Synthesis of Mesh Animation Sequences

Η Μεταπτυχιακή Διπλωματική Εργασία

υποβάλλεται στην ορισθείσα
από τη Συνέλευση
του Τμήματος Μηχανικών Η/Υ και Πληροφορικής
Εξεταστική Επιτροπή

από τον

Παύλο Χατζησάββα

ως μέρος των υποχρεώσεων για την απόκτηση του

ΔΙΠΛΩΜΑΤΟΣ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΗ ΜΗΧΑΝΙΚΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΩΝ
ΣΥΣΤΗΜΑΤΩΝ

ΜΕ ΕΙΔΙΚΕΥΣΗ
ΣΤΑ ΠΡΟΗΓΜΕΝΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

Πανεπιστήμιο Ιωαννίνων
Πολυτεχνική Σχολή
Ιωάννινα 2024

Εξεταστική Επιτροπή:

- **Ιωάννης Φούντος**, Καθηγητής, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων (Επιβλέπων)
- **Βασίλειος Δημακόπουλος**, Καθηγητής, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων
- **Βασίλειος Τενέντες**, Επίκ. Καθηγητής, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων

Αφιερωση

Στην οικογένειά μου.

Ευχαριστιες

Πρώτιστα, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ιωάννη Φούντο για την καθοδήγηση, την υπομονή και την αμέριστη βοήθειά του.

Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου, Αναστάσιο και Αρτεμησία, την αδελφή μου, Νικολέτα καθώς και τους φίλους μου για την κατανόηση που έδειξαν καθώς και για την στήριξή τους.

Περιεχομενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Περιληψη

Παύλος Χατζησάββας, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2024.
Synthesis of Mesh Animation Sequences.
Επιβλέπων: Ιωάννης Φούντος, Καθηγητής.

Στην παρούσα διατριβή, διερευνούμε καινοτόμες μεθόδους για το συνδυασμό ακολουθιών από πλέγματα (meshes) μέσω της εφαρμογής τεχνικών βαθιάς μάθησης. Η κύρια συνεισφορά μας είναι η ανάπτυξη μιας προηγμένης αρχιτεκτονικής κωδικοποιητή-αποκωδικοποιητή προσαρμοσμένης για τη σύνθεση κινούμενων σχεδίων. Ο κωδικοποιητής λαμβάνει και κωδικοποιεί τα χαρακτηριστικά των μεμονωμένων σχημάτων, ενώ ο αποκωδικοποιητής ανακατασκευάζει μια ολοκληρωμένη ακολουθία σχεδίων πλέγματος ενσωματώνοντας αυτά τα κωδικοποιημένα χαρακτηριστικά. Ο αποκωδικοποιητής είναι ιδιαίτερα ευέλικτος, επιτρέποντας την παραμετροποίηση με πολλαπλές εισόδους για την επίτευξη μεταφοράς στυλ, επιτρέποντας έτσι τη δημιουργία διαφορετικών στυλ κίνησης από τις εισόδους. Για να διασφαλίσουμε την ευρωστία και την πιστότητα των δημιουργούμενων πλεγμάτων σε σχέση με το αρχικό πλέγμα, εφαρμόσαμε και αξιολογήσαμε διάφορες τεχνικές που έχουν σχεδιαστεί για τη διατήρηση των βασικών χαρακτηριστικών του πλέγματος. Οι τεχνικές αυτές επικεντρώνονται στη διατήρηση τόσο τοπικών όσο και παγκόσμιων χαρακτηριστικών, συμπεριλαμβανομένων του όγκου, της καμπυλότητας και των μέσων κανονικών, τα οποία είναι ζωτικής σημασίας για τη διατήρηση της οπτικής και δομικής ακεραιότητας των κινούμενων σχεδίων. Τα πειραματικά μας αποτελέσματα καταδεικνύουν την αποτελεσματικότητα της προσέγγισής μας στην παραγωγή πλεγμάτων υψηλής ποιότητας που διατηρούν τα βασικά χαρακτηριστικά των αρχικών μοντέλων. Εντοπίζουμε επίσης διάφορες υποσχόμενες μεθόδους για μελλοντική έρευνα, όπως η ενίσχυση της αρχιτεκτονικής κωδικοποιητή-αποκωδικοποιητή,

η βελτίωση των μεθόδων παραμετροποίησης για πιο διαφοροποιημένες μεταφορές στυλ και η ανάπτυξη πιο εξελιγμένων τεχνικών για τη διατήρηση των πλεγμάτων.

Extended Abstract

Pavlos Chatzisavvas, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2024.
Synthesis of Mesh Animation Sequences.
Advisor: Ioannis Foudos, Professor.

In this thesis, we explore innovative methods for combining mesh animation sequences through the application of deep learning techniques. Our primary contribution is the development of an advanced encoder-decoder architecture tailored for animation synthesis. The encoder effectively captures and encodes the features of individual animation schemes, while the decoder reconstructs a comprehensive mesh animation sequence by integrating these encoded features. This decoder is highly versatile, allowing for parameterization with multiple inputs to achieve style transfer, thereby enabling the creation of diverse animation styles from given inputs. To ensure the robustness and fidelity of the generated mesh animations relative to the original rest pose mesh, we have implemented and evaluated various techniques designed to preserve essential mesh characteristics. These techniques focus on maintaining both local and global features, including volume, curvature, and average normals, which are crucial for preserving the visual and structural integrity of the animations. Our experimental results demonstrate the effectiveness of our approach in producing high-quality mesh animations that retain critical attributes of the original models. We also identify several promising avenues for future research, such as enhancing the encoder-decoder architecture, improving parameterization methods for more nuanced style transfers, and developing more sophisticated techniques for preserving mesh features during animation synthesis.

Κεφαλαιο 1

Εισαγωγη

1.1 Στόχοι

Στον παρόν κεφάλαιο παρουσιάζεται μια σύντομη επισκόπηση της διατριβής, οι στόχοι της καθώς και μία περιγραφή της δομής των κεφαλαίων της. Σκοπός της παρούσας διατριβής, είναι η μελέτη και η δημιουργία κινήσεων χαρακτήρων. Πιο συγκεκριμένα, με τη χρήση νευρωνικών δικτύων που αναπτύξαμε και με μεθόδους βαθειάς μάθησης, προσπαθήσαμε να αναπαράγουμε ή να δημιουργήσουμε νέες κινήσεις για τους χαρακτήρες μας. Για να το επιτύχουμε αυτό, αρχικά, εξάγαμε δεδομένα κινήσεων με τη χρήση του Blender [1]. Ειδικότερα, αντλήσαμε δεδομένα τόσο για την κίνηση του εκάστοτε χαρακτήρα, όσο και για τα χαρακτηριστικά του κάθε mesh που περιβάλλει τους χαρακτήρες. Στη συνέχεια, χρησιμοποιήσαμε τις πληροφορίες αυτές για την εκπαίδευση και την αξιολόγηση ενός νευρωνικού δικτύου αλλά, και για τη δημιουργία νέων, πιο σύνθετων κινήσεων. Για να επιτύχουμε όλα τα παραπάνω, χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Python η οποία με τις βιβλιοθήκες της, μας επιτρέπει τη δημιουργία και την εκπαίδευση του νευρωνικού δικτύου. Επίσης, η χρήση της κάρτας γραφικών [2], μας δίνει τη δυνατότητα να αναπτύξουμε και να αξιολογήσουμε ταχύτερα το νευρωνικό διότι, χρησιμοποιεί τις δυνατότητες της κάρτας γραφικών για υπολογισμούς. Τέλος, το Python API που

προσφέρει το Blender, μας βοήθησε στην αποκομιδή όλων των απαραίτητων δεδομένων, καθώς μας επιτρέπει να προγραμματίσουμε απευθείας σε Python μέσα στο περιβάλλον του Blender, ώστε να πάρουμε και να αποθηκεύσουμε σε όποια μορφή επιθυμούμε τις πληροφορίες που χρειαζόμαστε.

1.2 Δομή της Διατριβής

Η διατριβή περιέχει 5 κεφάλαια. Στο κεφάλαιο 2, θα προσδιοριστούν το θεωρητικό και τεχνολογικό υπόβαθρο που χρειάζεται προκειμένου να κατανοηθούν οι όροι που θα χρησιμοποιηθούν αλλά και για την πλήρη αντίληψη της μεθολογίας που υλοποιήθηκε.

Στο τρίτο κεφάλαιο, θα αποδώσουμε τον τρόπο ανάπτυξης και υλοποίησης του νευρωνικού δικτύου. Αναλύουμε τις μεθόδους με τις οποίες εξάγαμε τις απαραίτητες πληροφορίες αλλά και τον τρόπο με τον οποίο τις αξιοποιούμε αυτές.

Στο τέταρτο κεφάλαιο, θα παρουσιάσουμε και θα αναλύσουμε τα αποτελέσματα που προέκυψαν από την υλοποίηση της συγκεκριμένης διατριβής.

Τέλος, στο πέμπτο κεφάλαιο, θα ασχοληθούμε με τα συμπεράσματα στα οποία καταλήξαμε και τις πιθανές βελτιώσεις και επεκτάσεις που μπορούν να προκύψουν στο μέλλον.

Κεφαλαιο 2

Θεωρητικο και Τεχνολογικο Υποβαθρο

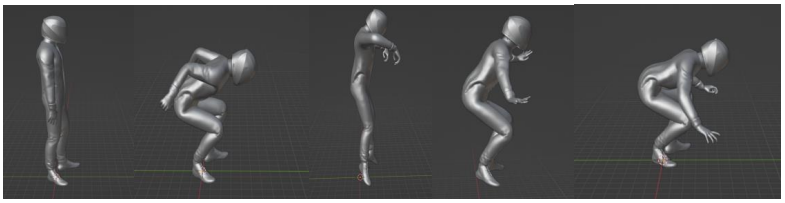
Στο παρόν κεφάλαιο θα παρουσιάσουμε το θεωρητικό και τεχνολογικό υπόβαθρο που χρειαζόμαστε έτσι ώστε να μπορούμε να κατανοήσουμε την υλοποίηση της εργασίας. Στις επόμενες ενότητες, θα αναφερθούμε σε μερικές βασικές έννοιες που σχετίζονται με τα νευρωνικά δίκτυα και ειδικότερα με το μοντέλου που χρησιμοποιήσαμε. Επίσης, θα γίνει μια σύντομη αναφορά στο Blender και τις λειτουργίες του καθώς αποτελεί σημαντικό κομμάτι στην εξαγωγή των δεδομένων.

2.1 Blender

Σε αυτή την ενότητα θα ασχοληθούμε με το εργαλείο σχεδίασης Blender [1]. Πρόκειται για ένα ελεύθερο και ανοιχτού κώδικα λογισμικό το οποίο μας επιτρέπει να σχεδιάσουμε 3D μοντέλα, οπτικά εφέ, animations και άλλα είδη 3D γραφικών. Το περιβάλλον του, αν και δεν είναι το πιο εύχρηστο, ειδικά για κάποιον αρχάριο χρήστη, προσφέρει πολλές δυνατότητες [3]. Πιο συγκεκριμένα, οι animators, μέσω

του texturing και του modeling που προσφέρει το Blender, μπορούν να σχεδιάσουν κάθε μορφής χαρακτήρα ο οποίος θα συνοδεύεται από κινήσεις άκρως ρεαλιστικές και δίχως ιδιαίτερες αλλοιώσεις στα μοντέλα των χαρακτήρων. Προκειμένου να επιτευχθούν όλα τα παραπάνω, το Blender προσφέρει διάφορους τύπους αντικειμένων όπως τα meshes (πλέγματα) και τα armatures τα οποία μπορούμε να επεξεργαστούμε ώστε να πάρουμε το επιθυμητό αποτέλεσμα [4].

Στις υποενότητες που ακολουθούν, θα αναλύσουμε τον τρόπο με τον οποίο μπορούμε να καταλήξουμε στη σύνθεση μίας κίνησης ενός χαρακτήρα [5] όπως φαίνεται στο Σχήμα 2.1 όπου ο χαρακτήρας εκτελεί ένα άλμα.



Σχήμα 2.1: Η κίνηση ενός χαρακτήρα

2.2 Meshes και Animations

Το πλέγμα (mesh) ενός τρισδιάστατου χαρακτήρα, αποτελεί ίσως το πιο σημαντικό στοιχείο στην τρισδιάστατη μοντελοποίηση. Μέσω αυτού, το μοντέλο μας παίρνει την τελική του μορφή που εν τέλει προβάλλεται στον εκάστοτε χρήστη. Η δημιουργία της μορφής ενός 3D μοντέλου, βασίζεται στην ένωση των κορυφών (vertices), των ακμών (edges) και των επιφανειών (faces) του στον τρισδιάστατο χώρο [6]. Γενικότερα, το πλέγμα σε συνδυασμό με διάφορες τεχνικές όπως το texturing δίνει στο μοντέλο την απαραίτητη λεπτομέρεια που χρειάζεται ωστέ να είναι όσο πιο ρεαλιστικό γίνεται πριν εφαρμοστεί πάνω του κάποιο animation. Παρακάτω, στο Σχήμα 2.2 βλέπουμε πως ένας σκελετός κατέληξε να έχει τη μορφή ενός οδηγού με κόκκινη στολή μέσα από την προσθήκη ενός mesh αλλά και textures. Τέλος, αξίζει να αναφέρουμε πως στις αρχικές του εκδόσεις, το Blender προσέφερε μόνο τη δυνατότητα δημιουργίας τριγώνων ενώ πλέον, ο χρήστης μπορεί να σχηματίσει οποιοδήποτε πολύγωνο.

Σχήμα 2.2: Το mesh ενός χαρακτήρα

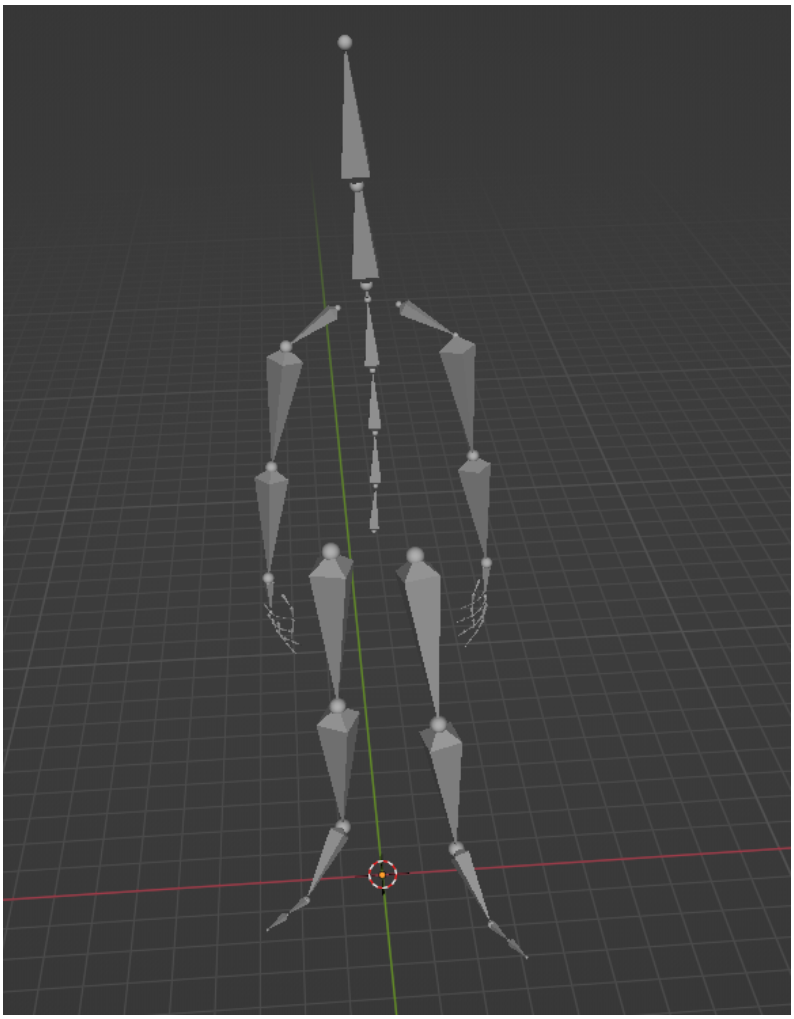
2.2.1 Animations

Όπως αναφέραμε, κάθε mesh αποτελείται από κορυφές. Κάθε κορυφή, χαρακτηρίζεται από τις συντεταγμένες x, y και z του τρισδιάστατου χώρου. Για να πετύχουμε την επιθυμητή κίνηση του χαρακτήρα μας, πρέπει κάθε φορά να μετατοπίζουμε τις συντεταγμένες αυτές στη θέση που θέλουμε για να σχηματιστεί μία διαφορετική στάση του πλέγματος μας. Η συνεχόμενη μετατόπιση των συντεταγμένων σε πολύ μικρά χρονικά διάστηματα, (frames) μας δημιουργεί την αίσθηση πως το μοντέλο μας κινείται στο χώρο.

2.2.2 Σκελετοί Τρισδιάστατων Μοντέλων

Σκελετός (armature) είναι ένα σύνολο από οστά τα οποία προσδίδουν στο μοντέλο μας τη λειτουργικότητα του σκελετού που θέλουμε να αναπαρίστουμε είτε αυτός

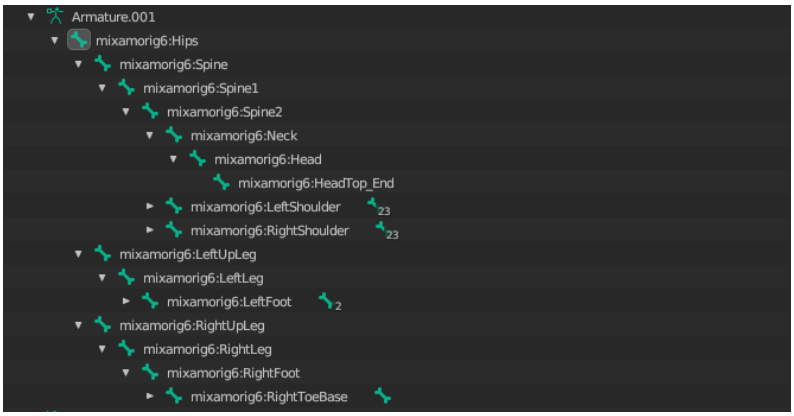
ανθρώπινος είτε όχι. Κάθε οστό, είναι υπεύθυνο για τη κίνηση ενός συγκεκριμένου τμήματος του mesh. Επιπρόσθετα, κάθε οστό ανήκει σε μια ομάδα από οστά όπως τα χέρια, τα πόδια ή το κεφάλι. Κατά κύριο λόγο, η ρίζα, το αρχικό οστό του σκελετού είναι οι γοφοί του και από ξεκινά ο σκελετός να χωρίζεται στις διάφορες υποομάδες του [7]. Παρακάτω φαίνεται ο διαχωρισμός του σκελετού στις ομάδες του (Σχήμα 2.3, Σχήμα 2.4).



