνο. Ο συγχρονισμός των αλλαγών του περιβάλλοντος με την δημιουργία ή αλλαγή των στοιχείων του τεχνητού κόσμου δεν είναι πάντα γρήγορη.

Ένα άλλο θέμα το οποίο παρουσιάζεται, είναι η έλλειψη αληθοφανείς παρουσίας των τεχνητών στοιχείων, σε σχέση με αυτά του πραγματικού κόσμου. Αυτό το πρόβλημα είναι πιο έντονο αυτή την εποχή, λόγω της ραγδαίας ανάπτυξης της τεχνολογίας. Αυτό το πρόβλημα αρχίζει να αντιμετωπίζεται, διότι μέρα με την μέρα τα γραφικά των υπολογιστών βελτιώνονται, και ο τρόπος με τον οποίο προβάλλονται και αποθηκεύονται γίνεται όλο και πιο γρήγορος.

Το θέμα που θα ασχοληθούμε με αυτή την εργασία, είναι η δημιουργία αληθοφανούς σκιάς. Λόγω του ότι τα εικονικά στοιχεία του AR πρέπει να συμβαδίζουν με τα στοιχεία του πραγματικού περιβάλλοντος, πρέπει όλα τα χαρακτηριστικά τους να είναι αληθοφανή. Ένα από τα στοιχεία που πρέπει να ληφθούν υπ' όψη είναι η σκίαση του αντικειμένου, η οποία πρέπει να συνάδει με την σκίαση των αντικειμένων στην πραγματικότητα. Επίσης, η σκιά πρέπει να εμφανίζεται και σε σημεία του πραγματικού κόσμου, εάν πρέπει να σκιάζονται.

* 1. **Σκοπός ΑΔΕ**

Ο σκοπός μας είναι να δημιουργήσουμε ένα πρόγραμμα το οποίο να μπορεί με το να του εισάγουμε σαν είσοδο μια εικόνα, να μπορεί να αναγνωρίσει πού βρίσκεται η φωτεινή της πηγή, έτσι ώστε στο μέλλον, να μπορεί να χρησιμοποιηθεί από εφαρμογές AR για τον σωστό φωτισμό και σκίαση του δημιουργημένου αντικειμένου.

Θεωρούμε ότι η φωτεινή πηγή στην εικόνα είναι μία και ότι είναι αρκετή για να φωτίζεται ολόκληρο το περιβάλλον του αντικειμένου.

* 1. **Περίγραμμα ΑΔΕ**

Σε αυτή την εργασία, θα ασχοληθούμε με την αναγνώριση του φωτός σε τεχνικό περιβάλλον, θα προσπαθήσουμε, δηλαδή, να ανιχνεύσουμε την πηγή φωτός από μια τεχνητή εικόνα, όπως αυτή φωτίζεται μέσω της μηχανής που θα την δημιουργήσουμε, στην περίπτωσή μας, στη Unity.

Οι ομάδες φωτογραφιών που θα τραβηχτούν θα μπορούν να χωριστούν με βάση τριών από τα χαρακτηριστικά τους: Τον αριθμό των διανυσμάτων των μοιρών που βρισκόταν η φωτεινή πηγή, το σχήμα ή τα σχήματα των αντικειμένων που τραβηχτήκαν, και την ύπαρξη ή όχι textures στην εικόνα.

Θα πάρουμε αρκετά στοιχεία, έτσι ώστε να μπορούμε να τα περάσουμε σε ένα νευρωνικό δίκτυο, το οποίο θα μάθει τις ομάδες των φωτογραφιών, και θα μπορεί να προβλέψει με τις νέες φωτογραφίες σε ποια κατηγορία ανήκουν.

**Κεφάλαιο 2**

**Προηγούμενη Εργασία**

2.1 Προηγούμενα Αποτελέσματα 4

2.2 Νέα Προσέγγιση 5

* 1. **Προηγούμενα Αποτελέσματα**

Υπάρχουν πολλές έρευνες οι οποίες αφοσιώθηκαν στην ανίχνευση της πηγής φωτισμού, και συγκεκριμένα στην αναγνώριση σκιών. Μια από τις εργασίες αυτές[1] αναλύει τις σκιές της φωτογραφίας, χωρίζοντας τα pixels σε τρείς κατηγορίες: στα pixels του ουρανού, στα pixels του εδάφους και στα pixels των κάθετων επιφανειών. Αυτό το πετυχαίνει με το να αναλύσει τα pixels της φωτογραφίας, και έχοντας δεδομένη την γωνιά της φωτογραφικής σε σχέση με τον ήλιο, και την ύπαρξη ή όχι του ήλιου. Στην συνέχεια, αφού κάνει τον διαχωρισμό, αναγνωρίζει σε ποια τοποθεσία θα μπει το εικονικό αντικείμενο.

Μία άλλη έρευνα που χρησιμοποιήθηκε για αναγνώριση σκιάς[2], χρησιμοποίησε μια άλλη προσέγγιση. Παίρνοντας σαν δεδομένα εικόνες από δορυφόρο, μετέτρεπε τις φωτογραφίες από κλίμακα RGB σε κλίμακα HSV. Αναγνωρίζοντας ποια είναι τα κτήρια, και αποκλείοντας τα pixels που αναγνώριζε σαν δρόμους και βλάστηση, υπολόγιζε τις τιμές των σκιών. Στην συνέχεια, "ζωγραφίζει" τις σκιές σαν ένα από τα περιβαλλόμενα pixels του εδάφους. Η φωτογραφία, μετά επανέρχεται από HSV σε RGB.

Και οι δύο εργασίες ανιχνεύουν επιτυχημένα τις σκιές, με πολύ θετικά αποτελέσματα. Παρ' όλα αυτά, οι αλγόριθμοι αυτοί δεν μπορεί να χρησιμοποιηθούν σε πραγματικό χρόνο, διότι ο χρόνος επεξεργασίας της εικόνας είναι αρκετά μεγάλος. Γι' αυτό, σε αυτή την έρευνα θα δοκιμάσουμε μια άλλη προσέγγιση.

* 1. **Νέα Προσέγγιση**

Οι προηγούμενες εργασίες είχαν σχετικά μικρό χρόνο επεξεργασίας, με πολύ υψηλό ποσοστό επιτυχίας. Όμως, λόγω του χρόνου προεπεξεργασίας, δεν μπορούμε να χρησιμοποιήσουμε κάποια από τις δύο εργασίες ως βασική, λόγω του ότι το πρόγραμμά μας πρέπει να εξάγει αποτελέσματα σε πραγματικό χρόνο. Γι' αυτό, ο τρόπος υλοποίησης αυτού του προγράμματος, πρέπει, εκτός από αποδοτικός και ακριβής, να είναι και γρήγορος.

Η προσέγγιση που θα επιχειρήσουμε σε αυτή την εργασία, είναι με την εκπαίδευση ενός νευρωνικού δικτύου, το οποίο θα παίρνει σαν δεδομένα αρκετές εικόνες που να έχουν διάφορες σκιάσεις, και στην συνέχεια να μπορεί να προβλέψει από ποια κατεύθυνση έρχεται το φώς.

Ο λόγος που επιλέξαμε σαν προσέγγιση εκπαίδευση νευρωνικού δικτύου, είναι επειδή στις προηγούμενες δύο εργασίες, το μεγαλύτερο πρόβλημα που αντιμετωπίζαμε, ήταν ο χρόνος προεπεξεργασίας των δεδομένων. Στα νευρωνικά δίκτυα, η προεπεξεργασία που δέχονται τα δίκτυα, είναι η εκπαίδευσή τους, η οποία μόλις ολοκληρωθεί, δεν χρειάζεται να εκτελείται για κάθε φωτογραφία που θα βάζουμε στο πρόγραμμα ως είσοδο.

Ένας άλλος λόγος που επιλέξαμε τα νευρωνικά δίκτυα, είναι διότι, σε σύγκριση με τις προηγούμενες δύο εργασίες, ο χρόνος εξαγωγής αποτελεσμάτων είναι πολύ μικρός, πράγμα το οποίο μας προσελκύει, λόγω της ανάγκης δημιουργίας εφαρμογής πραγματικού χρόνου.

Τέλος, θα χρησιμοποιήσουμε νευρωνικά δίκτυα, διότι, όπως και οι προηγούμενες δύο εργασίες που μελετήσαμε, έτσι και αυτή μπορεί να γενικευτεί σε πολύ περισσότερες φωτογραφίες με την ίδια εκπαίδευση των προηγούμενων φωτογραφιών. Με λίγα λόγια, θα μπορούμε να εξορύξουμε επιτυχημένα αποτελέσματα από ένα δίκτυο όπου οι φωτογραφίες που χρησιμοποιήθηκαν για εκπαίδευση ήταν διαφορετικές από αυτές που εισάγαμε στο πρόγραμμα.

**Κεφάλαιο 3**

**Νευρωνικά Δίκτυα**

3.1 Ιστορική Αναδρομή 7

3.2 Είδη Νευρωνικών Δικτύων 7

3.3 Νευρωνικά Δίκτυα Perceptron 8

3.4 Νευρωνικά Δίκτυα Kohonen 11

3.5 Νευρωνικά Δίκτυα Με Λήψη Αποφάσεων Markov 13

3.6 Convolutional Νευρωνικά Δίκτυα 15

3.7 Επιλογή Νευρωνικού Δικτύου 18

3.8 Επιλογή Εφαρμογής για Επίλυση Προβλήματος 19

* 1. **Ιστορική Αναδρομή**

Πριν αρχίσουμε με την εργασία, αξίζει να κάνουμε μια ιστορική αναδρομή στις τεχνολογίες των νευρωνικών δικτύων. Νευρωνικά δίκτυα είναι μία δομή επεξεργασίας πληροφοριών. Η δομή αυτή εμπνεύστηκε από τον τρόπο που δουλεύουν τα βιολογικά συστήματα νευρώνων, όπως ο εγκέφαλος. Τα νευρωνικά δίκτυα σχεδιάζονται για την εκπλήρωση συγκεκριμένων εργασιών, και μαθαίνουν τα επιθυμητά αποτελέσματα που πρέπει να εξάγουν μέσω μιας διαδικασίας εκπαίδευσής.

Τα νευρωνικά δίκτυα, σαν έννοια, σχηματίστηκαν το 1943 από τον νευροφυσικό Warren McCulloch και τον επιστήμονα της λογικής Walter Pits[3], αλλά δεν υπήρχε η τεχνολογία να τα δοκιμάσουν στην πράξη. Η λογική τους, ήταν η μίμηση της λειτουργίας των νευρώνων του εγκεφάλου, και με τον τρόπο με τον οποίο οι νευρώνες στέλνουν συνάψεις στους άλλους γειτονικούς νευρώνες του εγκεφάλου.

