Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

## Μεταπτυχιακή Διατριβή

| | |
|---|---|
| Τίτλος Διατριβής | **Ανάλυση συναισθήματος κατά τη διάρκεια εξ αποστάσεως εκπαίδευσης σε εφαρμογή Android και αποστολή σε RESTful Web Service**<br><br>**Sentiment analysis during online learning on Android application and sending to a RESTful Web Service** |
| Ονοματεπώνυμο Φοιτητή | **Γεώργιος Γαβριηλίδης** |
| Πατρώνυμο | **Γαβριήλ** |
| Αριθμός Μητρώου | **ΜΠΠΛ / 18009** |
| Επιβλέπων | **Ευθύμιος Αλέπης, Αναπληρωτής καθηγητής** |

Ημερομηνία Παράδοσης Σεπτέμβριος 2022

**Τριμελής Εξεταστική Επιτροπή**

| | | |
|---|---|---|
| (υπογραφή) | (υπογραφή) | (υπογραφή) |
| Ευθύμιος Αλέπης<br>Αναπληρωτής καθηγητής | Μαρία Βίρβου<br>Καθηγήτρια | Κωνσταντίνος Πατσάκης<br>Αναπληρωτής καθηγητής |

# ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία παρουσιάζεται η διαδικασία δημιουργίας μιας Android εφαρμογής για ανάλυση των συναισθημάτων των φοιτητών πανεπιστημίου κατά τη διάρκεια εξ αποστάσεως εκπαίδευσης. Συγκεκριμένα, η εφαρμογή προσομοιώνει τη διδασκαλία ενός μαθήματος μέσω κάμερας και πραγματοποιεί, ανά ορισμένο χρονικό διάστημα, ανάλυση συναισθήματος προσώπου όσων φοιτητών έχουν δώσει την απαραίτητη άδεια στην εφαρμογή, στέλνοντάς την πρόβλεψη του αλγόριθμου από τις κινητές συσκευές τους, που λειτουργούν ως clients, σε μια βάση δεδομένων μέσω web service. Από τα περιεχόμενα της βάσης δημιουργούνται ποσοτικά δεδομένα στα οποία έχει πρόσβαση ο διδάσκων καθηγητής, ο οποίος, παρατηρώντας τα, μπορεί να αντλήσει συμπεράσματα για την επίδραση του τρόπου παράδοσής του είτε σε συγκεκριμένη διάλεξη είτε και διαχρονικά, καθώς εξελίσσεται το εξάμηνο σπουδών. Μια υλοποίηση όπως η παραπάνω, με κάποιες επεκτάσεις, θα μπορούσε να χρησιμοποιηθεί είτε αυτόνομα είτε να προσαρτηθεί σε ήδη ευρέως χρησιμοποιούμενες πλατφόρμες, για την εξασφάλιση της ποιότητας του μαθήματος σε συνθήκες εξ αποστάσεως εκπαίδευσης, μέσω της σχεδόν άμεσης ανατροφοδότησης του καθηγητή ως προς την απήχηση του περιεχομένου και της μεθόδου διδασκαλίας του στους φοιτητές.

Στις επόμενες σελίδες γίνεται αρχικά μια επισκόπηση των νευρωνικών δικτύων και συγκεκριμένα μιας υποκατηγορίας τους, του συνελικτικού νευρωνικού δικτύου, ένα μοντέλο του οποίου χρησιμοποιήθηκε για την αναγνώριση συναισθήματος των φοιτητών, της βιβλιοθήκης Tensorflow Lite που επιτρέπει την ανάπτυξη μοντέλων μηχανικής μάθησης σε περιβάλλον Android συσκευής καθώς και του Spring Boot framework, με το οποίο δημιουργήθηκε το RESTful web service για την εφαρμογή. Στη συνέχεια παρουσιάζεται η διαδικασία της εκπαίδευσης του νευρωνικού δικτύου, του σχεδιασμού της βάσης δεδομένων, και της ανάπτυξης του κώδικα της εφαρμογής. Τέλος, γίνεται αναφορά στα συμπεράσματα που προέκυψαν από την υλοποίηση και σε πιθανές μελλοντικές επεκτάσεις.

# ABSTRACT

This thesis is about an Android application that predicts facial emotions of university students during distance learning. Specifically, the application simulates an online lecture, during which it periodically performs facial emotion recognition on snapshots of students who gave the required permission to the application and sends the algorithm's prediction from their online devices, who act as clients, to a database through a web service. From the contents of the database, quantitative data are created, which can be accessed by the teaching professor, enabling him to draw conclusions about the impact his style of teaching made either at a specific lecture or during the course of a semester. An implementation like this, after getting expanded, could be used either independently or as an add-on to widely used platforms, to help insure the quality of online learning by providing the professor with almost immediate feedback on the students' response to the content and the method of his teaching.

In the following pages an overview is being provided of neural networks, and specifically a subcategory of theirs, the convolutional neural network, a model of which was used for predicting the emotion of the students using the application, of the Tensorflow Lite library, which allows the deployment of machine learning models on an Android device and of the Spring Boot Framework, with which the RESTful web service for the application was created. Subsequently, a presentation is being given on the process of training the neural network, designing the database and developing the application. Lastly, a mention is being made of conclusions which came up after completing the implementation and of probable future expansions.

# Πίνακας περιεχομένων

# 1 ΕΙΣΑΓΩΓΗ

Η εξ αποστάσεως εκπαίδευση, είτε πρόκειται για ασύγχρονη παρακολούθηση οπτικοακουστικού υλικού, είτε για συμμετοχή σε ένα μάθημα που εκτυλίσσεται σε πραγματικό χρόνο, αποτελεί, εδώ και αρκετά χρόνια, μια διαδεδομένη εναλλακτική λύση για διεύρυνση των γνώσεων κάθε ενδιαφερόμενου. Αν εστιαστούμε στην δεύτερη περίπτωση, αυτή της πραγματοποίησης μιας πανεπιστημιακής διάλεξης μέσω Διαδικτύου, παρατηρούμε ότι, ενώ απλοποιεί τη διαδικασία παρακολούθησης του μαθήματος, παρουσιάζει το εξής πρόβλημα: απομακρύνοντας χωρικά τον καθηγητή από τους φοιτητές, καθιστά πιο δύσκολη την παρατήρηση και ερμηνεία των αντιδράσεών τους στη διδασκαλία και κατά συνέπεια της απήχησης του τρόπου παράδοσης του μαθήματος. Αυτό ενδέχεται να επηρεάσει τη συνολική επίδραση που θα έχει το μάθημα σε αυτούς, αφού δεν δίνεται στον καθηγητή η πληροφορία βάσει της οποίας θα κάνει τις απαραίτητες αναπροσαρμογές στη διδασκαλία του για να αφυπνίσει το ενδιαφέρον τους. Σκοπός της παρούσας διπλωματικής εργασίας είναι η αντιμετώπιση αυτού του προβλήματος αλλά και η εξέταση της πιθανότητας παροχής στον καθηγητή ακόμα καλύτερης ανατροφοδότησης από αυτή που θα μπορούσε να εξασφαλίσει μια φυσική παρουσία στην τάξη, κατά την οποία θα έπρεπε απαραίτητα να εστιαστεί οπτικά σε συγκεκριμένο υποσύνολο φοιτητών κάθε στιγμή.

Λαμβάνοντας υπόψη την διαρκώς αυξανόμενη μνήμη και υπολογιστική δύναμη των κινητών συσκευών που διατίθενται στην αγορά, η οποία πλέον καθιστά δυνατή την απρόσκοπτη παρακολούθηση μιας τηλεδιάσκεψης μέσω φορητού τηλεφώνου, αποφασίστηκε να δημιουργηθεί μια εφαρμογή για κινητά Android. Σε αυτή την απόφαση συντέλεσε και η δυνατότητα ενσωμάτωσης της βιβλιοθήκης Tensorflow Lite στο Android Studio, για χρησιμοποίηση ενός μοντέλου μηχανικής μάθησης που εκπαιδεύτηκε προηγουμένως με χρήση του Tensorflow και ενός αρκετά μεγάλου dataset έτσι ώστε να αναγνωρίζει πέντε βασικά συναισθήματα σε στιγμιότυπα ανθρώπινων προσώπων. Οι περιοδικές προβλέψεις του μοντέλου αυτού για το συναίσθημα που απεικονίζεται στην κάμερα του εκάστοτε φοιτητή στέλνονται μέσω Διαδικτύου σε ένα web service που υλοποιήθηκε με χρήση του Spring Boot Framework. Αυτό επικοινωνεί με μια βάση δεδομένων που δημιουργήθηκε με τη βοήθεια του open-source συστήματος βάσεων δεδομένων PostgreSQL, όπου και αποθηκεύονται τα δεδομένα για μετέπειτα αποστολή τους στην εφαρμογή και προβολή τους από τον καθηγητή με μορφή γραφημάτων.

Στις σελίδες που ακολουθούν, μετά από μια επισκόπηση του γνωστικού πεδίου των νευρωνικών δικτύων, θα αναλυθεί η διαδικασία μέσω της οποίας αυτά τα εργαλεία υλοποιήθηκαν το κάθε ένα ξεχωριστά και του τρόπου με τον οποίο συνδέθηκαν μεταξύ τους για την επίτευξη του σκοπού της διπλωματικής εργασίας.

# 2 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

## 2.1 Ιστορική Αναδρομή

Τα νευρωνικά δίκτυα γεννήθηκαν ως ιδέα το 1943. Ο νευροφυσιολόγος Warren McCulloch και ο μαθηματικός Walter Pitts, σε συνέχεια ενός άρθρου τους που περιέγραφε τον τρόπο με τον οποίο πιθανώς να λειτουργεί το ανθρώπινο μυαλό, μοντελοποίησαν ένα απλό νευρωνικό δίκτυο χρησιμοποιώντας ένα ηλεκτρικό κύκλωμα.

Η προσομοίωση ενός υποθετικού νευρωνικού δικτύου έγινε δυνατή με την εξέλιξη των υπολογιστών στις αρχές της δεκαετίας του ’50. Μια ομάδα της εταιρείας IBM, με υπεύθυνο τον ηλεκτρολόγο μηχανικό Nathaniel Rochester, προσπάθησε να μιμηθεί ένα νευρωνικό δίκτυο σε έναν υπολογιστή IBM 704. Η προσπάθεια ωστόσο δεν στέφτηκε με επιτυχία.

Η πρώτη επιτυχημένη προσπάθεια δημιουργίας νευρωνικού δικτύου και χρήσης του για την επίλυση ενός πραγματικού προβλήματος ήρθε το 1959. Οι καθηγητές του Stanford Bernard Widrow και Marcian Hoff δημιούργησαν δύο μοντέλα νευρωνικού δικτύου με τα ακρωνύμια "ADALINE" και "MADALINE". Ο σκοπός δημιουργίας του δεύτερου ήταν η μείωση του θορύβου στις τηλεφωνικές γραμμές, μέσω της λειτουργίας του ως προσαρμοστικού φίλτρου. Ο σκοπός επιτεύχθηκε και το σύστημα βρίσκεται σε χρήση ακόμη και σήμερα.

Ωστόσο, η περαιτέρω ανάπτυξη στο πεδίο των νευρωνικών δικτύων δεν ανταποκρίθηκε στις αυξημένες προσδοκίες που προέκυψαν μετά από τις πρώτες επιτυχίες. Ένα βασικό εμπόδιο αποτελούσε το hardware των υπολογιστών εκείνης της περιόδου, που καθιστούσε πολύ χρονοβόρα τη διαδικασία εκτέλεσης. Ταυτόχρονα, μια συνάρτηση μάθησης που χρησιμοποιούνταν ευρέως είχε δομικά ελαττώματα, κάτι που αναπόφευκτα οδηγούσε τις προσπάθειες για εξέλιξη στον τομέα σε συνεχή αδιέξοδα.

Εκείνη την περίοδο, και συγκεκριμένα το 1969, εκδόθηκε ένα βιβλίο με τίτλο "Perceptrons" από τον Marvin Minsky και τον Seymour Papert, ιδρυτή και διευθυντή αντίστοιχα του εργαστηρίου τεχνητής νοημοσύνης του πανεπιστημίου M.I.T. Στο βιβλίο διατυπωνόταν ο ισχυρισμός ότι δεν θα μπορούσε να υπάρξει επέκταση ενός νευρωνικού δικτύου ενός στρώματος (single layered) σε ένα δίκτυο πολλών στρωμάτων (multiple layered), λόγω του τεράστιου - ίσως και άπειρου - αριθμού επαναλήψεων που θα χρειάζονταν για να αποφασιστεί η σωστή αξία των βαρών (weights) των νευρώνων στα διάφορα επίπεδα ενός πολυστρωματικού νευρωνικού δικτύου. Η μεγάλη επίδραση του παραπάνω συγγράμματος, σε συνδυασμό με φιλοσοφικές θεωρήσεις επί του θέματος που γέννησαν ανησυχίες σχετικά με το μέλλον της ανθρωπότητας σε περίπτωση δημιουργίας "σκεπτόμενων μηχανών", οδήγησαν σε δραστική πτώση των χρηματοδοτήσεων και της έρευνας πάνω στο πεδίο για τα επόμενα 12 περίπου χρόνια.

Το 1982 αυτή η περίοδος στασιμότητας έλαβε τέλος. Ο John Hopfield παρουσίασε στην Εθνική Ακαδημία Επιστημών ένα άρθρο με τίτλο "Neural networks and physical systems with emergent collective computational abilities", γνωστό και ως "Δίκτυο Hopfield". Πρόκειται για ένα επαναλαμβανόμενο νευρωνικό δίκτυο με αμφίδρομες συνδέσεις μεταξύ νευρώνων, σε αντίθεση με τις μονόδρομες που χρησιμοποιούνταν μέχρι τότε.

Ένα άλλο γεγονός κομβικής σημασίας ήταν η πραγματοποίηση ενός συνεδρίου με συμμετοχή των Ηνωμένων Πολιτειών και της Ιαπωνίας με θέμα τα Συνεργατικά- Ανταγωνιστικά Νευρωνικά Δίκτυα, κατά το οποίο η Ιαπωνία ανακοίνωσε την πρόθεσή της να εκκινήσει μια προσπάθεια ανάπτυξης της επονομαζόμενης 5ης Γενιάς πληροφορικής, που αφορά στην τεχνητή νοημοσύνη με τη χρήση νευρωνικών δικτύων. Ο φόβος των Ηνωμένων Πολιτειών ότι θα μείνουν πίσω στις εξελίξεις πυροδότησε ένα νέο κύκλο χρηματοδοτήσεων και έρευνας.

Η μεγάλη ώθηση όμως στην εξέλιξη των νευρωνικών δικτύων δόθηκε από μια μέθοδο που εφαρμοζόταν ήδη από τη δεκαετία του 1960, γνωστή ως Μέθοδος Οπισθοδιάδοσης του Λάθους (Backpropagation Method), η οποία εφαρμόστηκε στα δίκτυα πολλαπλών επιπέδων που είχαν πλέον αναπτυχθεί. Η μέθοδος αυτή, μετά από ένα πέρασμα του νευρωνικού δικτύου από την αρχή προς το τέλος, πραγματοποιεί ένα πέρασμα προς την αντίθετη κατεύθυνση, προσαρμόζοντας παράλληλα τις παραμέτρους του δικτύου, οι οποίες είναι τα βάρη και οι πολώσεις (weights, biases).

Τρεις διαφορετικές ερευνητικές ομάδες διαπίστωσαν την ίδια περίπου περίοδο τον πολύ σημαντικό ρόλο που θα μπορούσε να διαδραματίσει η μέθοδος αυτή στην εξέλιξη των νευρωνικών δικτύων. Μια από αυτές απαρτιζόταν από τους Rumelhart, Hinton και Williams, οι οποίοι διατύπωσαν τις

ιδέες τους σε ένα λεπτομερές άρθρο, που σηματοδότησε την έναρξη της ευρείας χρήσης της Οπισθοδιάδοσης από την κοινότητα της πληροφορικής.

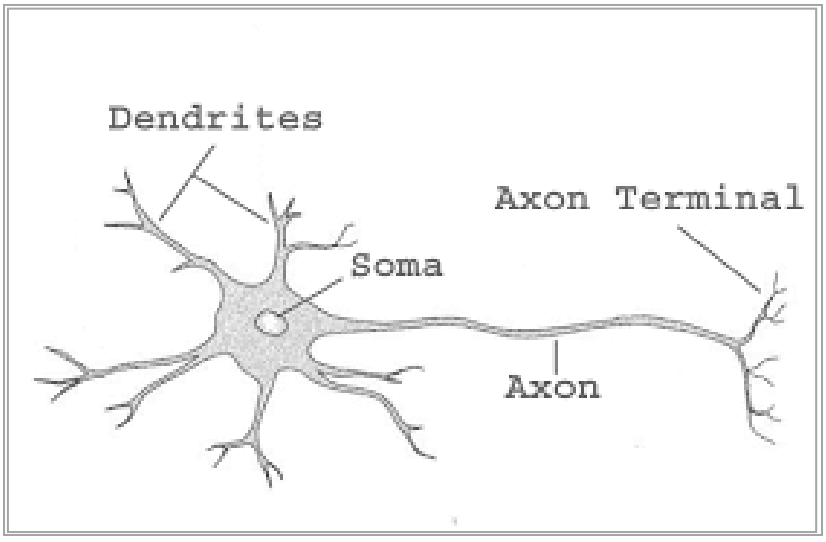
Από τα τέλη της δεκαετίας του ‘80, οπότε συνέβησαν τα παραπάνω, μέχρι σήμερα, τα νευρωνικά δίκτυα έχουν εξελιχθεί σε πολύ μεγάλο βαθμό και χρησιμοποιούνται πλέον από μια πληθώρα εφαρμογών που συναντιούνται στην καθημερινότητα, από τη μηχανική μετάφραση ενός κειμένου στο διαδίκτυο έως την λειτουργία ενός αυτόματου πιλότου σε ένα αεροσκάφος.

## 2.2 Βιολογία

Για τη δημιουργία των τεχνητών νευρωνικών δικτύων πηγή έμπνευσης αποτέλεσαν τα βιολογικά νευρωνικά δίκτυα που υπάρχουν στον ανθρώπινο (και όχι μόνο) εγκέφαλο. Η ανίχνευση των δομικών στοιχείων και η κατανόηση -έστω και περιορισμένη μέχρι και σήμερα- του τρόπου λειτουργίας ενός βιολογικού νευρωνικού δικτύου διαδραμάτισε πολύ σημαντικό ρόλο στην ανάπτυξη των αντίστοιχων τεχνητών.

Το νευρωνικό δίκτυο υπάγεται στον έλεγχο του ανθρώπινου μυαλού. Αυτό διαθέτει μικρότερα κέντρα ελέγχου που είναι υπεύθυνα για βασικές ανθρώπινες λειτουργίες, όπως οι πέντε αισθήσεις, η κίνηση, η όραση και η ακοή και συνδέεται με τα υπόλοιπα αισθητήρια και κινητήρια τμήματα του ανθρώπινου σώματος μέσω ενός πυκνού νευρικού δικτύου. Ένα νευρωνικό δίκτυο εμπλέκεται στις παραπάνω βασικές λειτουργίες καθώς και γενικότερα στη διαδικασία της εκμάθησης και της διαμόρφωσης εξαρτημένων αντανακλαστικών.

Στον πυρήνα του βιολογικού νευρωνικού δικτύου βρίσκεται το νευρικό κύτταρο, γνωστό και ως νευρώνας. Ο αριθμός τους εντός του ανθρώπινου εγκέφαλου είναι τεράστιος, συγκεκριμένα 86 δισεκατομμύρια νευρώνες κατά μέσο όρο. Το νευρικό κύτταρο, σε δομικό επίπεδο, χωρίζεται σε τρία βασικά μέρη: το σώμα(soma), τους δενδρίτες (dendrites) και τον άξονα(axon), όπως βλέπουμε και στην παρακάτω εικόνα:




*Εικόνα 1. Ένα νευρικό κύτταρο*

Οι δενδρίτες είναι απολήξεις του νευρικού κυττάρου που λειτουργούν ως σημεία υποδοχής. Το σώμα περιέχει τα οργανίδια του νευρικού κυττάρου. Ο άξονας πρόκειται για μια ίνα, μοναδική ανά

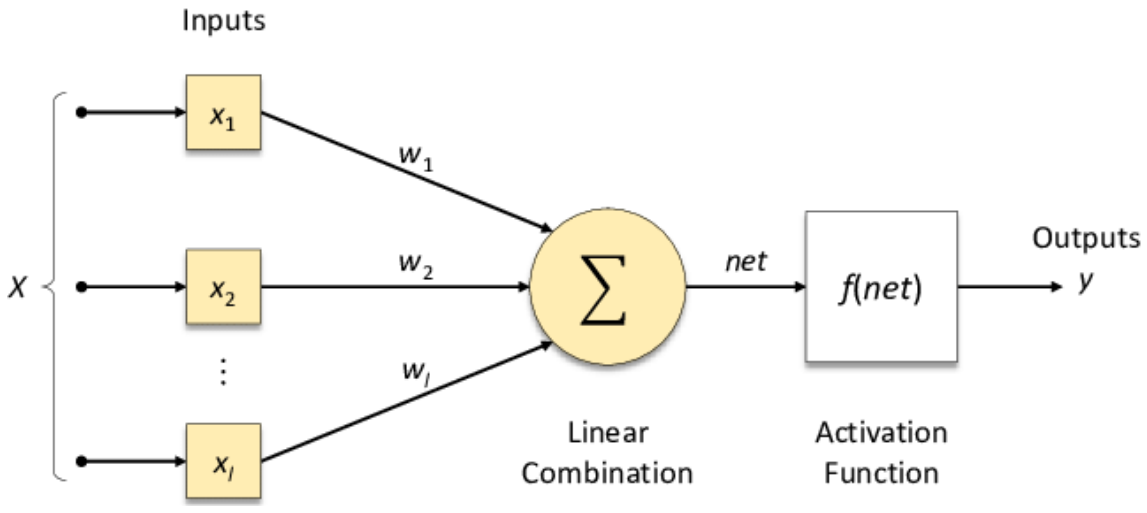
νευρώνα. Η διαδικασία στην οποία τα τρία αυτά τμήματα συμμετέχουν είναι η εξής: στους δενδρίτες ενός νευρώνα φτάνουν ηλεκτρικές ώσεις από διάφορους άλλους, με τους οποίους αυτός συνδέεται. Περνώντας στο σώμα του κυττάρου, μεταδίδονται μέσω του άξονα, στο τέλος του οποίου υπάρχει ένα σύνολο από παρακλάδια, σε δενδρίτες άλλων νευρώνων. Έτσι σχηματίζεται ένα τεράστιο δίκτυο μετάδοσης ηλεκτρικών ώσεων από νευρώνες που συνδέονται μεταξύ τους.

Ωστόσο οι νευρώνες δεν ενεργοποιούνται με κάθε σήμα που λαμβάνουν. Το αν ένας νευρώνας θα ενεργοποιηθεί αφού λάβει ένα σύνολο ηλεκτρικών παλμών μέσω των δενδριτών του, έτσι ώστε να ενεργοποιήσει με τη σειρά του άλλους συνδεδεμένους με αυτόν, εξαρτάται από τους εξής παράγοντες: το σήμα που δέχεται κάθε δενδρίτης μπορεί να διαφέρει ως προς την ένταση και τη συχνότητα. Όλα τα σήματα που δέχεται ο νευρώνας αθροίζονται και δίνονται ως είσοδος σε μια συνάρτηση κατωφλιού (threshold function). Αν η είσοδος αυτή βρίσκεται πάνω από το κατώφλι της συνάρτησης, παράγεται ένα ηλεκτρικό σήμα εξόδου, διαφορετικά ο νευρώνας δεν στέλνει ηλεκτρική ώση σε άλλους νευρώνες που συνδέονται με αυτόν. Στο αποτέλεσμα της συνάρτησης παίζει λοιπόν ρόλο και η συσχέτιση που έχει διαμορφωθεί μεταξύ συνδεδεμένων νευρώνων: το σήμα που λαμβάνεται από νευρώνες με ισχυρή συσχέτιση με τον παρόντα παίζει σημαντικότερο ρόλο στην ενεργοποίησή του, αφού η ηλεκτρική ώση που έρχεται από αυτούς έχει μεγαλύτερο βάρος στη διαμόρφωση του αποτελέσματος της συνάρτησης.

Με την παράλληλη χρονικά ενεργοποίηση διαφορετικών ομάδων νευρώνων μέσω της παραπάνω διαδικασίας δίνεται η δυνατότητα στον ανθρώπινο εγκέφαλο να διεκπεραιώνει πολύ γρήγορα μια σειρά από περίπλοκες λειτουργίες, όπως η αναγνώριση ενός οικείου του προσώπου μετά τη λήψη του οπτικού ερεθίσματος.

## 2.3 Perceptrons

Οι perceptrons αποτελούν τους προκατόχους των τεχνητών νευρωνικών δικτύων. Προέκυψαν ως ιδέα από τον ψυχολόγο Frank Rosenblatt, γνωστό για την προσφορά του στον τομέα της τεχνητής νοημοσύνης. Ο Rosenblatt προσπάθησε να μιμηθεί τη λειτουργία ενός βιολογικού νευρωνικού δικτύου μέσω ενός απλοποιημένου μαθηματικού μοντέλου, όπως το παρακάτω:




*Εικόνα 2 Perceptron*

Οι ηλεκτρικές ώσεις που φτάνουν ως είσοδος από συνδεδεμένους νευρώνες σε ένα βιολογικό νευρωνικό δίκτυο αντιπροσωπεύονται στον perceptron από τις μεταβλητές $x_{1...}x_{i,}$ που έχουν τιμή 0 ή 1. Τα βάρη(weights) $w_{1...}w_{i}$ προσομοιώνουν τον αυξημένη ή μειωμένη βαρύτητα που διαδραματίζει ο κάθε συνδεδεμένος νευρώνας στην ενεργοποίηση ή όχι του συγκεκριμένου νευρώνα, αναλόγως της ισχύος της μεταξύ τους συσχέτισης.

Οι τιμές εισόδου, αφού πολλαπλασιαστούν με τα αντίστοιχα βάρη, αθροίζονται (σύμβολο Σ) και το άθροισμά τους δίνεται ως όρισμα στην συνάρτηση κατωφλιού, γνωστή και ως συνάρτηση ενεργοποίησης (activation function). Αν το άθροισμα του γινομένου κάθε τιμής εισόδου επί το αντίστοιχο βάρος είναι αρκετά μεγάλο, η έξοδος (output) της συνάρτησης θα είναι 1, αντιπροσωπεύοντας την αποστολή ηλεκτρικής ώσης στους επόμενους συνδεδεμένους νευρώνες, διαφορετικά θα είναι 0, αντιπροσωπεύοντας την απουσία της.

Αυτό το απλοποιημένο μαθηματικό μοντέλο ήταν αρκετό ώστε να χρησιμοποιηθεί επιτυχώς σε απλές εργασίες ταξινόμησης. Ωστόσο, οι αδυναμίες του έγιναν γρήγορα αντιληπτές. Το μοντέλο αυτό δεν μπορούσε να δώσει λύση σε περιπτώσεις μη γραμμικής ταξινόμησης, μην έχοντας τη δυνατότητα για παράδειγμα να διδαχτεί μια απλή συνάρτηση XOR, η οποία είναι αληθής όταν οι 2 μεταβλητές εισόδου της έχουν τιμές που διαφέρουν (συνδυασμοί: 1,0 ή 0,1).

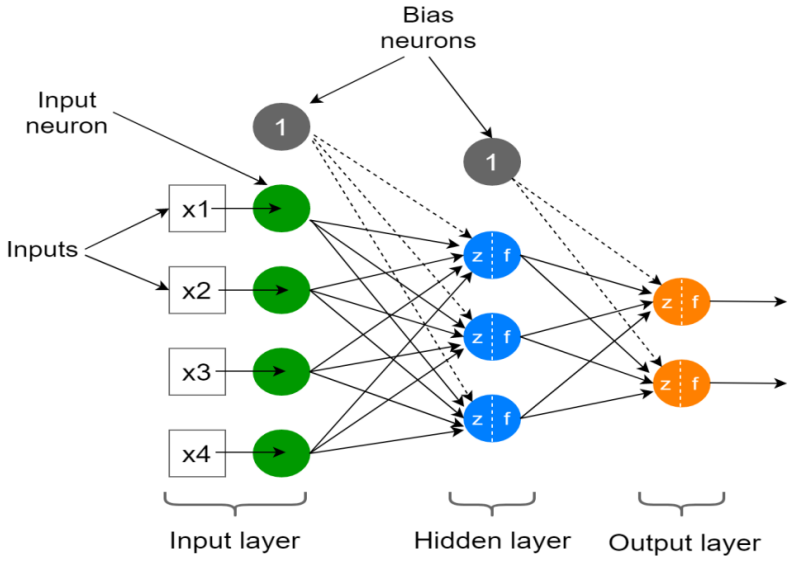
Προβλήματα όπως το παραπάνω απαιτούσαν τη δημιουργία ενός τεχνητού νευρωνικού δικτύου με περισσότερα στρώματα. Οι perceptrons όμως, δεν μπορούσαν να χρησιμοποιηθούν εντός ενός πολυστρωματικού δικτύου, λόγω των αδυναμιών της συνάρτησης ενεργοποίησης που χρησιμοποιούσαν. Μέσω αυτής οι perceptrons των ενδιάμεσων στρωμάτων θα δέχονταν πάντα τιμές 0 ή 1 και θα είχαν πάντα ως έξοδο τιμές 0 ή 1. Αυτό θα καθιστούσε κάθε αλλαγή που θα γινόταν στις τιμές των βαρών κατά την εκπαίδευση τυχαία, αφού δεν θα ήταν δυνατό να προβλεφθεί η επίδρασή της στην αύξηση της αποτελεσματικότητας του μοντέλου.