Σχήμα 2.3: Ένας σκελετός

Επίσης, αξίζει να σημειώσουμε πως οι ανθρώπινοι, κυρίως, σκελετοί έχουν δύο σημαντικές στάσεις, Την T-Pose και την Rest Pose. Η T-Pose, όπως είναι εύκολα αντιληπτό και από την ονομασία της, είναι μία στάση κατά την οποία τα χέρια του μοντέλου είναι τεντωμένα στο πλάι και ουσιαστικά ο σκελετός σχηματίζει ένα ”T”. Προσφέρει στο χρήστη ένα ξέκαθαρο οπτικό αποτέλεσμα μέσω του οποίου μπορεί να επεξεργαστεί όλα τα οστά όπως φαίνεται και στο Σχήμα 2.5.

Η Rest Pose, είναι η προεπιλεγμένη στάση που έχει ο χαρακτήρας μας πριν εφαρ-



Σχήμα 2.4: Οι ομάδες οστών στις οποίες χωρίζεται ο σκελετός

μοστούν πάνω του παραμορφώσεις ή animations. Αξίζει να σημειώσουμε πως πολλές φορές οι δύο στάσεις ταυτίζονται για πρακτικούς λόγους ώστε να μην υπάρχει σύγχυση κατά τη δημιουργία του σκελετού και των animations του.

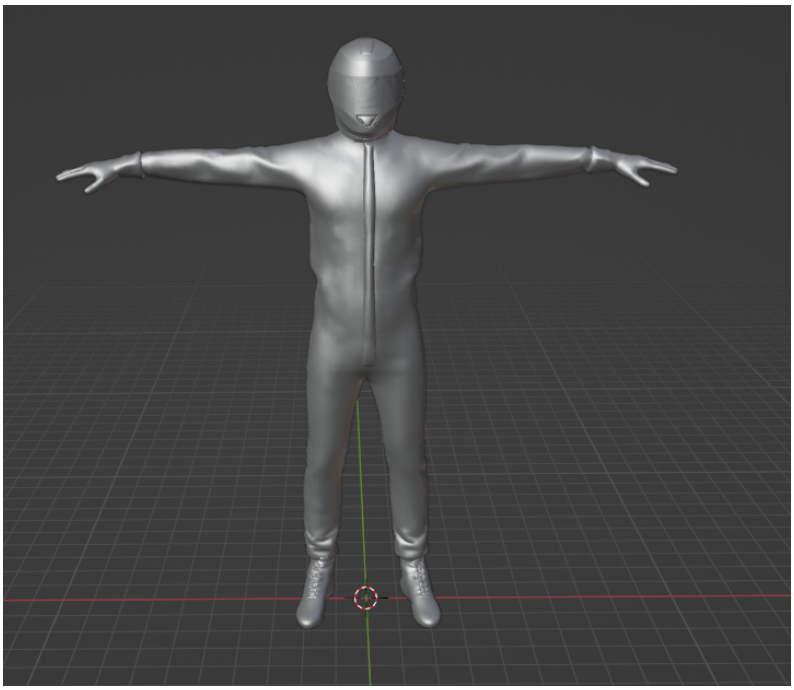
2.2.3 Rigging

Όταν ολοκληρωθεί η διαδικασία δημιουργίας του σκελετού και του mesh, θα πρέπει αυτά να συνδεθούν με κάποιο τρόπο. Η διαδικασία αυτή, ονομάζεται rigging και πέρα από την δημιουργία των οστέινων δομών, περιλαμβάνει και την τοποθέτηση controllers οι οποίοι θα διαχειρίζονται αλλά και θα περιορίζουν την κίνηση των οστών ώστε να έχουμε ένα όσο το δυνατόν πιο ρεαλιστικό και φυσικό αποτέλεσμα στην κίνηση του χαρακτήρα. Για παράδειγμα, με τη χρήση των Inverse Kinematics έχουμε τη δυνατότητα να ελέγξουμε την κίνηση των οστών. Αυτό συμβαίνει διότι, η συγκεκριμένη μέθοδος μας επιτρέπει να κινούμε τα άκρα του χαρακτήρα με βάση την τελική τους θέση.

2.3 Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα είναι ένας τύπος αλγόριθμου μηχανικής μάθησης (machine learning) και αποτελούν μια μέθοδο στη τεχνητή νοημοσύνη, που διδάσκει στους υπολογιστές να επεξεργάζονται δεδομένα με τρόπο εμπνευσμένο από τη δομή και τη λειτουργία του ανθρώπινου εγκεφάλου [8].

Τα νευρωνικά δίκτυα αποτελούνται από επίπεδα (layers) διασυνδεδεμένων τεχνητών νευρώνων, που επεξεργάζονται διάφορες πληροφορίες και εξάγουν ένα απο-



Σχήμα 2.5: Το T-Pose ενός μοντέλου

τέλεσμα με βάση ένα σύνολο από inputs, δηλαδή δεδομένα που εισάγονται.

Τα νευρωνικά δίκτυα μπορούν να βοηθήσουν σημαντικά τους υπολογιστές να λαμβάνουν έξυπνες αποφάσεις, περιορίζοντας τη συμμετοχή του ανθρώπινου παράγοντα.

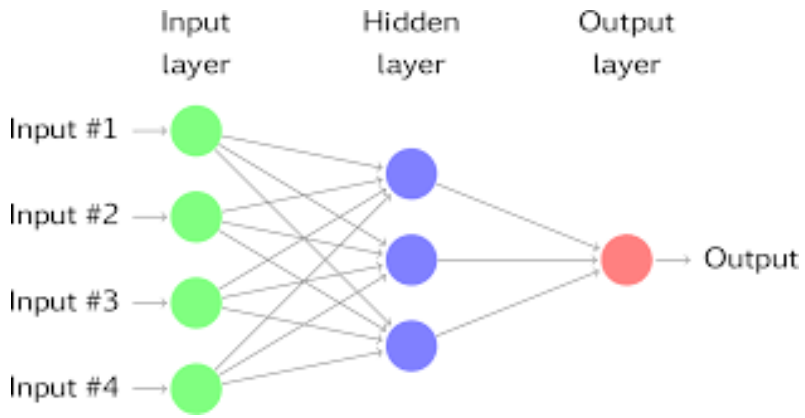
Αυτό πραγματοποιείται καθώς μπορούν να εκπαιδευτούν σε μεγάλα σύνολα δεδομένων και να μοντελοποιήσουν περίπλοκες σχέσεις μεταξύ δεδομένων εισόδου και εξόδου, βγάζοντας συμπεράσματα και κάνοντας διάφορες γενικεύσεις.

Στη συνέχεια, θα δούμε τον τρόπο με τον οποίο λειτουργεί ένα απλό νευρωνικό δίκτυο.

2.3.1 Τρόπος Λειτουργίας

Η πιο απλή αρχιτεκτονική ενός νευρωνικού δικτύου αποτελείται από τρεις ομάδες ή στρώματα[9] (layers): ένα στρώμα μονάδων «εισόδου» (input layer), που συνδέεται με ένα στρώμα «κρυφών» μονάδων (hidden layer), το οποίο συνδέεται με ένα επίπεδο μονάδων «εξόδου» (hidden layer) (Σχήμα 2.6).

- **Επίπεδο εισόδου (input layer)**: Πρόκειται για το πρώτο επίπεδο σε ένα νευρωνικό δίκτυο που λαμβάνει τα αρχικά δεδομένα εισόδου. Οι κόμβοι εισόδου



Σχήμα 2.6: Η αρχιτεκτονική ενός απλού νευρωνικού δικτύου

επεξεργάζονται τα δεδομένα αυτά, τα αναλύουν και τα μεταβιβάζουν στο επόμενο επίπεδο.

- **Κρυμμένο επίπεδο (hidden layer)**: Τα κρυμμένα στρώματα αποτελούν τα ενδιάμεσα επίπεδα ανάμεσα στο επίπεδο εισόδου και εξόδου και επιτελούν το μεγαλύτερο μέρος του υπολογισμού. Ενδέχεται να υπάρχουν πολλά κρυμμένα επίπεδα σε ένα νευρωνικό δίκτυο.

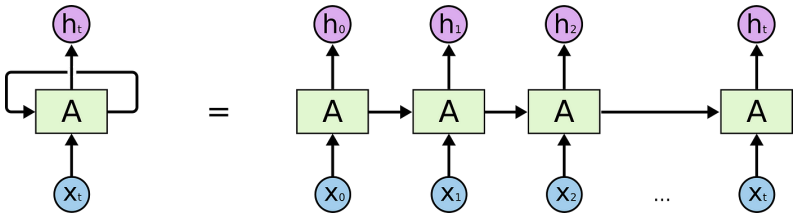
- **Επίπεδο εξόδου (output layer)**: Το επίπεδο εξόδου είναι το τελικό επίπεδο σε ένα νευρωνικό δίκτυο και οδηγεί στην έξοδο του δικτύου. Ο αριθμός των νευρώνων στο επίπεδο αυτό εξαρτάται από το εκάστοτε πρόβλημα που λύνεται.

2.3.2 Επαναλαμβανόμενα Νευρωνικά Δίκτυα (RNNs)

Τα επαναλαμβανόμενα νευρωνικά δίκτυα (recurrent neural networks) είναι μία κατηγορία νευρωνικών δικτύων, τα οποία έχουν σχεδιαστεί για την ανάλυση και επεξεργασία διαδοχικών, ακολουθιακών δεδομένων [10].

Η αρχιτεκτονική τους τους δίνει τη δυνατότητα να κρατούν αποτελέσματα από προηγούμενες εισόδους της ακολουθίας των δεδομένων. Το γεγονός αυτό, τα καθιστά ιδανικά για πληθώρα εφαρμογών που σχετίζονται με προβλέψεις και χρονοσειρές όπως για παράδειγμα, οι προβλέψεις των πωλήσεων σε ένα μελλοντικό χρονικό διάστημα ή η επεξεργασία φυσικών γλωσσών. Πιο συγκεκριμένα, η συγκεκριμένη αρχιτεκτονική δίνει τη δυνατότητα στο δίκτυο να επεξεργαστεί την ίδια είσοδο πολλές φορές. Όπως παρατηρούμε στο Σχήμα 2.7, η πληροφορία σχηματίζει

έναν κύκλο κατά τον οποίο συσχετίζεται η νέα πληροφορία με την παλιά, γεγονός που δεν συμβαίνει στα απλά Feed Forward δίκτυα.



Σχήμα 2.7: Η αρχιτεκτονική ενός RNN

2.3.3 Long Short Term Memory Νευρωνικά Δίκτυα (LSTMs)

Ένα Long Short Term Memory (LSTM) νευρωνικό δίκτυο είναι ένας συγκεκριμένος τύπος RNN, το οποίο δημιουργήθηκε με σκοπό να μοντελοποιεί τις χρονικές αλληλουχίες και τις μεγάλου εύρους εξαρτήσεις αυτών με μεγαλύτερη ακρίβεια από τα συμβατικά RNNs [11].

Η διαφορά της αρχιτεκτονικής τους σε σχέση με αυτή των RNNs έγκειται στο γεγονός πως κάθε κελί διαθέτει τρεις πύλες, οι οποίες ελέγχουν τη ροή των πληροφοριών. Αυτές είναι:

- **Πύλη εισόδου (input gate)**: Ελέγχει ποιες νέες πληροφορίες θα προστεθούν στο κύτταρο μνήμης. Αρχικά, μέσω ενός σιγμοειδούς επιπέδου (sigmoid), αποφασίζεται ποια πληροφορία θα κρατηθεί από την μνήμη. Στη συνέχεια μία συνάρτηση υπερβολικής εφαπτομένης (tanh), θα δημιουργήσει ένα διάνυσμα με υποψήφιες τιμές που θα μπορούσαν να εισαχθούν στη μνήμη.

- **Πύλη επιλεκτικής συγκράτησης (forget gate)**: Η συγκεκριμένη πύλη αποφασίζει ποιες από τις πληροφορίες δεν είναι πλέον σημαντικές και μπορούν να ”ξεχαστούν” από το κελί. Η απόφαση αυτή λαμβάνεται μέσω μια σιγμοειδούς συνάρτησης με είσοδο την κατάσταση του προηγούμενου κελιού και την είσοδο του τρέχοντος. Το αποτέλεσμά της, είναι ένα διάνυσμα με τιμές μεταξύ 0 και 1. Όσο πιο κοντά είναι μία τιμή στο 1 τόσο πιο πιθανό είναι να διατηρηθεί ενώ όσο πιο κοντά βρίσκεται στο 0, τότε είναι πιο πιθανό να ”ξεχαστεί”.

- **Πύλη εξόδου (output gate)**: Η πύλη αυτή αποφασίζει το αποτέλεσμα που θα εξαχθεί και έπειτα θα προωθηθεί στο επόμενο κελί. Αρχικά, με την εφαρμογή

μια σιγμοειδούς συνάρτησης θα αποφασιστεί πιο κομμάτι της κατάστασης θα διατηρηθεί ενώ μετέπειτα, όλη η κατάσταση του κελιού θα περάσει από μία υπερβολική εφαπτομένη συνάρτηση (tanh) προκειμένου οι τιμές να κλιμακωθούν στο πεδίο τιμών (-1,1). Τέλος, τα δύο αποτελέσματα πολλαπλασιάζονται μεταξύ τους ώστε να καταλήξουμε στην επιθυμητή έξοδο.

2.3.4 Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης αποτελούν ένα πολύ σημαντικό κομμάτι των νευρικών δικτύων, καθώς τους δίνουν τη δυνατότητα να μαθαίνουν πολύπλοκες, μη γραμμικές σχέσεις και μοτίβα ανάμεσα στα δεδομένα [12]. Δίχως αυτές, τα νευρωνικά δίκτυα θα έπρεπε να περιοριστούν μόνο σε γραμμικές σχέσεις με αποτέλεσμα να μπορούν να εκτελούν σχετικά απλούς υπολογισμούς.

Μια συνάρτηση ενεργοποίσης, υπολογίζει την έξοδο ενός νευρώνα συναρτήσει της εισόδου που θα δεχτεί αυτός. Η επιλογή της συνάρτησης που θα χρησιμοποιήσουμε εξαρτάται άμεσα από τις απαιτήσεις του προβλήματος και είναι ιδιαίτερα σημαντική διότι μπορεί να βελτιώσει την απόδοση του νευρωνικού δικτύου.

Στη συνέχεια της υποενότητας, θα αναλυθούν μερικές από τις συναρτήσεις ενεργοποίησης που χρησιμοποιήσαμε για την παρούσα διατριβή:

- **Σιγμοειδής συνάρτηση (sigmoid function)**: Πρόκειται για μία γραμμική συνάρτηση ενεργοποίησης η οποία μετατρέπει τις εισόδους σε εξόδους που ανήκουν στο διάστημα τιμών (0,1). Η εξίσωσή της είναι:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

- **Υπερβολική εφαπτομένη (tanh)**: Η συγκεκριμένη, μη γραμμική, συνάρτηση μετατρέπει τις εισόδους σε εξόδους με εύρος τιμών (-1,1). Γενικά, η συνάρτηση προτιμάται διότι τα δεδομένα κατανέμονται καλύτερα γύρω από το 0 με αποτέλεσμα το μοντέλο να συγκλίνει γρηγορότερα. Ο τύπος της είναι:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

- **Rectified Linear Unit (ReLU)**: Η ReLU, είναι μία απλή συνάρτηση η οποία παίρνει τη μεγαλύτερη τιμή ανάμεσα στο 0 και την είσοδο της. Είναι αρκετά

αποδοτική και λόγω της απλότητάς της προτιμάται στα κρυμμένα επίπεδα του νευρωνικού δικτύου. Ο τύπος της είναι ο παρακάτω:

$$\text{f(x)} = max(0, x) \tag{2.3}$$

- **Softmax συνάρτηση**: Χρησιμοποιείται κατά κύριο λόγο στην έξοδο ενός μοντέλου. Λαμβάνει ως είσοδο ένα διάνυσμα z και το μετατρέπει σε μία ακολουθία πιθανοτήτων με διάστημα τιμών (0,1). Η εξίσωση της είναι:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.4}$$

2.4 Αξιολόγηση Δικτύου και Σύνολα Εκπαίδευσης

Σκοπός της μηχανικής μάθησης, είναι η δημιουργία δικτύων τα οποία να μπορούν να μαθαίνουν μέσα από σύνολα δεδομένων. Επίσης, στόχος είναι να γίνονται προβλέψεις βάσει των δεδομένων με τα οποία εκπαιδεύτηκε ένα δίκτυο. Τα δεδομένα εκπαίδευσης των νευρωνικών δικτύων, αποτελούνται από τρία σύνολα: το σύνολο εκπαίδευσης (Training Dataset), το σύνολο δεδομένων επικύρωσης (Validation Dataset) και το σύνολο δοκιμής (Test Dataset).

Γενικά, όσα περισσότερα είναι τα δεδομένα που θα δώσουμε στο σύνολο εκπαίδευσης, τόσο καλύτερα εκπαιδεύεται το νευρωνικό δίκτυο σε αυτά και κάνει καλύτερες προβλέψεις. Βέβαια, κατά τη δημιουργία των παραπάνω συνόλων χρειάζεται ιδιαίτερη προσοχή διότι, μπορούμε να οδηγηθούμε σε φαινόμενα overfitting ή underfitting [13] [14].