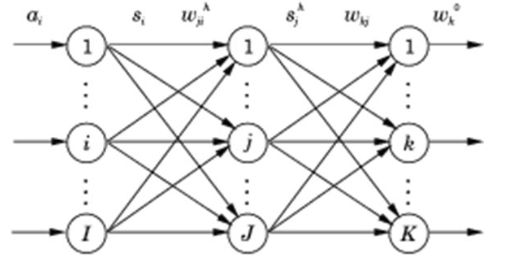
* 1. **Είδη Νευρωνικών Δικτύων**

Παρ' όλο που τα νευρωνικά δίκτυα χρησιμοποιούνται για επίλυση συγκεκριμένων προβλημάτων, μπορούμε να τα χωρίσουμε σε 4 βασικές κατηγορίες: Στα νευρωνικά δίκτυα επιβλεπόμενης μάθησης, στα νευρωνικά δίκτυα μη επιβλεπόμενης μάθησης, στα νευρωνικά δίκτυα ενισχυτικής μάθησης, και τέλος, στα convolutional νευρωνικά δίκτυα. Το κάθε ένα από τα δίκτυα θα μπορούσε να χρησιμοποιηθεί για την εκπαίδευση του δικτύου μας, γι' αυτό θα αναλύσουμε την κάθε κατηγορία για να δούμε ποιο απ' αυτά είναι το ιδανικότερο για την επίλυση του προβλήματός μας.

* 1. **Νευρωνικά Δίκτυα Perceptron**

Πρώτα, ας δούμε πώς δουλεύουν τα δίκτυα επιβλεπόμενης μάθησης, και συγκεκριμένα, το delta rule (Perceptron). Για να δουλέψει το δίκτυο αυτό, πρέπει να ξέρουμε από πριν τι τιμές πρέπει να εξάγουμε με τα αποτελέσματα.

Το δίκτυο αυτό, το οποίο είναι εύκολο να το σκεφτούμε σαν ένα γράφο, αποτελείται από 3 μέρη: τους νευρώνες εισόδου, τους κρυφούς νευρώνες, και τους νευρώνες εξόδου. Όλοι οι νευρώνες εισόδου έχουν ακμές που ενώνονται με όλους τους κρυφούς νευρώνες, και όλοι οι κρυφοί νευρώνες, έχουν ακμές που ενώνονται με όλους τους νευρώνες εξόδου.



*Σχήμα 3.1 - Εικονογραφημένη Παρουσίαση δικτύου Delta Rule (Perceptron)*

Αρχικά, όλες οι ακμές παίρνουν ένα τυχαίο βάρος από 0 (μη συμπεριλαμβανομένου) μέχρι και 1. Στην συνέχεια, το δίκτυό μας διαβάζει τις I τιμές εισόδου του διανύσματος α, και τις και τις εισχωρεί στους I νευρώνες εισόδου του δικτύου, μια τιμή σε κάθε νευρώνα.

Όπου αi το i-οστό στοιχείο εισόδου, το οποίο έχει τιμή από 0 μέχρι και 1, και ο i-οστός νευρώνας εισόδου. Στη συνέχεια, υπολογίζονται οι τιμές των J κρυφών νευρώνων, με το να προσθέτουμε το γινόμενο της τιμής του κάθε νευρώνα με το βάρος της ακμής που ενώνεται με τον νευρώνα ως εξής

Όπου το άθροισμα του j-οστού κρυφού νευρώνα του δικτύου και η i-οστή ακμή του νευρώνα εισόδου που ενώνεται με το j-οστό κρυφό νευρώνα. Στην συνέχεια, για να υπολογίσουμε εάν θα στείλουμε σήμα ή όχι, πρέπει να περάσουμε τα δεδομένα από την συνάρτηση ενεργοποίησης, έτσι ώστε το αποτέλεσμα του νευρώνα να είναι από 0 μέχρι 1. Ο τύπος ενεργοποίησης είναι

Όπου ο j-οστός κρυφός νευρώνας. Αφού υπολογίσουμε τις τιμές όλων των κρυφών νευρώνων, με παρόμοιο τρόπο υπολογίζουμε τις τιμές των K εξωτερικών νευρώνων

Όπου το άθροισμα του k-οστού νευρώνα εξόδου και η k-οστή ακμή του κρυφού νευρώνα που ενώνεται με το k-οστό νευρώνα εξόδου. Όπως και πριν, πρέπει οι τιμές εξόδου να είναι από το 0 μέχρι το 1, γι' αυτό, έχουμε

Όπου ο k-οστός νευρώνας εξόδου του δικτύου.

Στην συνέχεια, το δίκτυο μελετά τις τιμές εξόδου, και διορθώνει τα βάρη των ακμών, ανάλογα με την διαφορά τις τιμής κάθε νευρώνα και της διαφοράς της τιμής του αναμενόμενου αποτελέσματος. Για να βρούμε όμως κατά πόσο θα αλλάξει το κάθε βάρος, πρέπει να υπολογίσουμε το δ, το οποίο υπολογίζεται

Όπου bk το αποτέλεσμα που αναμέναμε από τον k-οστό νευρώνα εξόδου. Παρόμοια, για τους κρυφούς νευρώνες, βρίσκουμε το δ με τον τύπο

Στη συνέχεια, αφού βρούμε τα δ, πρέπει να αλλάξουμε τα βάρη. Η αλλαγή των βαρών γίνεται για τα βάρη των κρυφών νευρώνων. Πρώτα πρέπει να αλλάξουμε τα βάρη των ακμών των κρυφών νευρώνων. Αυτό γίνεται με τον ακόλουθο τύπο

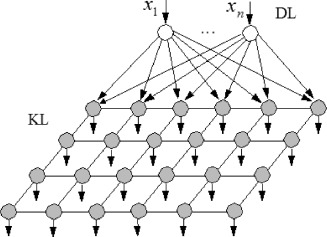
Όπου η ο ρυθμός μάθησης, ένας σταθερός αριθμός από το 0 μέχρι το 1 ο οποίος βοηθά την σωστή ρύθμιση των βαρών. Επίσης, ρυθμίζουμε τα βάρη των ακμών των νευρώνων εισόδου με παρόμοιο τρόπο

Η διαδικασία επαναλαμβάνεται μέχρι να διαβαστούν όλα τα στοιχεία εκμάθησης. Στη συνέχεια, εισάγουμε στο νευρωνικό δίκτυο τα στοιχεία ελέγχου, όπου θα συμβεί η ίδια διαδικασία με τα στοιχεία εκμάθησης, χωρίς ωστόσο να αλλάζουν τα βάρη μετά από κάθε εκτέλεση. Αυτό γίνεται για να δούμε εάν το νευρωνικό μας δίκτυο είναι στο σωστό δρόμο για να εκπαιδευτεί. Η διαδικασία εκπαίδευσης - ελέγχου, γίνεται για όσες επαναλήψεις θεωρηθούν ικανοποιητικές για την εκπαίδευση του δικτύου.

* 1. **Νευρωνικά Δίκτυα Kohonen**

Ένα από τα πιο γνωστά δίκτυα μη επιβλεπόμενης μάθησης, είναι τα νευρωνικά δίκτυα αυτό-οργάνωσης Kohonen[4]. Το δίκτυο αυτό, οργανώνει τα δεδομένα εισόδου από μόνο του, χωρίς να υπάρχει κάποιου είδος ένδειξης ορθότητας. Αυτό συμβαίνει, διότι το δίκτυο ομαδοποιεί τα στοιχεία εισόδου του με βάση τις κατηγορίες το οποίο το ίδιο δημιουργεί.

Το δίκτυο αυτό, αποτελείται από δύο μέρη: τους νευρώνες εισόδου, και τους νευρώνες εξόδου. Ο κάθε νευρώνας εισόδου, ενώνεται με ακμές με όλους τους νευρώνες εξόδου, και η κάθε ακμή, όπως και στα νευρωνικά δίκτυα perceptron, έχουν ένα βάρος. Ο κάθε νευρώνας εξόδου, έχει συντεταγμένες. Θα μπορούσαμε να χαρακτηρίσουμε τους νευρώνες εξόδου σαν ένα δισδιάστατο πίνακα με νευρώνες.



*Σχήμα 3.2 - Εικονογραφημένη Παρουσίαση δικτύου Kohonen*

Αρχικά, όπως και στα νευρωνικά δίκτυα perceptron, οι ακμές του δικτύου μας αρχικοποιούνται με μια τιμή από το -1 μέχρι το 1, χωρίς να συμπεριλαμβάνεται το 0. Στην συνέχεια, αρχικοποιούμε τους νευρώνες εισόδου με τις τιμές εισόδου μας

Όπου αi το i-οστό στοιχείο των I εισόδου και xi ο i-οστός νευρώνας των I εισόδου. Στην συνέχεια, υπολογίζουμε την απόσταση του κάθε νευρώνα, αθροίζοντας το τετράγωνο της διαφοράς της τιμής των νευρώνων εισόδου με το αντίστοιχο βάρος της κάθε ακμές.

Όπου ο νευρώνας εξόδου με συντεταγμένες [j,k] και το βάρος της ακμής του νευρώνα εισόδου προς τον νευρώνα . Αφού υπολογιστούν οι αποστάσεις όλων των νευρώνων εξόδου, δηλαδή όλα τα , υπολογίζουμε τον νευρώνα-νικητή, τον νευρώνα, δηλαδή με την μικρότερη απόσταση. Για τον νευρώνα αυτό, και για κάθε νευρώνα σε απόσταση R, αλλάζουμε τα βάρη των νευρώνων εξόδου

Όπου και το βάρος πριν και μετά την διόρθωση, ο ρυθμός μάθησης και η απόσταση των συντεταγμένων του νευρώνα από τον νευρώνα - νικητή. Αυτή η διαδικασία γίνεται για όλα τα δεδομένα εισόδου.

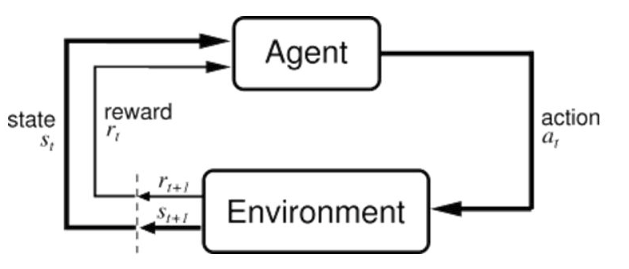
Στη συνέχεια, πριν περάσουμε στην επόμενη επανάληψη, πρέπει να μειώσουμε την εμβέλεια του R και τον ρυθμό μάθησης η. Το R αλλάζει, για να μπορέσουν κατά την εκτέλεση της εκμάθησης, οι ομάδες να γίνονται πιο συγκεκριμένες και μικρές, έτσι ώστε να μειωθεί η πιθανότητα επικάλυψης μεταξύ ομάδων. Το η μειώνεται, διότι με την πάροδο του χρόνου δεν χρειαζόμαστε μεγάλη αλλαγή των βαρών.