Το πρόβλημα αυτό λύθηκε με την επινόηση μιας νέας ομάδας συναρτήσεων ενεργοποίησης, με κοινό χαρακτηριστικό το ότι έδιναν ως έξοδο συνεχόμενες πραγματικές τιμές εντός ενός εύρους τιμών. Αυτή η αλλαγή σε συνδυασμό με την έναρξη χρησιμοποίησης της μεθόδου οπισθοδιάδοσης του λάθους (backpropagation method) που αναφέρθηκε και κατά την ιστορική αναδρομή, οδήγησαν στη μετάβαση στα μοντέρνα τεχνητά νευρωνικά δίκτυα, όπως τα γνωρίζουμε σήμερα.

## 2.4 Σύγχρονα τεχνητά νευρωνικά δίκτυα

Τα νευρωνικά δίκτυα που χρησιμοποιούνται πλέον για την επίλυση προβλημάτων διαφόρων ειδών, αποτελούνται από δύο ή και περισσότερα ενδιάμεσα στρώματα με δυνητικά διαφορετικό πλήθος κόμβων, ενώ και ο αριθμός των κόμβων του στρώματος εξόδου (output layer) είναι επίσης μεταβλητός. Έτσι χρησιμοποιούνται και για την ταξινόμηση ενός συνόλου τιμών εισόδου σε περισσότερες από μια ομάδες. Ένα παράδειγμα για το παρακάτω θα μπορούσε να αποτελέσει η ταξινόμηση του αυτοκινήτου που απεικονίζεται σε μια φωτογραφία σε μια από τέσσερις πιθανές κατηγορίες: τζιπ, φορτηγό, επιβατικό ή αγωνιστικό.

Η αρχιτεκτονική ενός μοντέρνου τεχνητού νευρωνικού δικτύου απεικονίζεται στην εικόνα που ακολουθεί:




*Εικόνα 3 Σύγχρονο τεχνητό νευρωνικό δίκτυο*

### 2.4.1 Αρχιτεκτονική

Έχοντας την παραπάνω εικόνα ως σημείο αναφοράς, θα γίνει μια παρουσίαση των μερών που απαρτίζουν το νευρωνικό δίκτυο και της λειτουργίας τους:

Είσοδος (Input)

Το σύνολο των τιμών εισόδου $\mathbf{x_{1..}x_i}$ ενός νευρωνικού δικτύου. Θα πρέπει να είναι πάντα πραγματικοί αριθμοί, ή, αν δεν πρόκειται για αριθμούς, να κωδικοποιηθούν σε τέτοιους. Αυτό είναι απαραίτητο ώστε να είναι δυνατές οι μαθηματικές πράξεις για την βελτιστοποίηση του δικτύου. Ένα παράδειγμα εισόδου αποτελεί το σύνολο των pixels μιας φωτογραφίας της οποίας το περιεχόμενο θέλουμε να αναγνωριστεί από το νευρωνικό δίκτυο.

Βάρη (Weights)

Επίσης πραγματικοί αριθμοί, ένας για κάθε μη διακεκομμένο βέλος που παρατηρείται στην παραπάνω εικόνα. Κάθε κόμβος (node) ή αλλιώς νευρώνας (neuron) που απεικονίζεται με σχήμα κύκλου, από το δεύτερο στρώμα του δικτύου και έπειτα, διαθέτει ένα σύνολο από βάρη. Κάθε ένα από αυτά πολλαπλασιάζεται με τον πραγματικό αριθμό που παράγεται ως έξοδος από τον αντίστοιχο κόμβο του προηγούμενου στρώματος . Επίσης υπάρχει ένας ακόμη κόμβος, ο κόμβος bias, η λειτουργία του οποίου θα επεξηγηθεί στην συνέχεια. Κάθε κόμβος ενός στρώματος λοιπόν διαθέτει αριθμό βαρών ίσο με τον αριθμό των κόμβων του προηγούμενου στρώματος του δικτύου. Οπότε το σύνολο των βαρών μεταξύ ενός στρώματος L και του προηγούμενου L-1 υπολογίζεται ως εξής:

$w_{L,\,L-1} = \Sigma n_L \times \Sigma n_{L-1}$ , όπου Σn το σύνολο των κόμβων κάθε στρώματος.

Κόμβος bias

Έχει λειτουργία παρόμοια με αυτή της σταθεράς σε μια γραμμική συνάρτηση. Η τιμή κάθε τέτοιου κόμβου αθροίζεται μαζί με αυτές των γινομένων των τιμών εξόδου των κανονικών κόμβων επί τα αντίστοιχα βάρη

κάθε επόμενου συνδεδεμένου κόμβου. Έτσι, η τιμή του καθορίζει το κατώτατο όριο των τιμών των υπόλοιπων κόμβων του στρώματος που θα είναι αρκετό για την ενεργοποίηση του επόμενου κόμβου.

Συνάρτηση Ενεργοποίησης

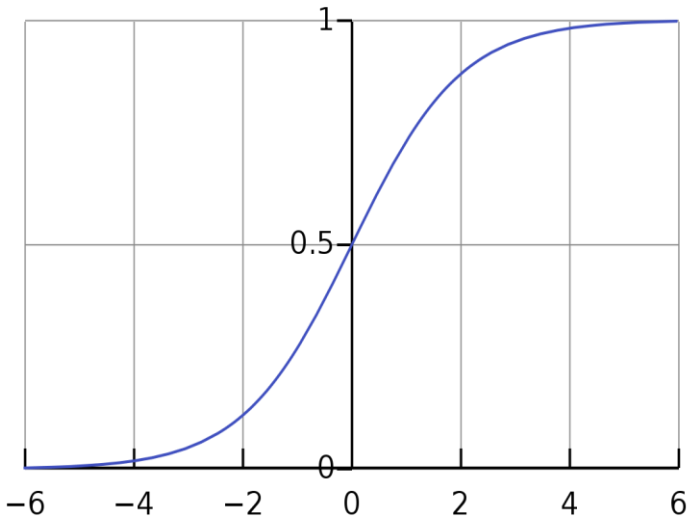
Πρόκειται για μια συνάρτηση (συμβολίζεται με f στην παραπάνω εικόνα) που δέχεται ως είσοδο το άθροισμα z του γινομένου των τιμών εξόδου των κανονικών κόμβων και αυτής του κόμβου bias επί τα αντίστοιχα βάρη και παράγει μια τιμή εξόδου, η οποία πολλαπλασιάζεται με τα βάρη του επόμενου κόμβου. Οι συναρτήσεις ενεργοποίησης που χρησιμοποιούνται πλέον στα νευρωνικά δίκτυα είναι μη γραμμικές, αφού αυτό το είδος συνάρτησης είναι το μόνο που επιτρέπει τη δημιουργία πολυστρωματικών μοντέλων και τη χρήση σε αυτά της συνάρτησης οπισθοδιάδοσης του λάθους. Ακολουθούν οι τέσσερις γνωστότερες μη γραμμικές συναρτήσεις ενεργοποίησης:

1) *Σιγμοειδής*

Δέχεται ως είσοδο οποιονδήποτε πραγματικό αριθμό και δίνει ως έξοδο αριθμούς μεταξύ 0 και 1. Όσο μεγαλύτερη τιμή έχει η είσοδος, τόσο πιο πολύ η έξοδος θα προσεγγίζει την τιμή 1, ενώ όσο μικραίνει η τιμή εισόδου, τόσο η έξοδος της συνάρτησης πλησιάζει προς το 0. Είναι η κατάλληλη επιλογή για περιπτώσεις όπου η έξοδος του νευρωνικού δικτύου αντιπροσωπεύει την πιθανότητα, για παράδειγμα τον βαθμό πιθανότητας του να ανήκει η είσοδος σε μια συγκεκριμένη ομάδα, ο οποίος πάντοτε θα έχει μια τιμή μεταξύ 0 και 1. Η μαθηματική αναπαράσταση της συνάρτησης είναι η παρακάτω, στην οποία το e πρόκειται για τον αριθμό Euler, μια μαθηματική σταθερά με τιμή περίπου ίση με 2,71828:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Το γράφημα της συνάρτησης είναι το ακόλουθο:




*Εικόνα 4 Γράφημα σιγμοειδούς συνάρτησης*

2) *Softmax*

Χρησιμοποιείται συνήθως στο στρώμα εξόδου του νευρωνικού δικτύου σε περιπτώσεις κατηγοριοποίησης μεταξύ πολλών τάξεων. Σε αυτή την περίπτωση η έξοδος δεν είναι ένας μοναδικός αριθμός αλλά ένα διάνυσμα με μήκος ίδιο με τον αριθμό των προς κατηγοριοποίηση τάξεων. Κάθε αριθμός στο διάνυσμα αντιπροσωπεύει την πιθανότητα του να ανήκει η είσοδος στην τάξη που αντιστοιχίζεται στη συγκεκριμένη θέση, υπολογιζόμενος με χρήση της σιγμοειδούς συνάρτησης. Ένα παράδειγμα διανύσματος εξόδου για την περίπτωση κατηγοριοποίησης μεταξύ τεσσάρων διαφορετικών ομάδων/ τάξεων είναι το ακόλουθο: [0.80, 0.55, 0.23, 0.17].

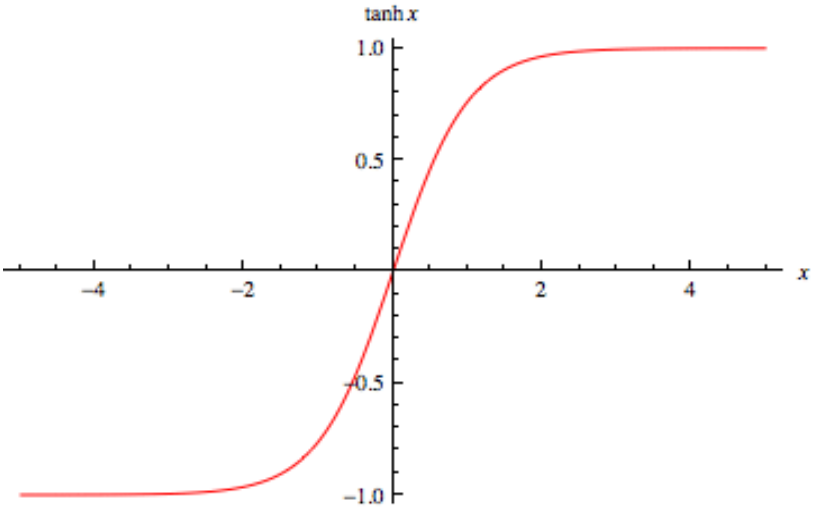
Ωστόσο, κάθε αριθμός του διανύσματος αυτού αντιπροσωπεύει την απόλυτη πιθανότητα του να ανήκει η είσοδος στην αντίστοιχη τάξη, και όχι την σχετική. Το πρόβλημα αυτό επιλύει η συνάρτηση Softmax, η οποία υπολογίζει αυτήν ακριβώς τη σχετική πιθανότητα ως εξής: Για τον υπολογισμό της κάθε τιμής του τελικού διανύσματος, η πιθανότητα του να ανήκει η είσοδος στην αντίστοιχη τάξη διαιρείται με το άθροισμα των πιθανοτήτων όλων των προς κατηγοριοποίηση τάξεων. Έτσι, το παραπάνω διάνυσμα, με την εφαρμογή της συνάρτησης Softmax, θα είχε τις ακόλουθες τιμές: [0.35, 0.27, 0.20, 0.18]. Το άθροισμα των τιμών του διανύσματος είναι πάντα ίσο με τη μονάδα. Ακολουθεί η μαθηματική αναπαράσταση της συνάρτησης, στην οποία επίσης γίνεται χρήση του αριθμού Euler:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

3) *Υπερβολική Εφαπτομένη (Hyperbolic Tangent ή Tahn)*

Παρόμοια με τη σιγμοειδή συνάρτηση, έχει πιο εκτεταμένο εύρος τιμών εξόδου, με τις πιθανές τιμές να βρίσκονται μεταξύ -1 και 1. Αυτό επιτρέπει μια πιο λεπτή κατηγοριοποίησή τους μεταξύ αρνητικών, θετικών και ουδέτερων. Χρησιμοποιείται κυρίως στα ενδιάμεσα στρώματα ενός νευρωνικού δικτύου. Ακολουθούν η μαθηματική αναπαράσταση και το γράφημα της συνάρτησης:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

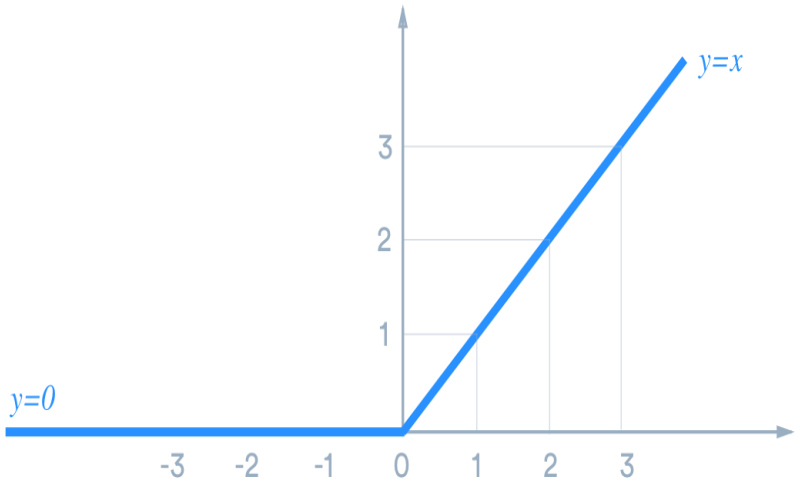



*Εικόνα 5 Γράφημα συνάρτησης Υπερβολικής Εφαπτομένης*

4) *Συνάρτηση Διορθωμένης Γραμμικής Μονάδας (Rectified Linear Unit ή ReLU)*

Η πιο συχνά χρησιμοποιούμενη συνάρτηση ενεργοποίησης στα πολυστρωματικά νευρωνικά δίκτυα. Το εύρος τιμών της είναι πιο ευρύ από των δύο προηγούμενων συναρτήσεων, καθώς δίνει ως έξοδο την μεγαλύτερη τιμή μεταξύ της εισόδου της και του μηδέν. Αυτό την καθιστά πιο αποδοτική από τις προηγούμενες, καθώς οι νευρώνες που έχουν μηδενικό ή αρνητικό άθροισμα των γινομένων των τιμών εισόδου επί τα βάρη δεν ενεργοποιούνται καθόλου, ενώ για όσους ενεργοποιούνται αρκεί μια σύγκριση του παραπάνω αθροίσματος με το μηδέν. Η μαθηματική αναπαράσταση και το γράφημα της συνάρτησης είναι τα ακόλουθα:

$$f(x) = \max(0, x)$$




*Εικόνα 6 Γράφημα συνάρτησης Διορθωμένης Γραμμικής Μονάδας*

Έξοδος (Output)

Αυτό το στρώμα του νευρωνικού δικτύου παράγει τις τιμές εξόδου του, για κάθε σύνολο συγκεκριμένων τιμών εισόδου. Λειτουργεί όπως ακριβώς και τα ενδιάμεσα στρώματα, με μια πιθανή διαφοροποίηση το ότι κάποιες φορές χρησιμοποιείται σε αυτό μια συνάρτηση ενεργοποίησης διαφορετική από αυτή των ενδιάμεσων στρωμάτων. Ο αριθμός κόμβων του στρώματος εξόδου είναι ίδιος με τον αριθμό των προς ταξινόμηση κατηγοριών σε περίπτωση που οι κατηγορίες είναι παραπάνω από μια, ενώ ένας και μοναδικός κόμβος χρησιμοποιείται σε περιπτώσεις όπου το νευρωνικό δίκτυο χρειάζεται να αποφασίσει το αν η είσοδος ανήκει σε μια συγκεκριμένη κατηγορία ή όχι.

Συνάρτηση κόστους (Cost function)

Κατά την επιβλεπόμενη μάθηση, η οποία χρησιμοποιείται για την εκπαίδευση των νευρωνικών δικτύων, είναι ήδη γνωστή η σωστή τιμή εξόδου για κάθε τιμή εισόδου. Κατά τη διάρκεια της εκπαίδευσης του νευρωνικού δικτύου, το σύνολο των τιμών εισόδου δίνονται στο νευρωνικό δίκτυο πολλές φορές, με κάθε φορά από αυτές να ισοδυναμεί με μια διαπέραση των δεδομένων εισόδου. Σκοπός της συνάρτησης κόστους είναι να υπολογίσει τον βαθμό απόκλισης των τιμών εξόδου του δικτύου από τις σωστές τιμές εξόδου και συνήθως εφαρμόζεται μετά από κάθε διαπέραση των δεδομένων εισόδου. Υπάρχουν διάφορες συναρτήσεις κόστους, τέσσερις από τις οποίες παρουσιάζονται παρακάτω:

1) *Μέσο Απόλυτο Σφάλμα (Mean Absolute Error)*

Η συγκεκριμένη συνάρτηση κόστους υπολογίζει το άθροισμα μιας σειράς από επιμέρους αφαιρέσεις. Κάθε αφαίρεση αντιπροσωπεύει την απόλυτη διαφορά μεταξύ κάθε τιμής εξόδου (η οποία συμβολίζεται με το $y_i$) που παράγει το νευρωνικό δίκτυο κατά τη διάρκεια μιας διαπέρασης και της σωστής τιμής εξόδου για κάθε είσοδο (που συμβολίζεται με το $\hat{y}_i$), που είναι γνωστή εκ των προτέρων. Το άθροισμα διαιρείται με τον αριθμό των δεδομένων εισόδου (γράμμα m) και η τελική τιμή αντιπροσωπεύει το μέσο σφάλμα του νευρωνικού δικτύου μετά τη συγκεκριμένη διαπέραση. Ακολουθεί η μαθηματική αναπαράσταση της συνάρτησης κόστους, η οποία αντιπροσωπεύεται από το γράμμα J:

$$J = \frac{1}{m}\sum_{i=1}^{m} |y_i - \widehat{y_i}|$$

2) *Μέσο Τετραγωνισμένο Σφάλμα (Mean Squared Error)*

Πρόκειται για μια από τις πιο διαδεδομένες συναρτήσεις κόστους, αποτελώντας παραλλαγή της παραπάνω συνάρτησης. Η διαφορά μεταξύ τους έγκειται στο ότι στη συγκεκριμένη, η τιμή που προκύπτει από την αφαίρεση της κάθε σωστής τιμής εξόδου από την αντίστοιχη τιμή εξόδου του νευρωνικού δικτύου υψώνεται στο τετράγωνο. Έτσι, τυχόν μεγάλες αποκλίσεις μεταξύ των δύο «τιμωρούνται» περισσότερο. Παρακάτω απεικονίζεται η μαθηματική αναπαράσταση της συνάρτησης:

$$J = \frac{1}{m}\sum_{i=1}^{m} (y_i - \widehat{y_i})^2$$

3) *Δυαδική Διασταυρούμενη Εντροπία (Binary Cross Entropy)*

Χρησιμοποιείται πολύ συχνά σε περιπτώσεις δυαδικής ταξινόμησης, όπου η σωστή τιμή εξόδου είναι 0 ή 1. Η συνάρτηση ενεργοποίησης του στρώματος εξόδου του δικτύου παράγει ένα δεκαδικό αριθμό με τιμή από 0.00 έως 1.00 (για παράδειγμα 0.72), ο οποίος αντιπροσωπεύει την πιθανότητα του να ανήκει η είσοδος στην προς ταξινόμηση κατηγορία. Ο αριθμός αυτός δίνεται ως όρισμα στη συνάρτηση κόστους, η οποία τον συγκρίνει με τη σωστή τιμή εξόδου, που είναι ένας ακέραιος αριθμός, είτε 0 είτε 1. Υπολογίζεται δηλαδή το πόσο απέχει η τιμή εξόδου του νευρωνικού δικτύου (y) από την σωστή τιμή, παίρνοντας τον λογάριθμο της τιμής εξόδου του δικτύου με αρνητικό πρόσημο όταν η σωστή τιμή είναι 1 ή τον λογάριθμο της τιμής εξόδου του δικτύου αφαιρούμενης από τη μονάδα όταν η σωστή τιμή είναι 0. Το σύνολο όλων των επιμέρους διαφορών διαιρούμενο από τον αριθμό m, που αντιπροσωπεύει το πλήθος των τιμών εισόδου, δίνει το κόστος για μια συγκεκριμένη διαπέραση του σετ δεδομένων εισόδου που χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου. Ακολουθεί η μαθηματική αναπαράσταση, με ανάλυση της συνάρτησης κόστους ανά περίπτωση σωστής τιμής εξόδου $\hat{y}$:

$$J = \frac{1}{m}\sum_{i=1}^{m} (\Sigma υν.κόστους(y_i, \widehat{y_i}))$$

Όπου η συνάρτηση κόστους ορίζεται ως εξής:

$$Συν.κόστους(y, \hat{y}) = \begin{cases} -\log(y)\,, αν\ \hat{y} = 1 \\ -\log(1-y)\,, αν\ \hat{y} = 0 \end{cases}$$

4) *Κατηγορηματική Διασταυρούμενη Εντροπία (Categorical Cross Entropy)*

Χρησιμοποιείται αποκλειστικά σε περιπτώσεις ταξινόμησης μεταξύ πολλών τάξεων. Σε αυτή την περίπτωση, αντί για έναν μοναδικό αριθμό εξόδου, χρησιμοποιείται ένα διάνυσμα αριθμών με μήκος όσο και οι τάξεις. Ο δείκτης της κάθε τάξης στο διάνυσμα έχει οριστεί εκ των προτέρων. Το διάνυσμα της σωστής εξόδου αποτελείται από 0 σε όλες τις θέσεις εκτός από αυτή της σωστής τάξης, όπου υπάρχει ο αριθμός 1. Το διάνυσμα της εξόδου του νευρωνικού δικτύου περιλαμβάνει μια κατανομή πιθανότητας, ένα σύνολο δεκαδικών αριθμών με τιμή μεταξύ 0.00 και 1.00, καθένας από τους οποίους αντιπροσωπεύει την πιθανότητα του να ανήκει η τιμή εισόδου του νευρωνικού δικτύου στην τάξη ο δείκτης της οποίας βρίσκεται στη συγκεκριμένη θέση του διανύσματος. Το άθροισμα των παραπάνω αριθμών που συμβολίζουν την εκάστοτε πιθανότητα, είναι 1. Η συγκεκριμένη συνάρτηση κόστους συγκρίνει το πόσο κοντά βρίσκεται αυτή η κατανομή πιθανοτήτων στην πραγματική κατηγοριοποίηση.

Λαμβάνοντας ένα συγκεκριμένο παράδειγμα, έστω τέσσερεις κατηγορίες οχήματος προς ταξινόμηση, με δεδομένα εισόδου τα pixels μιας φωτογραφίας: τζιπ, φορτηγό, επιβατικό αυτοκίνητο, αγωνιστικό αυτοκίνητο. Η σωστή κατηγορία του περιεχομένου μιας συγκεκριμένης φωτογραφίας που δίνεται ως είσοδος είναι φορτηγό, οπότε το διάνυσμα της σωστής τιμής θα ήταν το εξής : [0, 1, 0, 0]. Έστω ότι η έξοδος του νευρωνικού δικτύου για τη συγκεκριμένη είσοδο, κατά την διάρκεια μιας διαπέρασης του σετ δεδομένων εκπαίδευσης, είναι η εξής: [0.21, 0.67, 0.12, 0.00]. Η συνάρτηση πολλαπλασιάζει τον λογάριθμο κάθε τιμής του διανύσματος εξόδου($y_{ij}$, με το i να δηλώνει την i-οστή είσοδο του νευρωνικού δικτύου και το j τη θέση εντός του διανύσματος) με την αντίστοιχη τιμή του διανύσματος με τις σωστές τιμές ($\hat{y}_{ij}$). Επειδή το διάνυσμα των σωστών τιμών έχει 0 σε όλες τις θέσεις πλην αυτής της σωστής κατηγορίας, ο μόνος πολλαπλασιασμός που έχει σημασία στο άθροισμα του κόστους είναι αυτός της θέσης με τιμή 1 του διανύσματος σωστών τιμών με τον λογάριθμο του αριθμού της αντίστοιχης θέσης από το διάνυσμα εξόδου ή, απλοποιώντας, ο λογάριθμος του αριθμού αυτού, αφού πάντα πολλαπλασιάζεται με τη μονάδα. Η ίδια διαδικασία ακολουθείται και για τα υπόλοιπα διανύσματα, στο τέλος των οποίων έχει υπολογιστεί το συνολικό κόστος για τη συγκεκριμένη διαπέραση. Ακολουθεί ο μαθηματικός τύπος:

$$J = -\sum_{i=0}^{m}\sum_{j=0}^{n}(\hat{y}_{i,j} * \log(y_{i,j}))$$

Μέθοδος Οπισθοδιάδοσης του Λάθους (Backpropagation Method)

Η μέθοδος αυτή εφαρμόζεται κατά τη διάρκεια της εκπαίδευσης, μετά από τον υπολογισμό της εξόδου του νευρωνικού δικτύου για το εκάστοτε σετ τιμών εισόδου. Όπως προκύπτει και από το όνομά της, η μέθοδος εφαρμόζεται ανάποδα, ξεκινώντας από το στρώμα εξόδου με κατεύθυνση προς τα προηγούμενα, ως εξής: το διάνυσμα τιμών εξόδου συγκρίνεται με το σωστό διάνυσμα μέσω της επιλεγμένης συνάρτησης κόστους. Το αποτέλεσμα της παραπάνω συνάρτησης αντιπροσωπεύει το «λάθος» κάθε κόμβου του στρώματος εξόδου, πόσο δηλαδή απείχε από την τιμή που θα έπρεπε να έχει ώστε η πρόβλεψη να προσεγγίζει περισσότερο την σωστή για την συγκεκριμένη είσοδο. Στη συνέχεια υλοποιείται μια σειρά μαθηματικών πράξεων που έχει ως στόχο τον υπολογισμό του λάθους (δ) της εξόδου της συνάρτησης ενεργοποίησης για κάθε κόμβο κάθε ενδιάμεσου στρώματος. Το λάθος για κάθε κόμβο προκύπτει από έναν

πολλαπλασιασμό των βαρών που το συνδέουν με κάθε κόμβο του επόμενου στρώματος επί τα λάθη των κόμβων αυτών.