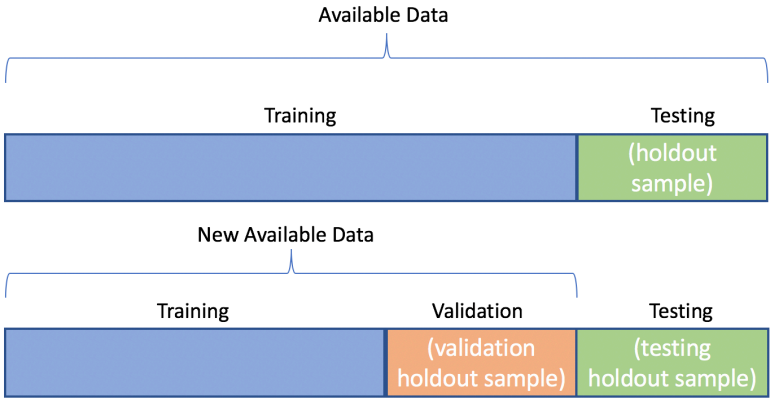
Κατόπιν της δημιουργίας των δεδομένων εκπαίδευσης, πρέπει να διασφαλίσουμε την ποιότητα αυτών. Για να το πετύχουμε αυτό, μπορούμε να αφαιρέσουμε θόρυβο ή ανωμαλίες (outliers) από τα δεδομένα μας.

Στη συνέχεια, οφείλουμε να δημιουργήσουμε ένα σύνολο δεδομένων για να προσμετρά την απόδοση του μοντέλου. Προκειμένου να επιτύχουμε έναν σωστό διαχωρισμό των δεδομένων μας, συνήθως χρησιμοποιούμε μία εκ των παρακάτω μεθόδων:

- **Train/Validation Split**: Πρόκειται για το διαχωρισμό των δεδομένων σε δύο υποσύνολα, ένα για εκπαίδευση και ένα για επικύρωση. Ο διαχωρισμός αυτός συνήθως γίνεται σε ένα ποσοστό 80% (backslash percent) ή 70% (backslash percent) για το σύνολο εκπαίδευσης και 20% (backslash percent) ή 30% (back-

slash percent) για το σύνολο επικύρωσης. Ωστόσο, τα ποσοστά αυτά μπορεί να τα καθορίσει ο χρήστης όπως επιθυμεί βάσει των αναγκών του (Σχήμα 2.8).

- **K-Fold Cross-Validation [15]**: Είναι μία μέθοδος όπου το σύνολο των δεδομένων, διαιρείται σε k ίσα υποσύνολα ίδιου μεγέθους. Η διαδικασία αυτή, εκτελείται k φορές με καθένα από τα υποσύνολα να χρησιμοποιείται μία φορά ως σύνολο δοκιμής και τα υπόλοιπα, k-1, να χρησιμοποιούνται για εκπαίδευση. Πρόκειται για μία αρκετά πιο αξιόπιστη μέθοδο της εκτίμησης της απόδοσης του μοντέλου συγκριτικά με την απλή διαίρεση των δεδομένων με βάση κάποια ποσοστά που ορίζει ο χρήστης. Αυτό συμβαίνει επειδή, κάθε υποσύνολο συμμετέχει τόσο στην εκπαίδευση όσο και στην επικύρωση. Τέλος, αξίζει να σημειώσουμε πως η εξαγωγή των μετρικών του μοντέλου γίνεται με τον μέσο όρο των μετρικών που προέκυψαν σε κάθε επανάληψη (Σχήμα 2.9).

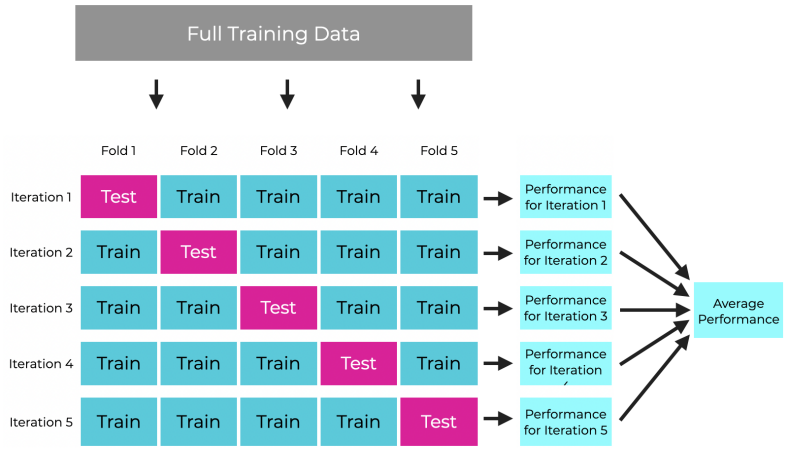


Σχήμα 2.8: Ένα απλό split

2.5 Μετρικές Υπολογισμού Απώλειας

Κατά τη διαδικασία της εκπαίδευσης ενός νευρωνικού δικτύου, είναι η όσο το δυνατόν μεγαλύτερη μείωση του σφάλματος πρόβλεψης. Για τον υπολογισμού του σφάλματος, χρησιμοποιούμε τις συναρτήσεις κόστους ή απώλειας οι οποίες υπολογίζουν τη διαφορά μεταξύ των πραγματικών τιμών των δεδομένων και των τιμών που προέκυψαν από την πρόβλεψη. Μερικές από αυτές τις συναρτήσεις είναι η συνάρτηση μέσου τετραγωνικού σφάλματος (Mean Squared Error), η συνάρτηση μέσου απόλυτου σφάλματος (Mean Absolute Error) η Categorical Cross-Entropy και

άλλες [16]. Παρακάτω, θα αναλύσουμε τις μετρικές που χρησιμοποιήσαμε για την αξιολόγηση του νευρωνικού δικτύου που αναλύσαμε.



Σχήμα 2.9: 5-Fold cross validation

2.5.1 Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error)

Η συγκεκριμένη συνάρτηση υπολογίζει το μέσο τετραγωνικό σφάλμα ανάμεσα στις τιμές που προέκυψαν από την πρόβλεψη και στις πραγματικές τιμές. Πρόκειται για το άθροισμα της τετραγωνικής διαφοράς μεταξύ των δύο τιμών διαιρώντας το με το συνολικό αριθμό των παραδειγμάτων. Η εξίσωση εκφράζεται με τον εξής τύπο:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.5}$$

Όπου, y_i είναι οι αρχικές τιμές και $\hat{y}_i$ είναι οι τιμές που προβλέφθηκαν.

2.5.2 Ποσοστό Παραμόρφωσης (Distortion Percentage)

Το Distortion Percentage, εκφράζει το λόγο παραμόρφωσης του πλέγματος που παράγει κατά την πρόβλεψη το μοντέλο σε σχέση με το πραγματικό mesh. Ο τύπος του είναι ο παρακάτω:

$$\text{DisPer} = 100 \cdot \frac{\|Aorig - Aapprox\|_F}{\|Aorig - Aavg\|_F} \tag{2.6}$$

Όπου, *Aorig* είναι ένας πίνακας που περιέχει όλες τις συντεταγμένες των κορυφών του mesh για κάθε frame του animation, *Aapprox* είναι ένας πίνακας που

περιέχει τις συντεταγμένες των κορυφών του mesh που προβλέφθηκαν από το μοντέλο για όλα τα frames και A_{avg} είναι ένας πίνακας που περιέχει σε κάθε στήλη του τη μέση τιμή των αρχικών συντεταγμένων των κορυφών σε όλα τα καρέ [17].

Κεφαλαιο 3

Υλοποιηση

Στο συγκεκριμένο κεφάλαιο θα ασχοληθούμε εκτενώς με τα βήματα και τους αλγορίθμους που υλοποιήθηκαν για την εκπόνηση της παρούσας διατριβής. Η λογική που ακολουθήσαμε για την παραγωγή αποτελεσμάτων είναι: η δημιουργία δεδομένων με τη χρήση του Blender, η δημιουργία των συνόλων δεδομένων train, validation και test. Στη συνέχεια, κατασκευάσαμε το μοντέλο το οποίο αφότου το εκπαιδεύσαμε στη συνέχεια το αξιολογήσαμε για διαφορετικές υπερπαραμέτρους και τέλος, αποθηκεύσαμε τα αποτελέσματα. Στις ενότητες που θα ακολουθήσουν, θα αναλύσουμε τα βήματα που αναφέραμε παραπάνω.

3.1 Προ-Επεξεργασία Δεδομένων

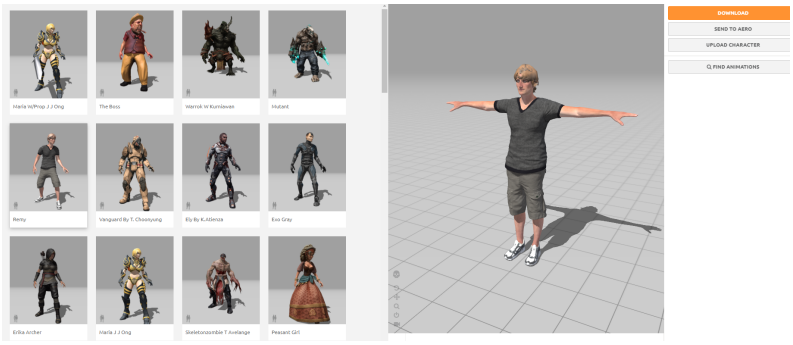
Ως προ-επεξεργασία δεδομένων (data preprocessing), ορίζουμε τη διαδικασία απόκτησης και επεξεργασίας των δεδομένων που θα τροφοδοτήσουμε ως είσοδο στο νευρωνικό μας δίκτυο [18]. Πρόκειται για ένα από τα πιο σημαντικά βήματα στην ανάπτυξη των δικτύων διότι μέσω αυτής, καθορίζεται η ποιότητα των δεδομένων.

Μερικές από τις τεχνικές προ-επεξεργασίας, είναι ο καθαρισμός των δεδομένων από θορύβους ή η κανονικοποίηση αυτών.

Στην παρούσα διατριβή, τα δεδομένα τα αντλούμε από την πλατφόρμα Mixamo και τα προ-επεξεργαζόμαστε στο Blender. Ειδικότερα, τα δεδομένα μας αποτελούνται από τα στοιχεία κινήσεων των χαρακτήρων.

3.1.1 Mixamo

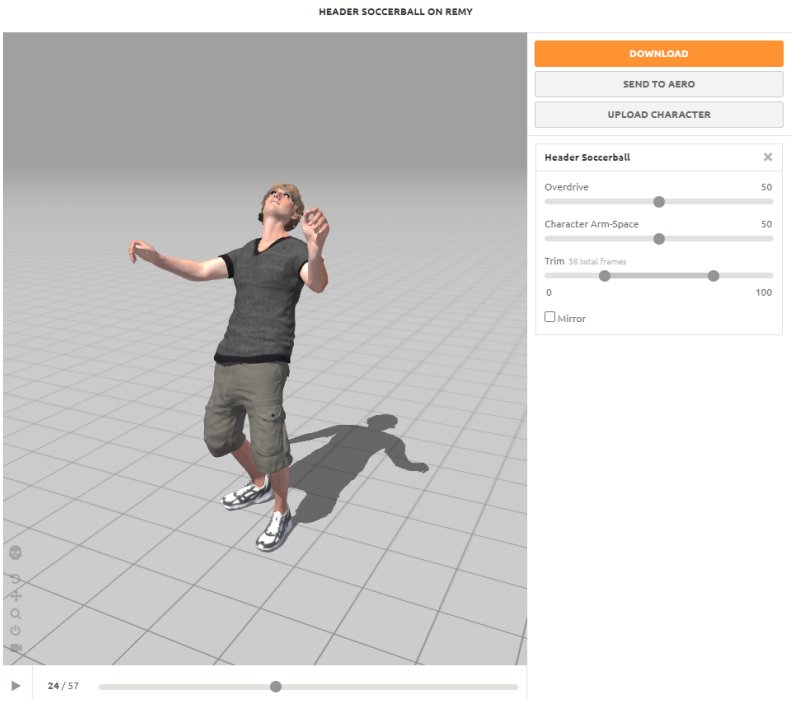
Το Mixamo, είναι μία online δωρεάν πλατφόρμα η οποία δίνει τη δυνατότητα στους χρήστες της να ανεβαίζουν ή να κατεβάζουν 3D μοντέλα χαρακτήρων και animations (Σχήμα 3.1). Η βιβλιοθήκη του παρέχει μία πληθώρα από χαρακτήρες και κινήσεις. Πιο συγκεκριμένα, κάθε μοντέλο της βιβλιοθήκης, μπορεί να συνδυαστεί με οποιοδήποτε animation επιλέξουμε.



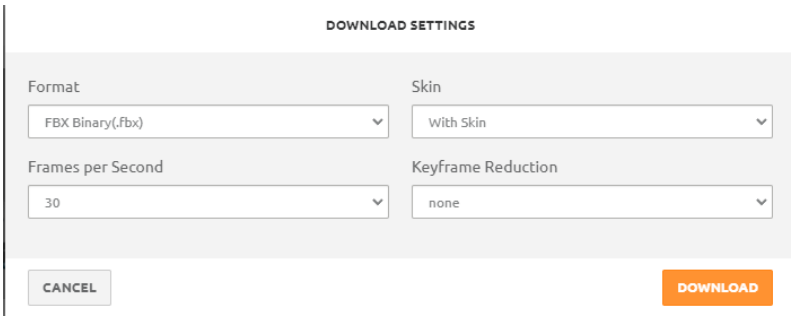
Σχήμα 3.1: Το περιβάλλον του mixamo

Στη συνέχεια, αφού διαλέξουμε το χαρακτήρα και το animation που θα τον συνοδεύει, καλούμαστε να ρυθμίσουμε μερικές από τις παραμέτρους του. Για παράδειγμα, ανάλογα με το animation που καλούμαστε να επεξεργαστούμε, μπορούμε να αλλάξουμε την απόσταση που έχουν τα χέρια από τον κορμό του σώματος, την στάση του σώματος, ακόμα και τα frames του animation που θα κρατήσουμε για αποθήκευση (Σχήμα 3.2). Όλα τα παραπάνω, είναι χαρακτηριστικά που προσδίδουν στο τρισδιάστατο μοντέλο μας ακόμα μεγαλύτερη ζωντάνια και αληθοφάνεια στις κινήσεις του.

Επίσης, η πλατφόρμα μας προσφέρει την επιλογή να δούμε σε προεπισκόπηση το τελικό animation πριν το κατεβάσουμε είτε με mesh είτε μόνο με τη σκελετική του δομή. Τέλος, καλούμαστε να ρυθμίσουμε παραμέτρους όπως τα frames και τον τύπο του αρχείου (όπως fbx, obj ή collada) πριν το κατεβάσουμε στον υπολογιστή μας όπως φαίνεται στο Σχήμα 3.3 που ακολουθεί.



Σχήμα 3.2: Οι παράμετροι που μπορούμε να αλλάξουμε στο animation



Σχήμα 3.3: Οι επιλογές για download

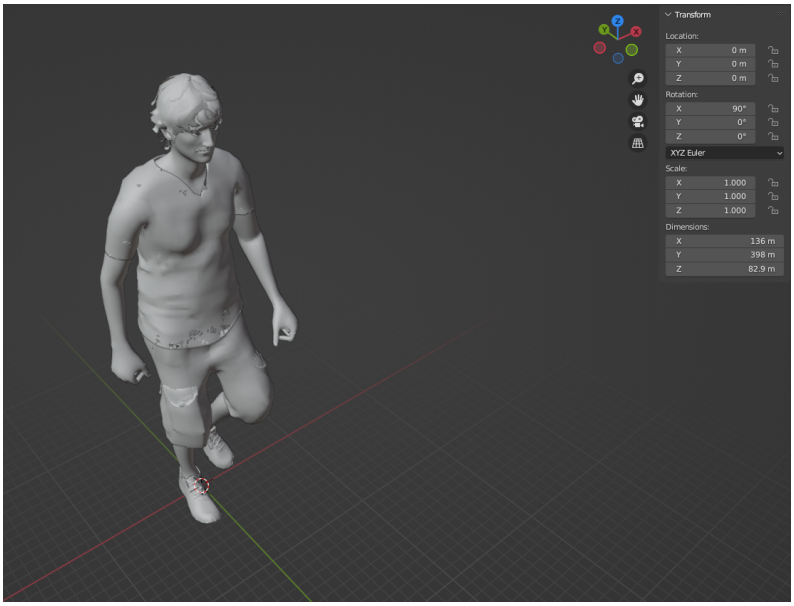
Όλα τα παραπάνω, καθιστούν το Mixamo μία εύκολα προσβάσιμη και εύχρηστη πλατφόρμα μέσω της οποίας θα αντλήσουμε τα δεδομένα μας τα οποία θα προ-επεξεργαστούμε στο Blender και ακολούθως θα τα δώσουμε ως είσοδο στο νευρωνικό μας δίκτυο.

3.1.2 Εισαγωγή Χαρακτήρων στο Blender

Παρακάτω, θα αναλύσουμε την επεξεργασία και παραγωγή των δεδομένων που θα αντλήσουμε από τα animations του Mixamo που χρησιμοποιήσαμε.

Αρχικά, πρέπει να εισάγουμε στο περιβάλλον του Blender το εκάστοτε animation. Για να το πετύχουμε αυτό, επιλέγουμε από τη γραφική διεπαφή *File* και στη

συνέχεια την επιλογή *Import*. Επιλέγουμε το αρχείο που θέλουμε να εισάγουμε και διαμορφώνουμε το μοντέλο ανάλογα με τις ανάγκες μας. Για πρακτικούς λόγους, στην παρούσα διατριβή, έχει οριστεί ως scale η τιμή 1 σε κάθε έναν από τους τρεις άξονες (x,y,z). Αυτό συμβαίνει έτσι ώστε, κάθε σκελετός να είναι πιο εμφανής (Σχήμα 3.4).