Ο αλγόριθμος των νευρωνικών δικτύων Kohonen επαναλαμβάνεται όσες φορές όσο θεωρήσει ο χρήστης.

* 1. **Νευρωνικά Δίκτυα Με Λήψη Αποφάσεων Markov**

Η τρίτη κατηγορία που θα ασχοληθούμε, είναι τα νευρωνικά δίκτυα ενισχυτικής μάθησης, και ένα καλό και διαδεδομένο παράδειγμα, είναι τα νευρωνικά δίκτυα με λήψη αποφάσεων Markov[5]. Τα νευρωνικά δίκτυα, αρχικά, βρίσκονται σε μια κατάσταση. Με κάθε τους πράξη, παίρνουν είτε μία ανταμοιβή, είτε μία τιμωρία, και μία ενημέρωση της κατάστασής τους.

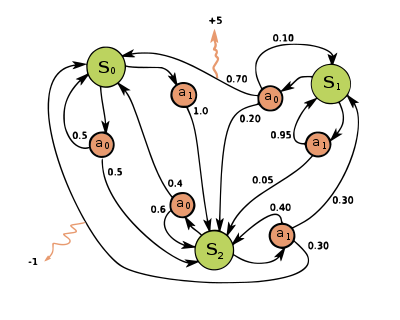
Μέσω της ανατροφοδότησης που του παρέχει ο αλγόριθμος, και σύμφωνα με τις τιμές που λαμβάνουν, και με την απόσταση που απομένει από την καινούρια κατάσταση μέχρι να βρεθεί στον στόχο του, μαθαίνει κατά πόσο η ενέργεια που έπραξε ήταν σωστή.



*Σχήμα 3.3 - Εικονογραφημένη Παρουσίαση Αλγόριθμου Λήψης Απόφασης Markov*

Η διαδικασία απόφασης Markov μπορεί να χαρακτηριστεί ως μια πενταπλή πλειάδα

Όπου ένα πεπερασμένο σύνολο από καταστάσεις, το πεπερασμένο σύνολο των ενεργειών από το οποίο η κατάσταση s μπορεί να προβεί, η πιθανότητα η ενέργεια α στην κατάσταση s να σε οδηγήσει στην κατάσταση s', η επιβράβευση της ενέργειας α από την κατάσταση s προς την κατάσταση s', και γ ο παράγοντας έκπτωσης, που αντιπροσωπεύει την διαφορά της σημαντικότητας μεταξύ μελλοντικών επιβραβεύσεων και τωρινών επιβραβεύσεων. Το γ παίρνει τιμές από 0 μέχρι και 1.



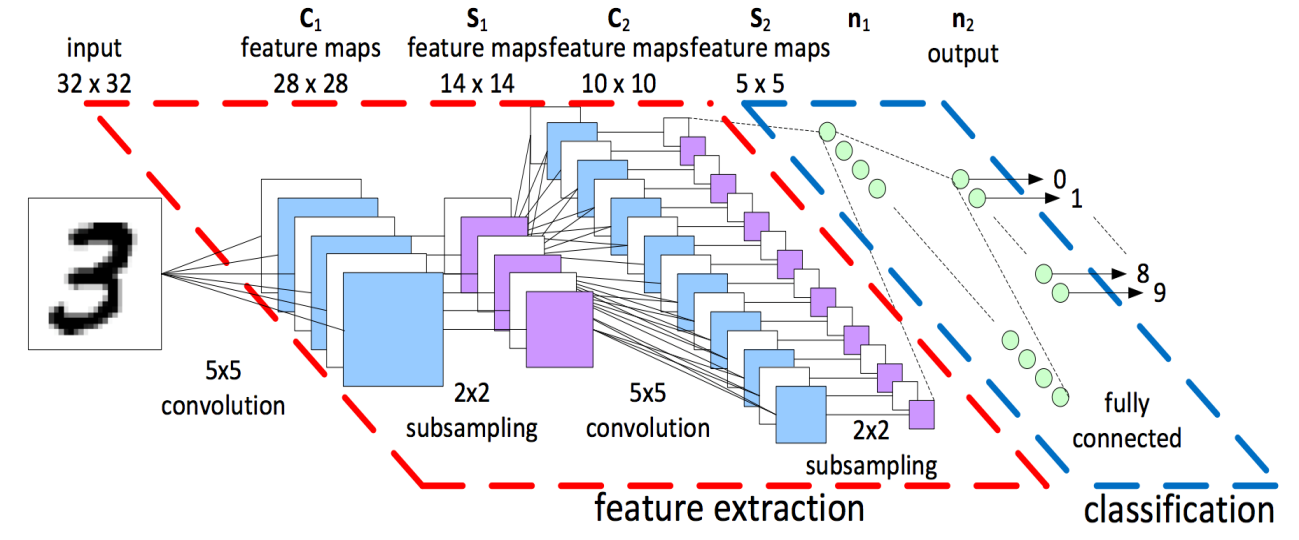
*Σχήμα 3.4 - Εικονογραφημένη Παρουσίαση ενός παραδείγματος διαδικασίας απόφασης Markov, με 3 καταστάσεις, 2 ενέργειες και 2 επιβραβεύσεις*

Οι αλγόριθμοι ενισχυτικής μάθησης χρησιμοποιούνται σε πληθώρα προβλημάτων, αλλά κυρίως σε προβλήματα εκμάθησης ρομπότ και άλλων ΑΙ. Για να μπορεί, όμως, να μάθει το δίκτυο, και για να μπορέσουμε να υλοποιήσουμε το δίκτυο, προϋποθέτετε πεπερασμένος αριθμός καταστάσεων, και πεπερασμένος αριθμός ενεργειών ανά κατάσταση κάθε φορά.

* 1. **Convolutional Νευρωνικά Δίκτυα**

Η τελευταία κατηγορία νευρωνικών δικτύων, που είναι και η πιο σύγχρονη, είναι τα convolutional νευρωνικά δίκτυα[6]. Η ειδικότητα των convolutional νευρωνικών δικτύων, είναι η αναγνώριση και κατηγοριοποίηση προτύπων. Γι' αυτό τον λόγο, χρησιμοποιείται ευρεία στην ανάλυση φωτογραφιών, διότι, αναλύοντάς τις φωτογραφίες, μπορεί να εξάγει δεδομένα σαν αριθμούς, τους οποίους μπορούμε να αναλύσουμε.

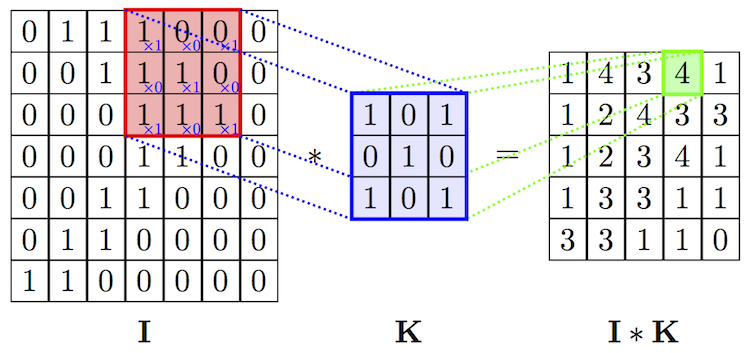
Το Convolutional νευρωνικό δίκτυο μπορεί να χωριστεί σε τρία μέρη: Τους νευρώνες εισόδου, την κρυφή στρώση, και τους νευρώνες εξόδου. Με την σειρά της, η κρυφή στρώση μπορεί να χωριστεί περεταίρω στα convolutional layers, στα pooling layers, στα fully connected layers και στα normalization layers.



*Σχήμα 3.5 - Εικονογραφημένη Παρουσίαση Ενός πλήρως υλοποιημένου Convolutional Neural Network*

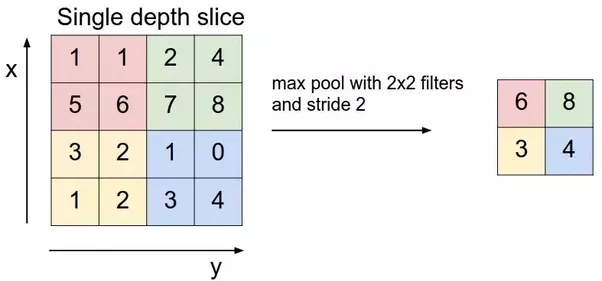
Σαν είσοδο, το Convolutional νευρωνικό δίκτυο, δέχεται ένα δισδιάστατο πίνακα. Ο σκοπός μας, είναι να περάσει από πολλαπλές διαδικασίες convolution, έτσι ώστε να ομαδοποιήσουμε καλύτερα τα δεδομένα του.

Το Convolution layer παίρνει μερικά φίλτρα, τα οποία φίλτρα είναι μικροί δισδιάστατοι τετράγωνοι πίνακες, οι οποίοι περιέχουν συγκεκριμένα μοτίβα των δεδομένων εισόδων, προς σύγκριση και ομαδοποίηση. Με το διάβασμα των εισόδων, και μετά την δημιουργία των Convolutions, το δίκτυο, με κάποιο συγκεκριμένο αλγόριθμο, δημιουργεί ένα άλλο πίνακα, ο οποίος είναι πιο μικρός από τον αρχικό. Αυτή η διαδικασία λέγεται pooling.



*Σχήμα 3.6 - Παράδειγμα Convolution μέσω dot product. Το δίκτυο παίρνει σαν είσοδο όλους τους υποπίνακες 3x3, τους πολλαπλασιάζει με το φίλτρο Κ, εκτελεί το dot product τους και το αποθηκεύει σε ένα πίνακα μικρότερου μεγέθους.*

Υπάρχουν μερικοί αλγόριθμοι pooling οι οποίοι μπορεί να υπερβούν τις τιμές της αρχικής εικόνας, είτε στα θετικά όρια, είτε στα αρνητικά. Για αυτές τις περιπτώσεις, την διαδικασία convolution - pooling πρέπει να διαδεχθεί η διαδικασία κανονικοποίησης (normalization layer), όπου κανονικοποιεί τις τιμές του pooling στις τιμές της πηγαίας εικόνας.