Συγκεκριμένα, το λάθος δ για έναν κόμβο i ενός ενδιάμεσου στρώματος L ενός νευρωνικού δικτύου υπολογίζεται μέσω του παρακάτω τύπου, όπου w ο αριθμός των βαρών που τον συνδέει με τον επόμενο:

$$\delta_i^{(L)} = \sum_{j=0}^{w^{(L,L+1)}} (w_{i,j} * \delta_j^{(L+1)})$$

Αφού υπολογιστούν τα λάθη των κόμβων του στρώματος εξόδου και των ενδιάμεσων στρωμάτων για τη συγκεκριμένη είσοδο, υπολογίζεται για κάθε μεμονωμένο βάρος του νευρωνικού δικτύου το παράγωγο του «λάθους» κάθε κόμβου με τον οποίο συνδέεται, σε συνάρτηση με το βάρος αυτό. Αρχικά υπολογίζεται μια ενδιάμεση τιμή που θα χρησιμοποιηθεί για τον υπολογισμό του τελικού παράγωγου. Για ένα βάρος i,j που συνδέει τον κόμβο i του στρώματος L με τον κόμβο j του στρώματος L+1, λαμβάνεται το γινόμενο του αποτελέσματος της συνάρτησης ενεργοποίησης (α) από τον κόμβο i επί το «λάθος» (δ) του κόμβου j και αποθηκεύεται σε μια μεταβλητή, αφού προστεθεί στην τιμή της, όπως φαίνεται στον παρακάτω μαθηματικό τύπο:

$$\Delta_{i,j}^{(L)} = \Delta_{i,j}^{(L)} + a_i^{(L)} * \delta_j^{(L+1)}$$

Στη μεταβλητή $\Delta_{ij}^{(L)}$ αποθηκεύεται λοιπόν το άθροισμα όλων των παραπάνω γινομένων. Κάθε ένα από αυτά υπολογίζεται μετά από την πρόβλεψη του νευρωνικού δικτύου για κάθε είσοδο που του δίνεται κατά τη διάρκεια μιας διαπέρασης του σετ δεδομένων εισόδου. Στο τέλος αυτής, η μεταβλητή χρησιμοποιείται για τον υπολογισμό του μερικού παραγώγου (D) για κάθε βάρος του νευρωνικού δικτύου, μέσω των παρακάτω τύπων, στον πρώτο από τους οποίους το λ συμβολίζει την παράμετρο κανονικοποίησης (regularization parameter) :

$$D_{i,j}^{(L)} = \frac{1}{m}\left(\Delta_{i,j}^{(L)} + \lambda * w_{i,j}^{(L)}\right), \alpha\nu\ j \neq 0$$

$$D_{i,j}^{(L)} = \frac{1}{m}\Delta_{i,j}^{(L)}, \alpha\nu\ j = 0$$

Μετά από τη ρύθμιση των βαρών όλων των στρωμάτων βάσει του αντίστοιχου μερικού παραγώγου, οι προβλέψεις του νευρωνικού δικτύου στις επόμενες διαπεράσεις των δεδομένων εισόδου θα προσεγγίσουν περισσότερο τις σωστές, που είναι γνωστές εκ των προτέρων κατά τη διάρκεια της εκπαίδευσης.

### 2.4.2 Ορισμός χαρακτηριστικών δεδομένων εισόδου

Ίσως το σημαντικότερο σημείο της διαδικασίας δημιουργίας και εκπαίδευσης ενός νευρωνικού δικτύου. Σε κάποιες περιπτώσεις, η επιλογή δεν είναι δύσκολη. Για παράδειγμα, η είσοδος ενός νευρωνικού δικτύου που εντοπίζει το αντικείμενο που απεικονίζεται σε μια φωτογραφία και το ταξινομεί μεταξύ διάφορων

κατηγοριών θα δέχεται ως είσοδο ένα διάνυσμα με τα pixels της φωτογραφίας και η μόνη απόφαση θα αφορά στον αριθμό των pixels που θα λαμβάνονται.

Σε άλλες περιπτώσεις όμως, απαιτείται συνεργασία μιας ομάδας επιστημόνων δεδομένων έτσι ώστε να αποφασιστούν τα σωστά χαρακτηριστικά (features) των δεδομένων εισόδου, ο αριθμός των οποίων πιθανόν να είναι πολύ μεγάλος. Ένα παράδειγμα αποτελεί η δημιουργία ενός νευρωνικού δικτύου χωρίς ενδιάμεσο στρώμα που θα δέχεται ως είσοδο κάποια χαρακτηριστικά μιας κατοικίας και θα αποφασίζει την τιμή στην οποία αυτή πιθανότατα θα διατίθεται προς αγορά. Σε αυτή την περίπτωση θα πρέπει να επιλεχθούν όλες οι ιδιότητες μιας κατοικίας που θα επηρεάζουν την τιμή της, κάποιες πολύ απλές από τις οποίες θα μπορούσαν να είναι τα τετραγωνικά μέτρα, το έτος στο οποίο κτίστηκε και ο αριθμός των υπνοδωματίων. Όσες από αυτές δεν αποτελούν αριθμούς, θα πρέπει να μετασχηματιστούν σε μια αριθμητική αντιπροσώπευση της πληροφορίας τους, αφού θα χρησιμοποιηθούν σε μαθηματικούς υπολογισμούς κατά την εκπαίδευση του δικτύου. Για παράδειγμα η τιμή του χαρακτηριστικού που θα αφορά στην εγγύτητα της κατοικίας σε σταθμό του μετρό, θα πρέπει από 'Ναι' ή 'Όχι' να αλλάξει σε 1 ή 0 αντιστοίχως.

Από την σωστή επιλογή των χαρακτηριστικών θα εξαρτηθεί σε πολύ μεγάλο βαθμό και η επιτυχία εκπαίδευσης και στη συνέχεια γενίκευσης του νευρωνικού δικτύου σε νέα παραδείγματα.

### 2.4.3 Επιλογή αρχιτεκτονικής μοντέλου

Πριν από την έναρξη της συλλογής δεδομένων και εκπαίδευσης ενός νευρωνικού δικτύου λαμβάνεται η απόφαση για το ποια θα είναι η αρχιτεκτονική του μοντέλου που θα χρησιμοποιηθεί: ο αριθμός των κόμβων του στρώματος εισόδου είναι ίδιος με αυτόν των χαρακτηριστικών των δεδομένων εισόδου που επιλέχθηκαν στο προηγούμενο βήμα. Ο αριθμός των κόμβων του στρώματος εξόδου είναι ισάριθμος με τις κατηγορίες προς ταξινόμηση σε περίπτωση χρήσης του δικτύου για κατηγοριοποίηση ή ένας μοναδικός κόμβος σε περιπτώσεις όπου το νευρωνικό δίκτυο χρησιμοποιείται για γραμμική παλινδρόμηση, όπως στο παράδειγμα της προηγούμενης παραγράφου με την πρόβλεψη της αξίας μιας κατοικίας. Στη συνέχεια επιλέγεται ο αριθμός των ενδιάμεσων στρωμάτων και το πλήθος των κόμβων σε κάθε ένα από αυτά. Τέλος, γίνεται επιλογή της συνάρτησης κόστους, της συνάρτησης ενεργοποίησης των ενδιάμεσων στρωμάτων όπως και αυτής του στρώματος εξόδου.

### 2.4.4 Δημιουργία του σετ δεδομένων εκπαίδευσης

Ένα επίσης πολύ σημαντικό τμήμα της διαδικασίας δημιουργίας ενός τεχνητού νευρωνικού δικτύου. Αφού αποφασιστούν τα χαρακτηριστικά των δεδομένων εισόδου και η αρχιτεκτονική του νευρωνικού δικτύου, θα πρέπει να αποφασιστεί από ποια δεδομένα θα απαρτίζεται το σετ δεδομένων εκπαίδευσης.

Η πιο σπάνια περίπτωση είναι τα δεδομένα αυτά να ξεκινήσουν να δημιουργούνται εκείνη τη στιγμή, με αποκλειστικό σκοπό την εκπαίδευση του νευρωνικού δικτύου. Μια άλλη περίπτωση είναι τα δεδομένα να είναι ήδη εσωτερικά διαθέσιμα στην εταιρεία ή τον οργανισμό που θέλει να εκπαιδεύσει το νευρωνικό δίκτυο. Πολύ συχνό ενδεχόμενο είναι τα δεδομένα να βρίσκονται ήδη διαθέσιμα σε μικρές ή μεγάλες ποσότητες στο Διαδίκτυο, για παράδειγμα χιλιάδες φωτογραφίες διαφόρων οχημάτων άμεσα προσβάσιμες μέσω της μηχανής αναζήτησης Google για ένα νευρωνικό δίκτυο που θα χρειαστεί να εκπαιδευτεί στην αναγνώριση τεσσάρων διαφορετικών τύπων οχημάτων εντός μιας φωτογραφίας. Συχνά επιλέγονται και συνδυασμοί των παραπάνω πηγών δεδομένων εισόδου.

Σε κάθε περίπτωση τα δεδομένα θα πρέπει να υποστούν κάποια επεξεργασία, για παράδειγμα «ξετύλιγμα» των αριθμών που αντιπροσωπεύουν τα pixels μιας φωτογραφίας για την κάθε στήλη κάθε σειράς σε ένα μονοδιάστατο διάνυσμα ή μετατροπή μη αριθμητικών τιμών σε αντίστοιχες αριθμητικές. Το σύνολο των δεδομένων που θα συλλεχθούν χωρίζονται σε δύο κατηγορίες: σετ εκπαίδευσης (training set) και σετ δοκιμής (test set), με συνήθη αναλογία 70-80% και 30-20% αντιστοίχως. Συχνά ένα τμήμα από το σετ εκπαίδευσης (συνήθως 20-30%) ορίζεται ως σετ επαλήθευσης (validation set). Ο ρόλος κάθε τμήματος του σετ δεδομένων περιγράφεται στην επόμενη παράγραφο.

### 2.4.5 Εκπαίδευση

Η εκπαίδευση ενός νευρωνικού δικτύου στο πλαίσιο επιβλεπόμενης μάθησης γίνεται με μια σειρά διαπεράσεων του σετ εκπαίδευσης και ανάλογη ρύθμιση των παραμέτρων του, δηλαδή των βαρών κάθε στρώματος. Στόχος της είναι η προοδευτική αύξηση της ακρίβειας του δικτύου τόσο αναφορικά με το σετ εκπαίδευσης όσο και με το σετ δοκιμής.

Συγκεκριμένα, κάθε είσοδος που ανήκει στο σετ δεδομένων εκπαίδευσης εισάγεται στο νευρωνικό δίκτυο και στη συνέχεια πραγματοποιούνται οι προσθέσεις, οι πολλαπλασιασμοί και εφαρμόζονται οι συναρτήσεις ενεργοποίησης σε κάθε ένα από τα στρώματα του δικτύου μέχρι και το στρώμα εξόδου, όπου δημιουργείται το διάνυσμα της πρόβλεψης του νευρωνικού δικτύου. Ακολουθεί η σύγκριση με το σωστό διάνυσμα εξόδου και η εφαρμογή της μεθόδου της οπισθοδιάδοσης του λάθους. Η παραπάνω διαδικασία ακολουθείται για όλες τις εισόδους του σετ δεδομένων εκπαίδευσης. Η εφαρμογή της στην τελευταία είσοδο του παραπάνω σετ σηματοδοτεί το τέλος της εκάστοτε διαπέρασής του.

Στη συνέχεια υπολογίζεται η ακρίβεια(accuracy) των προβλέψεων του νευρωνικού δικτύου πάνω στο σετ εκπαίδευσης , η οποία πρόκειται για το ποσοστό συμφωνίας με τις επιθυμητές προβλέψεις. Μετρώνται δηλαδή οι σωστές προβλέψεις του δικτύου και δίνεται ένα ποσοστό τους επί τοις εκατό. Επίσης υπολογίζεται η ακρίβειά του και πάνω στο σετ επαλήθευσης, το οποίο περιέχει νέα δεδομένα, που δεν είναι γνωστά στο νευρωνικό δίκτυο από την διαδικασία της εκπαίδευσης. Η ακρίβεια αυτή αποτελεί ένα δείγμα του πόσο καλά μπορεί να γενικεύσει τις προβλέψεις του το νευρωνικό δίκτυο. Χρησιμοποιώντας ένα προηγούμενο παράδειγμα, πρόκειται για το πόσο σωστά θα κατηγοριοποιήσει οχήματα που βρίσκονται σε φωτογραφίες που δεν έχει «ξαναδεί». Μέσω της εκτίμησης της ακρίβειας του νευρωνικού δικτύου στα δεδομένα του σετ επαλήθευσης δίνονται κάποιες ενδείξεις του αν το νευρωνικό δίκτυο έχει αρχίσει να εμφανίζει το φαινόμενο της υπερπροσαρμογής (overfitting), το οποίο, μαζί με αυτό της υποπροσαρμογής (underfitting), θα πρέπει να αποφεύγεται.

Υπερπροσαρμογή υφίσταται όταν τα βάρη του νευρωνικού δικτύου έχουν τιμές τέτοιες ώστε οι προβλέψεις του νευρωνικού δικτύου να προσεγγίζουν υπερβολικά τις σωστές για το σετ εκπαίδευσης αλλά να μην προσεγγίζουν επαρκώς αυτές για τα νέα, όχι γνωστά δεδομένα. Η υποπροσαρμογή αφορά στην αδυναμία προσέγγισης των σωστών τιμών των δεδομένων εκπαίδευσης, κάτι που οδηγεί και σε αντίστοιχη αδυναμία προσέγγισης των σωστών τιμών για νέα δεδομένα. Υπάρχουν διάφορες ενέργειες που μπορεί να ακολουθηθούν σε περίπτωση που διαπιστωθεί κάποιο από τα παραπάνω φαινόμενα, όπως για παράδειγμα η αύξηση των δεδομένων που απαρτίζουν το σετ εισόδου για την αντιμετώπιση του overfitting ή η δημιουργία πιο πολύπλοκων χαρακτηριστικών για τα δεδομένα εισόδου στην περίπτωση του underfitting.

Έπειτα από τον υπολογισμό της ακρίβειας των προβλέψεων για τα σετ εκπαίδευσης και επαλήθευσης, ρυθμίζονται αναλόγως τα βάρη σε κάθε στρώμα του νευρωνικού δικτύου και ξεκινά η επόμενη διαπέραση. Μετά από έναν συνήθως μεγάλο αριθμό διαπεράσεων, αν η ακρίβεια των προβλέψεων του νευρωνικού δικτύου επί του σετ εκπαίδευσης με εξέταση πάνω στο σετ επαλήθευσης κριθεί ικανοποιητική, η διαδικασία της εκπαίδευσης σταματάει. Τότε γίνεται ο υπολογισμός της ακρίβειας επί του σετ δοκιμής.

Το σετ αυτό απαρτίζεται από έναν μεγάλο αριθμό νέων για το νευρωνικό δίκτυο δεδομένων, για τα οποία επίσης υπάρχουν τα διανύσματα που αντιπροσωπεύουν τις σωστές κατηγοριοποιήσεις. Αν η ακρίβεια των προβλέψεων του δικτύου πάνω στο σετ δοκιμής είναι ικανοποιητικά υψηλή, συνήθως λήγει η διαδικασία της εκπαίδευσης. Αν δεν είναι στα επιθυμητά επίπεδα, η εκπαίδευση ξεκινά από την αρχή, αφού πρώτα γίνουν κάποιες ποσοτικές ή ποιοτικές αλλαγές, παραδείγματος χάριν προσθήκη νέων δεδομένων στο σετ εκπαίδευσης ή αλλαγές στην αρχιτεκτονική του νευρωνικού δικτύου.

Γενικότερα, κατά την εκπαίδευση του νευρωνικού δικτύου αναζητείται η χρυσή τομή μεταξύ των φαινομένων της υπερπροσαρμογής και της υποπροσαρμογής. Χρειάζεται δηλαδή το δίκτυο να έχει αρκετά μεγάλη ακρίβεια πρόβλεψης επί των δεδομένων του σετ εκπαίδευσης αλλά και αρκετά μεγάλη επί αυτών του σετ επαλήθευσης, και, στη συνέχεια, του σετ δοκιμής. Πρόκειται για μια επίπονη διαδικασία που απαιτεί μεθόδευση και μια σειρά δοκιμών μέχρι να επιτευχθεί το επιθυμητό αποτέλεσμα.

### 2.4.6 Πρόβλεψη

Μετά την ολοκλήρωση της διαδικασίας της εκπαίδευσης, το νευρωνικό δίκτυο εντάσσεται σε ένα ευρύτερο σύστημα λογισμικού όπου, δεχόμενο δεδομένα από τον πραγματικό κόσμο, πραγματοποιεί προβλέψεις.

Κάθε δεδομένο εισόδου δέχεται την απαραίτητη επεξεργασία ώστε να αποκτήσει δομή ίδια με αυτή των δεδομένων εισόδου με τα οποία εκπαιδεύτηκε το νευρωνικό δίκτυο. Στη συνέχεια γίνονται όλες οι

απαραίτητες αριθμητικές πράξεις εντός του δικτύου και το στρώμα εξόδου δίνει το τελικό διάνυσμα της πρόβλεψης, με την όλη διαδικασία να ολοκληρώνεται συνήθως σε κλάσματα του δευτερολέπτου για κάθε δεδομένο εισόδου.

Σε περιπτώσεις κατηγοριοποίησης, τα διανύσματα εξόδου συνήθως «μεταφράζονται» στην λέξη που αντιπροσωπεύει την κατηγορία στην οποία προβλέφτηκε ότι ανήκει η είσοδος που δόθηκε στο νευρωνικό δίκτυο. Για παράδειγμα, σε ένα νευρωνικό δίκτυο που ταξινομεί οχήματα που εμφανίζονται σε μια φωτογραφία μεταξύ τεσσάρων διαφορετικών κατηγοριών (1: τζιπ, 2: φορτηγό, 3: επιβατικό, 4: αγωνιστικό), μπορεί να υπάρχουν διαφορετικά πιθανά διανύσματα εξόδου, αναλόγως του σημείου του μέγιστου στοιχείου στο διάνυσμα. Έτσι, ένα διάνυσμα εξόδου της μορφής [0.07, 0.89, 0.01, 0.03], με τη μεγαλύτερη τιμή στη δεύτερη θέση, μεταφράζεται ως «φορτηγό».

## 2.5 Συνελικτικά Νευρωνικά Δίκτυα

### 2.5.1 Καινοτομία

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks ή CNNs) είναι μια υποκατηγορία των τεχνητών νευρωνικών δικτύων. Εφαρμόζονται με μεγάλη επιτυχία στο πεδίο της μηχανικής όρασης, στο οποίο ο υπολογιστής καλείται να κατηγοριοποιήσει το περιεχόμενο μιας εικόνας ή μιας ροής βίντεο λειτουργώντας σαν να το βλέπει, εξ ου και το όνομα του πεδίου αυτού.

Το βασικό πρόβλημα που έλυσαν είναι αυτό της εύρεσης των επιμέρους χαρακτηριστικών του αντικειμένου που απεικονίζεται στη φωτογραφία και των χωρικών εξαρτήσεων μεταξύ τους. Ένα απλό νευρωνικό δίκτυο με έναν αριθμό ενδιάμεσων στρωμάτων έχει μέτρια απόδοση στην κατηγοριοποίηση του περιεχομένου μιας εικόνας που του δίνεται ως είσοδος, με την απόδοση να μειώνεται όσο αυξάνεται η πολυπλοκότητα της εικόνας και η ανάγκη αλληλεξαρτήσεων μεταξύ τμημάτων της για τη σωστή ερμηνεία της.

Έστω ένα νευρωνικό δίκτυο με σκοπό την κατηγοριοποίηση του περιεχομένου φωτογραφιών μεταξύ δύο κατηγοριών: σκύλου ή γάτας. Ένα παραδοσιακό νευρωνικό δίκτυο θα εφάρμοζε την εκπαίδευσή του με αλλεπάλληλες ρυθμίσεις των βαρών του μέχρι να δοθεί σε κάθε ένα η τιμή που θα του αναλογεί για την σωστή ερμηνεία του περιεχομένου. Ωστόσο, η πληροφορία που οδηγεί στη σωστή κατηγοριοποίηση μιας φωτογραφίας δεν βρίσκεται πάντα στο ίδιο σημείο –άρα και στην ίδια ομάδα pixels- της εικόνας. Αντιθέτως, το κεφάλι, το σώμα ή η ουρά ενός σκύλου ή γάτας βρίσκονται σε διαφορετικό σημείο και πιθανώς απεικονιζόμενα και από διαφορετική οπτική γωνία σε κάθε φωτογραφία. Αυτό που θα οδηγήσει στην σωστή κατηγοριοποίηση της φωτογραφίας είναι η δυνατότητα εντοπισμού στο περιεχόμενό της μιας σειράς χαρακτηριστικών που έχουν σημασία για τη διαφοροποίηση μεταξύ γάτας και σκύλου, όπως για παράδειγμα τα μάτια, η μύτη, το κεφάλι ως σύνολο, ο κορμός και η ουρά, καθώς και η χωρική αλληλεξάρτηση μεταξύ τους και στη συνέχεια λήψη αυτών ως είσοδο για τη σωστή κατηγοριοποίηση.

Αυτή ακριβώς τη διαδικασία ακολουθούν τα συνελικτικά νευρωνικά δίκτυα. Η λειτουργία τους έχει ως πηγή έμπνευσης αυτή του οπτικού φλοιού του εγκεφάλου. Κατά την επεξεργασία ενός αντικειμένου μέσω της όρασης, μεμονωμένοι νευρώνες δέχονται ερέθισμα από μια συγκεκριμένη περιοχή του οπτικού πεδίου, γνωστή και ως πεδίο πρόσληψης. Η σύνθεση των επιμέρους πεδίων πρόσληψης δίνει την ολοκληρωμένη πληροφορία, καλύπτοντας το σύνολο του οπτικού πεδίου.

### 2.5.2 Αρχιτεκτονική

Η μεγάλη διαφοροποίηση λοιπόν ενός συνελικτικού νευρωνικού δικτύου από ένα απλό νευρωνικό δίκτυο είναι η ύπαρξη μιας σειράς από στρώματα που επεξεργάζονται τα δεδομένα εισόδου και τα δίνουν ως είσοδο στα επόμενα ενδιάμεσα στρώματα, τα οποία είναι ίδια σε λειτουργία με αυτά ενός απλού

νευρωνικού δικτύου, με το στρώμα εξόδου να κάνει την τελική κατηγοριοποίηση. Ακολουθούν τα δύο τμήματα των στρωμάτων που πραγματοποιούν την επεξεργασία αυτή.

Συνελικτικό στρώμα (Convolutional layer)

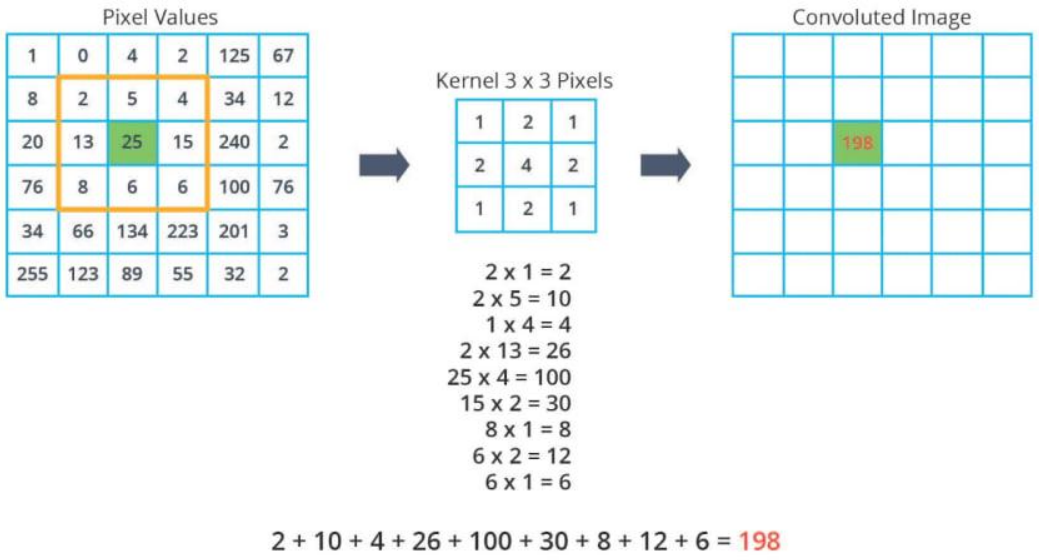
Αποτελεί πάντα το πρώτο στρώμα ενός συνελικτικού νευρωνικού δικτύου, δεχόμενο τα δεδομένα εισόδου, στα οποία υπάρχει μια διαφοροποίηση σε σχέση με τα απλά νευρωνικά δίκτυα. Στην περίπτωση των συνελικτικών νευρωνικών δικτύων, αποφεύγεται η κατασκευή ενός μονοδιάστατου διανύσματος με όλα τα pixels της φωτογραφίας. Αντιθέτως, η φωτογραφία εισόδου δίνεται ως ένας δισδιάστατος πίνακας, κάθε pixel δηλαδή παραμένει στην θέση στην οποία μπορεί να παρατηρηθεί και με το ανθρώπινο μάτι. Αν η φωτογραφία είναι έγχρωμη (συνήθως σε πρότυπο χρώματος RGB), δίνονται για κάθε μια ως είσοδος στο συνελικτικό νευρωνικό δίκτυο τρεις δισδιάστατοι πίνακες, ένας για κάθε ένα από τα τρία βασικά χρώματα, το κόκκινο (**R**ed), το πράσινο (**G**reen) και το μπλε (**B**lue). Σε περίπτωση ασπρόμαυρης φωτογραφίας, δίνεται για κάθε μια μόνο ένας δισδιάστατος πίνακας.

Ένα συνελικτικό στρώμα αποτελείται από έναν συγκεκριμένο αριθμό επίσης δισδιάστατων πινάκων, το πλήθος των οποίων ορίζεται από τους δημιουργούς του εκάστοτε συνελικτικού νευρωνικού δικτύου. Το μέγεθός τους είναι κατά πολύ μικρότερο από αυτό του πίνακα εισόδου, με τις πιο συνηθισμένες διαστάσεις να είναι 3 x 3.

Κάθε μία θέση του εκάστοτε τέτοιου πίνακα περιέχει έναν αρχικά τυχαίο αριθμό που αντιπροσωπεύει το αντίστοιχο βάρος (weight) .Οι πίνακες αυτοί ονομάζονται φίλτρα, λόγω της λειτουργίας τους: κάθε ένας από αυτούς, ξεκινώντας από την πάνω αριστερή γωνία, καλύπτει διαδοχικά τμήματα του δισδιάστατου πίνακα της εικόνας εισόδου, έτσι ώστε κάθε βάρος του να αντιστοιχίζεται με ένα pixel της συγκεκριμένης περιοχής της φωτογραφίας. Στη συνέχεια εφαρμόζεται πολλαπλασιασμός ενός προς ένα του κάθε pixel με το αντίστοιχο βάρος, και το άθροισμά τους τοποθετείται στην επόμενη διαθέσιμη θέση ενός επιπλέον δισδιάστατου πίνακα, ο ακριβής ρόλος του οποίου περιγράφεται παρακάτω. Το φίλτρο στη συνέχεια πραγματοποιεί μια μετακύλιση δεξιά για να καλύψει την περιοχή των αμέσως διπλανών pixels της φωτογραφίας, ενώ όταν καλύψει όλες τις θέσεις μιας σειράς μετακινείται στην αριστερή αρχή της πρώτης από τις επόμενες που δεν έχει καλυφτεί από αυτό. Οι μετακινήσεις αυτές των φίλτρων λέγονται και συνελίξεις, από όπου προκύπτει και το όνομα του στρώματος αυτού.

Σε κάθε συνέλιξη πραγματοποιείται η διαδικασία των πολλαπλασιασμών και της πρόσθεσης και η αποθήκευση στον αντίστοιχο πίνακα. Ο πίνακας αυτός, όταν συμπληρωθούν όλες οι θέσεις του, αντιπροσωπεύει το αποτέλεσμα της κάλυψης όλης της εικόνας από ένα συγκεκριμένο φίλτρο και έχει τόσες θέσεις όσες και οι συνελίξεις που θα πραγματοποιήσει αυτό στην αρχική εικόνα. Οι θέσεις κατά τις οποίες το φίλτρο μετακινείται προς τα δεξιά όταν πρόκειται για την ίδια σειρά ή προς τα κάτω όταν πρόκειται για αλλαγή σειρών καθορίζονται από μια ρυθμιζόμενη παράμετρο που λέγεται βήμα (stride).

Η διαδικασία που περιγράφηκε, ένα στιγμιότυπο της οποίας απεικονίζεται αμέσως παρακάτω, επαναλαμβάνεται για όλα τα επόμενα φίλτρα του συνελικτικού στρώματος, πάνω στην ίδια ακριβώς εικόνα εισόδου. Κάθε ένα από αυτά τα φίλτρα του πρώτου συνελικτικού στρώματος ερευνά κατά κάποιον τρόπο ένα συγκεκριμένο μοτίβο εντός της φωτογραφίας. Κάποιο για παράδειγμα μπορεί να εντοπίζει οριζόντιες γραμμές, ένα άλλο κάθετες, ενώ ένα τρίτο διαγώνιες.




*Εικόνα 7 Παράδειγμα λειτουργίας ενός φίλτρου του συνελικτικού στρώματος και του αντίστοιχου πίνακα*

Με τη συστηματική εφαρμογή του ίδιου φίλτρου σε όλα τα σημεία της εικόνας, επιτυγχάνεται ο εντοπισμός του συγκεκριμένου μοτίβου σε όποιο σημείο της εικόνας και αν βρίσκεται. Αυτό έχει πολλή μεγάλη σημασία, διότι, όπως αναφέρθηκε και κατά την περιγραφή του σκοπού της χρήσης των συνελικτικών νευρωνικών δικτύων, η σημαντική πληροφορία πιθανότατα βρίσκεται σε διαφορετικό σημείο σε κάθε εικόνα εισόδου. Αυτό δηλαδή που διαδραματίζει τον σημαντικότερο ρόλο στην σωστή κατηγοριοποίηση είναι όχι το σημείο στο οποίο βρίσκεται αλλά το αν υπάρχει ή όχι εντός της κάθε εικόνας.