Σχήμα 3.4: Η μορφή του χαρακτήρα μετά το import

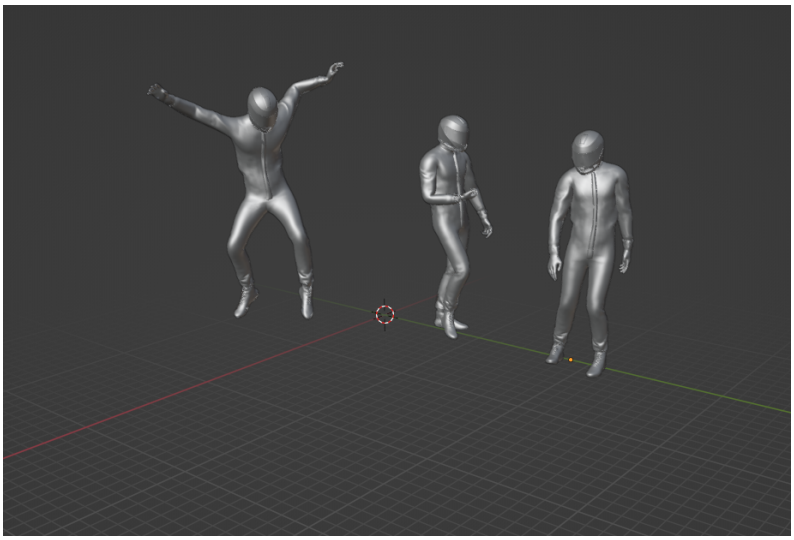
Επίσης, αξιοσημείωτο είναι το γεγονός πως το Mixamo μας προσφέρει μοντέλα με έτοιμους σκελετούς. Συνέπεια αυτού είναι αποφεύγουμε τη διαδικασία του rigging και της επεξεργασίας του σκελετού αφού η σκελετική δομή είναι ήδη έτοιμη.

Η διαδικασία εισαγωγής που περιγράψαμε, ακολουθήθηκε αρκετές φορές ώστε να αποκομίσουμε ένα σύνολο από χαρακτήρες οι οποίοι εκτελούν κινήσεις που μοιάζουν αρκετά μεταξύ τους (όπως διάφορες παραλλαγές όπου ο χαρακτήρας εκτελεί άλμα), είτε από κινήσεις που διαφέρουν (όπως ένα άλμα με το περπάτημα του χαρακτήρα).

Τα παραπάνω παραδείγματα όπως φαίνονται στο Σχήμα 3.5 χρησιμοποιήθηκαν για τη δημιουργία των συνόλων που στη συνέχεια θα αξιοποιηθούν από το νευρωνικό δίκτυο

3.1.3 Παραγωγή Δεδομένων από το Blender

Κάθε κίνηση που εφαρμόζεται σε έναν χαρακτήρα, είναι συνέπεια των κινήσεων των οστών του armature. Η απεικόνιση των μικρών κινήσεων και μετατοπίσεων



Σχήμα 3.5: Τρεις χαρακτήρες

που γίνονται στα οστά σε κάθε frame του animation γίνεται μέσω των πινάκων μετασχηματισμού. Ένας πίνακας μετασχηματισμού, είναι ένας 4X4 πίνακας όπου οι τρεις πρώτες γραμμές και στήλες μας παρέχουν πληροφορίες σχετικά με τη περιστροφική θέση του αντικειμένου, ενώ η τελευταία στήλη μας δίνει πληροφορίες για τη θέση του στον τρισδιάστατο χώρο. Η τέταρτη γραμμή του πίνακα, έχει συνήθως τις τιμές [0,0,0,1] και εξασφαλίζει πως ο εκάστοτε μετασχηματισμός θα εφαρμοστεί σωστά στις ομογενείς συντεταγμένες. Στην περίπτωσή μας, η τελευταία γραμμή του πίνακα μετασχηματισμού είναι περιττή καθώς δεν μας παρέχει κάποια πληροφορία χρήσιμη για το νευρωνικό μας δίκτυο, συνεπώς την αφαιρέσαμε.

Στη συνέχεια, αποκτώντας πρόσβαση στο mesh και τις πληροφορίες του, μέσω του Python API που προσφέρει το Blender, δημιουργούμε τους πίνακες των κορυφών του mesh. Για να το πετύχουμε αυτό, αρχικά υπολογίζουμε τους αντίστροφους πίνακες των τοπικών μετασχηματισμών των οστών, οι οποίοι με τη σειρά τους θα χρησιμοποιηθούν για να μετασχηματίσουν τις κορυφές του mesh από τον τοπικό χώρο των οστών, στον παγκόσμιο χώρο. Έπειτα, για κάθε κορυφή του mesh, υπολογίζουμε τα βάρη των οστών που επηρεάζουν την εκάστοτε κορυφή και υπολογίζουμε τις μετασχηματισμένες συντεταγμένες των κορυφών. Ακόμα, εφαρμόζουμε τους μετασχηματισμούς των οστών που έχουμε υπολογίσει στις συντεταγμένες των κορυφών μας και πολλαπλασιάζουμε την κάθε κορυφή με τον μετασχηματισμό που της αντιστοιχεί. Τέλος, πολλαπλασιάζουμε τις κορυφές που προέκυψαν με τα αντίστοιχα βάρη που υπολογίσαμε και καταλήγουμε στο επιθυμητό αποτέλεσμα.

Επιπλέον, για τον υπολογισμό του όγκου του πλέγματος (Σχήμα 3.6), εφαρμό-

```python
def volume_calculation(mesh):
    mesh_obj = bpy.data.objects[mesh]

    me=mesh_obj.data

    bm = bmesh.new()
    bm.from_mesh(me)
    bm.transform(mesh_obj.matrix_world)
    bmesh.ops.triangulate(bm, faces=bm.faces)

    volume = 0
    for f in bm.faces:
        v1 = f.verts[0].co
        v2 = f.verts[1].co
        v3 = f.verts[2].co
        volume += v1.dot(v2.cross(v3)) / 6

    print("Volume:", volume)
    bm.free()
    return volume


volume_calculation("Ch20")
```

Σχήμα 3.6: Block κώδικα υπολογισμού του όγκου

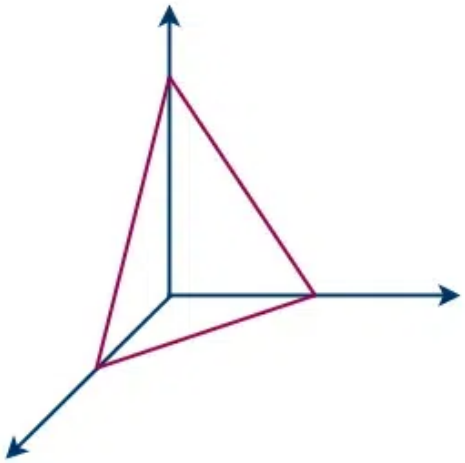
ζουμε στο πλέγμα τον πίνακα μετασχηματισμού και ακολούθως, τριγωνοποιούμε το πλέγμα. Τριγωνοποίηση, είναι η διαδικασία μετατροπής όλων των επιφανειών του πλέγματος σε τετράεδρα για τον καλύτερο υπολογισμό του όγκου. Τελικά, ο όγκος [19] υπολογίζεται με βάση την Εξίσωση 3.1 όπου $v1$,$v2$,$v3$ είναι οι συντεταγμένες του τρισδιάστατου διανύσματος που προκύπτει ανάμεσα σε κάθε κορυφή του τριγώνου και του σημείου (0,0,0) (Σχήμα 3.7).

$$\text{volume(V)} = \frac{1}{6} \sum v1 \cdot (v2 \times v3) \tag{3.1}$$

Κλείνοντας την ενότητα, αξίζει να αναφέρουμε πως όλα τα αποτελέσματα αποθηκεύονται σε αρχεία *npy* διότι, με αυτό τον τρόπο τα δεδομένα είναι πίνακες NumPy οι οποίοι εξυπηρετούν τις ανάγκες μας λόγω της ταχύτητας με την οποία γίνονται οι μαθηματικοί υπολογισμοί των πινάκων από τις έτοιμες συναρτήσεις της βιβλιοθήκης.

3.2 Δημιουργία του Νευρωνικού Δικτύου

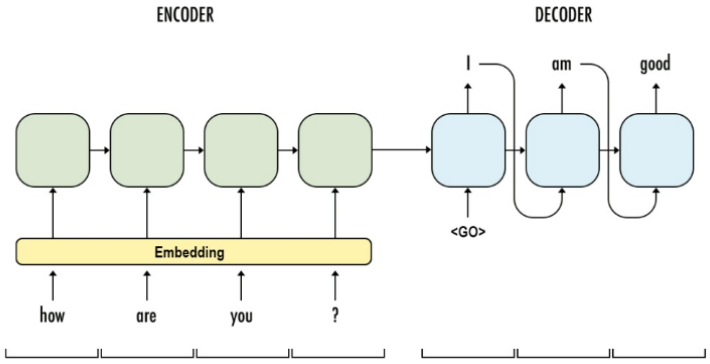
Στην παρούσα ενότητα, θα αναλύσουμε τον τρόπο με τον οποίο κατασκευάσαμε το νευρωνικό δίκτυο και εισάγαμε σε αυτό τα δεδομένα που αντλήσαμε από τα



Σχήμα 3.7: Ο σχηματισμός ενός τετράεδρου

animations.

Τα δεδομένα εισόδου και εξόδου του μοντέλου μας αφορούν τη μορφή και την κίνηση κάποιου χαρακτήρα γεγονός που τα καθιστά ακολουθιακής μορφής. Επιπλέον, η διατήρηση της πληροφορίας σε κάθε frame, είναι απαραίτητη και καθιστά την ανάγκη της ύπαρξης ενός επαναλαμβανόμενου νευρωνικού δικτύου απαραίτητη. Για αυτό το λόγο, επιλέξαμε μια αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή (encoder-decoder) σε συνδυασμό με χρήση ενός LSTM. Στο Σχήμα 3.8 που ακολουθεί, παρατηρούμε μία απλή αναπαράσταση της αρχιτεκτονικής ενός LSTM δικτύου για την επεξεργασία φυσικής γλώσσας.



Σχήμα 3.8: Δίκτυο LSTM

3.2.1 Αρχιτεκτονική του Δικτύου

Η αρχιτεκτονική που χρησιμοποιήσαμε για την επίτευξη των στόχων του μοντέλου μας αποτελείται από τέσσερα κύρια σημεία:

- **Κωδικοποιητής Οστών (Class Encoder_bones)**: Πρόκειται για μία κλάση η οποία κωδικοποιεί τα δεδομένα των οστών που εισάγονται στο δίκτυο. Αρχικά, παίρνει ως είσοδο τον αριθμό των features για κάθε χρονικό βήμα της ακολουθίας, τον αριθμό των units του κρυμμένου επιπέδου του δικτύου και τον αριθμό των επιπέδων του LSTM. Έπειτα, η μέθοδος *forward* αρχικοποιεί τις κρυφές καταστάσεις σε μηδενικά tensors μεγέθους όσο το batch size και εκτελεί την προώθηση των δεδομένων μέσω του LSTM. Τέλος, η έξοδος *output _encoder* που παίρνουμε από τον κωδικοποιητή περνά από μία συνάρτηση ενεργοποίησης tanh προκειμένου οι τιμές της να κανονικοποιηθούν στο διάστημα (-1,1) και το μοντέλο μας να γίνει πιο σταθερό και αποδοτικό.

- **Κωδικοποιητής Πρώτου Frame (Class Encoder_firstPose)**: Είναι μία κλάση η οποία υλοποιεί ένα FeedForward νευρωνικό δίκτυο για την επεξεργασία των δεδομένων της πρώτης πόζας του πρώτου καρέ του mesh του τρισδιάστατου χαρακτήρα που δέχεται το δίκτυό μας. Εφόσον η πληροφορία μας αφορά μόνο την πρώτη πόζα του πρώτου frame, καταλαβαίνουμε πως δεν πρόκειται για ακολουθιακά δεδομένα όπως θα συνέβαινε εάν, δεχόμασταν ως είσοδο την πρώτη πόζα για συνεχόμενα ή για όλα τα καρέ του animation. Για τον λόγο αυτό, επεξεργαζόμαστε τα δεδομένα με δύο γραμμικά επίπεδα τα αποτελέσματα των οποίων κανονικοποιούνται με τη χρήση της tanh συνάρτησης ενεργοποίησης.

- **Μηχανισμός Προσοχής (Class Attention Mechanism)**: Ο μηχανισμός προσοχής είναι ένα χαρακτηριστικό των δικτύων το οποίο χρησιμοποιείται για να επιτρέπει στο μοντέλο να επικεντρώνεται σε διαφορετικά σημεία των δεδομένων που δέχεται ως είσοδο. Στην παρούσα κλάση, ο μηχανισμός δέχεται ως είσοδο τα δεδομένα που παρήχθησαν από τους δύο κωδικοποιητές και τα συνενώνει. Έπειτα, η ενωμένη πληροφορία, περνά από ένα γραμμικό στρώμα και μία συνάρτηση ενεργοποίησης tanh. Τέλος, η κανονικοποιημένη πληροφορία περνά πάλι από ένα γραμμικό επίπεδο με σκοπό να παράξει τα *βάρη προσοχής*. Τα βάρη προσοχής είναι αυτά που καθορίζουν πόσο σημαντική είναι για τον

μηχανισμό μία πληροφορία. Επιπρόσθετα, τα βάρη περνούν και αυτά από τη συνάρτηση ενεργοποίησης Softmax ώστε να εξομαλυνθούν οι τιμές τους. Τέλος, τα κανονικοποιημένα βάρη πολλαπλασιάζονται με τις ενωμένες πληροφορίες έτσι ώστε να καθοριστεί το πόσο σημαντική ή όχι είναι η καθεμία από αυτές.

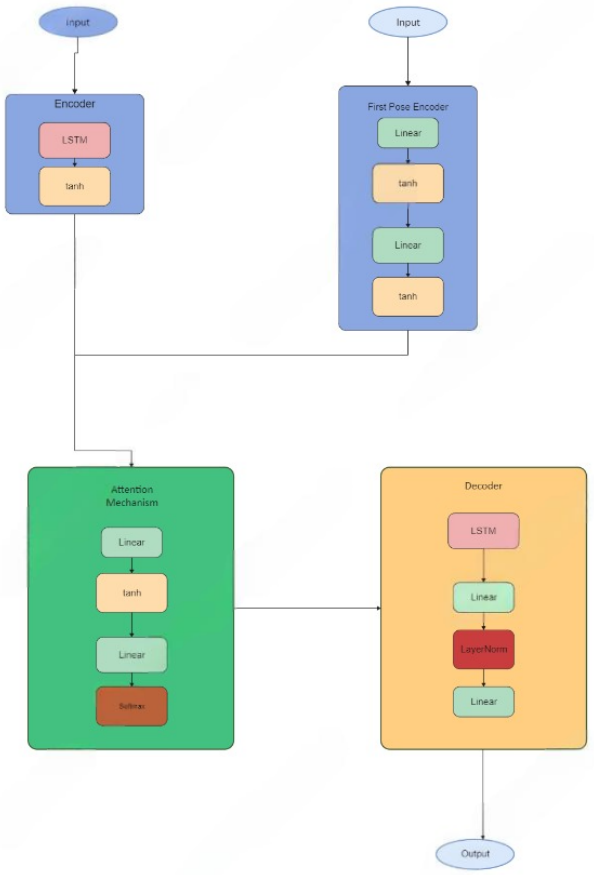
- **Αποκωδικοποιητής (Class Decoder)**: Κατά τη διαδικασία της αποκωδικόποιησης, το νευρωνικό μας δίκτυο μαθαίνει να παράγει τα δεδομένα που επιθυμούμε να λάβουμε ως αποτέλεσμα. Στη συγκεκριμένη περίπτωση, ο αποκωδικοποιητής δέχεται ως είσοδο το αποτέλεσμα που παράγεται από τον απολογισμό του μηχανισμού προσοχής. Όπως διακρίνουμε στο Σχήμα 3.9, η είσοδος μας θα περάσει αρχικά από ένα δίκτυο LSTM του οποίου η έξοδος θα χρησιμοποιηθεί από δύο γραμμικά επίπεδα για το μετασχηματισμού του μεγέθους της πληροφορίας. Η έξοδος του αποκωδικοποιητή πρέπει να έχει το κατάλληλη μέγεθος δηλαδή, πρέπει να είναι ίσο με το γινόμενο των καρέ που έχει το animation επί τον συνολικό αριθμό των κορυφών του.

3.2.2 Υπερπαράμετροι και Εκπαίδευση του Δικτύου

Για να αρχικοποιηθεί και να εκπαιδευτεί σωστά το νευρωνικό δίκτυο με τον μηχανισμό προσοχής που αναπτύξαμε, πρέπει να προσδιοριστούν οι υπερπαράμετροί του. Οι υπερπαράμετροι, είναι ορισμένα στοιχεία του νευρωνικού δικτύου των οποίων οι τιμές εξαρτώνται από το χρήστη και επηρεάζουν την ταχύτητα αλλά και την ποιότητα με την οποία προβλέπει το δίκτυο. Μερικές από αυτές είναι: το *batch_size*, ο *ρυθμός μάθησης* και ο *αριθμός των εποχών*.

Στη δική μας περίπτωση, οι παράγοντες που επηρεάζουν το αποτέλεσμα είναι το μέγεθος της εισόδου (Input_size), ο αριθμός των κρυμμένων επιπέδων (Hidden_size), το μέγεθος της εξόδου (Output_size) και ο αριθμός των επιπέδων του LSTM. Αναλυτικότερα, η είσοδος ισούται με το γινόμενο του αριθμού των οστών επί τις τιμές τους, η έξοδος είναι ο συνολικός αριθμός όλων των vertices του mesh ενώ τα κρυμμένα επίπεδα και τα επίπεδα του LSTM έχουν καθοριστεί με τις τιμές 40 και 2 αντίστοιχα.