*Σχήμα 3.7 - Παράδειγμα pooling με Max Pooling. Το δίκτυο χωρίζει τον πίνακα, και παίρνει από κάθε κομμάτι του την μεγαλύτερη τιμή.*

Η διαδικασία Convolution και Pooling γίνεται για όσες φορές κριθεί αναγκαίο. Εννοείται πως για κάθε επιπλέον κύκλο convolution-pooling που γίνεται σε κάθε εικόνα, τα μοτίβα που μπορούν να ανιχνευτούν και να κατηγοριοποιηθούν είναι πιο ειδικά και πολύπλοκα.

Μετά την εκτέλεση όλων των Convolution και Pooling, σειρά έχει το fully connected layer. Ο δισδιάστατος πίνακας που μας επιστρέφει η τελευταία εκτέλεση του Convolution-Pooling, θα είναι η είσοδος του fully connected layer, που δεν είναι τίποτα άλλο από ένα δίκτυο Perceptron(βλέπε κεφάλαιο 3.3). Το δίκτυό μας εκπαιδεύεται όπως προαναφέραμε. Σαν έξοδο, θα λάβουμε ένα διάνυσμα μήκους X, όπου Χ οι ομάδες οι οποίες προσπαθούμε να εξετάσουμε. Με βάση τα λάθη στις κατηγορίες, το fully connected layer εκπαιδεύει τους νευρώνες του, έχοντας σαν 1 την κατηγορία στην οποία θα έπρεπε να "επιλεχτεί", και 0 τις υπόλοιπες. Η έξοδος του Perceptron, είναι η έξοδος ολόκληρου του Convolutional νευρωνικού δικτύου.

Αυτή η διαδικασία γίνεται για όλα τα δεδομένα εισόδου, και για όσες επαναλήψεις επιλέξουμε, ανάλογα με τις πολυπλοκότητες των εικόνων, και τις ανάγκες της εργασίας μας.

* 1. **Επιλογή Νευρωνικού Δικτύου**

Με την ανάλυση όλων των νευρωνικών δικτύων, και έχοντας υπ' όψη την εφαρμογή που θέλουμε να υλοποιήσουμε, ήρθε η ώρα να επιλέξουμε το δίκτυο που θα μας βοηθήσει περισσότερο στην υλοποίηση του στόχου μας.

Η ιδέα είναι η εξής: Η εφαρμογή, έχοντας φυλαγμένα τα μοντέλα που θέλει να προβάλει στον πραγματικό κόσμο, θα προσπαθεί να τα βάλει σε λογικούς τόπους. Αναγνωρίζοντας την πηγή φωτός, θα επιλέγει την κατάλληλη σκιά να βάλει. Οι επιλογές από σκιές θα είναι πεπερασμένες, και θα είναι όλες φυλαγμένες στον υπολογιστή. Οι σκιές που θα αποθηκεύουμε για κάθε αντικείμενο θα είναι πεπερασμένες, έστω μια για κάθε 22.5o.

Η απορία που προκύπτει, είναι το εάν είναι δυνατό να δημιουργηθεί μια τέτοια εφαρμογή, η αναγνώριση του φωτισμού ως κατηγορίες μοιρών φωτός. Λόγω του ότι η προσέγγιση προς την λύση είναι πρόβλημα κατηγοριοποίησης, θα ήταν εξαιρετικά βοηθητικό να επιλέξουμε ένα νευρωνικό δίκτυο κατηγοριοποίησης.

Από όλες τις προαναφερθέντες επιλογές κατηγοριοποίησης, τα Convolutional Νευρωνικά δίκτυα θα είναι τα ιδανικότερα για την κοντινότερη προσέγγιση του στόχου μας. Ένας από τους λόγους που επιλέχτηκε, είναι διότι θα γνωρίζουμε εξ' αρχής σε ποια κατηγορία θα ανήκει η κάθε εικόνα εκπαίδευσης του δικτύου, κάνοντας την αλλαγή των βαρών των ακμών του fully connected layer πιο ακριβή.

Άλλος ένας λόγος που επιλέξαμε τα Convolutional Νευρωνικά Δίκτυα, και μάλλον ο κυριότερος, είναι το γεγονός ότι τα δεδομένα εισόδου μας είναι φωτογραφίες, δεδομένα τα οποία τα Convolutional νευρωνικά δίκτυα ειδικεύονται. Παρ' όλο που ο χρόνος εκμάθησης είναι σχετικά μεγάλος, ο χρόνος υπολογισμού της κατηγορίας μιας φωτογραφίας είναι πολύ μικρός, και θεωρητικά θα μπορούσε να χρησιμοποιηθεί για την εφαρμογή μας.

* 1. **Επιλογή Εφαρμογής Επίλυσης Προβλήματος**

Αφού ξεκαθαρίσαμε την προσέγγιση που θα ακολουθήσουμε, το τελευταίο στάδιο που χρειάζεται να αναφερθεί είναι οι εφαρμογές και τα εργαλεία που θα χρησιμοποιήσουμε για την υλοποίηση του περιβάλλοντος μας. Η γλώσσα προγραμματισμού που επιλέχτηκε για την υλοποίηση του Convolutional Νευρωνικού Δικτύου, είναι η Python, με την βοήθεια της διεπαφής Anaconda, για ευκολία εγκατάστασης plug-ins που μας βοηθούν στην λειτουργία του νευρωνικού μας δικτύου.

Ένα άλλο εργαλείο που θα χρησιμοποιήσουμε για την υλοποίηση του νευρωνικού δικτύου, θα είναι η βιβλιοθήκη TensorFlow[7], η οποία υλοποιήθηκε για την δημιουργία μεγάλων προγραμμάτων με Convolutional Νευρωνικά Δίκτυα. Ο κύριος λόγος που θα το χρησιμοποιήσουμε, όμως, είναι διότι δεν θα χρειαστεί να υλοποιήσουμε το νευρωνικό δίκτυο από την αρχή, διότι παρέχει έτοιμες συναρτήσεις οι οποίες έχουν υλοποιημένα όλα τα απαραίτητα συστατικά για την δημιουργία του νευρωνικού δικτύου μας.

Τέλος, για την εξαγωγή των φωτογραφιών, θα χρησιμοποιήσουμε το Unity Game Engine, για να παράγουμε φωτογραφίες του εικονικού περιβάλλοντος, ελέγχοντας έτσι την προσέγγιση εισαγωγής στοιχείου από τον εικονικό κόσμο σε παρόμοιο εικονικό κόσμο.

**Κεφάλαιο 4**

**Θεωρητική Λύση**

4.1 Θεωρία Λύσης 21

4.2 Τρόπος Υλοποίησης 22

4.3 Αναμενόμενα Αποτελέσματα 25

* 1. **Θεωρία Λύσης**

Αφού τελειώσαμε με την εύρεση του ιδανικότερου νευρωνικού δικτύου, σε αυτό το κομμάτι της εργασίας θα ασχοληθούμε με το θεωρητικό κομμάτι της έρευνας αυτής, να ορίσουμε πότε θα θεωρείται επιτυχημένη, και να καθορίσουμε καλύτερα τι αποτελέσματα πληρούν στόχους μας.

Κατ' αρχήν, θέλουμε το δίκτυό μας να εκτελείται σε πραγματικό χρόνο, άρα χρειαζόμαστε, μετά την εκπαίδευση του δικτύου, η "πρόβλεψη" της κατηγορίας της εικόνας να είναι γρήγορη. Γρήγορο, θα θεωρήσουμε το δίκτυο το οποίο μπορεί να προβλέψει την σκίαση του αντικειμένου σε λιγότερο από 800 ms

Επίσης, το πρόγραμμα πρέπει να προβλέπει επιτυχώς την κατηγορία που ανήκει η εικόνα. Θα ορίσουμε επιτυχές την πρόβλεψη που έχει ποσοστό επιτυχίας πάνω από 85%.

Τέλος, το πρόγραμμά μας πρέπει να είναι εύστοχο, δηλαδή η πρόβλεψη της σωστής κατηγορίας να ανέρχεται με σιγουριά 100%. Για να θεωρηθεί το πρόγραμμά μας εύστοχο, θα πρέπει να υπάρχει το πολύ 10% απόκλιση.

* 1. **Τρόπος Υλοποίησης**

Λόγω του ότι το δίκτυο μας θα δέχεται εικόνες από το πραγματικό περιβάλλον, στην εργασία αυτή, για να δούμε την πρακτική του μεριά, θα πάρουμε τις φωτογραφίες εισόδου από ένα εικονικό περιβάλλον, που θα είναι και πιο γνώριμο στον υπολογιστή. Θα προσπαθήσουμε να δούμε, δηλαδή, κατά πόσο ο υπολογιστής μπορεί να αναγνωρίσει τον φωτισμό σε εικονικό περιβάλλον, για να δούμε εάν στο μέλλον θα μπορέσει να αναγνωριστεί ο φωτισμός και σε πραγματικό περιβάλλον.

Το εικονικό περιβάλλον, πρέπει να δημιουργηθεί σε μία εφαρμογή η οποία να μπορεί να αναδημιουργήσει με ακρίβεια την συμπεριφορά του φωτός, και ποιο συγκεκριμένα, να αναπαριστά τον τρόπο με τον οποίο σκιάζονται τα αντικείμενα. Η λύση που προτάθηκε, είναι η δημιουργία του κόσμου αυτού στο Unity Game Engine.

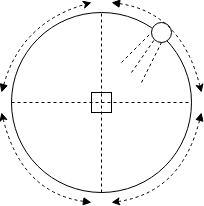
Ο λόγος που επιλέχτηκε το Unity Game Engine, εκτός του ότι πληρεί τις προδιαγραφές ακριβούς αναπαράστασης του φωτός που επεξηγήσαμε, είναι η ευκολία δημιουργίας του εικονικού κόσμου, και ο προγραμματισμός συμπεριφοράς του φωτισμού είναι στον μεγαλύτερο βαθμό έτοιμος. Η ευκολία δημιουργίας του περιβάλλοντος μας ενδιαφέρει, διότι η δημιουργία του περιβάλλοντος δεν είναι το κύριο κομμάτι αυτής της εργασίας.

Μετά την δημιουργία του περιβάλλοντος, θα πρέπει να δημιουργήσουμε τον φωτισμό. Η φωτεινή πηγή του κόσμου θα περιστρέφεται σε ένα κύκλο γύρω από το περιβάλλον και προς το κέντρο του περιβάλλοντος. Ο λόγος πίσω απ' αυτή την απόφαση, είναι να δούμε, εάν το Convolutional νευρωνικό δίκτυο μπορεί να αναγνωρίσει την κατηγορία της εικόνας σε ένα περιβάλλον που η πηγή φωτός δείχνει πάντα προς το αντικείμενο.