Το τι μοτίβο θα εντοπίζεται από κάθε φίλτρο εξαρτάται από τις τιμές των βαρών του. Πριν την έλευση των συνελικτικών νευρωνικών δικτύων, τα φίλτρα κατασκευάζονταν εκ των προτέρων από ειδικούς στη μηχανική μάθηση, αναλόγως του ποια μοτίβα ή συγκεκριμένα χαρακτηριστικά χρειαζόταν να εντοπιστούν εντός της φωτογραφίας για να κατηγοριοποιηθεί σωστά. Η εντυπωσιακή καινοτομία όμως των συνελικτικών νευρωνικών δικτύων είναι ότι εντός του δικτύου κατασκευάζονται σταδιακά τα κατάλληλα φίλτρα από μόνα τους, χωρίς καμία εκ των προτέρων παρέμβαση.

Ξεκινώντας δηλαδή με τυχαίες τιμές, κάθε πίνακας που αντιστοιχίζεται σε ένα φίλτρο διαμορφώνει κατά τη διάρκεια της εκπαίδευσης προοδευτικά τα βάρη του με τρόπο που να ερευνά για ένα συγκεκριμένο μοτίβο που κρίνεται ως σημαντικό για την σωστή τελική ταξινόμηση της εικόνας εισόδου. Το σύνολο των επιμέρους μοτίβων που τελικά ερευνούνται εντός της εικόνας είναι αυτό το οποίο οδηγεί στην ακριβέστερη πρόβλεψη της κατηγορίας του συνόλου των εισόδων. Το δίκτυο δηλαδή αυτορυθμίζεται μέσω μαθηματικών υπολογισμών ώστε να ψάχνει για το καταλληλότερο σύνολο των επιμέρους χαρακτηριστικών που κρίνουν περισσότερο από όλα τη σωστή κατηγοριοποίηση της εικόνας. Το μόνο που ορίζεται εκ των προτέρων είναι ο αριθμός των επιμέρους χαρακτηριστικών αυτών, μέσω της επιλογής του αριθμού των φίλτρων του συγκεκριμένου συνελικτικού στρώματος. Όπως θα περιγραφεί και στη συνέχεια, συνήθως δεν υπάρχει μόνο ένα συνελικτικό στρώμα αλλά περισσότερα, με σταδιακή αύξηση της πολυπλοκότητας των μοτίβων που αναζητούνται από τα φίλτρα του κάθε επόμενου συνελικτικού στρώματος.

Στρώμα υποδειγματοληψίας (Pooling layer)

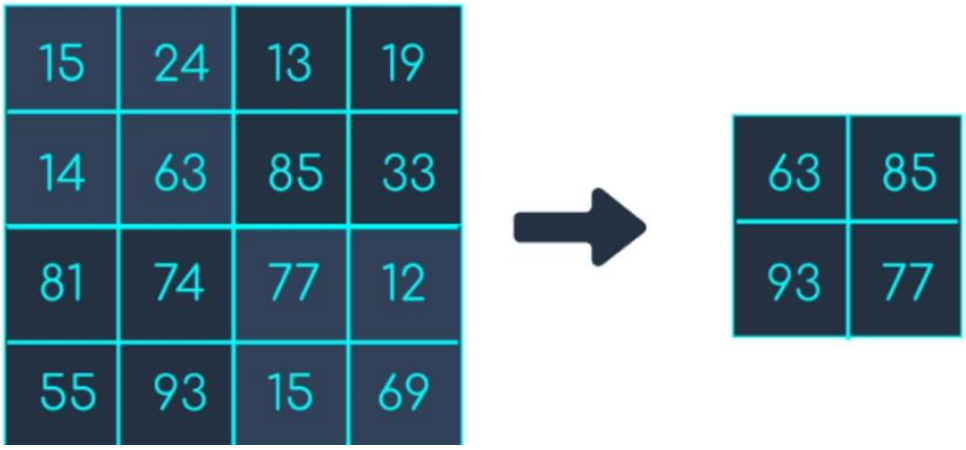
Και αυτό το στρώμα του συνελικτικού νευρωνικού δικτύου αποτελείται από μια σειρά δισδιάστατων πινάκων, με ακριβείς διαστάσεις ορισμένες εκ των προτέρων και πλήθος ίδιο με αυτό των πινάκων που προέκυψαν ως έξοδος από το αμέσως προηγούμενο συνελικτικό στρώμα. Η διαφορά τους από τα φίλτρα του συνελικτικού στρώματος είναι ότι δεν έχουν βάρη στο εσωτερικό τους. Κάθε τέτοιος πίνακας απλώς διατρέχει τον αντίστοιχο πίνακα που δόθηκε ως έξοδος από το συνελικτικό στρώμα, καλύπτοντας

σταδιακά διάφορα τμήματά του ξεκινώντας από τη πάνω αριστερή γωνία, όπως ακριβώς και τα φίλτρα του συνελικτικού στρώματος στην αρχική εικόνα εισόδου.

Για κάθε τμήμα που καλύπτεται, πραγματοποιείται ένας μαθηματικός υπολογισμός, το αποτέλεσμα του οποίου αποθηκεύεται στο επόμενο διαθέσιμο σημείο ενός ακόμη πίνακα, με τις επιλογές να είναι δύο: α) υπολογισμός του μέσω όρου όλων των τιμών pixels του τμήματος που καλύπτεται (average pooling) ή β) επιλογή της μέγιστης τιμής pixel του αντίστοιχου τμήματος (max pooling), με την δεύτερη επιλογή να χρησιμοποιείται στην πλειοψηφία των περιπτώσεων. Η διαδικασία συνεχίζεται μέχρι να καλυφτούν σταδιακά όλα τα τμήματα της εκάστοτε εικόνας εξόδου του συνελικτικού στρώματος και να γεμίσουν όλες οι θέσεις του αντίστοιχου πίνακα. Η παραπάνω ακολουθία συνελίξεων και αποθήκευσης της μέγιστης αξίας κάθε περιοχής επαναλαμβάνεται για όλες τις εικόνες εξόδου του συνελικτικού στρώματος, με τα αποτελέσματα να αποθηκεύονται κάθε φορά σε έναν αντίστοιχο νέο πίνακα.

Με αυτόν τον τρόπο πραγματοποιείται μια υποδειγματοληψία της κάθε εικόνας εξόδου του συνελικτικού στρώματος και προκύπτει μείωση διαστάσεων (dimensionality reduction), καθώς μειώνονται οι παράμετροι που προκύπτουν από το συνελικτικό στρώμα, ενώ παράλληλα διατηρείται η σημαντική πληροφορία. Έτσι αυξάνεται η αποδοτικότητα του συνελικτικού νευρωνικού δικτύου, ενώ παράλληλα μειώνεται η πιθανότητα υπερπροσαρμογής.

Σε ένα συνελικτικό νευρωνικό δίκτυο, ένα στρώμα υποδειγματοληψίας διαδέχεται πάντα ένα συνελικτικό στρώμα, αποτελώντας απαραίτητο συμπλήρωμά του. Στην παρακάτω εικόνα αποτυπώνεται ένα στιγμιότυπο της λειτουργίας του στρώματος αυτού:

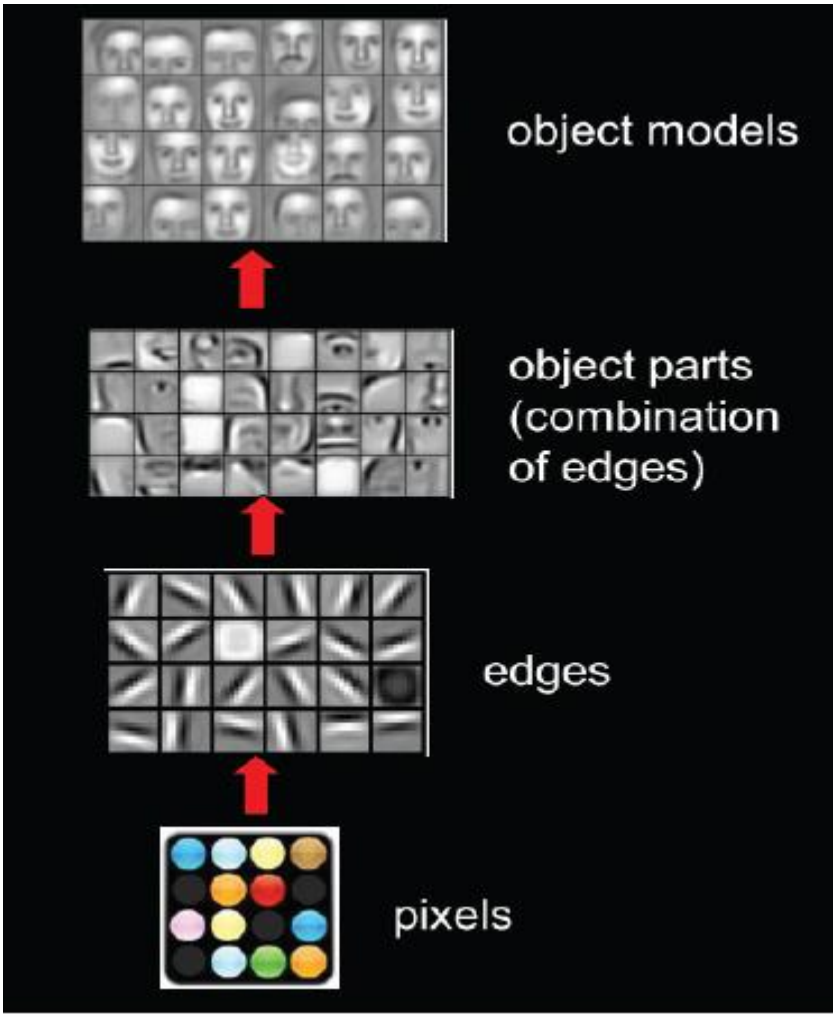



*Εικόνα 8 Max pooling με χρήση πίνακα υποδειγματοληψίας 2 X 2 και αποθήκευση σε νέο πίνακα*

Επιπλέον ενδιάμεσα στρώματα (συνελικτικά και υποδειγματοληψίας)

Στην πλειοψηφία των περιπτώσεων χρήσης συνελικτικών νευρωνικών δικτύων, χρειάζεται να εντοπιστούν περίπλοκα χαρακτηριστικά εντός των εικόνων εισόδου του σετ εκπαίδευσης. Στις περιπτώσεις αυτές, χρησιμοποιούνται επιπλέον συνελικτικά στρώματα, κάθε ένα από τα οποία ακολουθείται από το αντίστοιχο στρώμα υποδειγματοληψίας. Κάθε επόμενο συνελικτικό στρώμα εφαρμόζει φίλτρα στις εικόνες που προέκυψαν ως έξοδος από το αμέσως προηγούμενο συνελικτικό στρώμα, μετά και την εφαρμογή της υποδειγματοληψίας, εντοπίζοντας ολοένα και πιο περίπλοκα χαρακτηριστικά.

Για παράδειγμα, ενώ τα φίλτρα του πρώτου συνελικτικού στρώματος μπορεί να εντόπιζαν απλώς οριζόντιες, κάθετες και διαγώνιες γραμμές, τα φίλτρα του επόμενου μπορεί να εντοπίζουν γραμμές που τέμνονται, κοιλότητες και γωνίες. Η πολυπλοκότητα των χαρακτηριστικών που εντοπίζονται αυξάνεται προοδευτικά σε κάθε συνελικτικό στρώμα, με τα τελευταία στρώματα τέτοιου είδους να μπορούν να εντοπίσουν πολύ σύνθετα μοτίβα εντός της κάθε εικόνας:




*Εικόνα 9 Εντοπισμός χαρακτηριστικών εικόνας με χρήση συνελικτικών στρωμάτων*

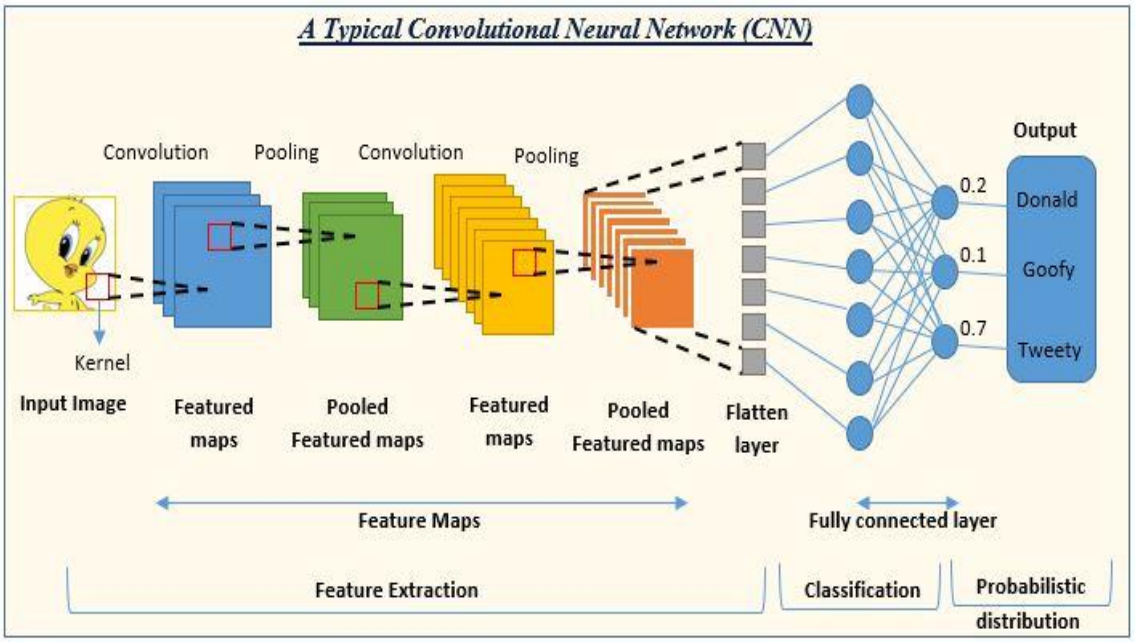
Πλήρως Συνδεδεμένο Στρώμα (Fully Connected Layer)

Η έξοδος του τελευταίου συνελικτικού στρώματος, μετά και την εφαρμογή σε αυτή της υποδειγματοληψίας, περιέχει τα πιο σύνθετα εντοπισμένα χαρακτηριστικά της εκάστοτε εικόνας εισόδου, αποθηκευμένα ως μια σειρά δισδιάστατων πινάκων. Σε αυτό το σημείο σχηματίζεται ένα μονοδιάστατο διάνυσμα που περιέχει τις τιμές όλων αυτών των πινάκων.

Το διάνυσμα αυτό δίνεται ως είσοδος σε ένα στρώμα ακριβώς ίδιο με το στρώμα εισόδου ενός σύγχρονου νευρωνικού δικτύου, όπως περιγράφηκε στην αντίστοιχη ενότητα.
Το στρώμα αυτό ακολουθούν ένα ή περισσότερα πλήρως συνδεδεμένα στρώματα, και εκείνα με ιδιότητες ακριβώς ίδιες με αυτές που έχουν ήδη περιγραφεί. Κάθε ένα διαθέτει μια σειρά από βάρη μέσω των οποίων συνδέεται με κάθε κόμβο του προηγούμενο στρώματος, συνδέεται δηλαδή πλήρως με αυτό. Το τελευταίο πλήρως συνδεδεμένο στρώμα είναι αυτό της εξόδου (σε ορισμένες περιπτώσεις είναι και το μοναδικό πλήρως συνδεδεμένο μετά από αυτό της εισόδου), το οποίο πραγματοποιεί την πρόβλεψη, δίνοντας το διάνυσμα με τη κατανομή των πιθανοτήτων του να ανήκει η είσοδος στην εκάστοτε κατηγορία. Για τον σκοπό αυτό χρησιμοποιείται στο στρώμα εξόδου η συνάρτηση ενεργοποίησης Softmax. Για την προσαρμογή των βαρών των πλήρως συνδεδεμένων στρωμάτων χρησιμοποιούνται όλα τα εργαλεία που

περιγράφηκαν στην ενότητα των νευρωνικών δικτύων, όπως οι συναρτήσεις κόστους και η μέθοδος οπισθοδιάδοσης του λάθους.

Μετά από πολλές διαπεράσεις του σετ εκπαίδευσης, το τμήμα του συνελικτικού δικτύου που πραγματοποιεί τον εντοπισμό και την εξαγωγή χαρακτηριστικών έχει ρυθμιστεί έτσι ώστε να εντοπίζει τα πιο σημαντικά χαρακτηριστικά και το δεύτερο τμήμα, αυτό που κάνει την τελική κατηγοριοποίηση, έχει ρυθμισμένες τις τιμές των βαρών του έτσι ώστε η πρόβλεψη να έχει τη μεγαλύτερη δυνατή ακρίβεια επί του σετ εκπαίδευσης με δοκιμή στο σετ επαλήθευσης αλλά και πολύ καλή απόδοση επί του σετ δοκιμής.

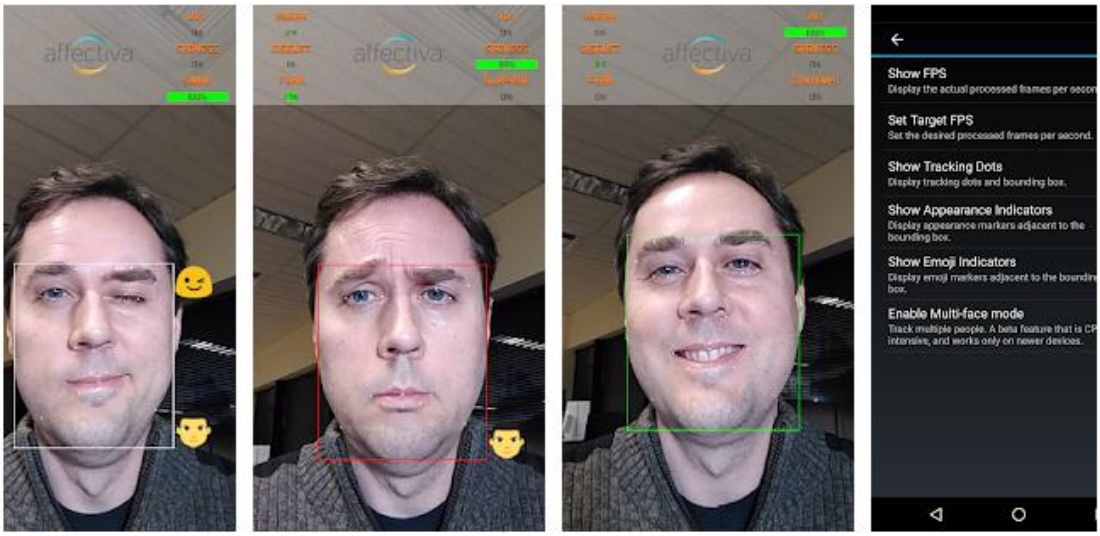



*Εικόνα 10 Αρχιτεκτονική ενός τυπικού Συνελικτικού Νευρωνικού Δικτύου*

# 3 ΠΑΡΕΜΦΕΡΕΙΣ ΕΦΑΡΜΟΓΕΣ

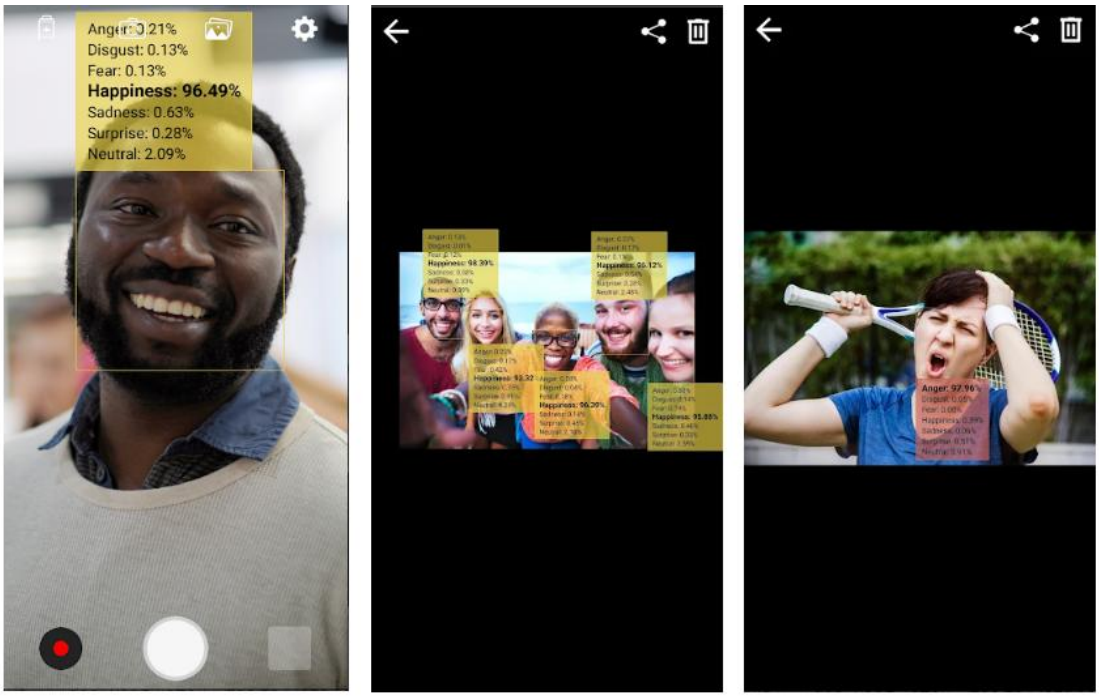
Έπειτα από έρευνα στο Google Play με τα κατάλληλα keywords, βρέθηκαν αρκετές εφαρμογές για Android οι οποίες παρέχουν τη δυνατότητα αναγνώρισης συναισθήματος σε φωτογραφία ή σε βίντεο. Ακολουθεί η παρουσίαση των περισσότερων από αυτές.

## 3.1 AffdexMe



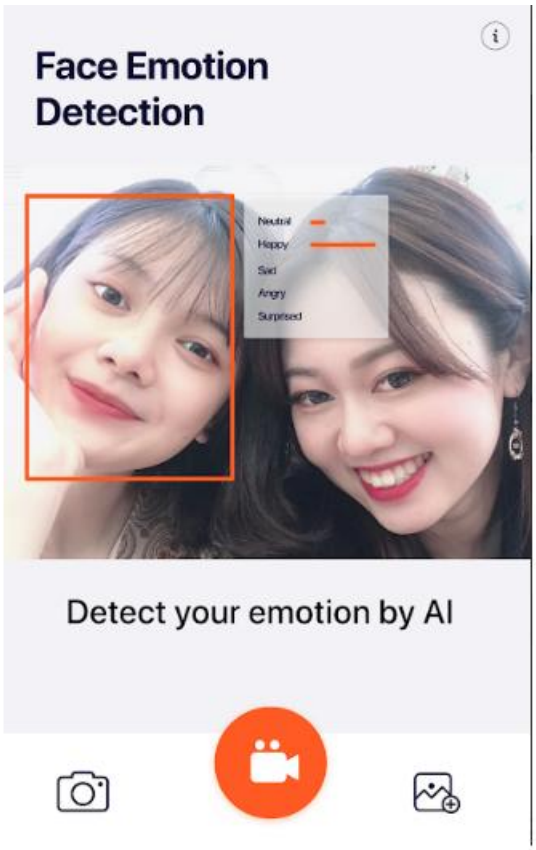

Με την είσοδο στην εφαρμογή ανοίγει το preview της κάμερας του χρήστη και εντοπίζεται το πρόσωπό του, ενώ στο πάνω μέρος της οθόνης απεικονίζονται 6 συναισθήματα, όπως φαίνεται και σε αυτή τη φωτογραφία από το Google Play. Κάθε στιγμή γίνεται μια πρόβλεψη του αλγόριθμου για το ποιο συναίσθημα απεικονίζεται στο πρόσωπο του χρήστη και ταυτόχρονα γίνεται πράσινο το κείμενο κάτω από το συναίσθημα, το οποίο δείχνει την σιγουριά (confidence) του για την πρόβλεψη. ενώ εμφανίζεται και το ανάλογο emoji δίπλα στο περίγραμμα του προσώπου. Μέσω των ρυθμίσεων υπάρχει η δυνατότητα επιλογής αναγνώρισης πιο εξειδικευμένων μορφασμών προσώπου, όπως ενός μειδιάματος, ύψωσης των φρυδιών ή κλεισίματος των ματιών, μεταξύ άλλων.

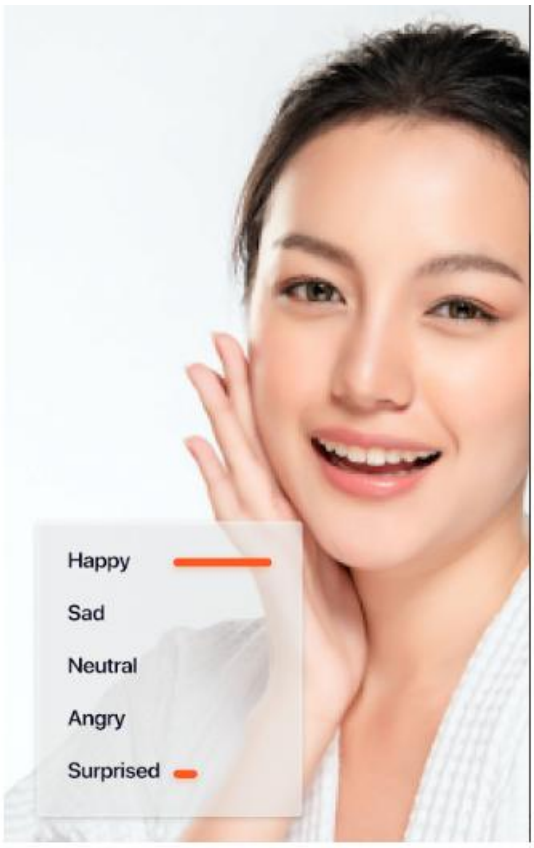
## 3.2 Emotimeter




Και αυτή η εφαρμογή πραγματοποιεί ανάλυση συναισθήματος βάσει της ροής που δίνεται από το preview της συσκευής του χρήστη. Κοντά στο περίγραμμα του εντοπισμένου προσώπου απεικονίζονται τα επτά βασικά συναισθήματα και, δίπλα σε κάθε ένα από αυτά, το αντίστοιχο ποσοστό της σιγουριάς του αλγόριθμου. Το συναίσθημα που προβλέφθηκε ότι απεικονίζεται, εμφανίζεται με bold γράμματα και μεγαλύτερη γραμματοσειρά, ενώ υπάρχει και ανάλογως χρωματισμός στο κουτάκι. Η εφαρμογή μπορεί να αναγνωρίσει συναισθήματα και σε ομαδικές φωτογραφίες.