Επίσης, σημαντικό ρόλο παίζει τόσο ο *optimizer* που έχουμε επιλέξει (*Adam*) όσο και ο ρυθμός με τον οποίο μαθαίνει αυτός. Στην περίπτωσή μας, ο ρυθμός μάθησης κυμαίνεται μεταξύ 0.00001 και 0.001 για όλα τα μοντέλα κίνησης είτε κρατάμε τον



Σχήμα 3.9: Διάγραμμα δικτύου

όγκο των mesh είτε όχι. Κατά την εκπαίδευση του εκάστοτε μοντέλου, έχουμε προσθέσει στον optimizer και μία ποινή (wight decay) για τα βάρη, της τάξης του 10^{-5}. Η ποινή αυτή, προστίθεται στην απώλεια που υπολογίζεται στο τέλος κάθε εποχής ώστε το νευρωνικό δίκτυο να μαθαίνει και να ενημερώνει με πιο ορθό τρόπο τα βάρη κατά την διαδικασία της εκπαίδευσης. Αποτέλεσμα της προσθήκης αυτής της ποινής είναι η αποφυγή φαινομένων, υπερεκπαίδευσης και η βελτίωση της γενίκευσης καθώς τα μεγάλα βάρη που μπορεί να προκύψουν, καταλήγουν να περιορίζονται σε κανονικοποιημένες τιμές ενώ, η μείωση του κόστους γίνεται σταθερά και με μικρότερες μεταβολές.

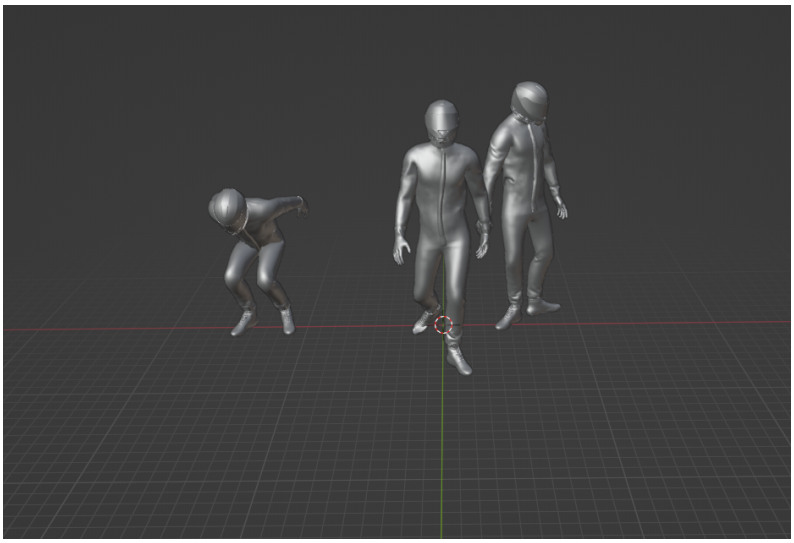
Ακόμη, ξεχωριστή είναι η σημασία του όγκου που έχει ένα mesh (Σχήμα 3.10) καθώς τον προσθέτουμε σε μεταγενέστερο στάδιο στον υπολογισμό της απώλειας ώστε το τελικό αποτέλεσμα που θα πάρουμε να είναι όσο το δυνατόν πιο εύρωστο.

Τέλος, σημειώνουμε ότι το σύνολο των εποχών κατά την εκπαίδευση και αξιολόγηση καθορίστηκε στην τιμή 10000 (*number of epochs = 10000*), καθώς μετά από

πειράματα καταλήξαμε στο γεγονός πως ο συγκεκριμένος αριθμός εποχών είναι ένα ιδανικό σημείο όπου η απώλεια έχει έρθει στα επιθυμητά επίπεδα και το μοντέλο έχει εκπαιδευτεί ικανοποιητικά στα δεδομένα που δέχεται ενώ και ο χρόνος εκτέλεσης της εκπαίδευσης είναι αρκετά καλός.

Σχήμα 3.10: Ο όγκος του mesh που χρησιμοποιήσαμε

Όπως αναφέραμε προηγουμένως, για να εκπαιδευτεί ένα δίκτυο, πρέπει να τροφοδοτηθεί με δεδομένα. Στην παρούσα διατριβή, τα δεδομένα που δόθηκαν στο δίκτυο έχουν χωριστεί με βάση τον τύπο κίνησης που εκτελεί κάθε σκελετός. Για παράδειγμα, μία κατηγορία κινήσεων είναι οι χαρακτήρες που περπατούν ενώ μία άλλη είναι αυτή των χαρακτήρων που εκτελούν κάποια στροφή. Στο Σχήμα 3.11 που ακολουθεί παρατηρούμε τρεις χαρακτήρες που εκτελούν από μία διαφορετική κίνηση.



Σχήμα 3.11: Διάγραμμα δικτύου

Γενικότερα, ένα σύνολο κινήσεων αποτελείται μόνο από κινήσεις του ίδιου τύπου, όπως το άλμα, και στη συνέχεια χωρίζεται σε σύνολα εκπαίδευσης, αξιολόγησης και επικύρωσης. Για να μην υπάρχει διαφορά ανάμεσα στα δεδομένα έχουμε προκαθορίσει μέσω του Blender πως όλες οι κινήσεις θα εκτελούνται για τον ίδιο αριθμό από frames. Επιπρόσθετα, αξίζει να σημειώσουμε πως τα σύνολα επικύρωσης και αξιολόγησης αποτελούνται από ίδιου τύπου κινήσεις με αυτή στην οποία εκπαιδεύεται

το δίκτυο ωστόσο, δεν ανήκουν στο σύνολο εκπαίδευσης.

Κατά την διαδικασία της τροφοδότησης, το δίκτυο δέχεται δεδομένα σε batches ώστε να λαμβάνει ταυτόχρονα πολλαπλά παραδείγματα. Με τον τρόπο αυτό, βοηθούμε το μοντέλο να αποφύγει φαινόμενο υπερπροσαρμογής όπου ουσιαστικά το μοντέλο μας έχει μάθει να προβλέπει πολύ σωστά μόνο για ένα συγκεκριμένο σύνολο παραδειγμάτων.

Μετά την τροφοδότηση, το μοντέλο εκπαιδεύεται με τα δεδομένα που το παρήχθησαν και προσπαθεί να παράγει αποτελέσματα που θα είναι όσο το δυνατόν πιο κοντά σε αυτά που στοχεύουμε. Στη συνέχεια, το μοντέλο ανάλογα με τις συναρτήσεις λάθους ενημερώνει τις παραμέτρους του. Όσο πιο μικρή είναι η απώλεια που προκύπτει τόσο καλύτερες είναι οι προβλέψεις που έχουν πραγματοποιηθεί. Σε αυτό το σημείο αξίζει να αναφέρουμε πως στη συνάρτηση λάθους έχει προστεθεί η διατήρηση του όγκου. Αναλυτικότερα, το μοντέλο μαζί με τις υπόλοιπες παραμέτρους, προσπαθεί να προβλέψει και τον όγκο του mesh που κάθε φορά προβλέπεται ο οποίος επιθυμούμε να είναι όσο πιο κοντά γίνεται στον πραγματικό όγκο. Έπειτα, ο όγκος προστίθεται μαζί με ένα βάρος w στην απώλεια της κάθε εποχής σύμφωνα με την εξίσωση Εξίσωση 3.2 που έπεται.

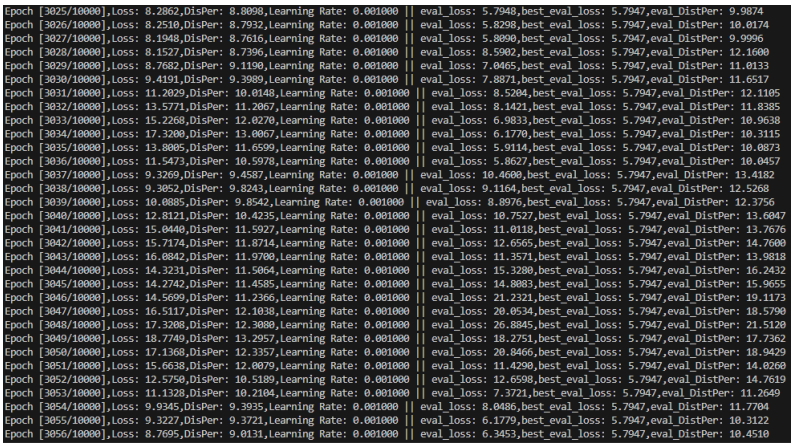
$$\text{total\_loss} = \text{epoch\_loss} + x \cdot w \tag{3.2}$$

Όπου x:

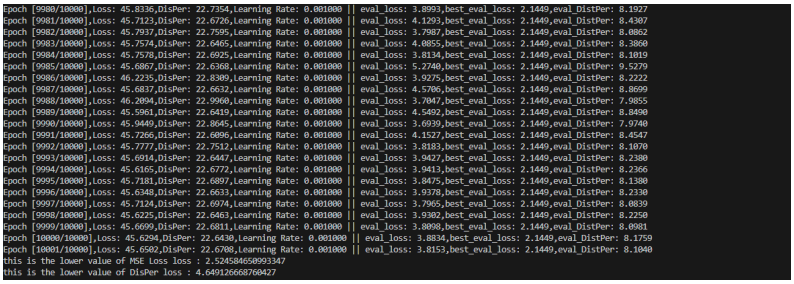
$$\text{x} = \frac{\text{volume\_predicted}}{\text{target \_volume}} \tag{3.3}$$

Σε κάθε εποχή, μετά τη διαδικασία της εκπαίδευσης πραγματοποιείται και μία διαδικασία αξιολόγησης του μοντέλου στο εκάστοτε σύνολο επικύρωσης. Το μοντέλο, καλείται να δημιουργήσει αποτελέσματα βάσει του συνόλου επικύρωσης. Έτσι, μπορούμε να αξιολογήσουμε την απόδοση του μοντέλου μας στο να πραγματοποιεί προβλέψεις με δεδομένα τα οποία του είναι άγνωστα. Ακόμα, τα αποτελέσματα κάθε εποχής εμφανίζονται στο τερματικό μας προκειμένου να έχουμε μία καλύτερη εικόνα του τι συμβαίνει σε κάθε εποχή (Σχήμα 3.12, Σχήμα 3.13).

Τέλος, έχουμε εφαρμόσει μία τεχνική κατά την οποία, αποθηκεύουμε το μοντέλο με την καλύτερη απόδοση στο σύνολο επικύρωσης. Η διαδικασία αυτή, επαναλαμβάνεται για όλες τις εποχές που εκπαιδεύεται το μοντέλο και στην περίπτωση που η απώλεια είναι μικρότερη συγκριτικά με την ήδη υπάρχουσα καλύτερη απώλεια, τότε



Σχήμα 3.12: Παράδειγμα terminal



Σχήμα 3.13: Το terminal στο τέλος της εκπαίδευσης

το μοντέλο αποθηκεύεται ως καλύτερο. Η συγκεκριμένη προσέγγιση διαφέρει από την τεχνική του early stopping επειδή, η διαδικασία εκπαίδευσης δεν διακόπτεται ποτέ.

3.3 Σύνθεση Κινήσεων

Ως σύνθεση κινήσεων, προσδιορίζουμε την ανάμειξη κινήσεων. Κατά τη διαδικασία της σύνθεσης, αντλούμε δεδομένα από διαφορετικές κινήσεις που θα συνδυαστούν με σκοπό να δημιουργηθεί μία νέα κίνηση για τον χαρακτήρα. Οι κινήσεις που θα χρησιμοποιήσουμε για τη σύνθεση, ανήκουν στα σύνολα των κινήσεων που έχουμε ήδη αντλήσει από το Blender και αποσκοπούν στη δημιουργία μιας άγνωστης και νέας κίνησης όπως το άλμα μαζί με το περπάτημα.

3.3.1 Υλοποίηση Σύνθεσης Δίχως Κατηγοριοποίηση

Για την υλοποίηση της σύνθεσης, χρησιμοποιήσαμε μία προσέγγιση κατά την οποία, εκμεταλλευόμαστε τα ήδη εκπαιδευμένα δίκτυα και τα αποτελέσματα τους. Ειδικότερα, πρέπει να διασφαλίσουμε πως τα μοντέλα που θα χρησιμοποιηθούν για τη σύνθεση, θα έχουν τις ίδιες παραμέτρους εισόδου και εξόδου.

Ακολούθως, η διαδικασία εκπαίδευσης για κάθε μοντέλο είναι ίδια με αυτή που έχουμε επεξηγήσει ανωτέρω. Μετά την εκπαίδευση των μοντέλων, ακολουθεί η τροφοδότηση των κωδικοποιητών με τα δεδομένα των κινήσεων που θέλουμε να χρησιμοποιήσουμε στη σύνθεση. Ο εκάστοτε κωδικοποιητής θα λάβει δεδομένα με τα οποία έχει εκπαιδευτεί και μέσω των οποίων θα εξάγει τη χρήσιμη πληροφορία.

Μετέπειτα, λαμβάνουμε την κωδικοποιημένη πληροφορία που παράγει ο κάθε κωδικοποιητής και συγχωνεύουμε τα χαρακτηριστικά που αυτά που έχουν εξαχθεί. Τελικά, τα συγχωνευμένα δεδομένα δίνονται στους ήδη υπάρχοντες αποκωδικοποιητές με σκοπό να παράξουμε το mesh του νέου animation που θα αποτελεί την καινούργια κίνηση.

3.3.2 Υλοποίηση Σύνθεσης με Κατηγοριοποίηση

Μία άλλη προσέγγιση στο ζήτημα της σύνθεσης των κινήσεων είναι με τη χρήση της κατηγοριοποίησης. Αυτή η μέθοδος, βασίζεται στην εισαγωγή ενός διανύσματος το οποίο κατηγοριοποιεί την κάθε κίνηση ανάλογα με την κατηγορία στην οποία ανήκει. Τα συγκεκριμένα διανύσματα, είναι της μορφής [χ,χ,χ,χ,χ] όπου το χ μπορεί να λάβει τιμές 0 ή 1. Αν κάποια το χ λάβει την τιμή 1, αυτό σημαίνει πως η αντίστοιχη κίνηση εκτελείται. Παραδείγματος χάριν, το άλμα αντιπροσωπεύεται από το διάνυσμα [0.0,0.0,0.0,1.0,0.0]

Για τη συγκεκριμένη προσέγγιση, η κύρια αλλαγή που έγινε στο μοντέλο του LSTM είναι η προσθήκη της πληροφορίας του διανύσματος στο σύνολο της εκπαίδευσης. Εν κατακλείδι, μπορούμε να καταλήξουμε στο συμπέρασμα πως η μοναδική διαφορά που έχει η συγκερκιμένη προσέγγιση συγκριτικά με την απλή αρχιτεκτονική LSTM που παρουσιάσαμε παραπάνω είναι η αντιμετώπιση των δεδομένων κατηγοριοποίησης ως ακολουθιακά δεδομένα.

Κεφαλαιο 4

Αποτελεσματα

Στο παρόν κεφάλαιο θα αναλύσουμε και θα σχολιάσουμε τα αποτελέσματα που προέκυψαν από την υλοποίηση της παρούσας διατριβής. Πρώτιστα, αξίζει να αναφερθούμε όμως στις βιβλιοθήκες και τα εργαλεία που χρησιμοποιήσαμε καθώς και στα χαρακτηριστικά του υπολογιστή στον οποίο έλαβε χώρα η ανάπτυξη του δικτύου. Χρησιμοποιήσαμε τις εκδόσεις 3.10 και 3.4 της γλώσσας Python και του Blender αντίστοιχα. Επίσης, έγινε χρήση της βιβλιοθήκης PyTorch η οποία είναι μια βιβλιοθήκη ανοιχτού κώδικα που χρησιμοποιείται για την ανάπτυξη εφαρμογών βαθειάς μάθησης.

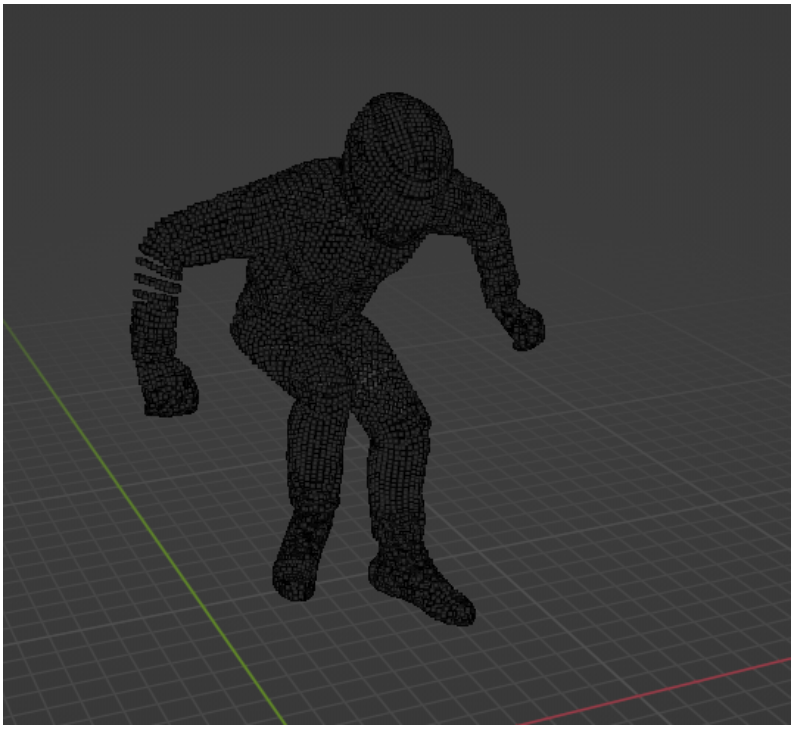
Σημειώνουμε, ακόμα, πως για την αύξηση της ταχύτητας των υπολογισμών, έγινε χρήση της βιβλιοθήκης CUDA η οποία έχει αναπτυχθεί από την Nvidia με σκοπό να εκτελούνται στις κάρτες γραφικών της υπολογισμοί, ώστε να γίνεται εκμετέλλευση της υπολογιστικής τους ισχύος. Ενδεικτικά, η χρήση της κάρτας γραφικών του συστήματος, επιταχύνει την διάρκεια εκπαίδευσης για 10000 εποχές από κατά προσέγγιση 50 λεπτά, σε περίπου 18 λεπτά για ένα μοντέλο εκπαίδευσης στην κίνηση του άλματος δίχως διατήρηση του όγκου. Επιπλέον, μπορούμε να σημειώσουμε ότι η διάρκεια της εκπαίδευσης ενός μοντέλου με διατήρηση του όγκου για 10000 εποχές χρειάζεται περίπου 6 ώρες με τη χρήση της GPU ενώ, δίχως αυτή απαιτούνται

περίπου 10 ώρες. Βέβαια, αυτό δεν σημαίνει πως η υλοποίηση λειτουργεί μόνο για συστήματα που έχουν μία κάρτα γραφικών της Nvidia. Στην περίπτωση αυτή, όλοι οι υπολογισμοί είναι ρυθμισμένοι να γίνονται μέσω του επεξεργαστή.