Στην συνέχεια, θα δούμε πώς θα δημιουργήσουμε το Convolutional νευρωνικό δίκτυο. Λόγω του ότι το δίκτυο μας θα πρέπει να αναγνωρίζει σκιές, το δίκτυό μας θα πρέπει να αναγνωρίζει τόσο την σκίαση, όσο και την φωτεινότητα του αντικειμένου. Και επειδή προσπαθούμε να το γενικεύσουμε σε όλα τα αντικείμενα και φωτισμούς, το δίκτυό μας θα πρέπει να αναγνωρίζει όλα τα μοτίβα. Γι' αυτό τον λόγο, το δίκτυό μας πρέπει να έχει τουλάχιστο 2 επίπεδα Convolution λόγω της πληθώρας σχημάτων και σκιών που θα εκπαιδευτεί το δίκτυο, αλλά όχι πάνω από 4, γιατί τα μοτίβα που θέλουμε να αναγνωρίσουμε δεν είναι τόσο πολύπλοκα. Επίσης, πάνω από 4 Convolution, θα καταστήσουν την εκπαίδευση του δικτύου πολύ πιο αργή, σε σύγκριση του ποσοστού επιτυχίας που θα λάβουμε.

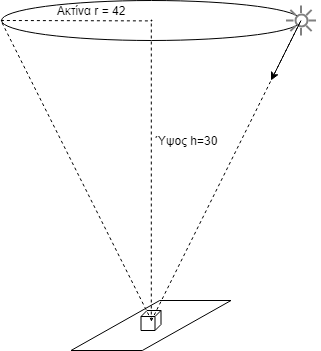
Η μέση λύση, είναι να δημιουργήσουμε 3 Convolutions, τα πρώτα δύο με 32 φίλτρα RGB μεγέθους 3x3, και το τελευταίο με 64 φίλτρα RGB μεγέθους 3x3. Το μέγεθος των φίλτρων, αν και μικρό, μπορεί να αναγνωρίσει πιο λεπτομερή μοτίβα. Η επιλογή 32 φίλτρων στα πρώτα 2 convolutions, είναι για την σωστή μεταφορά των σημαντικών δεδομένων των φωτογραφιών, και η επιλογή 64 φίλτρων στο τέλος, είναι για την εξόρυξη πιο συγκεκριμένης πληροφορίας για να εισαχτεί στο δίκτυο Perceptron.

Για σκοπούς μετρήσεων, οι φωτογραφίες εισόδου θα κατέχουν 3 χαρακτηριστικά. Το πρώτο χαρακτηριστικό, είναι η ύπαρξη texture ή όχι στα δεδομένα μας. Ο λόγος που εισάγουμε textures, είναι για να προσομοιώσουμε ένα αληθοφανές για το δίκτυο περιβάλλον. Το δεύτερο χαρακτηριστικό, είναι οι ομάδες κατηγοριών που θα αναγνωρίζει το δίκτυο. Θα χωρίσουμε το δίκτυο σε δύο κατηγορίες, σε τέσσερεις, οκτώ και τέλος, σε δεκαέξι. Η διαφορά της κάθε κατηγορίας, είναι ο τόπος από τον οποίο προέρχεται ο φωτισμός. Στις δύο κατηγορίες, ο φωτισμός θα έρχεται σε ένα από τα δύο ημικύκλια, στις 4 από ένα από τα 4 τεταρτημόρια του κύκλου και ούτω καθ' εξής. Το τελευταίο χαρακτηριστικό, είναι τα αντικείμενα που εκπαιδεύσαμε το δίκτυο. Αυτά τα δεδομένα μπορεί να είναι μόνο σφαίρες, μόνο κύβοι, μόνο κύλινδροι, ή συνδυασμός αντικειμένων.



*Σχήμα 4.1 - Η κάτοψη του τεχνητού μας περιβάλλοντος, στην περίπτωση που θα έχουμε 4 κατηγορίες. Όταν έχουμε 4 κατηγορίες, ο φωτισμός μπορεί να έρχεται από ένα από τα 4 τεταρτημόρια του κύκλου*

Τα δεδομένα ελέγχου μας, θα έχουν 2 χαρακτηριστικά, την ύπαρξη ή όχι textures από την εικόνα και την ύπαρξη ενός είδους αντικειμένου στις φωτογραφίες ή στην ύπαρξη συνδυασμού αντικειμένων. Σε αντίθεση με τα δεδομένα εισόδου, ο φωτισμός του αντικειμένου δεν θα έχει κάποιον περιορισμό, εκτός από το να βρίσκεται κάπου στον κύκλο και ο φωτισμός να έχει φορά προς το αντικείμενο.



*Σχήμα 4.2 - Γραφική απεικόνιση του τεχνητού μας περιβάλλοντος*

* 1. **Αναμενόμενα Αποτελέσματα**

Μετά την ολοκλήρωση του περιβάλλοντος, θα προβούμε σε μετρήσεις χρόνου για να ελέγξουμε την συμβατότητα του προγράμματός μας σε σχέση με τις ανάγκες των στόχων μας. Το αναμενόμενο αποτέλεσμα του χρόνου εκτέλεσης του ελέγχου, θα πρέπει να είναι μικρό, έτσι ώστε να μπορεί να χρησιμοποιηθεί από την εφαρμογή σε πραγματικό χρόνο. Περιμένουμε τα δεδομένα ελέγχου να μην ξεπερνούν το 1 δευτερόλεπτο, και τα δεδομένα ελέγχου να έχουν ποσοστό επιτυχίας 90% στα δίκτυα που εκπαιδευτήκαν με τα ίδια χαρακτηριστικά.. Το νευρωνικό δίκτυο, θα χρειαστεί περίπου 1 λεπτό για να εκτελέσει ένα ολοκληρωμένο iteration εκπαίδευσης - ελέγχου, δεδομένου ότι θα έχουμε 2000 φωτογραφίες, από τις οποίες τα δύο τρίτα θα χρησιμοποιηθούν για την εκπαίδευση του δικτύου, και το υπόλοιπο ένα τρίτο για τον έλεγχό του. Τα iterations που θα χρησιμοποιηθούν για κάθε δίκτυο θα είναι 500 για όλες τις εικόνες, άρα ο συνολικός χρόνος εκπαίδευσης θα ανέρχεται περίπου στις 8 ώρες.

**Κεφάλαιο 5**

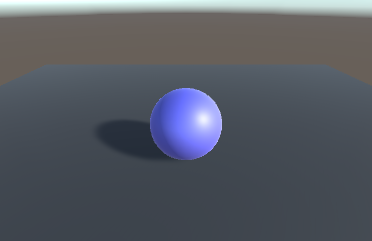
**Στήσιμο Λύσης**

5.1 Στήσιμο Unity 27

5.2 Στήσιμο Νευρωνικού Δικτύου 29

1. **Στήσιμο Unity**

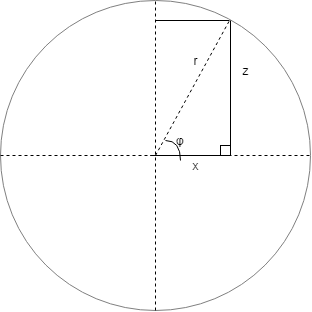
Για να μπορέσουμε να δημιουργήσουμε το δίκτυο, πρέπει πρώτα να μαζέψουμε τις εικόνες εισόδου μας από την μηχανή παιχνιδιών Unity. Πρώτα όμως, πρέπει να δημιουργήσουμε το περιβάλλον μας. Το περιβάλλον μας θα είναι απλό, διότι θα αποτελείται από την camera, την πηγή φωτός, ένα επίπεδο και τέλος το αντικείμενό μας το οποίο θα φωτίζεται και θα παράγει σκιά.



*Σχήμα 5.1 - Ένα τυχαίο δείγμα μιας φωτογραφίας εισόδου*

Αρχικά, τοποθετούμε το επίπεδο και το αντικείμενο που θα φωτογραφίσουμε στην αρχή των αξόνων, και την camera να βλέπει το αντικείμενο μας, όπως στο *Σχήμα 5.1*. Τοποθετούμε την πηγή φωτός στην αρχή των αξόνων, αλλά σε ύψος 30m.

Στην συνέχεια, γράφουμε τον κώδικα με τον οποίο θα κινείται το φώς πριν να βγάλει την φωτογραφία. Το φώς, θα κινείται σε ένα κύκλο. Για έκδοση καλύτερων αποτελεσμάτων, θα έχουμε τον κύκλο σε ακτίνα 42m από το (0,30,0). Για να υπολογιστεί αυτό, θα επιλέξουμε μια τυχαία τιμή για το z, ανάλογα με το τεταρτημόριο που βρίσκεται.



*Σχήμα 5.2 - Γραφική κάτοψη της πηγής φωτός*

Λόγω του ότι το φώς κινείται κυκλικά, μπορούμε να υπολογίσουμε το z μέσω του τριγώνου που δημιουργείται

Όπου Rand(a,b) ένας τυχαίος αριθμός από το a μέχρι το b, r η ακτίνα της πηγής φωτός από το C'(0,30,0), και η μικρότερη και μεγαλύτερη γωνιά που μπορεί να πάρει η παρούσα κατηγορία. Στην συνέχεια, αφού γνωρίζουμε την τιμή του z, και γνωρίζοντας τον τύπο του κύκλου που είναι

τότε μπορούμε να βρούμε την τιμή του x, λύνοντας την εξίσωση ως προς x, έχοντας

Έχοντας υπ' όψη μας ότι για γωνιές φ από 90 μέχρι 270, οι τιμή του x γίνεται αρνητική.

Τέλος, υπολογίζουμε την κατεύθυνση της camera, και τραβάμε την φωτογραφία. Η φωτογραφία είναι τύπου png και έχει διαστάσεις 372 x 241 pixels.

Επαναλαμβάνουμε την διαδικασία, αλλά αυτή τη φορά αντί τα στοιχεία μας να είναι μόνο ένα είδος αντικειμένου, έχουμε πολλά είδη αντικειμένων. Μετά, επαναλαμβάνουμε την διαδικασία, αλλά αυτή τη φορά, όλα τα στοιχεία θα έχουν textures.

Αυτή η διαδικασία γίνεται για όλες τις περιπτώσεις, μέχρι να μαζέψουμε αρκετά δείγματα για κάθε περίπτωση. Στο τέλος, για κάθε μια από τις περιπτώσεις εξόδου, δημιουργούμε 100 δείγματα με τυχαία τοποθεσία του φωτισμού για να δούμε τα ελέγξουμε στο μέλλον.