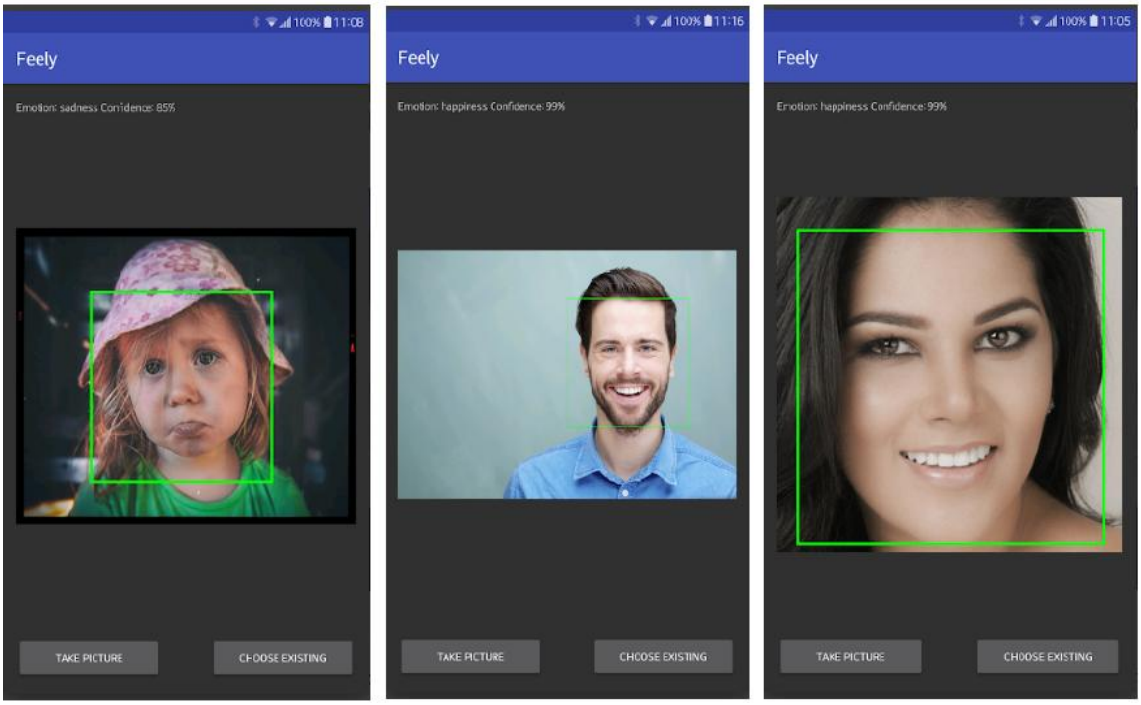
## 3.3 Face Emotion



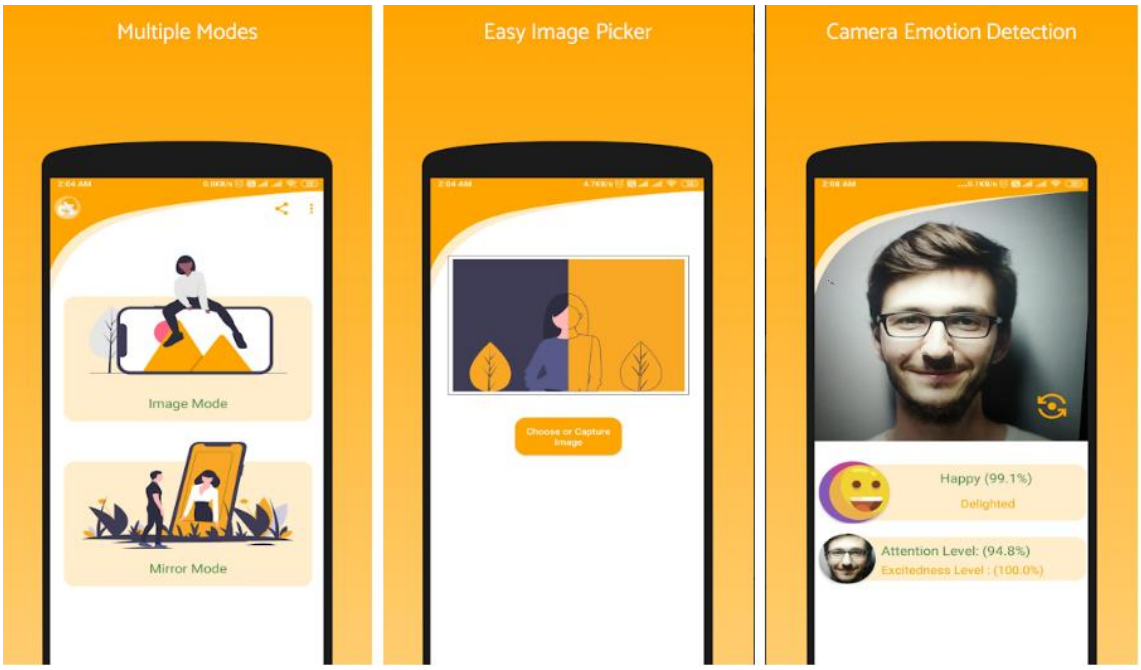

Έχει παρόμοια λειτουργία με τις παραπάνω εφαρμογές. Στην συγκεκριμένη γίνεται αναγνώριση πέντε βασικών συναισθημάτων, ενώ η πρόβλεψη στη ροή από το preview δεν είναι συνεχόμενη, καθώς υπάρχει μια μικρή παύση μεταξύ κάθε ανάλυσης συναισθήματος. Αυτό κρίνεται ως θετικό, καθώς ξεκουράζονται τα μάτια του χρήστη από τη συνεχή εναλλαγή των τιμών στην οθόνη της συσκευής. Δίπλα στα δύο συναισθήματα με την μεγαλύτερη πιθανότητα σωστής περιγραφής της συναισθηματικής κατάστασης του χρήστη εμφανίζεται μια μπάρα, το μήκος της οποίας εξαρτάται από το ποσοστό της σιγουριάς του αλγόριθμου για το εκάστοτε συναίσθημα.

## 3.4 Feely




Όπως φαίνεται και στις παραπάνω εικόνες, ο χρήστης μπορεί να τραβήξει μια φωτογραφία του προσώπου του εντός της εφαρμογής ή να επιλέξει μία από τη Συλλογή Φωτογραφιών της συσκευής. Αφού τραβηχτεί ή επιλεχτεί η φωτογραφία, πατώντας πάνω στο τετράγωνο εντοπισμού του προσώπου, εμφανίζεται στο πάνω μέρος της οθόνης η πρόβλεψη με το συναίσθημα και το ποσοστό σιγουριάς του αλγόριθμου. Σε αντίθεση με τις προηγούμενες εφαρμογές, στη συγκεκριμένη δεν υπάρχει η δυνατότητα αναγνώρισης συναισθήματος στη ροή του preview της οθόνης του κινητού. Ωστόσο, η ακρίβεια του αλγόριθμου στην αναγνώριση του συναισθήματος που απεικονίζεται είναι υψηλή.

## 3.5 Expression AI-Offline Emotion Finder-Actor Trainer




Αποτελεί την εφαρμογή με το πιο όμορφο γραφικό περιβάλλον από όσες εξετάστηκαν. Δίνεται η δυνατότητα ανάλυσης συναισθήματος είτε σε φωτογραφίες (με επιλογή μεταξύ ήδη αποθηκευμένων στη συσκευή ή με λήψη νέας εντός της εφαρμογής) είτε σε ζωντανή ροή στο preview της κάμερας. Ξεχωρίζει από τις προηγούμενες λόγω της απεικόνισης, εκτός από το συναίσθημα, και του επιπέδου προσοχής και έντασης βάσει της εικόνας του χρήστη.

# 4 ΛΟΓΙΣΜΙΚΟ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΕ

## 4.1 Jupyter Notebook

Πρόκειται για ένα περιβάλλον ανάπτυξης κώδικα βασισμένο στο Διαδίκτυο, που λειτουργεί μέσω browser σε localhost. Περιέχει υποστήριξη για την γλώσσα Python, με χρήση της οποίας έγινε όλη η προεπεξεργασία του, dataset το οποίο στη συνέχεια χρησιμοποιήθηκε για την εκπαίδευση του νευρωνικού δικτύου. Πολύ χρήσιμη η δυνατότητα εκτέλεσης του κώδικα σε τμήματα, με το κάθε κομμάτι να εκτελείται ως ξεχωριστό script με ταυτόχρονη αποθήκευση των αποτελεσμάτων εκτέλεσης. Η βασικότερη αιτία επιλογής του ήταν η δυνατότητα χρησιμοποίησης σε αυτό της βιβλιοθήκης Tensorflow. Η εγκατάστασή του έγινε μέσω της πλατφόρμας ανοιχτού κώδικα Anaconda.

## 4.2 Tensorflow

Μαθηματική βιβλιοθήκη που αναπτύχθηκε από τη Google Brain, την ομάδα τεχνητής νοημοσύνης της Google, σε αρχικό στάδιο για ιδία χρήση. Τον Νοέμβριο του 2015 διανεμήθηκε δημόσια κάτω από την άδεια ανοιχτού λογισμικού της Apache (Apache 2.0 Open Source License). Αποτελεί μια πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Μέσω μιας πληθώρας εργαλείων και βιβλιοθηκών, δίνει στους προγραμματιστές τη δυνατότητα δημιουργίας, εκπαίδευσης και μετέπειτα χρήσης μοντέλων νευρωνικών δικτύων, διευκολύνοντας τη δημιουργία εφαρμογών που βασίζονται στη μηχανική μάθηση. Τον Σεπτέμβριο του 2019 κυκλοφορήθηκε από την Google μια αναβάθμιση με όνομα Tensorlow 2.0, η οποία χρησιμοποιήθηκε μέσω του Jupyter Notebook και στην παρούσα διπλωματική εργασία.

## 4.3 Tensorflow Lite

Πρόκειται για μια βιβλιοθήκη για κινητές συσκευές που δίνει τη δυνατότητα χρήσης μοντέλων νευρωνικών δικτύων που εκπαιδεύτηκαν μέσω του Tensoflow. Η διαδικασία που ακολουθείται είναι η εξής: Αρχικά πραγματοποιείται εκπαίδευση του μοντέλου σε υπολογιστή με χρήση της βιβλιοθήκης Tensorflow. Στη συνέχεια, μέσω του Tensorflow Lite Converter, το μοντέλο μετατρέπεται σε ένα συμπιεσμένο flat buffer. Τέλος, το συμπιεσμένο αρχείο φορτώνεται στην κινητή συσκευή, όπου πραγματοποιεί προβλέψεις πάνω στα δεδομένα εισόδου της εφαρμογής από την οποία χρησιμοποιείται.

## 4.4 PostgreSQL

Ένα open-source σύστημα σχεσιακών βάσεων δεδομένων που χρησιμοποιεί τη γλώσσα SQL (Structured Query Language), γνωστό και ως Postgres, με ήδη πάνω από τριάντα χρόνια ανάπτυξης. Η διαχείριση μιας βάσης δεδομένων PostgreSQL γίνεται μέσω της πλατφόρμας διαχείρισης και ανάπτυξης pgAdmin, που προβάλλεται μέσω του browser. Η Postgres είναι συμβατή με όλα τα βασικά λειτουργικά συστήματα ενώ διαθέτει και JDBC (Java Database Connectivity) driver, κάτι που καθιστά δυνατή την εύκολη σύνδεσή της με το Spring Framework μέσω του JPA (Java Persistence API).

## 4.5 Spring Boot

Το Spring Boot αποτελεί μια εξέλιξη του δημοφιλούς Spring framework που απλοποιεί σε πολύ μεγάλο βαθμό τη διαδικασία δημιουργίας stand-alone εφαρμογών που λειτουργούν ως services, με χρήση του web εργαλείου Spring Initializr. Μέσω έτοιμου κώδικα δίνεται με πολύ λίγες ενέργειες η δυνατότητα δημιουργίας ενός RESTful web service, το οποίο μπορεί στη συνέχεια να παραμετροποιηθεί περαιτέρω με προσθήκη νέων modules. Ένα μεγάλο πλεονέκτημα χρήσης του Spring Boot είναι η δυνατότητα αλληλεπίδρασης με τη βάση δεδομένων με χρήση αποκλειστικά κώδικα Java, κάτι που καθιστά πολύ εύκολη και την αντικατάστασή της βάσης με άλλη χωρίς να χρειάζεται καμία αλλαγή στον κώδικα της εφαρμογής.

## 4.6 Android Studio

Το επίσημο προγραμματιστικό περιβάλλον για σχεδιασμό εφαρμογών για το λειτουργικό σύστημα Android, βασισμένο στο λογισμικό IntelliJ IDEA της JetBrains, παρέχει όλα τα εργαλεία που χρειάζονται για την ανάπτυξη μιας Android εφαρμογής. Παράλληλα διαθέτει μια σειρά από εξομοιωτές (emulators), για εύκολη προσομοίωση της εμφάνισης και λειτουργίας της εφαρμογής σε διαφόρων τύπων υποστηριζόμενες φορητές συσκευές, από smartwatch μέχρι και tablet. Χρησιμοποιήθηκε για τη συγγραφή του κώδικα που εκτελείται στις Android συσκευές των χρηστών της εφαρμογής, οι οποίες λειτουργούν ως clients του Spring Boot web service.

# 5 ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

## 5.1 Νευρωνικό Δίκτυο

### 5.1.1 Επιλογή του σετ δεδομένων

Ως βάση για το σετ δεδομένων επιλέχθηκαν δύο datasets από τη γνωστή διαδικτυακή πλατφόρμα kaggle.com, το FER-2013 και το MMA FACIAL EXPRESSION.

Το πρώτο αποτελείται από ασπρόμαυρες φωτογραφίες διαστάσεων 48x48 pixels. Περιλαμβάνει έναν φάκελο αρχείων για την εκπαίδευση του νευρωνικού δικτύου και έναν για τη δοκιμή της απόδοσής του. Και οι δύο φάκελοι περιέχουν επτά υποφακέλους ο καθένας, έναν για κάθε ένα από τα ακόλουθα επτά βασικά συναισθήματα: θυμό, χαρά, ουδετερότητα, θλίψη, έκπληξη, φόβο και αηδία. Το σετ δοκιμής αποτελείται από 3.589 αρχεία και το σετ εκπαίδευσης από 28.709.

Το δεύτερο αποτελείται από έγχρωμες φωτογραφίες ίδιων διαστάσεων και περιλαμβάνει τρεις φακέλους, έναν για το σετ εκπαίδευσης, έναν για το σετ επαλήθευσης και έναν για το σετ δοκιμής. Όλοι οι φάκελοι περιέχουν επτά υποφακέλους έκαστος, με τα ίδια βασικά συναισθήματα όπως και το παραπάνω dataset. Όλοι οι φάκελοι μαζί περιέχουν 128.000 αρχεία κατά προσέγγιση.

### 5.1.2 Επεξεργασία του σετ δεδομένων

Αρχικά τα σετ επαλήθευσης συγχωνεύτηκαν με τα σετ δοκιμής, για διευκόλυνση της περαιτέρω επεξεργασίας του σετ δεδομένων. Στη συνέχεια αφαιρέθηκαν από το σετ δεδομένων οι φάκελοι που περιείχαν φωτογραφίες για τα συναισθήματα της αηδίας και του φόβου. Η πλειοψηφία των δεδομένων στο εσωτερικό των φακέλων αυτών δεν ήταν ενδεικτικά του ανάλογου συναισθήματος, έτσι ώστε να χρησιμοποιηθούν για την εκπαίδευση του δικτύου, ενώ ταυτόχρονα κρίθηκε σχεδόν απίθανη η εμφάνιση τέτοιων συναισθημάτων σε ένα περιβάλλον εξ αποστάσεως εκπαίδευσης.

Για την περαιτέρω επεξεργασία του σετ δεδομένων δημιουργήθηκαν ορισμένα scripts με χρήση της γλώσσας Python, τα οποία εκτελέστηκαν στο περιβάλλον ανάπτυξης Jupyter Notebook.

Το πρώτο από αυτά χρησιμοποιήθηκε για τη μετατροπή των φωτογραφιών και του δεύτερου dataset σε ασπρόμαυρες. Λήφθηκε η απόφαση το συνελικτικό νευρωνικό δίκτυο που θα δημιουργηθεί να έχει ως είσοδο ασπρόμαυρες φωτογραφίες διαστάσεων 48x48 pixels. Το χρώμα δεν θα προσέδιδε παραπάνω χρήσιμη πληροφορία στη διαδικασία της αναγνώρισης συναισθήματος στο πρόσωπο που απεικονίζεται στην εικόνα, ενώ οι παραπάνω διαστάσεις είναι αρκετές ώστε να μεταδοθεί η πληροφορία αλλά σε ένα μέγεθος που μειώνει τις πιθανότητες υπερπροσαρμογής του δικτύου. Τέλος, με τη χρήση ενός όχι πολύ μεγάλου μεγέθους για τα δεδομένα εισόδου αυξήθηκε η αποδοτικότητα του δικτύου.

Το δεύτερο script περιλάμβανε κώδικα για τον εντοπισμό ίδιων φωτογραφιών εντός ενός φακέλου αλλά και μεταξύ δύο διαφορετικών φακέλων. Με τη χρήση του εντοπίστηκαν και διαγράφηκαν οι ίδιες φωτογραφίες εντός του κάθε φακέλου κάθε ενός από τα δύο σετ δεδομένων αλλά και μεταξύ των φακέλων των σετ εκπαίδευσης, επαλήθευσης και δοκιμής που αφορούσαν στο ίδιο συναίσθημα. Μετά τη δημιουργία και εκτέλεση των δύο παραπάνω scripts, ο συνολικός αριθμός των διαθέσιμων δεδομένων εισόδου μειώθηκε κατά μεγάλο ποσοστό.

Ακολούθησε εξέταση των περιεχομένων του κάθε φακέλου, κατά την οποία βρέθηκαν φωτογραφίες με απεικόνιση συναισθήματος διαφορετικού από αυτό του φακέλου στον οποίο βρίσκονταν, αλλά και πολλές οι οποίες δεν απεικόνιζαν κάποιο πρόσωπο ή αυτό δεν απεικονιζόταν με αρκετή ευκρίνεια ώστε να χρησιμοποιηθεί για την εκπαίδευση του συνελικτικού νευρωνικού δικτύου. Οι παραπάνω φωτογραφίες αφαιρέθηκαν από το σετ δεδομένων.

Για την εκ νέου αύξηση του μεγέθους του σετ δεδομένων βρέθηκαν κατάλληλες εικόνες από διάφορες ελεύθερες ιστοσελίδες, μερικές από τις οποίες περιείχαν και το σώμα του φωτογραφιζόμενου. Για τη μετατροπή τους έγινε χρήση ενός ακόμη script, ο κώδικας του οποίου έκανε εντοπισμό προσώπου, αποκοπή του τετραγώνου στο οποίο αυτό περιέχεται από τη φωτογραφία και αποθήκευση ως νέας

εικόνας. Σε όλες τις παραπάνω εικόνες έγινε προσαρμογή μεγέθους σε 48x48 και μετατροπή τους σε ασπρόμαυρες με χρήση των ήδη δημιουργημένων τμημάτων κώδικα.

Το τελικό σετ δεδομένων αποτελείται από δύο φακέλους, έναν με το σετ δεδομένων εκπαίδευσης, με 30.000 αρχεία και έναν με το σετ δοκιμής, με μέγεθος 13.000 αρχείων. Κάθε φάκελος περιέχει πέντε υποφακέλους, έναν για κάθε ένα από τα πέντε βασικά συναισθήματα που αποφασίστηκε να αναγνωρίζονται εντός της εφαρμογής. Το μέγεθος όλων των υποφακέλων του κάθε σετ είναι ακριβώς το ίδιο, έτσι ώστε να αποτραπεί το λεγόμενο bias του συνελικτικού νευρωνικού δικτύου προς κάποια συγκεκριμένα συναισθήματα εις βάρος των υπόλοιπων και κατά συνέπεια η μείωση της απόδοσής του σε νέα δεδομένα. Αν για παράδειγμα κάποιος φάκελος είχε αρκετά παραπάνω αρχεία σε σχέση με τους υπόλοιπους, το δίκτυο κατά τη διάρκεια της εκπαίδευσης θα συναντούσε πιο συχνά εικόνες για τις οποίες η σωστή πρόβλεψη θα ήταν αυτή η κατηγορία. Θα διαμορφωνόταν λοιπόν η τάση να επιλέγεται συχνότερα η συγκεκριμένη κατηγορία ως πρόβλεψη, αυξάνοντας τις πιθανότητες λάθος ταξινόμησης μιας νέας φωτογραφίας.

### 5.1.3 Δημιουργία του συνελικτικού νευρωνικού δικτύου

Για τη δημιουργία του μοντέλου του συνελικτικού νευρωνικού δικτύου χρησιμοποιήθηκε η βιβλιοθήκη Tensorflow 2.0 στο Jupyter Notebook. Έπειτα από τη δοκιμή διαφορετικών δομών για το δίκτυο και της αντίστοιχης απόδοσής τους, λήφθηκε η απόφαση για το ποια θα ήταν η αρχιτεκτονική του τελικού μοντέλου. Ο κώδικας που χρησιμοποιήθηκε για τη δημιουργία του παρουσιάζεται παρακάτω.

Αρχιτεκτονική μοντέλου

```
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(48,48,1)))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Dropout(0.3))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(64))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Dense(5))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Επεξήγηση κώδικα

Ακολουθεί η επεξήγηση των διαφορετικών στοιχείων του παραπάνω κώδικα:

- `model = Sequential()`

Δηλώνεται η δημιουργία ενός αρχικά άδειου μοντέλου της τάξης Sequential του Tensorflow και γίνεται ανάθεσή του στην μεταβλητή model. Κάθε ένα από τα επόμενα στρώματα προστίθεται διαδοχικά στο μοντέλο του συνελικτικού νευρωνικού δικτύου μέσω της μεθόδου .add();

- `model.add(Conv2D(64, (3, 3), input_shape=(48,48,1)))`

Πρόκειται για το συνελικτικό στρώμα (convolutional layer) του δικτύου, που δημιουργείται μέσω της τάξης Conv2D της βιβλιοθήκης Tensorflow. Ως πρώτο όρισμα (64) δίνεται ο αριθμός των φίλτρων του συγκεκριμένου συνελικτικού στρώματος, ενώ μέσα στις παρενθέσεις δίνονται οι διαστάσεις τους, συγκεκριμένα 3x3 pixels. Στο παραπάνω συνελικτικό στρώμα, που δέχεται κατευθείαν την είσοδο του δικτύου, δηλώνονται και οι διαστάσεις της εικόνας εισόδου, οι οποίες είναι 48x48. Αυτές πολλαπλασιάζονται επί 1 και όχι επί 3, καθώς οι φωτογραφίες του σετ δεδομένων είναι ασπρόμαυρες και όχι έγχρωμες. Στα επόμενα συνελικτικά στρώματα οι διαστάσεις της εισόδου προκύπτουν από τις εξόδους των προηγούμενων στρωμάτων, οπότε υπολογίζονται αυτόματα.

- `model.add(Activation('relu'))`

Είναι η συνάρτηση ενεργοποίησης Διορθωμένης Γραμμικής Μονάδας(Rectified Linear Unit ή ReLU). Εφαρμόζεται σε κάθε κόμβο κάθε δυσδιάστατου πίνακα εξόδου που προκύπτει από την εφαρμογή των φίλτρων σε κάθε εικόνα εισόδου, δίνοντάς του ως νέα τιμή τον μέγιστο αριθμό μεταξύ της τιμής του κόμβου και του αριθμού μηδέν.

- model.add(BatchNormalization())

Η τάξη αυτή του Tensorflow χρησιμοποιείται για την ομαλοποίηση των δεδομένων, μέσω της μετατροπής των τιμών τους σε τιμές που βρίσκονται εντός ενός ορισμένου εύρους. Κάποιοι από τους κόμβους στους οποίους έχει εφαρμοστεί προηγουμένως η συνάρτηση ενεργοποίησης reLU έχουν πιθανώς μεγάλες τιμές, καθώς η συνάρτηση αυτή, όπως αναφέρθηκε στο αμέσως προηγούμενο τμήμα της επεξήγησης του κώδικα, δίνει ως έξοδο την μεγαλύτερη τιμή μεταξύ της εισόδου της και του αριθμού μηδέν. Ωστόσο, το εύρος τιμών εντός του νευρωνικού δικτύου προτιμάται να είναι μικρό, καθώς αυτό εξασφαλίζει την σταθερότητα του δικτύου και μειώνει τον χρόνο εκμάθησης των σωστών παραμέτρων στα πλαίσια της εκπαίδευσής του. Για αυτόν τον σκοπό χρησιμοποιείται η συγκεκριμένη τάξη, η οποία διατηρεί την μέση τιμή εξόδου κοντά στο 0, με μια τυπική απόκλιση κοντά στο 1.

- model.add(Dropout(0.3))

Με την εφαρμογή του dropout σε κάποιο στρώμα του νευρωνικού δικτύου κατά τη διάρκεια της εκπαίδευσης, γίνεται τυχαία αφαίρεση κάποιων κόμβων του, καθώς και των βαρών που τους συνδέουν με τους κόμβους του προηγούμενου και του επόμενου στρώματος. Με τη χρήση αυτής της τεχνικής μειώνεται η πιθανότητα υπερπροσαρμογής του δικτύου στα δεδομένα εκπαίδευσης. Η συχνότητα με την οποία εφαρμόζεται το dropout σε κάθε διαπέραση του σετ εκπαίδευσης ορίζεται από την παράμετρο που δίνεται, εν προκειμένω 0.3.

- model.add(MaxPooling2D(pool_size=(2, 2)))

Πρόκειται για το στρώμα υποδειγματοληψίας (pooling layer) του συνελικτικού νευρωνικού δικτύου. Δημιουργείται μέσω της τάξης MaxPooling2D της βιβλιοθήκης Tensorflow. Μια σειρά από φίλτρα, οι διστάσεις των οποίων ορίζονται μέσω της παραμέτρου pool_size, διατρέχουν τους πίνακες που δόθηκαν ως έξοδος από τα προηγούμενα στρώματα, αποθηκεύοντας τη μέγιστη (max) τιμή pixel του κάθε τμήματος που καλύπτεται στην αντίστοιχη θέση ενός νέου πίνακα.

- model.add(Flatten())

Μετά από την εφαρμογή τεσσάρων στρωμάτων συνέλιξης επί της αρχικής εικόνας εισόδου, οι τιμές που έχουν προκύψει και βρίσκονται στα επιμέρους δυσδιάστατα διανύσματα, αποθηκεύονται σε ένα μοναδικό μονοδιάστατο ή επίπεδο (flat) διάνυσμα, από όπου και το όνομα της συγκεκριμένης τάξης, μέσω της οποίας πραγματοποιείται η παραπάνω διαδικασία. Αποτελεί το στρώμα εισόδου του ενσωματωμένου νευρωνικού δικτύου που θα πραγματοποιήσει την τελική κατηγοριοποίηση.

- model.add(Dense(128))

Πρόκειται για ένα πλήρως συνδεδεμένο στρώμα του ενσωματωμένου νευρωνικού δικτύου, με όρισμα τον αριθμό των κόμβων που το αποτελούν, στην προκειμένη περίπτωση 128. Κάθε στρώμα της τάξης Dense εντός του δικτύου συνδέεται μέσω βαρών με κάθε κόμβο του προηγούμενου στρώματος.

- model.add(Activation('softmax'))

Είναι η συνάρτηση ενεργοποίησης softmax, η οποία περιγράφηκε στο αντίστοιχο κεφάλαιο. Εφαρμόζεται στο στρώμα εξόδου του δικτύου, το οποίο αποτελείται από πέντε κόμβους, όσα και τα πιθανά συναισθήματα εντός της εικόνας που δίνεται ως είσοδος. Η μέγιστη τιμή του διανύσματος εξόδου μετά την εφαρμογή της συνάρτησης αυτής αποτελεί την πρόβλεψη του δικτύου για το συναίσθημα που απεικονίζεται στη συγκεκριμένη είσοδο.

- model.compile(loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy'])

Αφού ορίστηκαν όλα τα επιμέρους τμήματα του συνελικτικού νευρωνικού δικτύου, καλείται η μέθοδος compile για τη δημιουργία του, με τρία ορίσματα:

α) loss='categorical_crossentropy'

Με το πρώτο αυτό όρισμα, ορίζεται ως συνάρτηση κόστους η κατηγορηματική διασταυρούμενη εντροπία (Categorical Cross Entropy).

β) optimizer='adam'

Με αυτή την γραμμή κώδικα επιλέγεται για το συνελικτικό νευρωνικό δίκτυο ένας από τους έτοιμους optimizers του Tensorflow. Ο optimizer της τάξης Adam χρησιμοποιεί τον ομώνυμο αλγόριθμο για τη βελτίωση της ταχύτητας και της απόδοσης του δικτύου κατά τη διάρκεια της εκπαίδευσής του.

γ) metrics=['accuracy']

Με την παραπάνω παράμετρο ορίζεται ότι θα μετράται η ακρίβεια των προβλέψεων του συνελικτικού νευρωνικού δικτύου στο τέλος κάθε διαπέρασης του σετ εκπαίδευσης. Πρόκειται για τη συχνότητα συμφωνίας της πρόβλεψης του δικτύου για την εκάστοτε εικόνα εισόδου με την σωστή κατηγορία στην οποία αυτή ανήκει, η οποία είναι γνωστή εκ των προτέρων σε περιπτώσεις επιβλεπόμενης μάθησης.

### 5.1.4 Εκπαίδευση

Αρχικά, σχηματίστηκε ένα πολυδιάστατο διάνυσμα με όλα τα δεδομένα του σετ εκπαίδευσης. Σε κάθε σειρά του βρίσκονταν οι τιμές των pixels για την αντίστοιχη εικόνα του σετ εκπαίδευσης, ενώ στην τελευταία στήλη της σειράς βρισκόταν η σωστή κατηγορία της συγκεκριμένης εικόνας, αντιπροσωπευόμενη από έναν αριθμό από 1 έως 5, με κάθε αριθμό να αντιστοιχίζεται με ένα συγκεκριμένο συναίσθημα.

Στη συνέχεια, με τη χρήση μιας μεθόδου της γλώσσας Python με το όνομα .shuffle() και όρισμα το προηγούμενο διάνυσμα, ανακατεύτηκαν τα δεδομένα του διανύσματος, τα οποία προηγουμένως ήταν ταξινομημένα ανά κατηγορία. Οι φωτογραφίες δηλαδή που αφορούσαν σε ένα συγκεκριμένο συναίσθημα βρίσκονταν όλες μαζί, η μια κάτω από την άλλη μέχρι να εξαντληθούν και να αρχίσουν αυτές του επόμενου συναισθήματος. Αυτό είναι ένα απαραίτητο βήμα για τη σωστή εκπαίδευση του συνελικτικού νευρωνικού δικτύου, καθώς κατά τη διάρκειά της θα πρέπει αυτό να συναντά συνεχώς παραδείγματα από διαφορετικές κατηγορίες.

Το διάνυσμα με τις ανακατεμένες τιμές χωρίστηκε σε δύο επιμέρους διανύσματα. Το πρώτο, ένα διάνυσμα τριών διαστάσεων με όνομα xTrain, περιείχε σε κάθε σειρά του ένα δισδιάστατο διάνυσμα 48x48 με τις τιμές των pixels κάθε φωτογραφίας ενώ το δεύτερο, ένα μονοδιάστατο διάνυσμα με όνομα yTrain, περιείχε την σωστή κατηγορία για τη φωτογραφία με τον αντίστοιχο δείκτη στο πρώτο διάνυσμα.

Η ίδια διαδικασία ακολουθήθηκε και για το σετ δοκιμής, αφού πρώτα χωρίστηκε στη μέση ώστε να δημιουργηθεί και το σετ επαλήθευσης για χρήση του κατά την εκπαίδευση του δικτύου. Έτσι, προέκυψαν τα διανύσματα xTest, yTest και xVal, yVal.

Για την αύξηση της ταχύτητας εκπαίδευσης του συνελικτικού νευρωνικού δικτύου χρησιμοποιήθηκε μια ακόμα τάξη του Tensorflow, η EarlyStopping, μέσω του παρακάτω κώδικα:

- earlyStopping = EarlyStopping(monitor='val_accuracy', patience=4, verbose=1, mode='max')

Η τάξη αυτή δίνει τη δυνατότητα πρόωρης παύσης της εκπαίδευσης του δικτύου. Ως πρώτο όρισμα δίνεται η παράμετρος η οποία θα αξιολογείται μετά από κάθε διαπέραση του σετ. Στην

συγκεκριμένη περίπτωση ορίστηκε να παρακολουθείται η ακρίβεια των προβλέψεων στο σετ επαλήθευσης (validation set). Με την παράμετρο patience ορίζεται ο αριθμός των διαπεράσεων για τις οποίες θα περιμένει η τάξη EarlyStopping να αυξηθεί η απόδοση στο σετ επαλήθευσης. Αν δεν αυξηθεί κατά τη διάρκεια αυτών των διαπεράσεων, η εκπαίδευση σταματάει. Αν αυξηθεί, αποθηκεύεται το μοντέλο με την αυξημένη απόδοση και η διαδικασία συνεχίζεται για τις επόμενες τέσσερις διαπεράσεις. Η τρίτη παράμετρος αφορά στο αν θα εμφανίζεται μήνυμα όταν ενεργοποιείται το early stopping και η τέταρτη αφορά στο ότι παρακολουθείται αν υπάρχει αύξηση (max) στην παράμετρο που έχει οριστεί μέσω της μεταβλητής monitor.

Ακόμη, χρησιμοποιήθηκε η τάξη ModelCheckPoint, για την αποθήκευση του μοντέλου με την μεγαλύτερη μέχρι εκείνη τη στιγμή ακρίβεια στο σετ επαλήθευσης. Ο κώδικας με τον οποίο πραγματοποιήθηκε είναι ο παρακάτω, με πρώτη παράμετρο το όνομα του αρχείου με επέκταση .h5. Πρόκειται για αρχεία που αποθηκεύονται με χρήση του Hierarchical Data Format (HDF) και περιέχουν πολυδιάστατα διανύσματα, στη συγκεκριμένη περίπτωση τα διανύσματα με τις τιμές των βαρών του εκπαιδευμένου συνελικτικού νευρωνικού δικτύου:

```
modelCheckPointSave = ModelCheckpoint("cnnModel.h5",
                                      monitor='val_accuracy',
                                      verbose=1,
                                      save_best_only=True,  mode='max')
```

Έχοντας ρυθμίσει όλες τις απαραίτητες μεταβλητές, κλήθηκε η μέθοδος .fit() για την έναρξη της διαδικασίας εκπαίδευσης του συνελικτικού νευρωνικού δικτύου:

```
model.fit(xTrain, yTrain,
          batch_size=64 ,
          epochs=20,
          verbose=1,
         callbacks=[earlyStopping, modelCheckPointSave],
         validation_data=(xVal, yVal), shuffle=True)
```

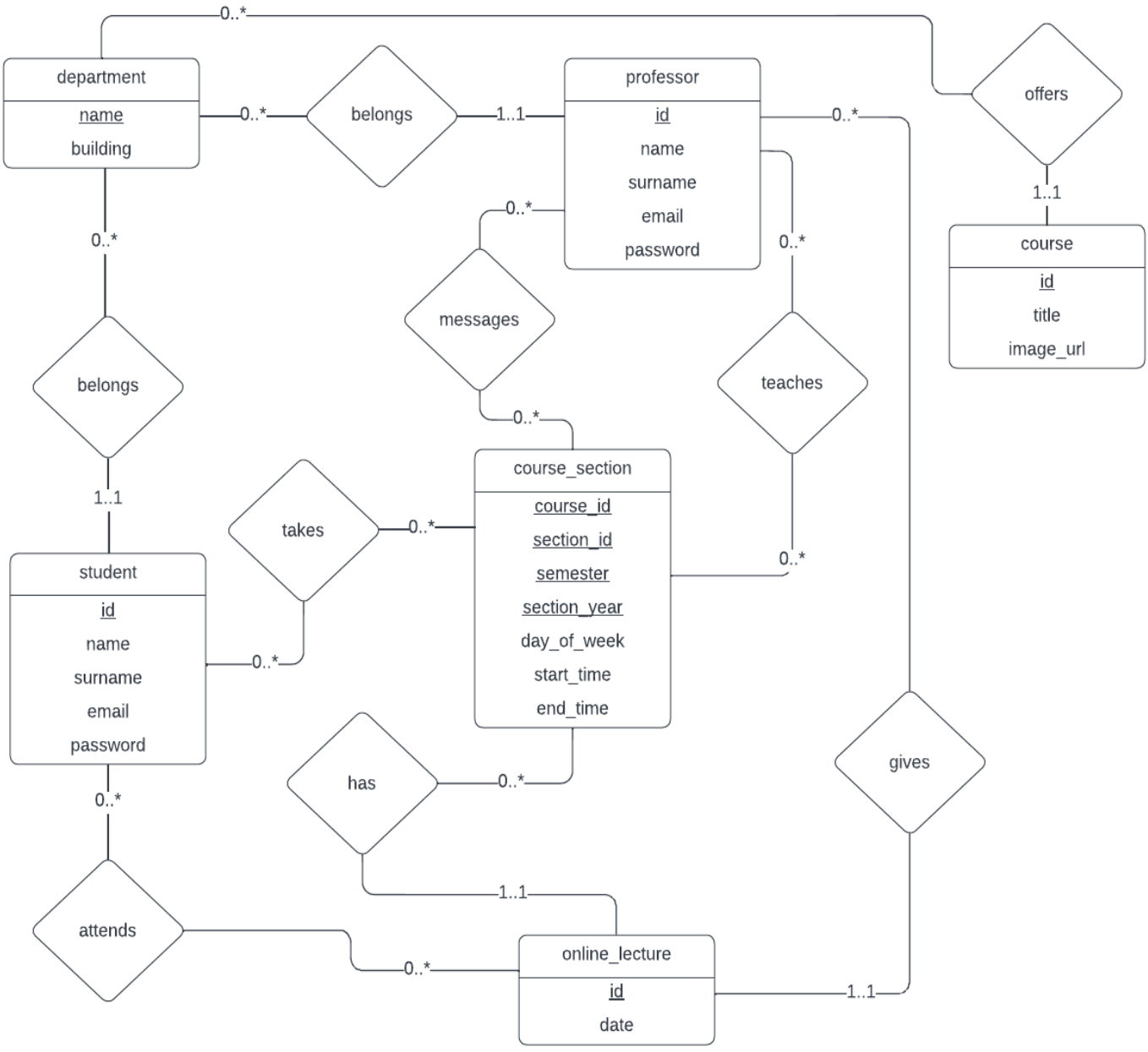
Αρχικά δόθηκαν ως παράμετροι τα διανύσματα xTrain και yTrain, τα περιεχόμενα των οποίων περιγράφηκαν παραπάνω. Με την παράμετρο batch_size ορίζεται ο αριθμός των εικόνων μετά από τις οποίες θα πραγματοποιηθεί ρύθμιση των βαρών του δικτύου, ενώ μέσω της παραμέτρου epochs δίνεται ο αριθμός των διαπεράσεων που θα γίνουν στο σετ εκπαίδευσης, αν δεν υπάρξει πρόωρος τερματισμός μέσω του early stopping. Εντός του διανύσματος callbacks τοποθετήθηκαν τα αντικείμενα των τάξεων EarlyStopping και ModelCheckPoint. Τέλος, δόθηκαν τα διανύσματα του σετ επαλήθευσης και η παράμετρος shuffle πήρε τιμή true. Μέσω της τελευταίας ρύθμισης ορίζεται ότι το σετ εκπαίδευσης και επαλήθευσης ανακατεύεται ξανά μετά από κάθε διαπέραση.

Η εκπαίδευση σταμάτησε μετά την όγδοη διαπέραση μέσω του early stopping, με ακρίβεια 0.7526 στο σετ επαλήθευσης. Στη συνέχεια, η απόδοση του συνελικτικού νευρωνικού δικτύου αξιολογήθηκε στο σετ δοκιμής, έτσι ώστε να διαπιστωθεί κατά πόσο μπορεί να γενικεύσει σωστά τις προβλέψεις του σε δεδομένα που δεν έχει ξανασυναντήσει. Η απόδοση ήταν σχεδόν ίδια με αυτή στο σετ επαλήθευσης, με τιμή 0.7498.

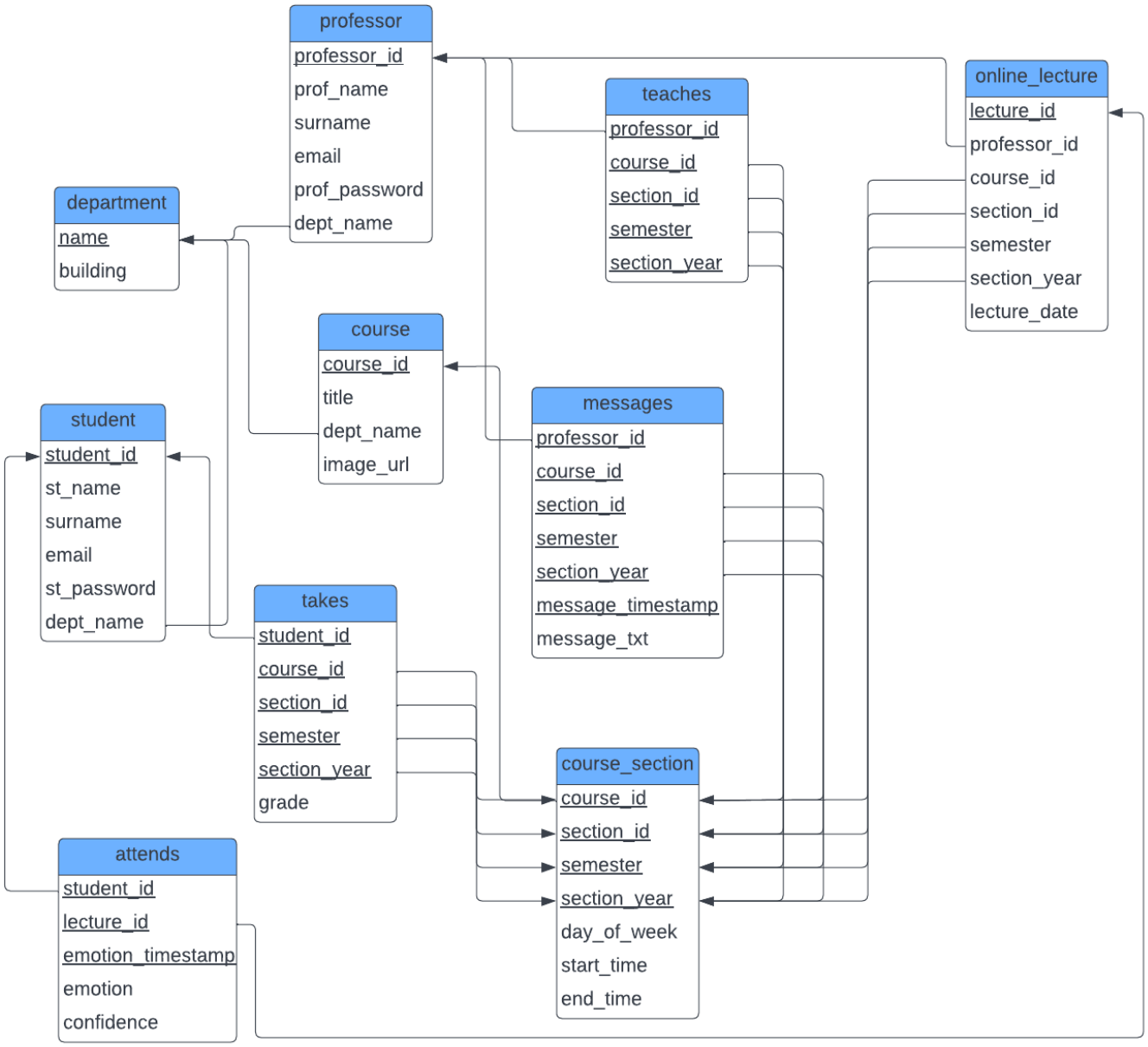
## 5.2 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Η βάση δεδομένων για τις ανάγκες της παρούσας διπλωματικής εργασίας δημιουργήθηκε με χρήση του open-source συστήματος σχεσιακών βάσεων δεδομένων PostgreSQL. Κατά τη διαδικασία του σχεδιασμού της εξασφαλίστηκε ότι θα συμπεριληφθούν στους πίνακες της βάσης όλες οι απαραίτητες πληροφορίες για μια εφαρμογή με δυνατότητα αποστολής μηνυμάτων από τον καθηγητή προς τα διδασκόμενα τμήματα και πραγματοποίησης διαδικτυακών διαλέξεων μέσω κινητής συσκευής. Ακολουθούν το διάγραμμα ER καθώς και το σχεσιακό σχήμα με βάση τα οποία δημιουργήθηκε η βάση δεδομένων εντός της πλατφόρμας διαχείρισης και ανάπτυξης pgAdmin:

### 5.2.1 Διάγραμμα ER



### 5.2.2 Σχεσιακό σχήμα



## 5.3 RESTFULL WEB SERVICE

### 5.3.1 Dependencies

Για τη δημιουργία της δομής του Spring Boot Web Service χρησιμοποιήθηκε το εργαλείο Spring Initializr. Πρόκειται για μια διεπαφή χρήστη που επιτρέπει την εύκολη και γρήγορη παραμετροποίηση της δομής της Spring Boot εφαρμογής. Εκτός από τα προεπιλεγμένα dependencies, προστέθηκαν δύο ακόμη:

α) *PostgreSQL Driver*

Ένας JDBC (Java Database Connectivity) driver που επιτρέπει σε ένα πρόγραμμα Java να συνδεθεί με μια βάση δεδομένων PostgreSQL και να αλληλεπιδράσει με αυτή χρησιμοποιώντας ανεξάρτητο από τη βάση δεδομένων κώδικα Java.

β) *Spring Security*

Πρόκειται για ένα εύκολα παραμετροποιήσιμο framework του Spring Boot για αυθεντικοποίηση, έλεγχο πρόσβασης και προστασία ενάντια σε επιθέσεις.

### 5.3.2 Πρωτόκολλο HTTPS

Για την ασφάλεια των δεδομένων που θα ανταλλάσσονται από τις κινητές συσκευές που λειτουργούν ως clients προς το web service και αντίστροφα, αποφασίστηκε να χρησιμοποιηθεί το πρωτόκολλο HTTPS (Hypertext Transfer Protocol Secure) εντός της εφαρμογής. Καθώς το Spring Boot χρησιμοποιεί ως προεπιλογή το πρωτόκολλο HTTP, χρειάστηκαν κάποιες ενέργειες και παραμετροποιήσεις για να επιτευχθεί ο παραπάνω σκοπός.

Αρχικά δημιουργήθηκε μέσω του command line ένα ιδιωτικό πιστοποιητικό SSL (Secure Sockets Layer) για χρήση στην εφαρμογή. Φυσικά η ενδεδειγμένη λύση, αν επρόκειτο για εφαρμογή που θα λειτουργούσε κανονικά σε κάποιον server στο διαδίκτυο με πραγματικούς clients, θα ήταν η απόκτηση ενός πιστοποιητικού από μια αρχή έκδοσης πιστοποιητικών (Certification Authority ή CA). Τα πιστοποιητικά που πωλούνται από αυτούς τους παρόχους χρησιμοποιούν μια πιο πρόσφατη και ασφαλή έκδοση του SSL, γνωστή ως TLS (Transport Layer Security). Ωστόσο, στα πλαίσια της διπλωματικής εργασίας, όπου η εφαρμογή του web service εκτελέστηκε στον localhost, κρίθηκε επαρκής η δημιουργία ενός αυτοϋπογεγραμμένου (self-signed) πιστοποιητικού.

Το πιστοποιητικό δημιουργήθηκε με χρήση του φορμάτ PKCS12 (Public Key Cryptographic Standards 12), που επιτρέπει την αποθήκευση πολλών κλειδιών εντός ενός αρχείου και της εντολής keytool, που παρέχεται μέσω του Java Runtime Environment. Ακολουθεί η εντολή που χρησιμοποιήθηκε, με όλες τις παραμέτρους και την επεξήγησή τους. Οι αγκύλες δηλώνουν ενδεικτικές τιμές, που χρησιμοποιούνται απλώς για την παρουσίαση της εντολής:

```
keytool -genkeypair -alias [springboot] -keyalg RSA -keysize 4096 -storetype PKCS12 -keystore
[springboot].p12 -validity [365] -storepass password –ext SAN=IP:[192.168.1.7]
```

- -genkeypair: δημιουργεί ένα ζεύγος κλειδιών (ένα δημόσιο κλειδί και ένα αντίστοιχο ιδιωτικό κλειδί). Το δημόσιο κλειδί εντάσσεται σε ένα self-signed πιστοποιητικό με χρήση του standard X.509 v3. Το πιστοποιητικό αυτό και το ιδιωτικό κλειδί αποθηκεύονται σε έναν αποθηκευτικό χώρο, γνωστό ως keystore.
- -alias springboot: το alias όνομα του παραπάνω ζεύγους κλειδιών, στη συγκεκριμένη περίπτωση το ενδεικτικό όνομα “springboot”.
- -keyalg RSA: δηλώνει τον αλγόριθμο που χρησιμοποιείται για τη δημιουργία του ζεύγους κλειδιών. Επιλέχθηκε ο αλγόριθμος κρυπτογράφησης RSA.

- -keysize 4096: δηλώνεται το μέγεθος σε bit του κάθε κλειδιού που θα δημιουργηθεί.
- -storetype PKCS12: επιλογή του φορμάτ PKCS12, που αναφέρθηκε παραπάνω.
- -keystore springboot.p12: το όνομα του keystore.
- -validity 365: ορισμός των ημερών ισχύος του πιστοποιητικού.
- -storepass password: ορισμός του κωδικού για το keystore. Χρήση του ενδεικτικού ονόματος “password”.
- -ext SAN=IP:192.168.1.7: ορισμός της IP του web service ως Εναλλακτικό Όνομα Θέματος (Subject Alternative Name ή SAN). Με αυτόν τον τρόπο η διεύθυνση IP συσχετίζεται με το πιστοποιητικό SSL και δηλώνεται ότι προστατεύεται από αυτό.

Το πιστοποιητικό που δημιουργήθηκε μέσω της παραπάνω εντολής αποθηκεύτηκε στον φάκελο των resources της Spring Boot εφαρμογής.

Το επόμενο βήμα της διαδικασίας ήταν η ενημέρωση του αρχείου application.properties του Spring Boot με τις παρακάτω τιμές. Μέσω αυτών δηλώνεται αρχικά το port 8443, που είναι αυτό μέσω του οποίου πραγματοποιείται η επικοινωνία με το TomCat Servlet της εφαρμογής με χρήση του πρωτοκόλλου HTTPS, καθώς και ότι ενεργοποιείται η επικοινωνία με χρήση SSL. Ακολουθεί η τοποθεσία του keystore με το πιστοποιητικό και τέλος οι βασικές παράμετροι που ορίστηκαν για το πιστοποιητικό SSL μέσω του command που παρουσιάστηκε παραπάνω:

```
server.port=8443
server.ssl.enabled=true
server.ssl.key-store=src/main/resources/springboot.p12
server.ssl.key-store-password=[password]
server.ssl.key-store-type=pkcs12
server.ssl.key-alias=[springboot]
server.ssl.key-password=[password]
```

Κατά τη χρήση ενός αυτοϋπογεγραμμένο πιστοποιητικού SSL, οι clients, για λόγους ασφαλείας, δεν εμπιστεύονται την Spring Boot εφαρμογή η οποία το διαθέτει. Για αυτόν τον λόγο, το SSL διαμοιράστηκε και στους clients που αντάλλαξαν δεδομένα με το web service κατά τη δοκιμή της λειτουργίας της εφαρμογής σε περιβάλλον Android.

### 5.3.3 Δομή κώδικα εφαρμογής

Μέσω του κώδικα που γράφτηκε για το Spring Boot web service υλοποιήθηκε η σύνδεση μεταξύ των συσκευών Android που λειτουργούσαν ως clients και της βάσης δεδομένων που δημιουργήθηκε μέσω της PostgreSQL.

Για κάθε πίνακα της βάσης δεδομένων, δημιουργήθηκε ένα ξεχωριστό ομώνυμο πακέτο με τρία ή τέσσερα .java αρχεία εντός της Spring Boot εφαρμογής, αναλόγως του εάν ο αντίστοιχος πίνακας διέθετε σύνθετο κλειδί ή όχι. Στην πρώτη περίπτωση δημιουργήθηκε και αρχείο με μια ξεχωριστή τάξη που περιέχει τα πεδία που συμμετέχουν στο κλειδί του αντίστοιχου πίνακα, όπως θα δούμε και στη συνέχεια. Τα αρχεία που βρίσκονταν απαραίτητα εντός όλων των πακέτων είναι τα ακόλουθα:

α) αυτό της τάξης, με τη δήλωση των ιδιοτήτων, κάποιες από τις οποίες εκφράζουν τα πεδία του πίνακα, ενώ κάποιες άλλες την σχέση του πίνακα που αντιστοιχίζεται στην συγκεκριμένη τάξη με άλλες τάξεις-πίνακες, μέσω των κατάλληλων annotations. Εκτός από τις ιδιότητες περιέχονται constructors, getters και setters.

β) ένα αρχείο για το interface, με όνομα αυτό της τάξης με προσθήκη της κατάληξης Repository, που χειρίζεται την αλληλεπίδραση, μέσω CRUD operations, με τον αντίστοιχο πίνακα της βάσης δεδομένων PostgreSQL.

γ) αρχείο με το όνομα της τάξης και την κατάληξη Service, που διαθέτει μεθόδους οι οποίες καλούν αυτές του αντίστοιχου αρχείου με κατάληξη Repository, δίνοντας και λαμβάνοντας πληροφορίες από την βάση δεδομένων.

Εκτός των παραπάνω πακέτων, δημιουργήθηκε και ένα για το αρχείο του Controller, στο οποίο ορίστηκαν οι μέθοδοι που θα διαχειριστούν την αλληλεπίδραση με τους clients, με τα ανάλογα URLs και παραμέτρους.

Τέλος, δημιουργήθηκε ένα ακόμη πακέτο, με όνομα joinedTableClasses, στο οποίο αποθηκεύτηκαν τα αρχεία κάποιων τάξεων που διαθέτουν ιδιότητες από παραπάνω από μια τάξεις-πίνακες, αντιπροσωπεύοντας αποτελέσματα αναζήτησης στη βάση δεδομένων σε παραπάνω από έναν πίνακα.

### 5.3.4 Ανάλυση τμημάτων κώδικα

Στην ενότητα αυτή θα παρουσιαστούν τρία αντιπροσωπευτικά τμήματα κώδικα για την υλοποίηση ισάριθμων διαφορετικών διαδικασιών εντός της Spring Boot εφαρμογής.

Αποστολή συναισθήματος φοιτητή

Στο παρακάτω τμήμα παρουσιάζεται ο απαραίτητος κώδικας για τον χειρισμό της διαδρομής ενός POST request από κάποιον Android client με το συναίσθημα του φοιτητή-χρήστη που αναγνωρίστηκε από το συνελικτικό νευρωνικό δίκτυο σε μια συγκεκριμένη χρονική στιγμή, από την στιγμή που φτάνει στο URL της αντίστοιχης μεθόδου του controller μέχρι και την αποθήκευσή του στη βάση δεδομένων μέσω της μεθόδου της τάξης Repository:

*Attends.java*

```
@Entity
@IdClass(AttendsId.class)
@Table
public class Attends
{
    @Id
    @Column(name = "student_id")
    private String studentId;

    @Id
    @Column(name = "lecture_id")
    private Integer lectureId;

    @Id
    @Column(name = "emotion_timestamp")
    private LocalDateTime emotionTimestamp;

    @Column(name = "emotion", nullable = false)
    private String emotion;

    @Column(name = "confidence", nullable = true)
    private float confidence;


    @ManyToOne
    @JoinColumn(name = "lecture_id", insertable = false, updatable = false)
    private OnlineLecture attendedOnlineLecture;
```

```
    //constructors
    public Attends(){}


    public Attends(String studentId, Integer lectureId, LocalDateTime
emotionTimestamp, String emotion, float confidence)
    {
        this.studentId = studentId;
        this.lectureId = lectureId;
        this.emotionTimestamp = emotionTimestamp;
        this.emotion = emotion;
        this.confidence = confidence;
    }


    //getters
    public String getStudentId()
    {
        return studentId;
    }

    public Integer getLectureId()
    {
        return lectureId;
    }

    public LocalDateTime getEmotionTimestamp()
    {
        return emotionTimestamp;
    }

    public String getEmotion()
    {
        return emotion;
    }

    public float getConfidence() {
        return confidence;
    }


    //setters
    public void setStudentId(String studentId)
    {
        this.studentId = studentId;
    }

    public void setLectureId(Integer lectureId)
    {
        this.lectureId = lectureId;
    }

    public void setEmotionTimestamp(LocalDateTime emotionTimestamp)
    {
        this.emotionTimestamp = emotionTimestamp;
    }

    public void setEmotion(String emotion)
    {
        this.emotion = emotion;
    }
```

```
    public void setConfidence(float confidence)
    {
        this.confidence = confidence;
    }
}
```

Μέσω του κώδικα που βρίσκεται στο συγκεκριμένο αρχείο, ορίζεται η τάξη που αντιστοιχίζεται στον ομώνυμο πίνακα της βάσης δεδομένων. Μέσω του πρώτου annotation δηλώνεται στο Hibernate, το framework του Spring Boot που είναι υπεύθυνο για τις αντιστοιχίσεις μεταξύ τάξεων και πινάκων, ότι η τάξη αυτή είναι οντότητα και συνδέεται με έναν πίνακα της βάσης δεδομένων. Μέσω του δεύτερου annotation δηλώνεται το όνομα της τάξης που περιέχει τις πληροφορίες για το σύνθετο κλειδί της συγκεκριμένης τάξης, ενώ μέσω του τρίτου η εφαρμογή εντοπίζει τον πίνακα που έχει όνομα ίδιο με αυτό της τάξης, έτσι ώστε να πραγματοποιήσει τη σύνδεση μεταξύ τους.

Ακολουθούν οι ιδιότητες της τάξης, οι πρώτες πέντε από τις οποίες εκφράζουν τα αντίστοιχα πεδία του πίνακα. Όσες από αυτές συμμετέχουν στο κλειδί του πίνακα, έχουν το annotation @Id, ενώ με το annotation @Column δηλώνεται ότι αντιστοιχίζονται σε στήλες του πίνακα, με την τιμή της παραμέτρου name να ορίζει το ακριβές όνομα της αντίστοιχης στήλης στην PostgreSQL βάση δεδομένων. Τέλος, με την παράμετρο nullable ορίζεται για τα πεδία που δεν συμμετέχουν στο κλειδί του πίνακα, το αν θα μπορούν να έχουν τιμές null ή όχι.

Η έκτη ιδιότητα είναι ένα αντικείμενο της τάξης OnlineLecture, μέσω του οποίου ορίζεται η συσχέτιση με την τάξη αυτή, που συνδέεται με τον ομώνυμο πίνακα και βρίσκεται σε διαφορετικό αρχείο της εφαρμογής. Με το πρώτο annotation δηλώνεται το είδος της συσχέτισης, στην συγκεκριμένη περίπτωση πολλά (από την πλευρά του Attends) προς ένα (από την πλευρά του OnlineLecture), ενώ με το δεύτερο δηλώνεται το πεδίο της τάξης-πίνακα που λειτουργεί ως ξένο κλειδί και αναφέρεται στο πρωτεύον κλειδί της τάξης-πίνακα OnlineLecture. Ακολουθούν οι constructors και οι getters-setters για τις ιδιότητες της τάξης.

*AttendsId.java*