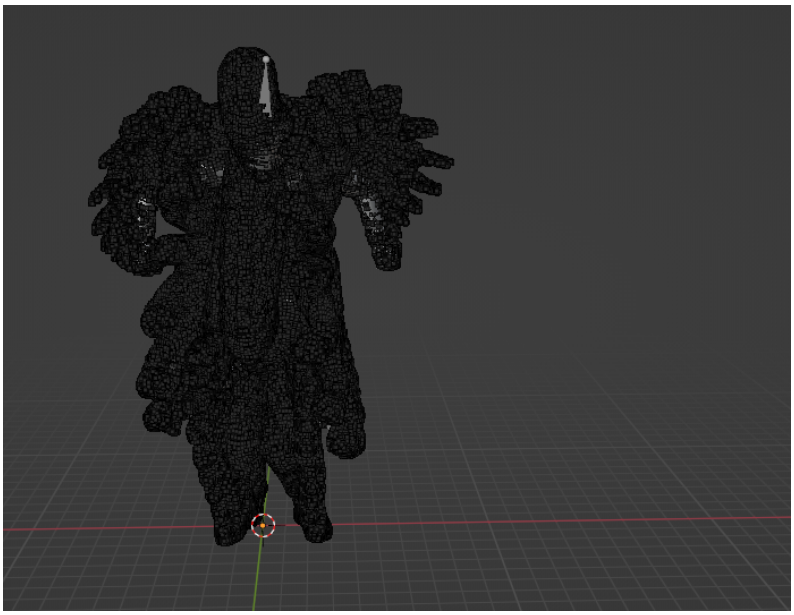
Τέλος, σημειώνουμε ότι η εργασία καθώς και όλα της τα πειράματα, έγιναν σε έναν υπολογιστή με έναν επεξεργαστή Intel i7-4790 ο οποίος λειτουργεί στα 3.6 GHz, μία κάρτα γραφικών Gigabyte GeForce GTX 1660 Super στα 6GB και 16 GB DDR3 μνήμη RAM.

4.1 Αξιολόγηση Κίνησης Χαρακτήρων

Όπως έχουμε αναφέρει σε προηγούμενο κεφάλαιο, με τη χρήση του Python API που προσφέρει το Blender, αναπτύξαμε κώδικα, μέσω του οποίου μπορούμε να ελέγξουμε την ορθότητα των αποτελεσμάτων μας. Στο Σχήμα 4.1, βλέπουμε τα αποτελέσματα που παράγονται κατά τη διαδικασία της εξαγωγής των πληροφοριών του mesh για όλα τα καρέ. Πιο συγκεκριμένα, κάθε κουκίδα που βλέπουμε είναι η αντίστοιχη κορυφή του πλέγματος του τρισδιάστατου χαρακτήρα. Επιπλέον, στο Σχήμα 4.2 παρατηρούμε την απεικόνιση των κουκίδων για όλα τα frames.



Σχήμα 4.1: Η απεικόνιση για ένα frame



Σχήμα 4.2: Η απεικόνιση για πολλαπλά frames

4.2 Αξιολόγηση Αποτελεσμάτων Δικτύου

Κατά την εκπόνηση της διατριβής, η απεικόνιση και αξιολόγηση των αποτελεσμάτων των νευρονικών δικτύων, αποτελεί σημαντικό παράγοντα διότι, με βάση τα αποτελέσματα που προέκυπταν μπορούσαμε να διακρίνουμε αν τα δίκτυα και οι εκάστοτε υπερπαράμετροί τους, ήταν σωστές ή χρειάζονταν διόρθωση.

Η αξιολόγηση της εκπαιδευτικής διαδικασίας γίνεται με τα αποτελέσματα των μετρικών απώλειας που προκύπτουν στο τέλος κάθε εποχής. Σε αυτό το σημείο, άξιο αναφοράς είναι το γεγονός πως για την εκπαίδευση του εκάστοτε μοντέλου, τα σύνολα εκπαίδευσης αποτελούνταν από κινήσεις της ίδιας κατηγορίας, όπως το άλμα, αλλά διαφορετικές μεταξύ τους. Στη συνέχεια, η αξιολόγηση των αποτελεσμάτων, έγινε με σύνολα τα οποία περιέχουν δεδομένα που ανήκουν στην ίδια κίνηση αλλά το μοντέλο δεν έχει εκπαιδευτεί σε αυτά. Μόνο στην περίπτωση της σύνθεσης, τα σύνολα εκπαίδευσης και επικύρωσης εμπεριέχουν λίστες δεδομένων και από τις δύο κινήσεις που συνδυάζουμε.

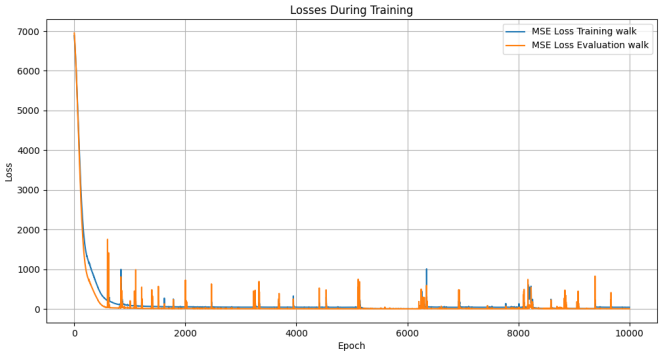
Οι δύο μετρικές που χρησιμοποιούμε είναι το MSE και το Distortion Percentage τα οποία τα αναλύσαμε προηγουμένως. Τέλος, αξίζει να αναφέρουμε πως στη συνέχεια θα παρουσιάσουμε και αποτελέσματα για τις ίδιες μετρικές με την τεχνική της διατήρησης του όγκου.

4.2.1 Αποτελέσματα Δίχως Διατήρηση Όγκου

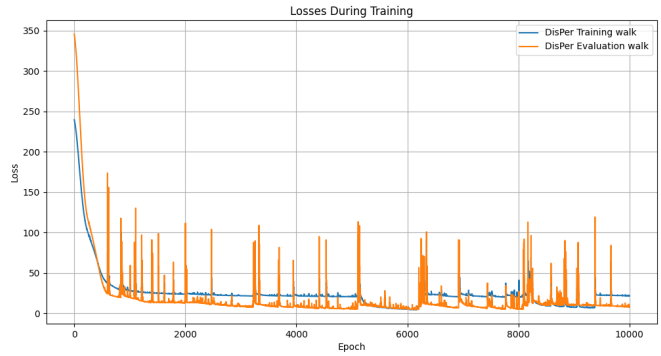
Στη συγκεκριμένη ενότητα, θα προβάλλουμε τα αποτελέσματα του δικτύου κατά την αξιολόγηση δίχως την διατήρηση του όγκου. Στο Σχήμα 4.3 βλέπουμε τον τρόπο με τον οποίο μειώνεται το μέσο τετραγωνικό λάθος στο σύνολο των δεδομένων για το περπάτημα. Επιπλέον, στο Σχήμα 4.4 παρατηρούμε τη μείωση του Distortion Percentage στο σύνολο των εποχών για τις οποίες εκπαιδεύτηκε το μοντέλο.

Βλέπουμε από τις δύο γραφικές παραστάσεις και από τα αποτελέσματα του τερματικού πως η ελάχιστη τιμή που πήρε το MSE ήταν: 1.7641946375370026 ενώ το DisPer:4.424110925577921.

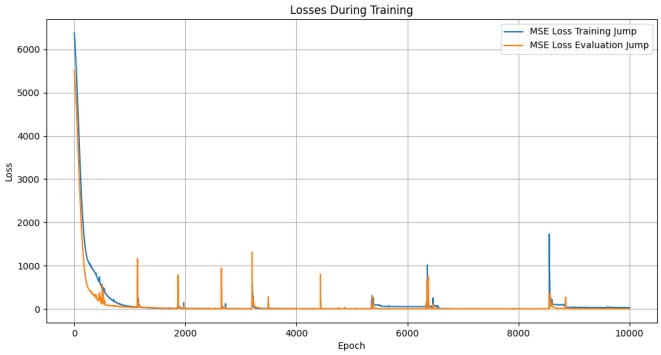
Κατόπιν στο Σχήμα 4.5 και στο Σχήμα 4.6 παρουσιάζουμε τα αντίστοιχα αποτελέσματα που προκύπτουν από την κίνηση που εκτελεί ο χαρακτήρας για να πραγματοποιήσει άλμα.



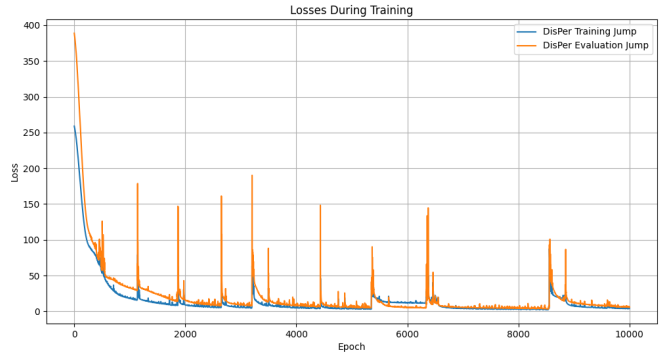
Σχήμα 4.3: MSE loss για περπάτημα



Σχήμα 4.4: Distortion percentage για περπάτημα



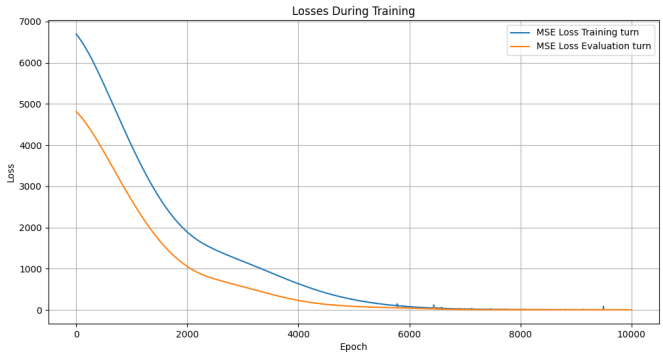
Σχήμα 4.5: MSE loss για άλμα



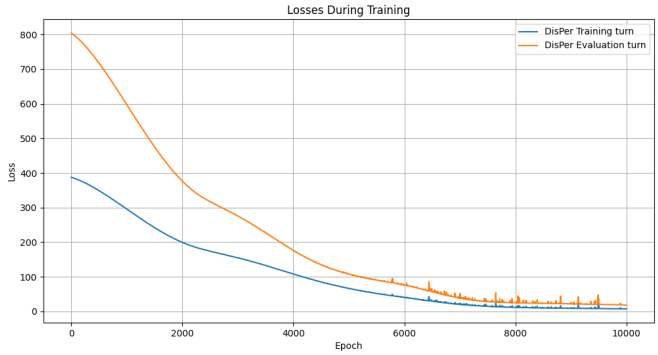
Σχήμα 4.6: Distortion percentage για άλμα

Πιο συγκεκριμένα, η ελάχιστη τιμή του MSE είναι: 0.4677031859755516 και του DisPer: 2.1740565069863047. Παρατηρούμε πως οι τιμές στη συγκεκριμένη περίπτωση είναι μικρότερες γεγονός που σημαίνει πως το δίκτυο έφτασε να εκπαιδεύεται καλύτερα.

Τέλος, παρουσιάζουμε τις απώλειες που προέκυψαν στην κίνηση όταν το mesh εκτελεί στροφή προς κάποια κατεύθυνση (Σχήμα 4.7, Σχήμα 4.8).



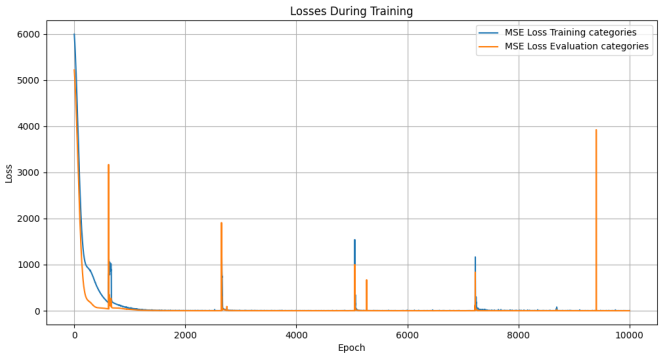
Σχήμα 4.7: MSE loss για στροφή



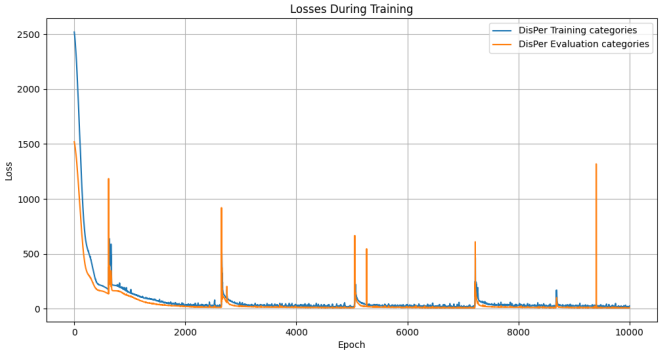
Σχήμα 4.8: Distortion percentage για στροφή

Για κάθε μία από τις παραπάνω εκπαιδεύσεις, αξίζει να σημειώσουμε πως το εκάστοτε μοντέλο είχε ως υπερπαραμέτρους: *batch_size = 2*, *Input_size = 624*, *Output_size = 21729*, *Hidden layers = 40* και *learning rate* μεταξύ των τιμών 10^{-5} έως 10^{-3}.

Στη συνέχεια, θα παρουσιάσουμε τα αποτελέσματα που προέκυψαν από τη σύνθεση κίνησης με την κατηγοριοποίηση αυτών (Σχήμα 4.9, Σχήμα 4.10). Όπως βλέπουμε, η ελάχιστη τιμή του MSE είναι: 0.18996027261018752 και του DisPer: 14.040560302718252.



Σχήμα 4.9: MSE loss για σύνθεση



Σχήμα 4.10: Distortion percentage για σύνθεση

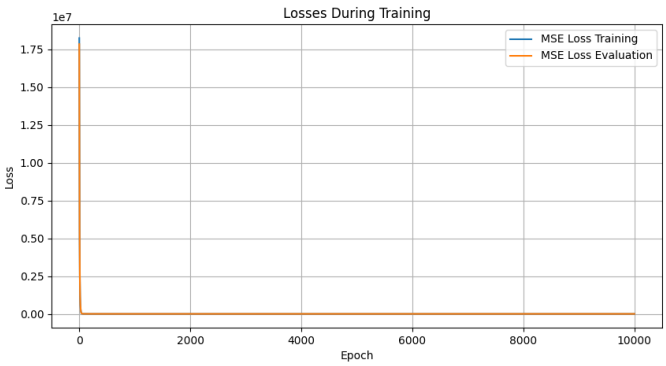
Στον παρακάτω Πίνακα 4.1, θα δούμε εκτεταμένα τις δύο μετρικές που χρησιμοποιήσαμε για την επικύρωση των αποτελεσμάτων μας

Πίνακας 4.1: Πίνακας τιμών MSE loss και distribution percentage.

Κίνηση	MSE	DisPer
Περπάτημα	1.764	4.424
Άλμα	0.468	2.174
Στροφή	2.519	18.394

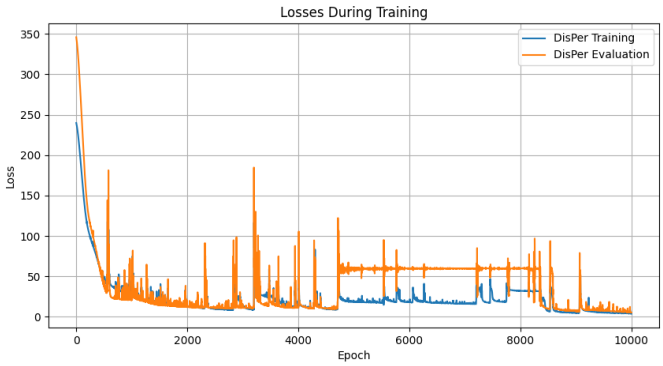
4.2.2 Αποτελέσματα με Διατήρηση Όγκου

Στην υποενότητα που ακολουθεί, θα παρουσιάσουμε μερικά από τα αποτελέσματα που προέκυψαν στις προβλέψεις όπου διατηρήσαμε τον όγκο στη συνάρτηση απώλειας. Στο Σχήμα 4.11 βλέπουμε τον τρόπο με τον οποίο μειώνεται το μέσο τετραγωνικό λάθος στο σύνολο των δεδομένων για το περπάτημα. Επιπλέον, στο Σχήμα 4.12 παρατηρούμε τη μείωση του Distortion Percentage στο σύνολο των εποχών για τις οποίες εκπαιδεύτηκε το μοντέλο.