1. **Στήσιμο Νευρωνικού Δικτύου**

Το νευρωνικό δίκτυο που θα υλοποιήσουμε, θα περιέχει 3 Convolutional Layers. Ο αρχικός μας πίνακας, θα έχει μέγεθος 372 x 241. Μετά το πρώτο convolution των 32 3x3 φίλτρων, θα έχει μέγεθος 369 x 238, μετά το δεύτερο convolution των 32 3x3 φίλτρων θα έχει μέγεθος 366 x 235, και στο τελευταίο convolution των 64 3x3 φίλτρων θα έχει μέγεθος 363 x 232. Στην συνέχεια, μετατρέπει τον 363 x 232 πίνακά μας σε ένα μονοδιάστατο διάνυσμα μεγέθους 84216. Αφού χρησιμοποιήσουμε την μέθοδο ReLU[8], όπου όλοι οι αρνητικοί αριθμοί του διανύσματός μας παίρνουν την τιμή 0, το διάνυσμά μας μπαίνει σε ένα fully connected Perceptron, με 84216 νευρώνες εισόδου, 160000 κρυφούς νευρώνες, και x νευρώνες εξόδου, όπου x οι κατηγορίες που θέλουμε να χωρίσουμε τις φωτογραφίες εξόδου.

Κάθε φορά που διορθώνονται τα βάρη των ακμών, το πρόγραμμά μας αποθηκεύει την κατάσταση που έχει το δίκτυο, έτσι ώστε να μπορεί σε μελλοντικό στάδιο να χρησιμοποιηθεί από την συνάρτηση πρόβλεψης άλλων εικόνων.

**Κεφάλαιο 6**

**Αποτελέσματα**

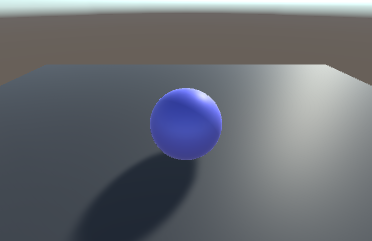
6.1 Δείγματα 31

6.2 Έλεγχος 32

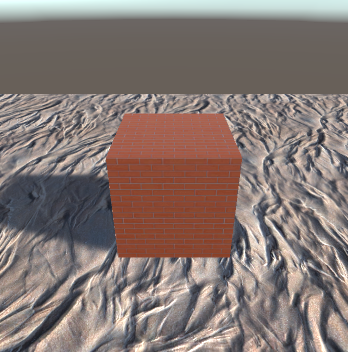
6.3 Σύγκριση Αποτελεσμάτων 36

1. **Δείγματα**

Με την εκτέλεση του προγράμματος στην Unity, μαζέψαμε τις εικόνες όπου θα εκπαιδεύσουμε το δίκτυο. Το δίκτυό μας, είχε ποσοστό επιτυχίας εκπαίδευσης 96.9%, και ποσοστό επιτυχίας ελέγχου 90.89% και validation error 0.0096. Τα νευρωνικά δίκτυα χρειάστηκαν περίπου 2 ώρες το καθ' ένα για να εκπαιδευτούν, και περίπου το 1% του χρόνου αυτού χρησιμοποιούταν για αποθήκευση.



*Σχήμα 6.1 - Δείγμα μιας εικόνας χωρίς textures*



*Σχήμα 6.2 - Δείγμα μιας εικόνας με textures*

1. **Έλεγχος**

Μετά την εκπαίδευση, μετά αρχίσαμε τον έλεγχο του δικτύου, τρέξαμε, δηλαδή στο δίκτυο τις φωτογραφίες ελέγχου, και εστιάσαμε στο πόσο ποσοστό επιτυχίας είχαν τα δίκτυα, το περιθώριο λάθους και τον χρόνο ολοκλήρωσης του προγράμματος.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Success Rate  No textures, Single Object | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Ttextures, Mixed Objects | 0.975 | 0.95 | 0.95 | 0.725 |
| With Textures, Mixed Objects | 0.95 | 0.55 | 0.5 | 0.15 |
| No Textures, Single Object | 1 | 0.925 | 0.825 | 0.775 |
| With Textures, Single Object | 1 | 0.4 | 0.35 | 0.125 |

*Σχήμα 6.3.1 - Πίνακας Success Rate σε δίκτυο με στοιχεία εκπαίδευσης χωρίς texture με ένα αντικείμενο σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Success Rate  No textures, Mixed Objects | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 1 | 0.95 | 0.875 | 0.85 |
| With Textures, Mixed Objects | 1 | 0.55 | 0.275 | 0.15 |
| No Textures, Single Object | 1 | 0.95 | 0.875 | 0.775 |
| With Textures, Single Object | 1 | 0.4 | 0.125 | 0.025 |

*Σχήμα 6.3.2 - Πίνακας Success Rate σε δίκτυο με στοιχεία εκπαίδευσης χωρίς texture με μεικτά αντικείμενα σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Success Rate  With textures, Mixed Objects | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 1 | 0.875 | 0.85 | 0.35 |
| With Textures, Mixed Objects | 1 | 0.9 | 0.825 | 0.8 |
| No Textures, Single Object | 1 | 0.8 | 0.75 | 0.325 |
| With Textures, Single Object | 1 | 1 | 0.75 | 0.825 |

*Σχήμα 6.3.3 - Πίνακας Success Rate σε δίκτυο με στοιχεία εκπαίδευσης με texture με ένα αντικείμενα σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Success Rate  With textures, Mixed Objects | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 1 | 0.65 | 0.775 | 0.275 |
| With Textures, Mixed Objects | 1 | 0.85 | 0.8 | 0.875 |
| No Textures, Single Object | 1 | 0.6 | 0.525 | 0.25 |
| With Textures, Single Object | 1 | 0.775 | 0.775 | 0.825 |

*Σχήμα 6.3.4 - Πίνακας Success Rate σε δίκτυο με στοιχεία εκπαίδευσης με texture με μεικτά αντικείμενα σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Margin  No textures, Single Object | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 3.63E-07 | 3.94E-18 | 2.97E-09 | 3.48E-03 |
| With Textures, Mixed Objects | 1.20E-06 | 2.49E-02 | 2.49E-02 | 2.49E-02 |
| No Textures, Single Object | 2.86E-04 | 5.16E-06 | 2.51E-05 | 4.42E-04 |
| With Textures, Single Object | 1.03E-06 | 2.49E-02 | 2.49E-02 | 2.49E-02 |

*Σχήμα 6.4.1 - Πίνακας Error Margin σε δίκτυο με στοιχεία εκπαίδευσης χωρίς texture με ένα αντικείμενο σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Margin  No textures, Mixed Objects | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 8.94E-09 | 3.48E-06 | 1.99E-07 | 1.16E-02 |
| With Textures, Mixed Objects | 1.32E-04 | 2.45E-02 | 2.49E-02 | 2.50E-02 |
| No Textures, Single Object | 6.85E-04 | 1.44E-03 | 6.96E-05 | 4.01E-03 |
| With Textures, Single Object | 9.36E-05 | 2.48E-02 | 2.49E-02 | 2.50E-02 |

*Σχήμα 6.4.2 - Πίνακας Error Margin σε δίκτυο με στοιχεία εκπαίδευσης χωρίς texture με μεικτά αντικείμενα σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Margin  With textures, Single Object | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 5.47E-05 | 2.15E-04 | 2.00E-06 | 3.59E-04 |
| With Textures, Mixed Objects | 5.61E-06 | 1.60E-04 | 1.25E-06 | 4.19E-06 |
| No Textures, Single Object | 1.13E-03 | 1.05E-02 | 2.18E-02 | 1.77E-01 |
| With Textures, Single Object | 3.07E-06 | 1.86E-03 | 1.27E-03 | 6.02E-03 |

*Σχήμα 6.4.3 - Πίνακας Error Margin σε δίκτυο με στοιχεία εκπαίδευσης με texture με ένα αντικείμενο σαν είσοδο*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Margin  With textures, Mixed Objects | | Training Data | | | |
| 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Test Data | No Textures, Mixed Objects | 1.33E-05 | 2.72E-06 | 2.68E-08 | 2.49E-02 |
| With Textures, Mixed Objects | 1.03E-06 | 2.78E-03 | 2.09E-02 | 4.59E-06 |
| No Textures, Single Object | 2.47E-03 | 2.49E-02 | 2.49E-02 | 2.50E-02 |
| With Textures, Single Object | 6.31E-07 | 2.35E-02 | 2.49E-02 | 1.60E-03 |

*Σχήμα 6.4.4 - Πίνακας Error Margin σε δίκτυο με στοιχεία εκπαίδευσης με texture με μεικτά αντικείμενα σαν είσοδο*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Calculation Time | 2 shadows | 4 shadows | 8 shadows | 16 shadows |
| Without Texture | 403.34 | 502.55 | 790.43 | 943.55 |
| With Texture | 413.55 | 497.19 | 800.97 | 935.75 |

*Σχήμα 6.5 - Πίνακας που περιέχει τον μέσο χρόνο εκτέλεσης του κάθε δικτύου*

Παρατηρούμε ότι, στα δίκτυα με 2 κατηγορίες σκιών, η πρόβλεψη για την σωστή κατηγορία είναι σχεδόν άψογη, με πολύ μικρά error margins. Στα δίκτυα με 4 και 8 κατηγορίες σκιών, παρατηρούμε σημαντική μείωση της απόδοσης των δεδομένων ελέγχου που ήταν ξένα στα δίκτυα, δηλαδή, στα δίκτυα που εκπαιδευτήκαν με textures είχαν σοβαρή μείωση στα δεδομένα ελέγχου με textures, και το αντίστροφο. Αυτό ήταν αναμενόμενο, διότι τα δίκτυα δεν είχαν γνώση των άλλων δεδομένων.

Παρ' όλα αυτά, βλέπουμε ότι τα δίκτυα που εκπαιδευτήκαν με textures απέδωσαν καλύτερα από αυτά που εκπαιδεύτηκαν χωρίς, πράγμα που μας δείχνει ότι τα δίκτυα που εκπαιδευτήκαν με textures είναι πιο εύκολο να προβλέψουν τον φωτισμό της εικόνας με άλλα δεδομένα απ' αυτά που εκπαιδευτήκαν.