```
class AttendsId implements Serializable
{
    private String studentId;
    private Integer lectureId;
    private LocalDateTime emotionTimestamp;

    // constructors
    public AttendsId(){}


    public AttendsId(String studentId, Integer lectureId, LocalDateTime
emotionTimestamp) {
        this.studentId = studentId;
        this.lectureId = lectureId;
        this.emotionTimestamp = emotionTimestamp;
    }


    //override 2 μεθόδων της class Object
    @Override
    public boolean equals(Object obj)
    {
        return super.equals(obj);
    }

    @Override
    public int hashCode()
```

```
    {
        return super.hashCode();
    }
}
```

Η τάξη αυτή αντιπροσωπεύει το σύνθετο κλειδί της τάξης-πίνακα Attends. Επειδή το κλειδί μιας οντότητας στο Spring Boot είναι ένας primitive τύπος με δυνατότητα σειριοποίησης, πρέπει και η τάξη που περιέχει τα πεδία που συμμετέχουν στο σύνθετο κλειδί να κάνει implement το interface Serializable.

Οι ιδιότητες της τάξης είναι οι ίδιες με τις στήλες που συμμετέχουν στο σύνθετο κλειδί του πίνακα στην PostgreSQL βάση δεδομένων. Ακολουθούν δύο constructors και δύο μέθοδοι της τάξης Object που γίνονται override, κάτι επίσης απαραίτητο για μια τάξη που αντιπροσωπεύει ένα σύνθετο κλειδί. Μέσω των δύο αυτών μεθόδων δίνεται η απαραίτητη πληροφορία στο Hibernate για τον τρόπο με τον οποίο θα συγκρίνει τις εγγραφές του πίνακα που περιέχει το σύνθετο κλειδί, έτσι ώστε να εξασφαλίζεται η μοναδικότητα της κάθε καταχώρισης εντός του πίνακα.

*AttendsRepository.java*

```
@Repository
public interface AttendsRepository extends JpaRepository<Attends, AttendsId>
{

}
```

Στο αρχείο αυτό δηλώνεται ένα interface που κληρονομεί όλες τις μεθόδους του Generic interface JpaRepository (Java Persistence API Repository). Το interface JpaRepository διαθέτει μια μεγάλη γκάμα μεθόδων για αλληλεπίδραση με τη βάση δεδομένων μέσω CRUD operations, που μπορούν να κληθούν από κάθε αντικείμενο ενός interface που κληρονομεί το συγκεκριμένο. Επίσης δίνεται η δυνατότητα ορισμού και νέων μεθόδων για πραγματοποίηση πιο πολύπλοκων αναζητήσεων στη βάση δεδομένων. Στην περίπτωση της τάξης-πίνακα Attends δεν χρειάστηκε να οριστεί κάποια νέα μέθοδος, για αυτό και δεν υπάρχουν περιεχόμενα εντός των αγκυλών του interface.

Το μόνο που δίνεται στο Generic interface JpaRepository ως παράμετρος είναι η τάξη που σχετίζεται με τον αντίστοιχο πίνακα και το πρωτεύον κλειδί της, στη συγκεκριμένη περίπτωση η τάξη που εκφράζει το σύνθετο κλειδί. Μέσω του annotation @Repository δηλώνεται ότι το interface που ακολουθεί θα είναι υπεύθυνο για την υλοποίηση του μηχανισμού αλληλεπίδρασης με τη βάση δεδομένων μέσω των διαδικασιών αποθήκευσης, ανάκτησης, ανανέωσης και διαγραφής δεδομένων.

*AttendsService.java*

```
@Service
public class AttendsService
{
    @Autowired
    private AttendsRepository attendsRepository;


    public Attends setStudentEmotion(Attends newEmotion)
    {
        return attendsRepository.save(newEmotion);
    };
}
```

Η τάξη αυτή λειτουργεί ως ενδιάμεσος μετξύ του controller και του interface που λειτουργεί ως repository. Μέσω του annotation @Service δηλώνεται ότι η συγκεκριμένη τάξη παρέχει ένα service, στη συγκεκριμένη

περίπτωση εφαρμόζει το business logic της εφαρμογής. Επίσης, μέσω του συγκεκριμένου keyword, όπως και με αυτό του @Repository της προηγούμενης τάξης, δηλώνεται ότι, με το κατάλληλο annotation, ένα object της τάξης αυτής θα μπορεί να γίνει αρχικοποιηθεί αυτόματα, χωρίς την κλήση constructor.

Το annotation που πραγματοποιεί την παραπάνω λειτουργία είναι το @Autowired, που βρίσκεται πάνω από την ιδιότητα attendsRepository, το οποίο πρόκειται για ένα αντικείμενο του interface AttendsRepository. Μέσω αυτού, το αντικείμενο αρχικοποιείται αυτόματα και μπορεί στη συνέχεια να κληθεί οποιαδήποτε μέθοδος της τάξης του.

Αυτό συμβαίνει εντός της μεθόδου setStudentEmotion: Η μέθοδος δέχεται ως παράμετρο ένα αντικείμενο της τάξης Attends και καλεί μια από τις έτοιμες μεθόδους του interface AttendsRepository, που έχει κληρονομηθεί από το JpaRepository, την μέθοδο .save. Η μέθοδος αυτή αποθηκεύει τις τιμές των ιδιοτήτων του αντικειμένου newEmotion σε μια νέα γραμμή στον πίνακα attends της PostgreSQL βάσης δεδομένων.

*Controller.java*

```
@RestController
@RequestMapping(path = "api/v1/university")
public class Controller
{
    @Autowired
    private AttendsService attendsService;
    .
    .
    .
    .
    @PostMapping("/setEmotion")
    public Attends setStudentEmotion(@Validated @RequestBody Attends
newEmotion)
    {
        return attendsService.setStudentEmotion(newEmotion);
    }
```

Με τα annotations που υπάρχουν πάνω από την εμφάνιση της τάξης του controller, δηλώνονται δύο βασικές πληροφορίες προς την Spring Boot εφαρμογή. Πρώτον, ότι η τάξη που ακολουθεί θα λειτουργεί ως controller του RESTful web service. Δεύτερον, δηλώνεται το βασικό τμήμα του URL στο οποίο ο controller θα δέχεται τα αιτήματα των clients. Κάθε μέθοδος του controller, προσθέτει ένα τμήμα στο παραπάνω URL, έτσι ώστε να διαμορφωθεί ένα μοναδικό endpoint για κάθε μία από αυτές. Ο controller δέχεται τα αιτήματα των clients και τα ανακατευθύνει στην κατάλληλη μέθοδο βάσει του URL τους.

Η private ιδιότητα της τάξης του controller είναι ένα αντικείμενο της τάξης AttendsService που αρχικοποιείται αυτόματα μέσω του annotation @Autowired, όπως περιγράφηκε και παραπάνω. Μετά από αυτήν ακολουθούν και άλλα αντικείμενα, για κάθε ένα από τα Services που αντιστοιχίζονται στην ομώνυμη τάξη-πίνακα, στη θέση των οποίων έχουν μπει οι τελείες, καθώς δεν θα τα εξετάσουμε στο συγκεκριμένο παράδειγμα κώδικα.

Στη συνέχεια ακολουθεί, πρώτη μεταξύ πολλών άλλων, η μέθοδος του controller που λαμβάνει τα δεδομένα από τον client. Με το annotation @PostMapping δηλώνεται ότι τα δεδομένα θα έρθουν στον controller μέσω ενός αιτήματος POST, με σκοπό δηλαδή να αποθηκευτούν στη βάση, ενώ το string εντός της παρένθεσης πρόκειται για το κομμάτι του URL που προστίθεται στο βασικό και διαμορφώνει το endpoint της συγκεκριμένης μεθόδου.

Το annotation @Validated εξετάζει τα δεδομένα που φτάνουν στην εφαρμογή μέσω του POST request, τα οποία στέλνονται εντός ενός JSON string. Το JSON string περιέχει μια σειρά από ζεύγη key-value, κάθε ένα από τα οποία έχει ως κλειδί το όνομα της αντίστοιχης ιδιότητας της τάξης Attends και ως value μια τιμή για αυτή την ιδιότητα. Με το παραπάνω annotation, ο controller εξασφαλίζει ότι τα δεδομένα που βρίσκονται εντός του JSON είναι έγκυρα, έχουν δηλαδή σωστές τιμές ως προς τον τύπο δεδομένων, έτσι όπως αυτός έχει οριστεί για την αντίστοιχη ιδιότητα και ότι ικανοποιούνται τυχόν προϋποθέσεις που ορίστηκαν για κάποιες ιδιότητες στο αρχείο της τάξης.

Το annotation @RequestBody δηλώνει στον controller ότι θα βρει το JSON string με τις ιδιότητες του αντικειμένου στο σώμα (body) του request. Αν το JSON string βρεθεί και είναι έγκυρο, η μέθοδος κατασκευάζει αυτόματα, μέσω των τιμών εντός του JSON, ένα αντικείμενο της τάξης Attends, το οποίο αποτελεί παράμετρο της μεθόδου.

Στη συνέχεια καλείται από αυτήν η κατάλληλη μέθοδος του attendsService, στην οποία περνιέται ως όρισμα το παραπάνω αντικείμενο.

Κρυπτογράφηση κωδικού πρόσβασης χρήστη

Για την προστασία των κωδικών πρόσβασης που θα επιλέγονταν από τους χρήστες και θα αποθηκεύονταν στη βάση δεδομένων, γράφτηκε ο ακόλουθος κώδικας εντός δύο αρχείων για την κρυπτογράφησή τους στην Spring Boot εφαρμογή, πριν από την πρώτη αποστολή τους στη βάση δεδομένων:

*PasswordHasher.java*

```
public interface PasswordHasher
{
    public String hashPassword(String passwordToBeHashed);

    public boolean passwordMatchesHashedPassword
            (String passwordToBeCheckedForMatch, String hashedPasswordFromDb);
}
```

Αρχικά δημιουργήθηκε το παραπάνω interface, που περιλαμβάνει δύο μεθόδους οι οποίες θα πρέπει να γίνουν implement από οποιαδήποτε τάξη κάνει implement το interface. Η πρώτη μέθοδος δέχεται τον δημιουργημένο από τον χρήστη κωδικό σε μορφή String και τον επιστρέφει αφού θα έχει κρυπτογραφηθεί με την χρήση ενός αλγόριθμου κατακερματισμού.

Η δεύτερη κάνει τη σύγκριση του κωδικού που της δίνεται από τον χρήστη κατά τη διαδικασία της εισόδου στην client εφαρμογή με τον αποθηκευμένο στη βάση δεδομένων κωδικό για τον ίδιο χρήστη. Τα δύο Strings αποτελούν τις παραμέτρους της μεθόδου.

*Argon2PasswordHasher.java*

```
@Service
public class Argon2PasswordHasher implements PasswordHasher
{
    private int saltLength = 16;
    private int hashLength = 32;
    private int parallelism = 1;
    private int memory = 4096;
    private int iterations = 3;
    Argon2PasswordEncoder argon2PasswordEncoder =
            new Argon2PasswordEncoder(saltLength, hashLength, parallelism,
memory, iterations);

    public String hashPassword(String passwordToBeHashed)
    {
        return argon2PasswordEncoder.encode(passwordToBeHashed);
    }

    public boolean passwordMatchesHashedPassword(String
passwordToBeCheckedForMatch, String hashedPasswordFromDb)
    {
```

```
        return argon2PasswordEncoder.matches(passwordToBeCheckedForMatch,
hashedPasswordFromDb);
    }
}
```

Η παραπάνω τάξη με το annotation @Service, η λειτουργία του οποίου περιγράφηκε παραπάνω, κάνει implement το interface PasswordHasher. Οι πρώτες πέντε ιδιότητές της αποτελούν παραμέτρους που δίνονται στην έκτη. Πρόκειται κατά σειρά για τις ακόλουθες:
α) το μέγεθος του salt σε bytes. Το salt πρόκειται για μια τυχαία ακολουθία αλφαριθμητικών που προστίθεται στον κωδικό που θα δοθεί ως είσοδος στον αλγόριθμο κρυπτογράφησης. Χρησιμοποιείται για την εξασφάλιση της μοναδικότητας του κρυπτογραφημένου κωδικού που θα δημιουργηθεί κάθε φορά, ακόμα και με τη ίδια ακριβώς είσοδο, και την αύξηση της προστασίας του έναντι μιας σειράς επιθέσεων που επιχειρούν να τον αποσπάσουν.
β) το μέγεθος του hash σε bytes. Το hash πρόκειται για τον κρυπτογραφημένο κωδικό που δημιουργείται μετά την προσθήκη του salt στον κωδικό που παρέχεται από τον χρήστη κατά την εγγραφή του και τη κρυπτογράφηση του αλφαριθμητικού που προκύπτει από την παραπάνω διαδικασία με έναν αλγόριθμο κατακερματισμού.

γ) ο αριθμός των threads που θα χρησιμοποιηθούν από τον αλγόριθμο κατακερματισμού.

δ)το κόστος της εφαρμογής του αλγόριθμου κατακερματισμού για τη μνήμη σε kibibytes(1 kibibyte = 1024 bytes).

ε) το πλήθος των φορών εφαρμογής του αλγόριθμου κατακερματισμού.

Η διαδικασία της κρυπτογράφησης ενός plaintext κωδικού με τη χρήση ενός αλγόριθμου κατακερματισμού είναι μονής κατεύθυνσης, καθώς δεν είναι δυνατόν να αναδημιουργηθεί ο αρχικός κωδικός με γνώση μόνο του κατακερματισμένου. Υπάρχουν ωστόσο μέθοδοι μέσω των οποίων ένας επιτιθέμενος προσπαθεί είτε να αποκτήσει τον κατακερματισμένο κωδικό μέσω SQL injection και στη συνέχεια να προσπαθήσει να βρει τον αρχικό μέσω για παράδειγμα έτοιμων πινάκων αντιστοίχισης, μέθοδος από την οποία προστατεύει τα δεδομένα της βάσης η χρήση του salt, είτε να παρακάμψει την παραπάνω διαδικασία, δοκιμάζοντας, μέσω της λεγόμενης brute-force attack, τη συνεχή αποστολή διάφορων κωδικών προς την εφαρμογή, με την προσδοκία ότι κάποιος από αυτούς θα χρησιμοποιείται από κάποιον χρήστη και έτσι θα καταφέρει να τον αποσπάσει.

Από τα παραπάνω προκύπτει ότι ένας αλγόριθμος κατακερματισμού θα πρέπει να είναι αρκετά αργός ώστε να αποτρέπει διάφορες επιθέσεις τέτοιου τύπου, χωρίς ταυτόχρονα να είναι τόσο αργός ώστε να καθυστερεί σημαντικά μια απλή είσοδο ενός χρήστη στην εφαρμογή. Κάποιες παράμετροι από τις παραπάνω, όπως για παράδειγμα η παράμετρος iterations, αυξάνουν τον χρόνο που θα χρειαστεί για την ολοκλήρωση της διαδικασίας εκτέλεσης του αλγόριθμου κατακερματισμού, ενώ μέσω της τιμής της παραμέτρου memory μπορεί να αυξηθεί το κόστος αυτής για τη μνήμη, καθυστερώντας σημαντικά τυχόν επιθέσεις.

Η έκτη ιδιότητα της τάξης πρόκειται για ένα αντικείμενο της τάξης Argon2PasswordEncoder που δημιουργείται περνώντας στον constructor της τις τιμές των παραπάνω πέντε ιδιοτήτων ως ορίσματα. Η τάξη αυτή χρησιμοποιεί τον αλγόριθμο κατακερματισμού Argon2, που πήρε την πρώτη θέση στον Διαγωνισμό Κατακερματισμού Κωδικού το 2015.

Ακολουθούν οι δύο μέθοδοι της τάξης, που κάνουν implement τις αντίστοιχες μεθόδους που ορίστηκαν στο interface. Στο εσωτερικό τους, περνούν τις παραμέτρους που τους δίνονται σε δύο μεθόδους του αντικειμένου της τάξης Argon2PasswordEncoder, στην .encode και στην .matches, η πρώτη από τις οποίες κάνει την κρυπτογράφηση μέσω του αλγόριθμου κατακερματισμού Argon2 του κωδικού που της δίνεται και η δεύτερη ελέγχει αν ο plaintext κωδικός που αποτελεί την πρώτη παράμετρο της μεθόδου ταιριάζει με αυτόν που της δίνεται ως δεύτερο, ο οποίος είναι ο κρυπτογραφημένος κωδικός που έχει αποθηκευτεί στη βάση δεδομένων, μέσω της ακόλουθης διαδικασίας: στο εσωτερικό του αλφαριθμητικού του αποθηκευμένου στη βάση κατακερματισμένου κωδικού υπάρχει αποθηκευμένο και το salt που χρησιμοποιήθηκε για την κρυπτογράφησή του. Το salt αυτό προστίθεται στον plaintext κωδικό και το αλφαριθμητικό που προκύπτει δίνεται ως είσοδος στον αλγόριθμο κατακερματισμού. Αν η έξοδος του παραπάνω αλγόριθμου είναι ίδια με τον αποθηκευμένο κατακερματισμένο κωδικό, τότε ο κωδικός που δόθηκε από το χρήστη είναι σωστός και η μέθοδος επιστρέφει τη boolean τιμή true. Διαφορετικά επιστρέφεται false, σηματοδοτώντας το ότι δόθηκε λάθος κωδικός πρόσβασης.

Οι δύο αυτές μέθοδοι της τάξης Argon2PasswordEncoder χρησιμοποιούνται τόσο κατά τη διαδικασία της αρχικής εγγραφής ενός φοιτητή ή καθηγητή στην εφαρμογή όσο και κατά τη διαδικασία της μετέπειτα εισόδου τους σε αυτή, με τα αλφαριθμητικά των κωδικών πρόσβασης να φτάνουν από τον εκάστοτε client στον controller της Spring Boot εφαρμογής.

Δημιουργία λίστας επερχόμενων διαλέξεων φοιτητή

Στο παρακάτω τμήμα παρουσιάζεται ο κώδικας που γράφτηκε για την ανάκτηση από τη βάση δεδομένων της επόμενης προγραμματισμένης διάλεξης για κάθε μάθημα που έχει επιλέξει ο φοιτητής και τη δημιουργία μιας λίστας με τις απαραίτητες πληροφορίες για κάθε μια από αυτές, η οποία αποστέλλεται στην client συσκευή του εκάστοτε φοιτητή-χρήστη.

*Takes.java*

```
@Entity
@IdClass(TakesId.class)
@Table
public class Takes
{
    @Id
    @Column(name = "student_id")
    private String studentId;

    @Id
    @Column(name = "course_id")
    private String courseId;

    @Id
    @Column(name = "section_id")
    private String sectionId;

    @Id
    @Column(name = "semester")
    private int semester;

    @Id
    @Column(name = "section_year")
    private String sectionYear;


    @OneToOne
    @JoinColumn(name = "student_id", insertable = false, updatable = false)
    private Student taughtStudent;


    @OneToOne
    @JoinColumns({
            @JoinColumn(name = "course_id", insertable = false, updatable =
false),
            @JoinColumn(name = "section_id", insertable = false, updatable =
false),
            @JoinColumn(name = "section_year", insertable = false, updatable =
false),
            @JoinColumn(name = "semester", insertable = false, updatable =
false)
    })
    private CourseSection takenCourseSection;


    //constructors
    public Takes(){};


    public Takes(String studentId, String courseId, String sectionId, int
semester, String sectionYear) {
        this.studentId = studentId;
```

```
        this.courseId = courseId;
        this.sectionId = sectionId;
        this.semester = semester;
        this.sectionYear = sectionYear;
    }

    //getters
    public String getStudentId() {
        return studentId;
    }

    public String getCourseId() {
        return courseId;
    }

    public String getSectionId() {
        return sectionId;
    }

    public int getSemester() {
        return semester;
    }

    public String getSectionYear() {
        return sectionYear;
    }

    public CourseSection getTakenCourseSection() {
        return takenCourseSection;
    }



    //setters
    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }

    public void setCourseId(String courseId) {
        this.courseId = courseId;
    }

    public void setSectionId(String sectionId) {
        this.sectionId = sectionId;
    }

    public void setSemester(int semester) {
        this.semester = semester;
    }

    public void setSectionYear(String sectionYear) {
        this.sectionYear = sectionYear;
    }
}
```

Στο παραπάνω αρχείο περιέχεται ο κώδικας της τάξης Takes, η οποία λειτουργεί ως οντότητα και συνδέεται με τον ομώνυμο πίνακα της βάσης δεδομένων, ο οποίος έχει σύνθετο πρωτεύον κλειδί. Τα παραπάνω καθορίζονται μέσω των annotations που αναλύθηκαν κατά την παρουσίαση του πρώτου τμήματος κώδικα της ενότητας, που αφορά στην αποστολή του συναισθήματος των φοιτητών στη βάση δεδομένων.

Οι πρώτες πέντε ιδιότητες συμμετέχουν στο σύνθετο κλειδί και αντιπροσωπεύουν τις στήλες του αντίστοιχου πίνακα στη βάση δεδομένων, όπως δηλώνεται μέσω των annotations, ενώ με την τιμή που δίνεται στο name ορίζεται το ακριβές όνομα της αντίστοιχης στήλης στη βάση δεδομένων PostgreSQL. (Η στήλη grade του πίνακα Takes της βάσης δεδομένων, στην οποία αποθηκεύεται ο βαθμός που θα δοθεί στον φοιτητή για το συγκεκριμένο μάθημα, έχει ως default τιμή το null, καθώς θα συμπληρωθεί στο τέλος του εξαμήνου, οπότε δεν εντάχτηκε στις ιδιότητες της ομώνυμης τάξης στο Spring Boot.

Η έκτη και έβδομη ιδιότητα είναι αντικείμενα της τάξης Student και CourseSection αντίστοιχα, εκφράζοντας τη σχέση της τάξης Takes με αυτές τις τάξεις-πίνακες. Ο πίνακας takes προέκυψε στη βάση δεδομένων από τη συσχέτιση πολλών προς πολλά μεταξύ των πινάκων student και course_section, οπότε μέσω αυτών των αντικειμένων εκφράζεται συμβατικά μια συσχέτιση ενός προς ένα της τάξης-πίνακα Takes με κάθε μια από τις δύο τάξεις-πίνακες. Ως παράμετρος του annotation @JoinColumn δίνεται το όνομα της στήλης του πίνακα takes που λειτουργεί ως ξένο κλειδί. Η σύνδεση με τον πίνακα student γίνεται μέσω ενός μοναδικού κλειδιού, ενώ αυτή με τον πίνακα course_section μέσω ενός σύνθετου κλειδιού που περιλαμβάνει τέσσερις στήλες.

Ακολουθούν δύο constructors, καθώς και οι getters και setters για τις ιδιότητες της τάξης.

*TakesId.java*

```
public class TakesId implements Serializable
{
    private String studentId;
    private String courseId;
    private String sectionId;
    private int semester;
    private String sectionYear;


    //constructors
    public TakesId(){}


    public TakesId(String studentId, String courseId, String sectionId, int
semester, String sectionYear)
    {
        this.studentId = studentId;
        this.courseId = courseId;
        this.sectionId = sectionId;
        this.semester = semester;
        this.sectionYear = sectionYear;
    }


    //override 2 μεθόδων της class Object
    @Override
    public boolean equals(Object obj) {
        return super.equals(obj);
    }

    @Override
    public int hashCode() {
        return super.hashCode();
    }
}
```

Στο παραπάνω αρχείο ορίζεται η τάξη που αντιπροσωπεύει το σύνθετο κλειδί της τάξης Takes. Όπως επεξηγήθηκε και κατά την αναφορά στην αντίστοιχη τάξη στο πρώτο τμήμα κώδικα της ενότητας, η τάξη αυτή είναι απαραίτητο να κάνει implement το interface Serializable και να κάνει override δύο από τις μεθόδους της τάξης Object, από την οποία κληρονομούν εμμέσως όλες οι τάξεις στη γλώσσα Java.

Επίσης ορίζονται οι ιδιότητες, με τύπο δεδομένων και ονόματα ίδια με αυτές της τάξης Takes, που αντιστοιχίζονται στις στήλες που συμμετέχουν στο σύνθετο κλειδί του ομώνυμου πίνακα, καθώς και δύο constructors.

*TakesRepository.java*

```
@Repository
public interface TakesRepository extends JpaRepository<Takes, TakesId>
{
    List<Takes> findByStudentId(String studentId);
}
```

Στο αρχείο αυτό δηλώνεται το interface που σχετίζεται με την τάξη Takes και κληρονομεί όλες τις μεθόδους του Generic interface JpaRepository. Δίνονται και εδώ ως παράμετροι το όνομα της τάξης και το πρωτεύον κλειδί της, στη συγκεκριμένη περίπτωση η τάξη που εκφράζει το σύνθετο πρωτεύον κλειδί.

Σε αυτό το repository, σε αντίθεση με αυτό της τάξης Attends που παρουσιάστηκε σε προηγούμενο τμήμα της ενότητας, χρειάστηκε να οριστεί μια νέα μέθοδος για αλληλεπίδραση με τη βάση δεδομένων. Το Jpa Repository δίνει τη δυνατότητα ορισμού μεθόδων για ανάκτηση δεδομένων από τη βάση με χρήση της σύνταξης findBy[όνομα της στήλης]. Το όνομα της στήλης πρέπει να είναι το όνομα της ιδιότητας της τάξης-παραμέτρου του JpaRepository που αντιστοιχίζεται με την ανάλογη στήλη του ομώνυμου πίνακα της βάσης δεδομένων. Αν το όνομα που θα δοθεί είναι έγκυρο, όταν κληθεί η συγκεκριμένη μέθοδος, το Spring κάνει αυτόματα αναζήτηση στη συγκεκριμένη στήλη στον πίνακα της βάσης δεδομένων, ψάχνοντας βάσει της τιμής που δόθηκε ως παράμετρος στη μέθοδο και γυρνώντας τα αποτελέσματα σε μια λίστα με αντικείμενα της ανάλογης τάξης.

*TakesService.java*

```
@Service
public class TakesService
{
    @Autowired
    private TakesRepository takesRepository;

    public List<Takes> findByStudentId(String studentId)
    {
        int currentYear = LocalDate.now().getYear();
        int currentMonth = LocalDate.now().getMonthValue();


        List<Takes> activeTakenCourses =
takesRepository.findByStudentId(studentId)
                .stream()
                .filter(tc ->
((Integer.valueOf(tc.getSectionYear().substring(0, 4)) == currentYear)
                        && (currentMonth >= 10 && currentMonth <= 2)
                        && (tc.getSemester() % 2 != 0))
                        || (Integer.parseInt(tc.getSectionYear().substring(0,
4)) + 1 == currentYear)
                        && (currentMonth >= 3 && currentMonth <= 9)
                        && tc.getSemester() % 2 == 0)
                .collect(Collectors.toList());

        return activeTakenCourses;
    };
}
```

Στην παραπάνω τάξη, που λειτουργεί ως Service, ορίζεται αρχικά ως ιδιότητα ένα @Autowired αντικείμενο της TakesRepository. Στη συνέχεια καλείται μια μέθοδος για την εύρεση των μαθημάτων που διδάσκεται ο φοιτητής-χρήστης της client συσκευής στο τρέχον ακαδημαϊκό έτος.

Στην αρχή της μεθόδου, αρχικοποιούνται δύο μεταβλητές ακέραιων αριθμών, η πρώτη με το τρέχον έτος και η δεύτερη με τον τρέχοντα μήνα ως αριθμό από 1 έως 12. Και για τις δύο χρησιμοποιείται η στατική μέθοδος .now() της τάξης LocalDate, που επιστρέφει την τοπική ημερομηνία του συστήματος του χρήστη.

Στη συνέχεια καλείται, μέσω του αντικειμένου takesRepository, η μέθοδος του ομώνυμου Repository, η οποία επιστρέφει όλα τα μαθήματα στα οποία έχει εγγραφεί ο συγκεκριμένος φοιτητής από την αρχή των σπουδών του. Για να επιστραφούν μόνο αυτά του τρέχοντος ακαδημαϊκού έτους και του σωστού εξαμήνου βάσει του μήνα που διανύουμε, η λίστα μετατρέπεται σε stream και εφαρμόζεται σε αυτό ένα filter, το οποίο θα φιλτράρει τα δεδομένα μέσω της συνάρτησης lambda που ορίζεται στο εσωτερικό των παρενθέσεων:

Στη βάση δεδομένων, το εξάμηνο του μαθήματος αποθηκεύεται στη στήλη semester του πίνακα course_section ως ακέραιος αριθμός μεταξύ 1 και 8 και η ακαδημαϊκή χρονιά ως αλφαριθμητικό με την ακόλουθη μορφή: ‘2020/21’. Εντός της συνάρτησης lambda, μέσω του getter της τάξης Takes λαμβάνεται για κάθε αντικείμενο αυτής εντός της λίστας το αλφαριθμητικό αυτό, από το οποίο αποκόπτονται οι πρώτοι τέσσερεις χαρακτήρες μέσω της μεθόδου .substring() της τάξης String και στη συνέχεια μετατρέπεται σε ακέραιο αριθμό μέσω της στατικής μεθόδου valueOf() της τάξης Integer.

Το έτος αυτό συγκρίνεται με το τρέχον έτος, που είναι αποθηκευμένο στη μεταβλητή currentYear. Αν υπάρξει ισότητα, ελέγχεται αν ο τρέχων μήνας είναι μεγαλύτερος από τον Σεπτέμβριο, αφού από τον Οκτώβριο ξεκινά η διδασκαλία των μαθημάτων. Αν και η δεύτερη συνθήκη είναι αληθής, σημαίνει ότι βρισκόμαστε εντός του χειμερινού εξαμήνου. Τέλος, ελέγχεται αν και το συγκεκριμένο μάθημα της λίστας διδάσκεται στο χειμερινό εξάμηνο, μέσω του ελέγχου του υπολοίπου της διαίρεσης του semester με τον αριθμό 2. Αν η διαίρεση έχει υπόλοιπο, σημαίνει ότι ο αριθμός του semester είναι μονός, άρα πρόκειται για χειμερινό εξάμηνο. Αν και οι τρεις συνθήκες είναι αληθείς, το συγκεκριμένο αντικείμενο της τάξης Takes πληροί τις προϋποθέσεις του φίλτρου και θα εξεταστεί το επόμενο κατά σειρά. Διαφορετικά πραγματοποιείται ένας ακόμη έλεγχος, του αν ο τρέχων μήνας ανήκει στο δεύτερο μισό της ακαδημαϊκής χρονιάς και το μάθημα ανήκει σε εαρινό εξάμηνο, προσθέτοντας ένα έτος στο αποκομμένο αλφαριθμητικό που αντιπροσωπεύει το πρώτο μισό της ακαδημαϊκής χρονιάς, αφού πρώτα αυτό μετατραπεί σε ακέραιο, και ελέγχοντας αν η διαίρεση του εξαμήνου με τον αριθμό 2 δεν έχει υπόλοιπο. Το stream, μετά την εφαρμογή του φίλτρου, ξαναμετατρέπεται σε λίστα που περιέχει μόνο όσα αντικείμενα πληρούν τις προϋποθέσεις που ορίστηκαν εντός του, μέσω της μεθόδου .collect() του stream, με όρισμα την στατική μέθοδο της τάξης Collectors με όνομα .toList(). Η τελική λίστα επιστρέφεται από τη μέθοδο μέσω του keyword return.

*Controller.java*

```
@RestController
@RequestMapping(path = "api/v1/university")
public class Controller
{
    .
    .
    @Autowired
    private TakesService takesService;
    .
    .
    .
    .
    .