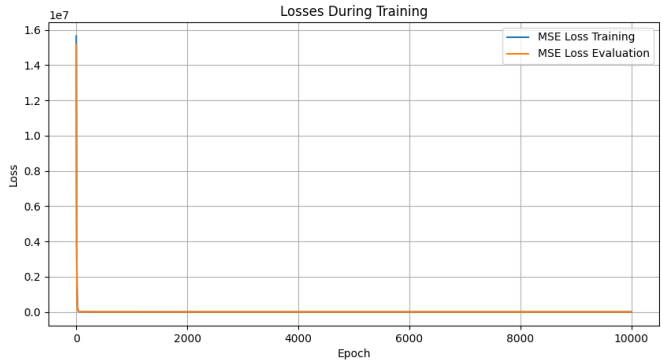


Σχήμα 4.11: MSE loss για περπάτημα

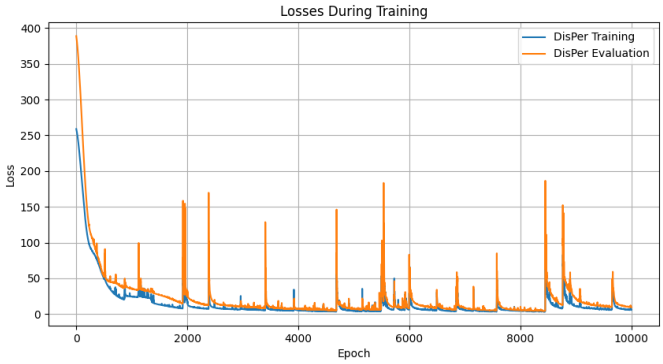
Στη συνέχεια, βλέπουμε πως διαμορφώθηκαν οι δύο μετρικές στα σχήματα που ακολουθούν παρακάτω για την περίπτωση του άλματος (Σχήμα 4.13,Σχήμα 4.14)



Σχήμα 4.12: Distortion percentage για περπάτημα



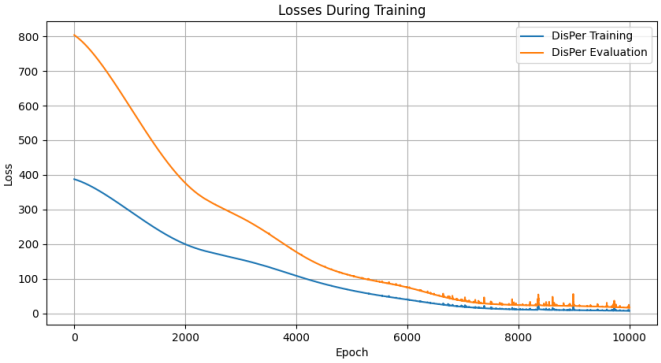
Σχήμα 4.13: MSE loss για άλμα



Σχήμα 4.14: Distortion percentage για άλμα

Τέλος, στο Σχήμα 4.15 και στο Σχήμα 4.16 παρατηρούμε τις μετρικές όπως διαμορφώθηκαν για το

Σχήμα 4.15: MSE loss για στροφή



Σχήμα 4.16: Distortion percentage για στροφή

4.3 Οπτικοποίηση Αποτελεσμάτων

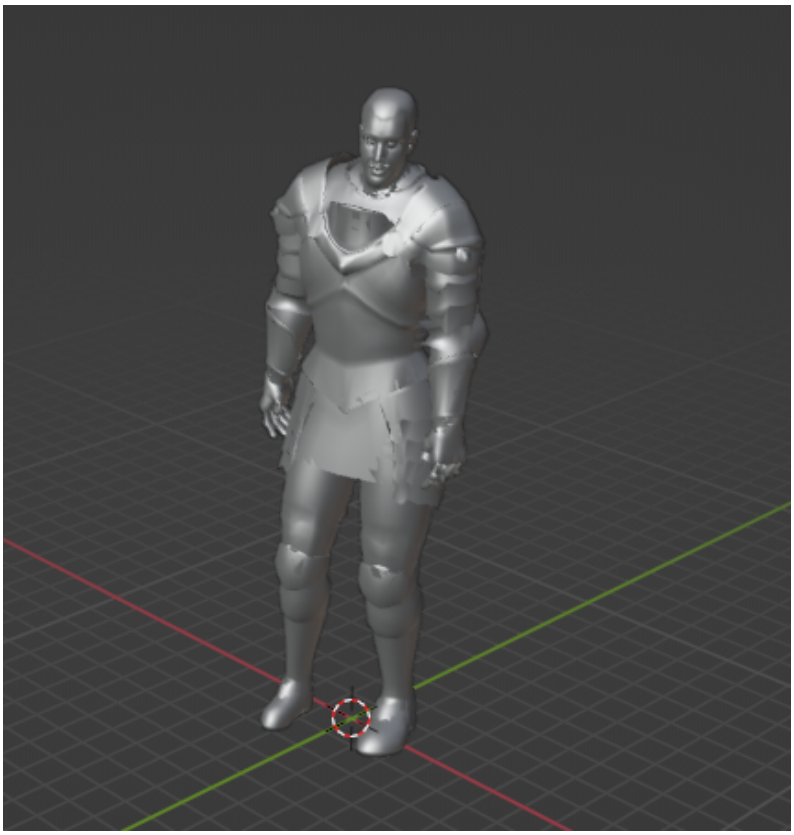
Μετά από την εκπαίδευση και την αξιολόγηση των δικτύων με τις εκάστοτε τιμές των μετρικών, δημιουργήσαμε αρχεία NumPy με τα δεδομένα που εξήχθησαν από τα δίκτυα κατά τη διάρκεια της αξιολόγησης. Ως συνέπεια του παραπάνω, είναι η δυνατότητα να οπτικοποιήσουμε τα αποτελέσματα στο Blender. Στα σχήματα που θα ακολουθήσουν, θα δούμε τα αποτελέσματα που προέκυψαν μετά τις προβλέψεις του μοντέλου LSTM με μηχανισμό προσοχής. Για παράδειγμα, στα Σχήματα 4.17, 4.18, 4.19 βλέπουμε πως το περπάτημα του χαρακτήρα, είναι αρκετά ικανοποιητικό με μια μικρή αλλοίωση στο κεφάλι και ειδικότερα στο μάτι του χαρακτήρα.

Σχήμα 4.17: Η κίνηση βαδίσματος που εκτελεί το mesh που προβλέφθηκε

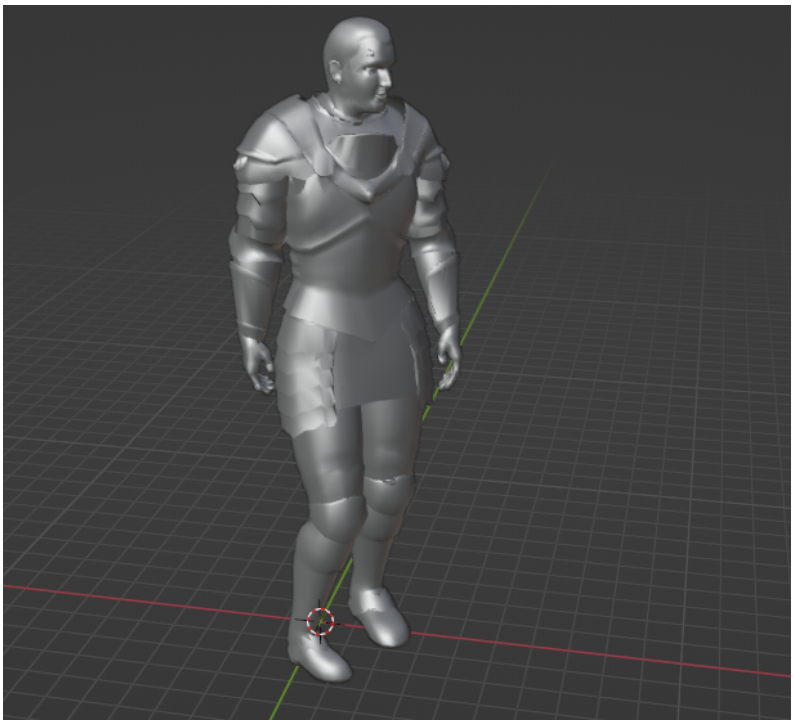
Σχήμα 4.18: Η κίνηση Βαδίσματος που εκτελεί το mesh που προβλέφθηκε

Σχήμα 4.19: Η κίνηση βαδίσματος που εκτελεί το mesh που προβλέφθηκε

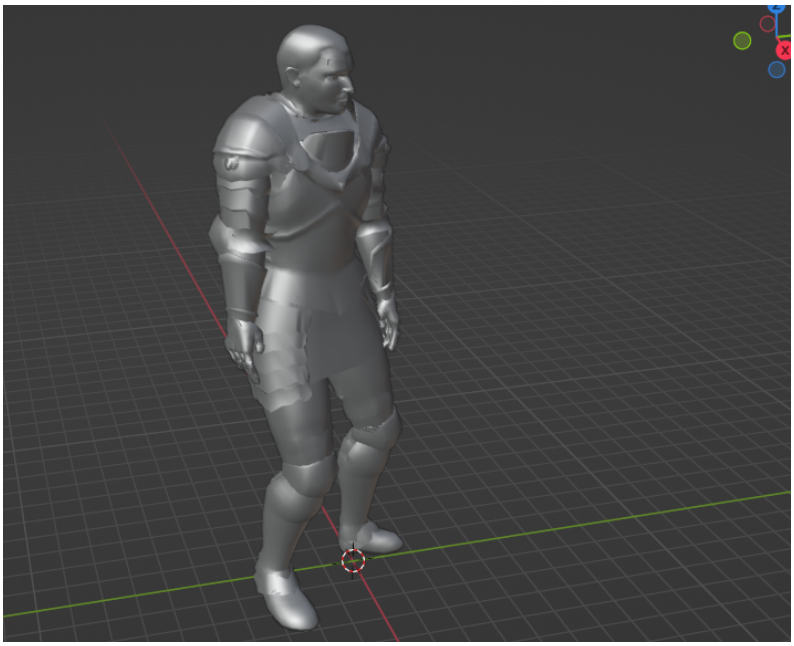
Στη συνέχεια, παραθέτουμε στα Σχήματα 4.20, 4.21, 4.22 την κίνηση που πραγματοποιεί ο χαρακτήρας αφού εκπαιδευτεί για μία περιστροφική κίνηση. Όπως διακρίνουμε, ο σκελετός πραγματοποιεί την κινήση σχεδόν πανομοιότυπα με την αρχική δίχως αλλοιώσες στα σημεία του mesh.



Σχήμα 4.20: Η περιστροφική κίνηση του mesh



Σχήμα 4.21: Η περιστροφική κίνηση του mesh



Σχήμα 4.22: Η περιστροφική κίνηση του mesh

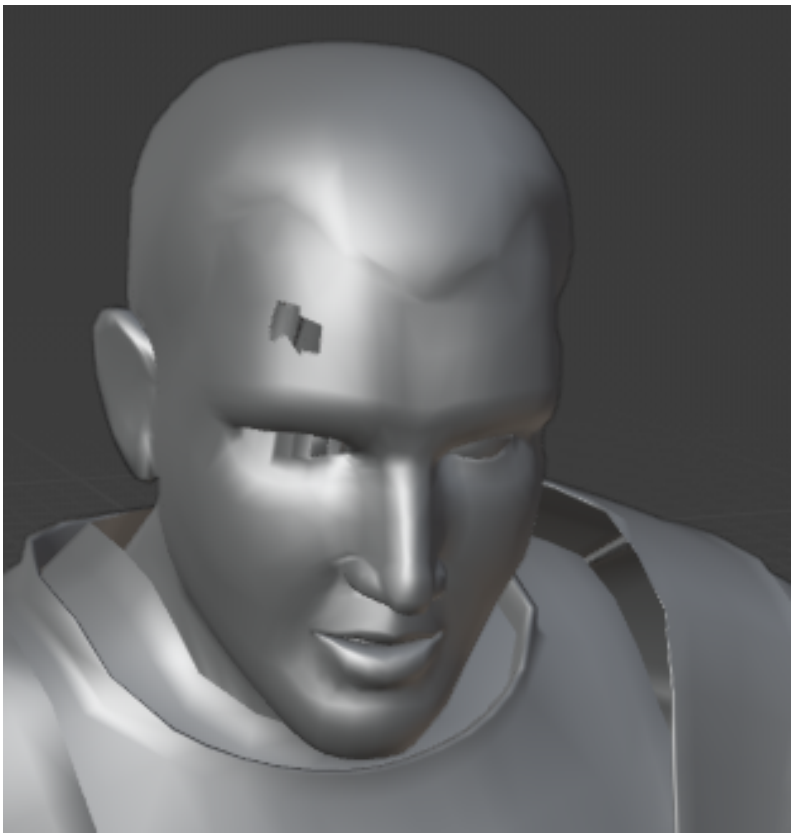
Επιπροσθέτως, στα σχήματα που ακολουθούν, εξετάζουμε την περίπτωση που ο χαρακτήρας εκτελεί ένα άλμα (Σχήματα 4.23, 4.24, 4.25).

Σχήμα 4.23: Άλμα του χαρακτήρα

Σχήμα 4.24: Άλμα του χαρακτήρα

Σχήμα 4.25: Άλμα του χαρακτήρα

Όπως μπορούμε να κατανοήσουμε εστιάζοντας στο mesh, η κίνηση μοιάζει αρκετά με την πραγματική με μία μικρή στροφή του κεφαλιού η οποία δεν αλλοιώνει όμως τη μορφή του και με μία μικρή παραμόρφωση στο δεξί μάτι (Σχήμα 4.26).

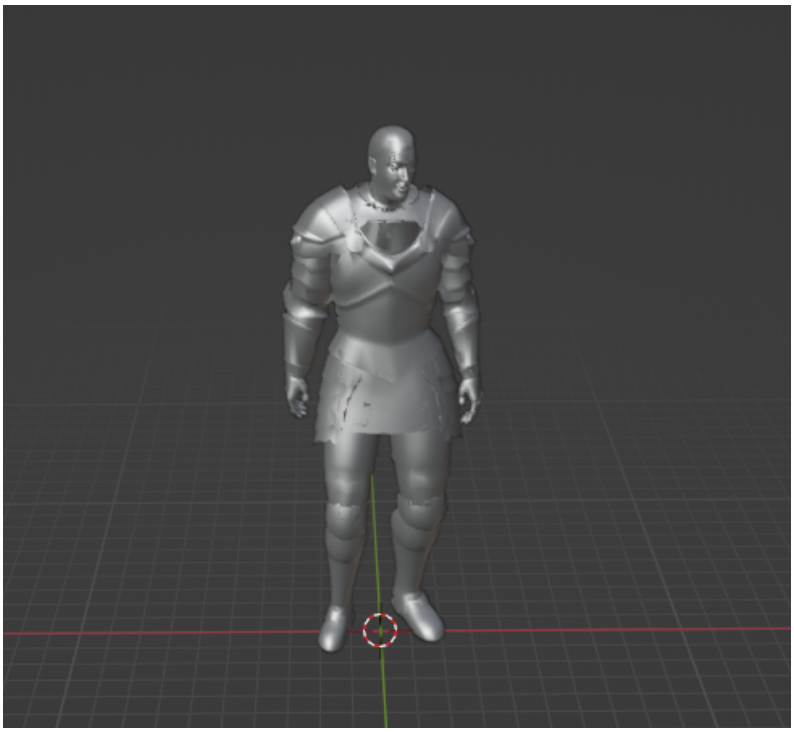


Σχήμα 4.26: Η αλλοίωση στο πρόσωπο

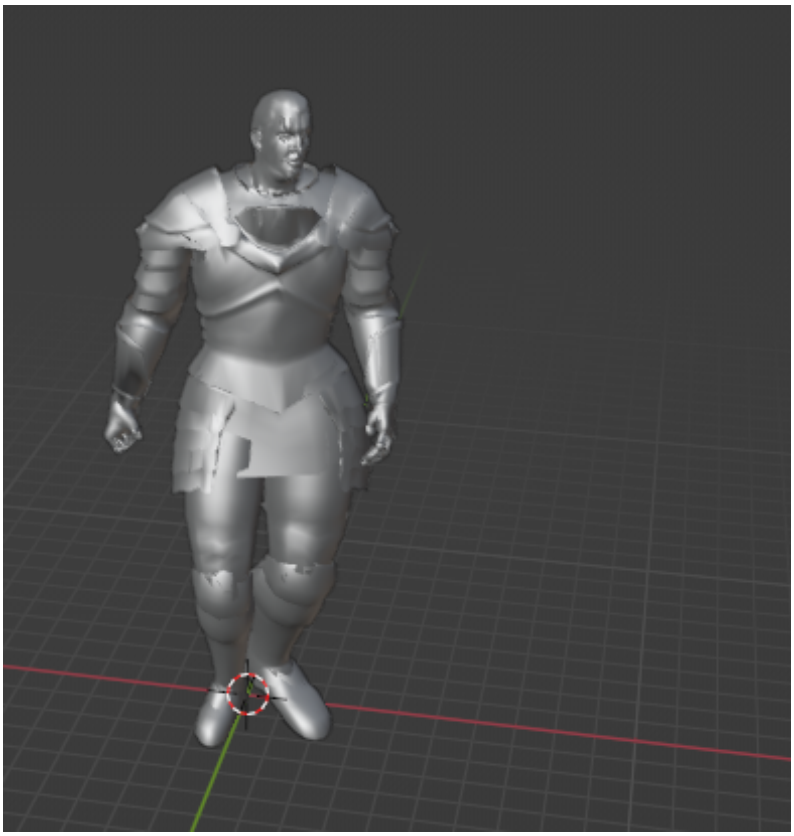
Από όλα τα παραπάνω αποτελέσματα, μπορούμε να συμπεράνουμε πως οι παραμορφώσεις των τρισδιάστατων πλεγμάτων για τις κινήσεις που εξετάστηκαν είναι μηδαμινές. Συνεπώς, μπορούμε να χαρακτηρίσουμε την ποιότητα των προβλέψεων ικανοποιητική.