Μία άλλη σημαντική και ενδιαφέρουσα παρατήρηση, είναι ότι τα δίκτυα που εκπαιδευτήκαν με ένα αντικείμενο αντί με μείξη πολλών αντικειμένων, είχαν ψηλότερη επίδοση στα δεδομένα που ήταν ξένα προς το δίκτυο, αλλά με μεγαλύτερο error margin. Αυτό μπορεί να συμβαίνει λόγω του ότι τα δίκτυα που εκπαιδευτήκαν με μείξη στοιχείων, να παραεκπαιδευτήκαν, και να "παπαγάλισαν" τα δεδομένα εκπαίδευσης. Αυτό το φαινόμενο, μας δείχνει ότι τα δίκτυα που εκπαιδευτήκαν με ένα αντικείμενο, παρ' όλο που είναι πιο πιθανό να κάνουν λάθος, έχουν μια πιο γενική και πολύπλευρη λύση στο πρόβλημα αναγνώρισης των σκιών.

Τέλος, θα μελετήσουμε την σημαντικότερη ομάδα δεδομένων, αυτή που έχει 16 κατηγορίες σκίασης. Όπως ήταν αναμενόμενο, έχουν τον πιο αργό χρόνο εξαγωγής αποτελεσμάτων, λόγω του μεγαλύτερου αριθμού εξόδων που έχει το fully connected layer του δικτύου. Παρατηρούμε τα ίδια φαινόμενα που παρατηρήσαμε στα προηγούμενα δίκτυα, δηλαδή το ότι τα δίκτυα που εκπαιδευτήκαν με ένα αντικείμενο έχουν πιο υψηλή επίδοση από τα δίκτυα που εκπαιδευτήκαν με μείξη αντικειμένων, και το φαινόμενο καλύτερης απόδοσης στα δίκτυα που εκπαιδευτήκαν με textures.

Κάτι επιπλέων που παρατηρήθηκε, είναι ότι η επίδοση των δικτύων στα δεδομένα ελέγχου που έχουν τα ίδια χαρακτηριστικά με τα στοιχεία εκπαίδευσης ήταν υψηλότερα από τα δίκτυα που έχουν λιγότερες κατηγορίες, και η επίδοση στα δεδομένα ελέγχου που είχαν τα αντίθετα χαρακτηριστικά με τα στοιχεία εκπαίδευσης ήταν σημαντικά χαμηλότερα. Αυτό επιβεβαιώνει τις υποψίες για εξειδικευμένη εκπαίδευση των δικτύων, το ότι δηλαδή τα δίκτυα ειδικεύονται στην αναγνώριση μόνο των στοιχείων που εκπαιδευτήκαν.

1. **Σύγκριση Αποτελεσμάτων**

Τα αποτελέσματα, κατά κύριο λόγο, επιβεβαίωσαν τις προβλέψεις μας. Τα αποτελέσματα που επέστρεψαν τα δίκτυα είναι απολύτως λογικά, και τα λάθη των δικτύων ήταν μέσα στα αποδεκτά όρια.

Το φαινόμενο, όμως, που δεν αναμέναμε, είναι το φαινόμενο της έκδοσης καλύτερων αποτελεσμάτων από το δίκτυο που είχε 16 κατηγορίες, σε σύγκριση με τα δίκτυα που είχαν λιγότερες κατηγορίες. Είναι ένα ενδιαφέρον και αξιόλογο φαινόμενο, το οποίο δείχνει ότι με περισσότερα δεδομένα εκπαίδευσης, το δίκτυο θα μπορούσε να πετύχει υψηλότερα ποσοστά απόδοσης, με μικρότερο error margin, και μια πιο ολοκληρωμένη και πολύπλευρη λύση στο πρόβλημά μας.

Αξιοσημείωτο επίσης, είναι το φαινόμενο της καλύτερης επίδοσης από τα δίκτυα τα οποία εκπαιδευτήκαν με ένα αντικείμενο από αυτά που εκπαιδευτήκαν με μίξη στοιχείων. Μπορεί σε αυτό τον τομέα να ερευνηθεί κατά πόσο ένα αντικείμενο, μπορεί να λειτουργήσει σαν καλούπι, και έτσι, ένα δίκτυο να πρέπει να εκπαιδευτεί μόνο από αυτό, αλλά να αναγνωρίζει όλα τα αντικείμενα ελέγχου.

**Κεφάλαιο 7**

**Συμπεράσματα**

Κλείνοντας την έρευνά μας, θα πρέπει να δούμε κατά πόσο υλοποιήσαμε τους στόχους μας. Ο στόχος μας ήταν η δημιουργία μιας εφαρμογής, η οποία να μπορεί να αναγνωρίσει τις φωτεινές πηγές σε πραγματικό χρόνο. Αν και τα αποτελέσματά μας δεν είναι όσο εύστοχα όσο υπολογίσαμε αρχικά, με περισσότερη μελέτη και έρευνα, η υλοποίηση της εφαρμογής αυτής δεν είναι τόσο εξεζητημένη όσο υπολογίζαμε.

Μια από τις αιτίες που πιστεύουμε αυτό, είναι το ότι τα δίκτυα είναι ευπροσάρμοστα στα δεδομένα των εισόδων τους, σε σχέση με τα στοιχεία με τα οποία εκπαιδευτήκαν. Με αυτή την ιδιότητα, μπορεί το δίκτυο να εκπαιδευτεί με τα στοιχεία που του παρέχονται, και να μπορέσει να λύσει προβλήματα και με εντελώς διαφορετικά δεδομένα ελέγχου, έχοντας μία πολύ μεγάλη πιθανότητα το αποτέλεσμα το οποίο θα εξάγει να είναι σωστό.

Επίσης, παρ' όλο που η διαδικασία εκπαίδευσης είναι μια από τις πιο αργές διαδικασίες στον τομέα της πληροφορικής, όταν το δίκτυο εκπαιδευτεί, τότε μπορεί να εξάγει τα αποτελέσματα των δεδομένων σε πραγματικό χρόνο, πράγμα που προοδεύει δραματικά την ανάπτυξη αρκετών τεχνολογιών, όπως το AR που έχουμε μελετήσει σε αυτή την έρευνα

Το σημαντικότερο, όμως, σημείο, είναι, ότι δείχνουν τάσεις για γενικοποίηση των στοιχείων εκπαίδευσής τους, γεγονός που μπορεί να βρει λύσεις και, στην περίπτωση των convolutional νευρωνικών δικτύων, πολύπλοκες κατηγοριοποιήσεις, που δεν θα μπορούσαν να βρεθούν από τον άνθρωπο. Αυτός είναι και ο λόγος που μας ώθησε εξ' αρχής να δοκιμάσουμε την προσέγγιση λύσης στο πρόβλημά μας.

Αυτή η μελέτη, ανέδειξε την σημαντικότητα και την χρησιμότητα των νευρωνικών δικτύων σε προβλήματα τα οποία πρόσφατα θεωρούνταν άλυτα. Η πληροφορία που μπορείς να εξάγεις, και η γρηγοράδα και η αμεσότητα που μπορείς να την συλλέξεις είναι συναρπαστική. Και, καθώς η τεχνολογία αναπτύσσεται, η αποτελεσματικότητα των νευρωνικών δικτύων θα συνεχίζει να αυξάνεται

**Κεφάλαιο 8**

**Μελλοντική Δουλειά**

Μετά από τα προαναφερθέντα συμπεράσματα, και μετά από την ανάλυση των αποτελεσμάτων της θεωρητικής λύσης, είναι σημαντικό να αναφερθεί η μελλοντική δουλειά που θα γίνει πάνω σε αυτή την εργασία. Η εργασία, αφού είδαμε ότι έχει δυνητικότητα για εφαρμογή στον πραγματικό κόσμο, θα πρέπει να την ελέγξουμε και με δεδομένα από τον πραγματικό κόσμο. Στα πλαίσια της εργασίας, πήραμε μικρό αριθμό δειγμάτων από τον πραγματικό κόσμο, και ελέγξαμε τα αποτελέσματά του με το νευρωνικό δίκτυο που εκπαιδεύτηκε σε 16 πιθανές σκιές με textures και με μίξη αντικειμένων. Το ποσοστό επιτυχίας δεν ήταν μεγαλύτερο από 20%, αλλά στην πρόβλεψη των σκιών, η ορθή σκιά ήταν η 2η ή 3η επιλογή του δικτύου. Τα αποτελέσματα αυτά ενισχύουν τις υποψίες μας για ορθή λειτουργία του δικτύου και στον πραγματικό κόσμο, δημιουργώντας, παράλληλα την ανάγκη για δημιουργία ενός δικτύου το οποίο θα είναι πιο γενικευμένο, και να έχει την δυνατότητα αναγνώρισης φωτεινές πηγές πέραν της μίας.

*Σχήμα 8.1 - Ένα από τα δείγματα φωτογραφίας παρμένα από τον πραγματικό κόσμο με τα οποία ελέγχτηκε το δίκτυο*



Στην μελλοντική εργασία, επίσης, θα πρέπει να ασχοληθούμε με την εκπαίδευση των δικτύων με πραγματικά στοιχεία. Λόγω της πληθώρας των φωτογραφιών που θα υπάρξουν, και λόγω της μοναδικότητας της κάθε φωτογραφίας, τα μελλοντικά νευρωνικά δίκτυα πρέπει να αλλάξουν με τέτοιο τρόπο, έτσι ώστε να αναγνωρίζουν τις ομοιότητες που έχουν οι φωτογραφίες που θα είναι στην ίδια κατηγορία σκίασης.

Τέλος, θα πρέπει να μελετηθεί κατά πόσο οι χρόνοι αναγνώρισης των σκιών μετά την προεπεξεργασία μπορεί να μειωθούν περαιτέρω, για πιο ομαλή λειτουργία του προγράμματος σε κινητές συσκευές, έτσι ώστε να δημιουργηθεί μια αποδοτική και ακριβής λύση.

**Βιβλιογραφία**

[1] J.-F.Lalonde, A. A. Efros, and S. G. Narasimhan. *Estimating natural illumination from a single outdoor image*. (ICCV, 2009. 1).

# [2] H. Ma, Q, Qin and X. Shen. *Detection of and compensation for shadows in colored urban aerial images*. (IEEE, Intelligent Control and Automation, 2004)

[3] W. McCulloch, W. Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. (Bulletin of Mathematical Biophysics, Vol. 5, 1943, p. 115-133)

[4] T. Kohonen. *Self-Organization and Associative Memory* (Springer-Verlag, Berlin, 1988).

[5] R.A. Howard. *Dynamic Programming and Markov Processes.* (Willey, The Massachusetts Institute of Technology, 1964)

[6] M. Matsugu, K. Mori, Y Mitari, Y. Kaneda. *Subject Independent Facial Expression Recognition With Robust Face Detection Using a Convolutional Neural Network.* (Neural Network, vol. 16, no. 5–6, pp. 555–559, 2003)

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane,´ R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-scale machine learning on heterogeneous systems* (Software available from tensorflow.org. 7 2015.)

[8] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, H. N Seung. *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit.* (Nature 405, 947–951, 2000)

**Παράρτημα Α - Κώδικας Python**

Ο κώδικας αυτός, είναι τροποποιημένος κώδικας από την hackCyprus.

import dataset

import tensorflow as tf

import time

from datetime import timedelta

import math

import random

import numpy as np

from datetime import datetime

from numpy.random import seed

seed()

from tensorflow import set\_random\_seed

set\_random\_seed(seed())

batch\_size = 32

#Prepare input data

classes = ['left','right']

num\_classes = len(classes)

# 33.3% of the data will automatically be used for validation

validation\_size = 0.333

img\_size = 128

num\_channels = 3

train\_path='training\_data'

# We shall load all the training and validation images and labels into memory using openCV and use that during training

data = dataset.read\_train\_sets(train\_path, img\_size, classes, validation\_size=validation\_size)

print("Complete reading input data. Will Now print a snippet of it")

print("Number of files in Training-set:\t\t{}".format(len(data.train.labels)))

print("Number of files in Validation-set:\t{}".format(len(data.valid.labels)))

session = tf.Session()

x = tf.placeholder(tf.float32, shape=[None, img\_size,img\_size,num\_channels], name='x')

## labels

y\_true = tf.placeholder(tf.float32, shape=[None, num\_classes], name='y\_true')

y\_true\_cls = tf.argmax(y\_true, dimension=1)

##Network graph params

filter\_size\_conv1 = 3

num\_filters\_conv1 = 32

filter\_size\_conv2 = 3

num\_filters\_conv2 = 32

filter\_size\_conv3 = 3

num\_filters\_conv3 = 64

fc\_layer\_size = 160000

def create\_weights(shape):

return tf.Variable(tf.truncated\_normal(shape, stddev=0.05))

def create\_biases(size):

return tf.Variable(tf.constant(0.05, shape=[size]))

def create\_convolutional\_layer(input,

num\_input\_channels,

conv\_filter\_size,

num\_filters):

## We shall define the weights that will be trained using create\_weights function.

weights = create\_weights(shape=[conv\_filter\_size, conv\_filter\_size, num\_input\_channels, num\_filters])

## We create biases using the create\_biases function. These are also trained.

biases = create\_biases(num\_filters)

## Creating the convolutional layer

layer = tf.nn.conv2d(input=input,

filter=weights,

strides=[1, 1, 1, 1],

padding='SAME')

layer += biases

## We shall be using max-pooling.

layer = tf.nn.max\_pool(value=layer,

ksize=[1, 2, 2, 1],

strides=[1, 2, 2, 1],

padding='SAME')

## Output of pooling is fed to Relu which is the activation function for us.

layer = tf.nn.relu(layer)

return layer

def create\_flatten\_layer(layer):

#We know that the shape of the layer will be [batch\_size img\_size img\_size num\_channels]

# But let's get it from the previous layer.

layer\_shape = layer.get\_shape()

## Number of features will be img\_height \* img\_width\* num\_channels. But we shall calculate it in place of hard-coding it.

num\_features = layer\_shape[1:4].num\_elements()

## Now, we Flatten the layer so we shall have to reshape to num\_features

layer = tf.reshape(layer, [-1, num\_features])

return layer

def create\_fc\_layer(input,

num\_inputs,

num\_outputs,

use\_relu=True):

#Let's define trainable weights and biases.

weights = create\_weights(shape=[num\_inputs, num\_outputs])

biases = create\_biases(num\_outputs)

# Fully connected layer takes input x and produces wx+b.Since, these are matrices, we use matmul function in Tensorflow

layer = tf.matmul(input, weights) + biases

if use\_relu:

layer = tf.nn.relu(layer)

return layer

layer\_conv1 = create\_convolutional\_layer(input=x,

num\_input\_channels=num\_channels,

conv\_filter\_size=filter\_size\_conv1,

num\_filters=num\_filters\_conv1)

layer\_conv2 = create\_convolutional\_layer(input=layer\_conv1,

num\_input\_channels=num\_filters\_conv1,

conv\_filter\_size=filter\_size\_conv2,

num\_filters=num\_filters\_conv2)

layer\_conv3= create\_convolutional\_layer(input=layer\_conv2,

num\_input\_channels=num\_filters\_conv2,

conv\_filter\_size=filter\_size\_conv3,

num\_filters=num\_filters\_conv3)

layer\_flat = create\_flatten\_layer(layer\_conv3)

layer\_fc1 = create\_fc\_layer(input=layer\_flat,

num\_inputs=layer\_flat.get\_shape()[1:4].num\_elements(),

num\_outputs=fc\_layer\_size,

use\_relu=True)

layer\_fc2 = create\_fc\_layer(input=layer\_fc1,

num\_inputs=fc\_layer\_size,

num\_outputs=num\_classes,

use\_relu=False)

y\_pred = tf.nn.softmax(layer\_fc2,name='y\_pred')

y\_pred\_cls = tf.argmax(y\_pred, dimension=1)

session.run(tf.global\_variables\_initializer())

cross\_entropy = tf.nn.softmax\_cross\_entropy\_with\_logits(logits=layer\_fc2,

labels=y\_true)

cost = tf.reduce\_mean(cross\_entropy)

optimizer = tf.train.AdamOptimizer(learning\_rate=1e-4).minimize(cost)

correct\_prediction = tf.equal(y\_pred\_cls, y\_true\_cls)

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

session.run(tf.global\_variables\_initializer())

def show\_progress(epoch, feed\_dict\_train, feed\_dict\_validate, val\_loss):

acc = session.run(accuracy, feed\_dict=feed\_dict\_train)

val\_acc = session.run(accuracy, feed\_dict=feed\_dict\_validate)

msg = "Training Epoch {0} --- Training Accuracy: {1:>6.1%}, Validation Accuracy: {2:>6.1%}, Validation Loss: {3:.3f}"

print(msg.format(epoch + 1, acc, val\_acc, val\_loss))

total\_iterations = 0

saver = tf.train.Saver()

def train(num\_iteration):

global total\_iterations

for i in range(total\_iterations,

total\_iterations + num\_iteration):

x\_batch, y\_true\_batch, \_, cls\_batch = data.train.next\_batch(batch\_size)

x\_valid\_batch, y\_valid\_batch, \_, valid\_cls\_batch = data.valid.next\_batch(batch\_size)

feed\_dict\_tr = {x: x\_batch,

y\_true: y\_true\_batch}

feed\_dict\_val = {x: x\_valid\_batch,

y\_true: y\_valid\_batch}

session.run(optimizer, feed\_dict=feed\_dict\_tr)

if i % int(data.train.num\_examples/batch\_size) == 0:

val\_loss = session.run(cost, feed\_dict=feed\_dict\_val)

epoch = int(i / int(data.train.num\_examples/batch\_size))

show\_progress(epoch, feed\_dict\_tr, feed\_dict\_val, val\_loss)

saver.save(session, './left\_right.ckpt')

total\_iterations += num\_iteration

train(num\_iteration=1000)

**Παράρτημα Β - Κώδικας Unity (C#)**

using System.IO;

using UnityEngine;

public class LightMovement : MonoBehaviour {

public GameObject sphere;

public GameObject cylinder;

public GameObject cube;

int frame = 0;

int itter = 0;

int max\_itter = 40;

float radius = 42;

float speed = 0.1f;

float posx = 0;

float posy = 30;

float posz = 42;

float rotx = 30;

float roty = 0;

float rotz = 0;

void Start() {

sphere.SetActive(false);

cylinder.SetActive(false);

cube.SetActive(false);

for (int i = 0; i < 16; i++) {

Directory.CreateDirectory("C:\\Users\\User\\Desktop\\samples\\training\_data\\" + i);

}

}

// Update is called once per frame

void Update () {

frame++;

if (frame==5&&itter < max\_itter) {

frame = 0;

//detecting what range the circle will take

switch (itter%16) {

case 0:

case 7:

case 8:

case 15:

posz = Random.Range(39.07f, 41.98f);

break;

case 1:

case 6:

case 9:

case 14:

posz = Random.Range(30.21f, 38.51f);

break;

case 2:

case 5:

case 10:

case 13:

posz = Random.Range(16.74f, 29.17f);

break;

default:

posz = Random.Range(0.73f, 15.39f);

break;

}

//calculating x with circle's equation

posx = Mathf.Sqrt((radius \* radius) - (posz \* posz));

//finding if x must be negative

if (itter % 16 >= 8) {

posx \*= -1;

}

//finding if z must be negative

if (itter % 16 <= 3 || itter % 16 >= 12) {

posz \*= -1;

}

posz = Random.Range(0f, 42f);

posx = Mathf.Sqrt((radius \* radius) - (posz \* posz));

//DELETE

if (Random.Range(1, 3) == 1) {

posz \*= -1;

}

if (Random.Range(1, 3) == 1) {

posx \*= -1;

}

//find the rotation of the light, in order to be faced at the centre

this.transform.position = new Vector3(posx, posy, posz);

float k = Mathf.Asin(posx / radius);

roty = 0;

if (posz > 0) {

roty = 180 \* k / Mathf.PI;

roty += 180;

}

else {

roty = -180 \* k / Mathf.PI;

}

this.transform.eulerAngles = new Vector3(rotx, roty, rotz);

//Creating the file's name

string i\_text;

string light;

if (itter / 10 == 0) {

i\_text = "000" + itter;

}

else if (itter / 100 == 0) {

i\_text = "00" + itter;

}

else if (itter / 1000 == 0) {

i\_text = "0" + itter;

}

else {

i\_text = "" + itter;

}

if (posx > 0) {

light = "right";

}

else {

light = "left";

}

if (posz > 0) {

light += "\_back";

}

else {

light += "\_front";

}

string name = ((itter%16) + "\\"+i\_text + "\_screenshot.png");

name = "test\_"+itter + ".png";

int number = Random.Range(1, 4);

number = 3;

itter++;

switch (number) {

case 1:

sphere.SetActive(true);

cylinder.SetActive(false);

cube.SetActive(false);

break;

case 2:

sphere.SetActive(false);

cylinder.SetActive(true);

cube.SetActive(false);

break;

case 3:

sphere.SetActive(false);

cylinder.SetActive(false);

cube.SetActive(true);

break;

default:

sphere.SetActive(false);

cylinder.SetActive(false);

cube.SetActive(true);

break;

}

ScreenCapture.CaptureScreenshot("C:\\Users\\User\\Desktop\\samples\\Testing\_data\\" + name);

print("itteration \t"+itter+"\n");

}

else {

Application.Quit();

}

}

}