Επιπρόσθετα, στόχος μας είναι η αξιολόγηση της σύνθεσης κίνησης με τρόπο όμοιο με τον παραπάνω. Προκειμένου να μπορέσουμε να φτάσουμε σε ένα ικανοποιητικό αποτέλεσμα σύνθεσης θα πρέπει να βελτιώσουμε την εκπαίδευση. Το γεγονός αυτό το επιτύχαμε με τον εμπλουτισμό των συνόλων εκπαίδευσης. Πιο συγκεκριμένα, για κάθε συνδυασμό που θέλουμε να δημιουργήσουμε προσθέτουμε στο σύνολο εκπαίδευσης της μίας κίνησης, μερικές από τις κινήσεις της άλλης. Για παράδειγμα, στο σύνολο των κινήσεων άλματος, προσθέτουμε μερικά δείγματα από στροφές του χαρακτήρα. Όλα αυτά, έχουν ως αποτέλεσμα την όσο πιο ρεαλιστική δημιουργία μιας νέας, σύνθετης κίνησης.

Στη συνέχεια, θα αξιολογήσουμε την ποιότητα της σύνθεσης μίας κίνησης στροφής με το βάδισμα. Τα Σχήματα 4.27, 4.28, 4.29 που ακολουθούν μας δείχνουν πως η κίνηση πετυχαίνει από την στροφή του κορμού ενώ παράλληλα περπατά, αλλά οι παραμορφώσεις στο σκελετό είναι αρκετά εμφανείς.



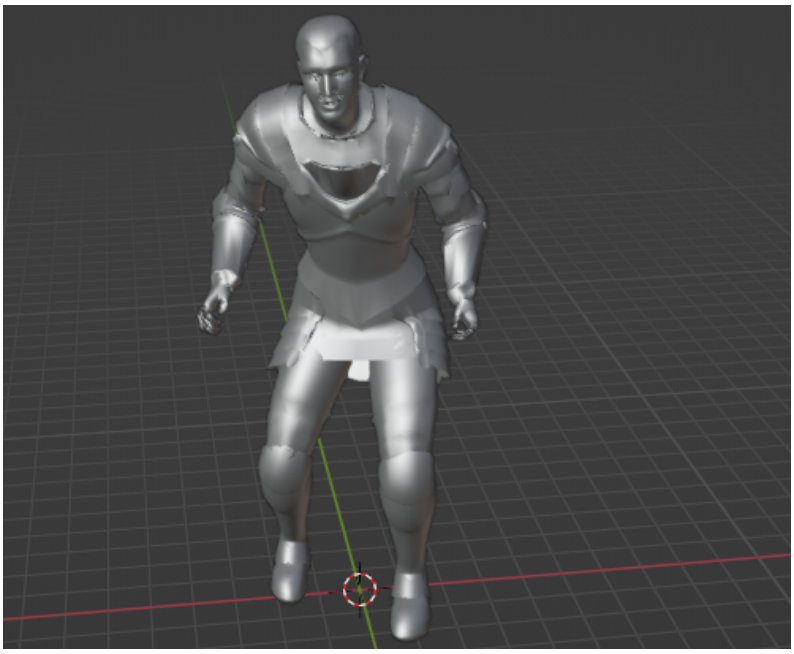
Σχήμα 4.27: Συνδυασμός βαδίσματος και στροφής



Σχήμα 4.28: Συνδυασμός βαδίσματος και στροφής

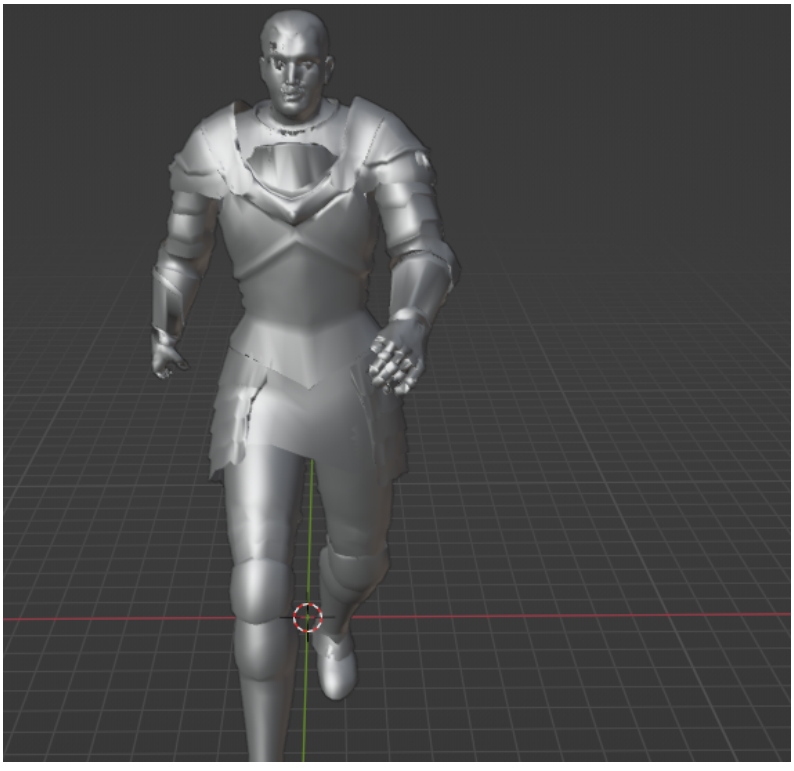
Σχήμα 4.29: Συνδυασμός βαδίσματος και στροφής

Όπως παρατηρούμε στα Σχήματα 4.30, 4.31, 4.32, η νέα κίνηση που δημιουργήσαμε ως συνδυασμό του άλματος και του βαδίσματος, εκτελείται κανονικά. Ωστόσο, παρατηρούμε μερικές αλλοιώσεις στο πλέγμα. Οι παραμορφώσεις αυτές, εντοπίζονται κατά κύριο λόγο στο δεξί χέρι του χαρακτήρα και στο δεξί του μάτι.



Σχήμα 4.30: Συνδυασμός άλματος και βαδίσματος

Σχήμα 4.31: Συνδυασμός άλματος και βαδίσματος



Σχήμα 4.32: Συνδυασμός άλματος και βαδίσματος

Επιπρόσθετα, θα ασχοληθούμε με τη σύνθεση κινήσεων μέσω της κατηγοριοποίησης των κινήσεων από τα διανύσματα που προσδιορίζουν την κίνηση που εκτελείται. Σε αυτή την περίπτωση, ο συνδυασμός δύο κινήσεων, επιτυγχάνεται με τον συνδυασμό των δύο διανυσμάτων που εκφράζουν αυτές τις κινήσεις. Συνεπώς, περιμένουμε ένα διάνυσμα το οποίο θα έχει σε δύο μη μηδενικές τιμές. Στο παράδειγμα που βλέπουμε στα Σχήματα 4.33, 4.34, 4.35, παρατηρούμε το συνδυασμό του άλματος με την κίνηση της υπόδειξης. Σε αυτή την περίπτωση, η κίνηση πετυχαίνει εν μέρει, ωστόσο, η παραμόρφωση πάλι είναι μεγάλη ειδικά στο δεξί χέρι του χαρακτήρα.

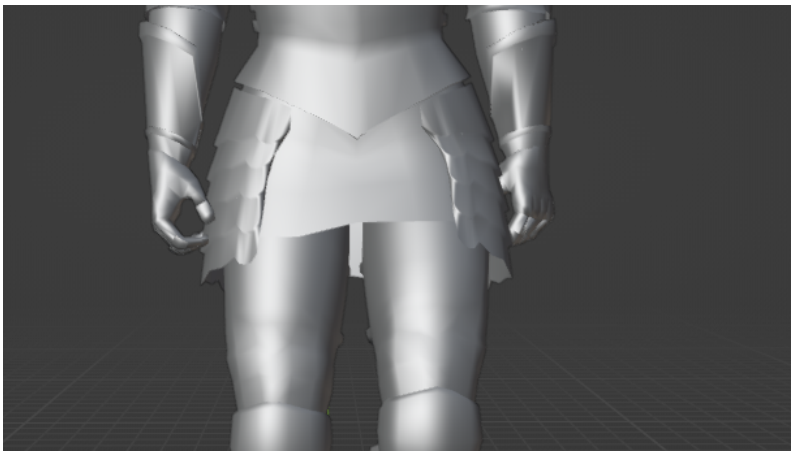
Σχήμα 4.33: Συνδυασμός άλματος και υπόδειξης

Σχήμα 4.34: Συνδυασμός άλματος και υπόδειξης

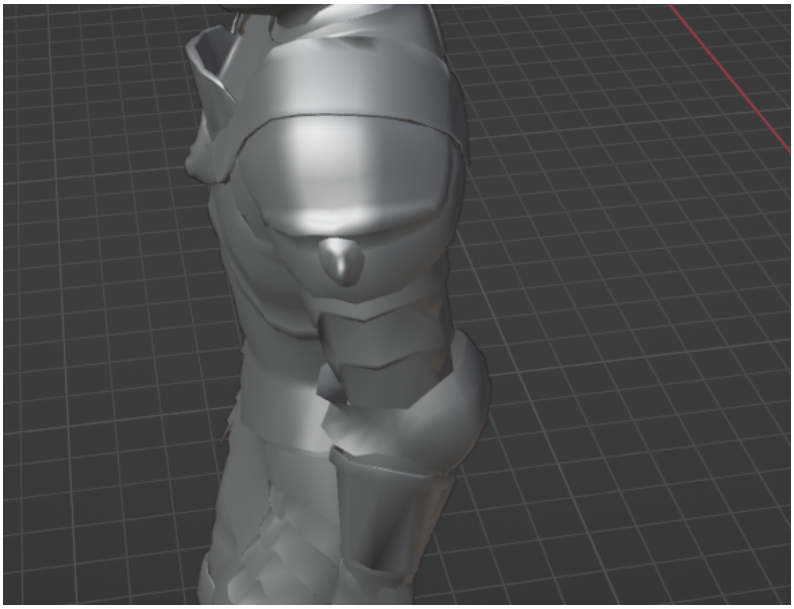
Σχήμα 4.35: Συνδυασμός άλματος και υπόδειξης

Παράλληλα, θα ασχοληθούμε με τον τρόπο που συνέβαλε η διατήρηση του όγκου στις προβλέψεις μας. Αρχικά, αξίζει να σημειώσουμε πως οι διαφορές μεταξύ των απλών κινήσεων είτε λογίζοντας είτε όχι τον όγκο στη συνάρτηση απώλειας, είναι απειροελάχιστες και αυτό συμβαίνει διότι δεν υπάρχουν ιδιαίτερες μεταβολές στο mesh κατά τη διάρκεια της εκτέλεσης μίας μόνο κίνησης. Όπως θα δούμε στα Σχήματα 4.36, 4.37, 4.38 που ακολουθούν, η διαφορά στο συνδυασμό βαδίσματος και άλματος είναι εμφανής. Πιο συγκεκριμένα αν εστιάσουμε στα δάχτυλα των χεριών, μπορούμε να παρατηρήσουμε πως το αποτέλεσμα είναι πιο φυσικό συγκριτικά με τον απλό συνδυασμό κινήσεων. Ειδικότερα, αν δώσουμε σημασία στο δεξί χέρι παρατηρούμε πως δίχως τη διατήρηση του όγκου το χέρι ήταν σαν μην έχει καθόλου δάχτυλα ενώ, στη δεύτερη περίπτωση έχει ένα αρκετά πιο εύρωστο αποτέλεσμα. Γενικά φαίνεται πως τα σημεία που σχηματίζουν γωνίες από τον τρόπο με τον οποίο είναι κατασκευασμένος ο σκελετός, εξομαλύνονται αν λάβουμε υπόψιν μας τον όγκο.

Σχήμα 4.36: Τα δάχτυλα στο δεξί χέρι είναι παραμορφωμένο



Σχήμα 4.37: Τα δάχτυλα στο δεξί χέρι είναι πιο εύρωστα



Σχήμα 4.38: Το οπτικό αποτέλεσμα στις γωνίες που σχηματίζονται είναι καλύτερο

Κλείνοντας, θα αναφερθούμε στο τρόπο που ο όγκος επηρεάζει τη σύνθεση που γίνεται όταν χρησιμοποιούμε κατηγοριοποίηση των κινήσεων. Εδώ υπάρχουν σημαντικές διαφορές (Σχήματα 4.39, 4.40), καθώς από τη μία παρατηρείται το ίδιο φαινόμενο σχετικά με τις γωνίες (ειδικά στις αρθρώσεις) που απαλύνονται, ωστόσο το οπτικό αποτέλεσμα είναι χειρότερο διότι και τα δύο χέρια είναι αρκετά αλλοιωμένα. Παρόλα αυτά, μπορούμε να σημειώσουμε ότι παρατηρώντας το animation σε όλη τη διάρκεια, ο όγκος του κεφαλιού και του κορμού είναι σχεδόν αμετάβλητος.

Σχήμα 4.39: Το αριστερό χέρι είναι αλλοιωμένο

Σχήμα 4.40: Το αριστερό χέρι είναι τελείως παραμορφωμένο αλλά τα πόδια έχουν μεγαλύτερη λεπτομέρεια

Κεφαλαιο 5

Συμπερασματα και Μελλοντικες Επεκτασεις

Ο βασικός στόχος της παρούσας διατριβής, ήταν η δημιουργία ενός νευρωνικού δικτύου το οποίο θα μπορούσε να προβλέψει την κίνηση ενός χαρακτήρα μέσω του πλέγματός του και η δημιουργία νέων κινήσεων οι οποίες θα έχουν ως βάση τις αρχικές. Το δίκτυο που αναπτύχθηκε, είναι αρκετά αποτελεσματικό ειδικά στον τομέα της πρόβλεψης των κινήσεων.

Με άλλα λόγια, το νευρωνικό δίκτυο έχει μια πολύ καλή απόδοση το οποίο σημαίνει πως οι μετρικές της απώλειας πρόβλεψης, είναι χαμηλές σε σημείο που μπορούν να θεωρηθούν μηδαμινές. Στην επιτυχία αυτής της απόδοσης, τον μεγαλύτερο ρόλο παίζουν οι υπερπαράμετροι του μοντέλου όπως για παράδειγμα το learning rate που παίρνει ο optimizer ή ο αριθμός των εποχών που εκτελούνται. Ειδική μνεία αξίζει να γίνει στη βιβλιοθήκη CUDA της NVIDIA, η οποία μας επιτρέπει να εκμεταλλευτούμε την υπολογιστική δύναμη της κάρτας γραφικών που μπορεί να διαθέτει ο υπολογιστής.

Επίσης, η σύνθεση κινήσεων με ένα LSTM δίκτυο με μηχανισμό προσοχής αποτελεί μία πολύ ενδιαφέρουσα προσέγγιση για τη δημιουργία νέων κινήσεων η οποία είναι αρκετά αποδοτική.

Ωστόσο, όπως συμβαίνει στα περισσότερα προβλήματα τεχνητής νοημοσύνης, έτσι και σε αυτό, υπάρχει η δυνατότητα βελτίωσης των αποτελεσμάτων. Πιθανώς, η χρήση άλλων κωδικοποιητών-αποκωδικοποιητών θα μπορούσε να βελτιώσει την

ποιότητα των προβλέψεων που παράγονται από τα δεδομένα εκπαίδευσης. Επιπλέον, η χρήση της καμπυλότητας (curvature) αλλά και μία αλλαγή στις παραμέτρους της διατήρησης του όγκου, θα μπορούσαν να ελαχιστοποιήσουν ακόμα περισσότερο την παραμόρφωση κατά τη σύνθεση των κινήσεων. Τέλος, μια διαφορετική μέθοδος τριγωνοποίησης ή η ολοκληρωτική μεταφορά του δικτύου στο περιβάλλον του Blender, θα μπορούσαν να αποτελέσουν καθοριστικούς παράγοντες στην διατήρηση των χαρακτηριστικών του μοντέλου κατά τη σύνθεση.

Βιβλιογραφια

[1] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[2] NVIDIA, P. Vingelmann, and F. H. Fitzek, "Cuda, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[3] T. Mullen, *Mastering blender*. John Wiley & Sons, 2011.

[4] B. R. Kent, "Meshes, models and textures," *3D Scientific Visualization with Blender®*, pp. 3–1, 2015.

[5] E. T. M. Guevarra, *Modeling and animation using blender: blender 2.80: the rise of Eevee*. Apress, 2019.

[6] R. Caudron, P.-A. Nicq, and E. Valenza, *Blender 3D: Designing Objects*. Packt Publishing Ltd, 2016.

[7] J. Eiderbäck and F. Jahnke, "A new approach for positioning human body models utilising the 3d-graphics program blender," 2023.

[8] N. Kriegeskorte and T. Golan, "Neural network models and deep learning," *Current Biology*, vol. 29, no. 7, pp. R231–R236, 2019.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[10] S. Grossberg, "Recurrent neural networks," *Scholarpedia*, vol. 8, no. 2, p. 1888, 2013.

[11] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[12] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *2018 Chinese control and decision conference (CCDC)*. IEEE, 2018, pp. 1836–1841.

[13] H. Jabbar and R. Z. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)," *Computer Science, Communication and Instrumentation Devices*, vol. 70, no. 10.3850, pp. 978–981, 2015.

[14] S. Pothuganti, "Review on over-fitting and under-fitting problems in machine learning and solutions," *Int. J. Adv. Res. Electr. Electron. Instrum. Eng*, vol. 7, pp. 3692–3695, 2018.

[15] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," in *2016 IEEE 6th International conference on advanced computing (IACC)*. IEEE, 2016, pp. 78–83.

[16] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[17] A. Moutafidou, V. Toulatzis, and I. Fudos, "Deep fusible skinning of animation sequences," *Visual Computer*, vol. 39, no. 9, pp. 1967–1980, 2023. [Online]. Available: https://doi.org/10.1007/s00371-023-03130-3

[18] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big data analytics*, vol. 1, pp. 1–22, 2016.

[19] Wikipedia contributors. (2024) Tetrahedron. Accessed: 2024-07-25. [Online]. Available: https://en.wikipedia.org/wiki/Tetrahedron

Σύντομο Βιογραφικο

Ο Χατζησάββας Παύλος γεννήθηκε στην Πτολεμαΐδα το 1997. Σπούδασε στο τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής στο Πανεπιστήμιο Ιωαννίνων και αποφοίτησε το 2022. Αυτό τον καιρό, προσπαθεί να αποκτήσει το Μεταπτυχιακό με τίτλο ''Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων'' από το ίδιο Τμήμα.