@GetMapping("/getOnlineLecturesByStudentId/{id}")
public List<OnlineLectureWithCourseSection>
getOnlineLecturesByStudentId(@PathVariable("id") String id)
```

```
{
    List<Takes> activeSelectedCourses = takesService.findByStudentId(id);

    List<OnlineLectureWithCourseSection> upcomingOnlineLectures = new
ArrayList<>();

    for(Takes activeSelectedCourse : activeSelectedCourses)
    {

if(activeSelectedCourse.getTakenCourseSection().getOnlineLectures().size() >
0)
        {
            OnlineLectureWithCourseSection lectureWithCourseSection = new
OnlineLectureWithCourseSection();

            Optional<OnlineLecture> onlineLectureOptional =
activeSelectedCourse.getTakenCourseSection().getOnlineLectures()
                    .stream()
                    .filter(
                            (ol) ->
ol.getLectureDate().isAfter(LocalDate.now())
                            || (

ol.getLectureDate().isEqual(LocalDate.now())
                            &&
activeSelectedCourse.getTakenCourseSection().getEndTime().isAfter(LocalTime.no
w())
                            )
                    )
                    .sorted()
                    .findFirst();


            if(onlineLectureOptional.isPresent())
            {

lectureWithCourseSection.setLectureId(onlineLectureOptional.get().getLectureId
());

lectureWithCourseSection.setLectureDate(onlineLectureOptional.get().getLecture
Date());

lectureWithCourseSection.setStartTime(activeSelectedCourse.getTakenCourseSecti
on().getStartTime());

lectureWithCourseSection.setEndTime(activeSelectedCourse.getTakenCourseSection
().getEndTime());

lectureWithCourseSection.setImageUrl(activeSelectedCourse.getTakenCourseSectio
n().getCourse().getImageUrl());

lectureWithCourseSection.setTitle(activeSelectedCourse.getTakenCourseSection()
.getCourse().getTitle());
                upcomingOnlineLectures.add(lectureWithCourseSection);
            }
        }
    }
    Collections.sort(upcomingOnlineLectures);

    return upcomingOnlineLectures;
}
```

Στο αρχείο του controller της Spring Boot εφαρμογής ορίζεται, μεταξύ άλλων, ως ιδιότητα της τάξης ένα αντικείμενο της τάξης TakesService, με το annotation @Autowired. Στη συνέχεια ακολουθούν διάφορες μέθοδοι, εκ των οποίων παρουσιάζεται αυτή που επιλέγει τις επερχόμενες διαλέξεις που θα παρακολουθήσει ο εκάστοτε φοιτητής.

Μέσω του annotation @GetMapping, δηλώνεται ότι η μέθοδος αυτή θα διαχειρίζεται αιτήματα των clients για αποστολή σε αυτούς πληροφορίας από τη βάση δεδομένων. Με το βασικό τμήμα του URL του web service να έχει δηλωθεί πάνω από τον ορισμό της τάξης Controller, το string εντός της παρένθεσης δηλώνει το κομμάτι του URL που προστίθεται στο βασικό και διαμορφώνει το endpoint της συγκεκριμένης μεθόδου. Η λέξη εντός των αγκυλών μετά την κάθετο πρόκειται για μια παράμετρο εντός του URL, που στη συγκεκριμένη περίπτωση αφορά στο id του φοιτητή. Η συγκεκριμένη σύνταξη δηλώνει ότι στο τελευταίο τμήμα του URL στο οποίο θα κάνει αίτημα μια client συσκευή θα τοποθετείται το id του φοιτητή-χρήστη της συγκεκριμένης συσκευής. Η μέθοδος getOnlineLecturesByStudentId έχει ως παράμετρο ένα String με όνομα ίδιο με αυτό της παραμέτρου στο τέλος του URL. Μέσω του annotation @PathVariable και την εμφάνιση της λέξης id εντός της παρένθεσής του δηλώνεται ότι η παράμετρος αυτή θα αρχικοποιηθεί με την τιμή που θα βρεθεί στο URL εντός αγκυλών, στην θέση της λέξης id.

Η παράμετρος id δίνεται ως όρισμα στη μέθοδο .findByStudentId() του αντικειμένου takesService, η οποία επιστρέφει μια λίστα με τα μαθήματα που παρακολουθεί ο φοιτητής- χρήστης της client συσκευής που έκανε το αίτημα. Το τελευταίο βήμα είναι το να βρεθεί η επερχόμενη διάλεξη για κάθε ένα από αυτά τα μαθήματα.

Για την πραγματοποίηση αυτής της διαδικασίας, χρησιμοποιείται μια από τις τάξεις του πακέτου joinedTableClasses που αναφέρθηκε παραπάνω, με όνομα OnlineLectureWithCourseSection. Η τάξη αυτή διαθέτει ιδιότητες από τρεις διαφορετικές τάξεις, συγκεκριμένα το lectureId και lectureDate από την τάξη OnlineLecture, το startTime και endTime από τη τάξη CourseSection και το title από την τάξη Course. Δημιουργείται αρχικά μια κενή λίστα με αντικείμενα της παραπάνω τάξης.

Στη συνέχεια εξετάζονται με τη σειρά τα αντικείμενα της λίστας με τα μαθήματα που παρακολουθεί ο φοιτητής-χρήστης. Όπως αναφέρθηκε και παραπάνω, η ιδιότητα της τάξης Takes με όνομα takenCourseSection είναι ένα αντικείμενο της τάξης CourseSection. Η τάξη αυτή έχει με τη σειρά της ως ιδιότητα μια λίστα με αντικείμενα της τάξης OnlineLecture, καθώς η σύνδεση μεταξύ αυτών των τάξεων-πινάκων είναι ένα προς πολλά. Εξετάζεται λοιπόν, για κάθε μάθημα, το αν έχουν προγραμματιστεί διαδικτυακές διαλέξεις, μέσω της εξέτασης του μήκους της παραπάνω λίστας.

Αν ναι, δημιουργείται ένα αντικείμενο της τάξης OnlineLectureWithCourseSection. Η λίστα των διαδικτυακών διαλέξεων για το μάθημα αυτό μετατρέπεται σε stream και ελέγχεται μέσω φίλτρου το αν η διάλεξη έχει ημερομηνία μεταγενέστερη της ισχύουσας, ή, αν έχει την ίδια, αν η ώρα λήξης του μαθήματος έπεται της τωρινής, στην οποία περίπτωση θα πρόκειται για διάλεξη σε εξέλιξη. Όσες διαλέξεις ικανοποιούν τη συνθήκη του φίλτρου ταξινομούνται μέσω της μεθόδου .sorted() σε αύξουσα ημερομηνιακή σειρά, κάτι το οποίο έχει καθοριστεί στο αρχείο της τάξης OnlineLecture: η τάξη αυτή κάνει implement το interface Comparable και κατά συνέπεια override τη μέθοδο .compareTo(), μέσω της οποίας καθορίζεται το είδος ταξινόμησης που θα εφαρμόζεται σε αντικείμενα της συγκεκριμένης τάξης. Από τις διαλέξεις που ταξινομήθηκαν σε αύξουσα σειρά επιλέγεται η πρώτη, που πρόκειται για την πρώτη επερχόμενη ή σε εξέλιξη διάλεξη. Αυτή αποθηκεύεται εντός ενός αντικειμένου της Generic τάξης Optional που λειτουργεί ως container ενός αντικειμένου που δεν είναι βέβαιο το αν θα υπάρχει ή θα έχει τιμή null. Η παρουσία ή όχι του αντικειμένου της τάξης OnlineLecture ελέγχεται μέσω της μεθόδου .isPresent(). Αν υπάρχει το αντικείμενο, γίνεται το mapping, με χρήση getters και setters, των τιμών των απαραίτητων ιδιοτήτων στις αντίστοιχες ιδιότητες του αντικειμένου της τάξης OnlineLectureWithCourseSection και αυτό προστίθεται στη λίστα των επερχόμενων διαλέξεων που θα επιστραφεί στον χρήστη της client συσκευής.

Η διαδικασία συνεχίζεται και για τα υπόλοιπα μαθήματα που παρακολουθεί ο χρήστης. Αφού προστεθούν στη λίστα όλες οι επερχόμενες διαλέξεις, ταξινομούνται σε αύξουσα σειρά βάσει ημερομηνίας και ώρας, με τη χρήση της static μεθόδου .sort() της τάξης Collections, έτσι ώστε η πλησιέστερη χρονικά διάλεξη να βρίσκεται στην κορυφή της λίστας. Τέλος, η επεξεργασμένη λίστα επιστρέφεται μέσω του controller στην client συσκευή.

## 5.4 ANDROID CLIENT

Ο κώδικας για τις Android συσκευές που θα λειτουργήσουν ως clients του web service γράφτηκε με χρήση της γλώσσας Java εντός του προγραμματιστικού περιβάλλοντος Android Studio. Στο τμήμα της διπλωματικής εργασίας που ακολουθεί, θα παρουσιαστούν οι διαφορετικές οθόνες της εφαρμογής καθώς και η λειτουργικότητα που διαμορφώθηκε μέσω κώδικα για την αλληλεπίδραση του χρήστη με κάθε μία από αυτές.

### 5.4.1 Γραφικό περιβάλλον – Λειτουργικότητα

*1. Redirect activity*

Η πρώτη οθόνη (ή activity στα πλαίσια μιας Android εφαρμογής) δεν είναι ορατή στον χρήστη, καθώς έχει τεθεί ως ημιδιαφανής μέσω της επιλογής του theme Translucent.NoTitleBar στο αρχείο AndroidManifest.xml. Στο αρχείο της αντίστοιχης τάξης με όνομα Redirect, υπάρχει κώδικας μέσω του οποίου γίνεται, μεταξύ άλλων, η επιλογή της οθόνης που θα εμφανιστεί στη συσκευή του χρήστη κατά το άνοιγμα της εφαρμογής.

Αρχικά, με χρήση ενός αντικειμένου της τάξης SharedPreferences, αποθηκεύεται τοπικά σε ένα xml αρχείο το βασικό τμήμα του URL του web service που υλοποιήθηκε μέσω Spring Boot, έτσι ώστε να είναι προσβάσιμο από τον κώδικα όλων των activities της εφαρμογής. Οι πληροφορίες αποθηκεύονται στο αρχείο αυτό με τη μορφή ζεύγους κλειδιού – τιμής.

Στη συνέχεια ελέγχεται αν έχει γίνει επιλογή γλώσσας και χρώματος για την εφαρμογή μέσω ελέγχου του αν υπάρχουν συγκεκριμένα κλειδιά στο xml αρχείο των shared preferences. Αν δεν υπάρχουν τα παραπάνω κλειδιά, δημιουργούνται δύο νέα ζεύγη κλειδιού-τιμής, μέσω των οποίων τίθεται ως χρώμα το μπλε και ως γλώσσα τα αγγλικά.

Τέλος, μέσω του ελέγχου της παρουσίας ενός ακόμη κλειδιού αποφασίζεται ποια θα είναι η οθόνη που θα απεικονιστεί στον χρήστη. Όπως θα παρουσιαστεί και κατά την ανάλυση του κώδικα για το αντίστοιχο activity, κατά την ολοκλήρωση της εγγραφής ενός χρήστη στην εφαρμογή δημιουργείται ένα ζεύγος κλειδιού-τιμής που έχει αποθηκευμένη την ιδιότητά του, αν πρόκειται δηλαδή για φοιτητή ή για καθηγητή. Εκτός αυτού, στο activity όπου πραγματοποιείται η είσοδος στην εφαρμογή μέσω email και κωδικού πρόσβασης, ένας ήδη εγγεγραμμένος χρήστης μπορεί να αποφασίσει το αν η εφαρμογή θα «θυμάται» τα στοιχεία του για κάποιο χρονικό διάστημα μετά από την είσοδό του. Αν ο χρήστης επιλέξει έστω και μια φορά να παραμείνει συνδεδεμένος στην εφαρμογή, δημιουργείται ένα ζεύγος κλειδιού-τιμής με την ισχύουσα ημερομηνία και ώρα.

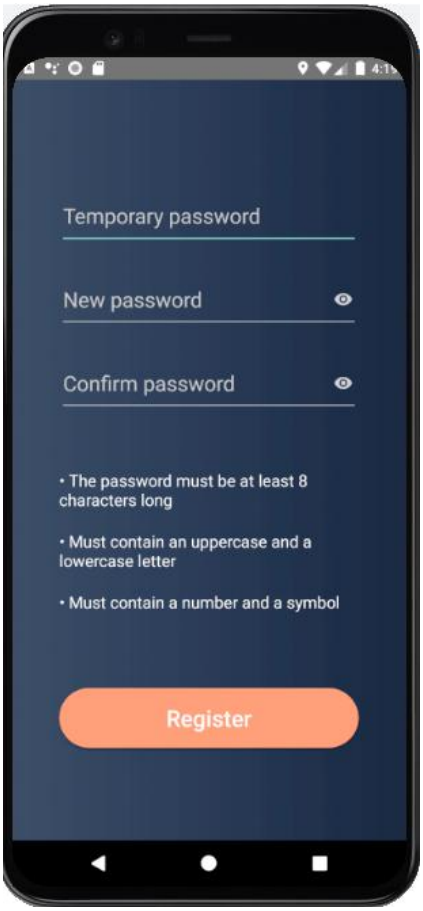
Μέσω του κώδικα της τάξης Redirect εξετάζεται το αν υπάρχει το πρώτο από τα δύο κλειδιά που αναφέρθηκαν παραπάνω, κάτι που θα σημαίνει ότι ο χρήστης έχει ολοκληρώσει την εγγραφή του και η ιδιότητά του έχει καθοριστεί. Αν το παραπάνω δεν ισχύει, τότε μεταφέρεται στην οθόνη που απεικονίζεται παρακάτω, με όνομα Welcome. Εάν ισχύει, ελέγχεται αν υπάρχει το κλειδί που αφορά στην ημερομηνία και ώρα επιλογής από τον χρήστη της παραμονής του στην εφαρμογή ως συνδεδεμένος. Αν το κλειδί δεν υπάρχει ή αν υπάρχει αλλά μέσω του ελέγχου της τιμής του διαπιστωθεί ότι έχει λήξει η περίοδος κατά την οποία παραμένει αυτόματα συνδεδεμένος, τότε μεταφέρεται στην οθόνη πραγματοποίησης εισόδου στην εφαρμογή. Αντιθέτως, αν η περίοδος δεν έχει λήξει, μεταφέρεται στην οθόνη με το αρχικό μενού της εφαρμογής.

*2. Welcome activity*


Η παραπάνω πρόκειται για την πρώτη ορατή οθόνη που συναντά ο χρήστης κατά την αρχική είσοδό του στην εφαρμογή. Η τάξη που περιλαμβάνει τον κώδικα για τη συγκεκριμένη οθόνη έχει όνομα Welcome και κληρονομεί από την τάξη Activity, όπως όλες οι τάξεις που περιλαμβάνουν κώδικα για την απεικόνιση των δεδομένων και τη διαχείριση των events εντός μιας οθόνης της Android εφαρμογής.

Μέσω ενός RadioButton ο χρήστης επιλέγει την ιδιότητά του μεταξύ αυτών του φοιτητή και του καθηγητή. Κατά το πάτημα του TextView με τίτλο “NEXT”, δημιουργείται μέσω του κώδικα ένα αντικείμενο της τάξης Intent για μετάβαση στην οθόνη πραγματοποίησης της εγγραφής του χρήστη στην εφαρμογή. Στο αντικείμενο αυτό δίνεται ως επιπλέον δεδομένο η ιδιότητα που επέλεξε ο χρήστης μέσω του RadioButton και στη συνέχεια εμφανίζεται η αντίστοιχη οθόνη.

*3. Register activity*




Μέσω της παραπάνω οθόνης πραγματοποιείται η εγγραφή του χρήστη στην εφαρμογή παρακολούθησης διαλέξεων μέσω διαδικτύου.

Η διαδικασία που προσομοιώνεται είναι η εξής: κάθε καθηγητής ή φοιτητής έχει λάβει στη διεύθυνση ηλεκτρονικού ταχυδρομείου που του έχει δοθεί από το πανεπιστήμιο ένα μήνυμα από την γραμματεία με έναν προσωρινό εξαψήφιο κωδικό, ο οποίος είναι αποθηκευμένος και στη βάση δεδομένων στην στήλη του κωδικού της γραμμής που αφορά στον συγκεκριμένο χρήστη. Για την ολοκλήρωση της εγγραφής, ο χρήστης αρχικά πληκτρολογεί τον συγκεκριμένο κωδικό στο πεδίο “Temporary Password”. Στη συνέχεια πληκτρολογεί τον κωδικό που θα επιλέξει για την είσοδό του στην εφαρμογή και επιβεβαιώνει πληκτρολογώντας τον ξανά. Με πάτημα του σημείου της οθόνης με το εικονίδιο του ματιού σε κάθε ένα από τα δύο EditTexts, ο χρήστης μπορεί να δει τον κωδικό που έχει πληκτρολογήσει, ενώ με ένα δεύτερο πάτημα ο κωδικός αποκρύπτεται ξανά. Αυτό επιτυγχάνεται μέσω δημιουργίας listener για το event της επαφής του δακτύλου του χρήστη με κάποιο από τα δύο εικονίδια και ενεργοποίησης της δυναμικής αλλαγής του inputType του εκάστοτε EditText κατά την παύση της επαφής του δακτύλου με το εικονίδιο μέσω του MotionEvent *ACTION_UP*.Ο νέος κωδικός που θα επιλεχθεί από τον χρήστη θα πρέπει να πληροί τις προϋποθέσεις ασφαλείας που προσδιορίζονται στο κάτω μέρος της οθόνης.

Με το πάτημα του κουμπιού “Register” καλείται μια μέθοδος της τάξης με όνομα arePasswordsValidAndSecure(), στην οποία δίνονται ως ορίσματα οι τρεις κωδικοί που πληκτρολογήθηκαν στα ισάριθμα EditTexts της οθόνης. Εντός της μεθόδου, καλείται μια σειρά από άλλες μεθόδους που ελέγχουν διαδοχικά τα αλφαριθμητικά των κωδικών, η κάθε μία βάσει διαφορετικών κριτηρίων. Αρχικά ελέγχεται αν κάποιο από τα τρία αλφαριθμητικά είναι κενό ή περιέχει μόνο τον

χαρακτήρα κενού διαστήματος. Δευτερευόντως εξασφαλίζεται ότι ο νέος κωδικός χρήστη και ο κωδικός επιβεβαίωσης είναι ίδιοι. Στη συνέχεια ελέγχεται αν ο νέος κωδικός έχει το κατάλληλο μήκος και τέλος αν περιέχει τον τύπο των χαρακτήρων που διευκρινίζεται στο κάτω μέρος της οθόνης. Αν κάποιος από τους παραπάνω ελέγχους αποτύχει γυρνώντας την αντίστοιχη Boolean τιμή στη μέθοδο arePasswordsValidAndSecure(), εκείνη επιστρέφει με τιμή false και το ανάλογο μήνυμα μέσω Toast, χωρίς να εκτελέσει τους υπόλοιπους κατά σειρά ελέγχους. Αν συμβεί αυτό, επιστρέφει μέσω του keyword return και η μέθοδος register(), που κλήθηκε κατά το πάτημα του κουμπιού “Register” από τον χρήστη.

Αντιθέτως, αν όλοι οι έλεγχοι ολοκληρωθούν επιτυχώς, εκτελείται ο υπόλοιπος κώδικας της μεθόδου, μέσω του οποίου πραγματοποιείται η αποστολή δεδομένων στο web service. Συγκεκριμένα, καλείται μια μέθοδος για τη δημιουργία ενός αντικειμένου της τάξης JSONObject με δύο ζεύγη κλειδιού-τιμής που θα περιέχουν τον προσωρινό και τον νέο κωδικό, και μια ακόμη που προσθέτει το κατάλληλο τελευταίο τμήμα στο URL του web service, το οποίο ανακτάται από τα shared preferences, αναλόγως της ιδιότητας (καθηγητή ή φοιτητή) που έχει επιλεχθεί από τον χρήστη. Αυτό το τελευταίο τμήμα, που προστίθεται στο βασικό URL, καθορίζει ποια μέθοδος της τάξης Controller της Spring Boot εφαρμογής θα κληθεί για να διαχειριστεί τα δεδομένα που θα σταλθούν από τη συγκεκριμένη client συσκευή. Τέλος, καλείται μια μέθοδος που θα πραγματοποιήσει την αποστολή των δεδομένων στο web service. Εντός της μεθόδου, αρχικά δημιουργείται ένα αντικείμενο της τάξης JsonObjectRequest, το οποίο θα περιέχει όλα τα δεδομένα που θα σταλθούν στο web service. Συγκεκριμένα, καλείται ο constructor της τάξης με πέντε ορίσματα: Τα τρία πρώτα ορίσματα είναι ο τύπος του CRUD operation, στη συγκεκριμένη περίπτωση το POST καθώς θα γίνει αποστολή δεδομένων προς το web service, το URL και το JSONObject με τους κωδικούς. Ως τέταρτο όρισμα δίνεται ένα ανώνυμο instance του static interface Response.Listener, στη μέθοδο του οποίου με όνομα onResponse(), η οποία γίνεται override, καθορίζεται η διαχείριση της απάντησης που θα επιστραφεί ασύγχρονα στη μέθοδο από το Spring Boot.

Στο Spring Boot, μέσω της αντίστοιχης μεθόδου της τάξης Controller, γίνεται αναζήτηση στη βάση δεδομένων βάσει του προσωρινού κωδικού που δόθηκε από τον χρήστη της Android συσκευής. Αν βρεθεί σειρά στον πίνακα της ιδιότητας χρήστη που επιλέχθηκε (φοιτητής ή καθηγητής) η οποία περιέχει τον συγκεκριμένο προσωρινό κωδικό, γίνεται update της τιμής του κωδικού με τον νέο, αφού αυτός κρυπτογραφηθεί με χρήση του αλγόριθμου κατακερματισμού που περιγράφτηκε στην ενότητα ανάλυσης του κώδικα του web service. Αν ο κωδικός δεν εντοπιστεί στον ανάλογο πίνακα της βάσης δεδομένων, επιστρέφεται null. Διαφορετικά, επιστρέφεται ένα αντικείμενο της τάξης Student ή Professor με ιδιότητες τις τιμές όλων των στηλών για τη συγκεκριμένη σειρά του πίνακα στη βάση δεδομένων.

Αν επιστραφεί null από το web service, η μέθοδος του listener ενημερώνει τον χρήστη με ανάλογο μήνυμα Toast. Αν τα στοιχεία που δόθηκαν είναι σωστά και αποθηκευτεί ο νέος κωδικός στη βάση δεδομένων, εκτελούνται κάποιες ενέργειες: αρχικά δημιουργούνται δυο νέα ζεύγη κλειδιού-τιμής και αποθηκεύονται μέσω SharedPreferences στο αντίστοιχο αρχείο xml της συσκευής. Το πρώτο αφορά στο μόνιμη πλέον ιδιότητα του χρήστη, όπως αποφασίστηκε μετά την εύρεση του προσωρινού του κωδικού στον ανάλογο πίνακα και την αποθήκευση του νέου του κωδικού. Το δεύτερο λαμβάνεται από το αντικείμενο της τάξης Student ή Professor που επιστρέφει από το web service και πρόκειται για το id του χρήστη στον αντίστοιχο πίνακα, το οποίο επίσης αποθηκεύεται στο xml αρχείο για χρησιμοποίηση κατά τη διαδικασία εισόδου του χρήστη στην εφαρμογή. Επίσης, εμφανίζεται το μήνυμα ενημέρωσης της επιτυχούς εγγραφής στην εφαρμογή.

Ως πέμπτο και τελευταίο όρισμα της μεθόδου αποστολής των δεδομένων στο web service, δίνεται ένα ανώνυμο instance του static interface Response.ErrorListener, και γίνεται override η μέθοδός του με όνομα onErrorResponse(). Η μέθοδος αυτή ενεργοποιείται σε περίπτωση που συνέβη κάποιο σφάλμα κατά την αλληλεπίδραση με το web service. Αναλόγως με το είδος του σφάλματος που θα επιστραφεί, γίνεται και η ενημέρωση του χρήστη με το κατάλληλο Toast μήνυμα.

Τέλος, το αντικείμενο της τάξης JsonObjectRequest που δημιουργήθηκε προστίθεται στην ουρά με τα προς αποστολή αιτήματα στο web service μέσω της κλήσης της μεθόδου .add() ενός αντικειμένου της τάξης RequestQueue που έχει οριστεί ως ιδιότητα της τάξης Register. Η τάξη RequestQueue ανήκει στη βιβλιοθήκη Volley του Android, που απλοποιεί την αποστολή HTTP αιτημάτων από μια εφαρμογή Android. Όταν βρεθεί πρώτο στη λίστα, θα σταλθεί στο web service.

*4. Login activity*


Μέσω του παραπάνω activity πραγματοποιείται η είσοδος του χρήστη στην εφαρμογή. Αρχικά πληκτρολογείται το email και ο νέος κωδικός που επιλέχθηκε κατά την εγγραφή του στην εφαρμογή. Επίσης, μέσω ενός CheckBox, δίνεται και η δυνατότητα επιλογής του να παραμείνει συνδεδεμένος στην εφαρμογή για ορισμένο χρονικό διάστημα.

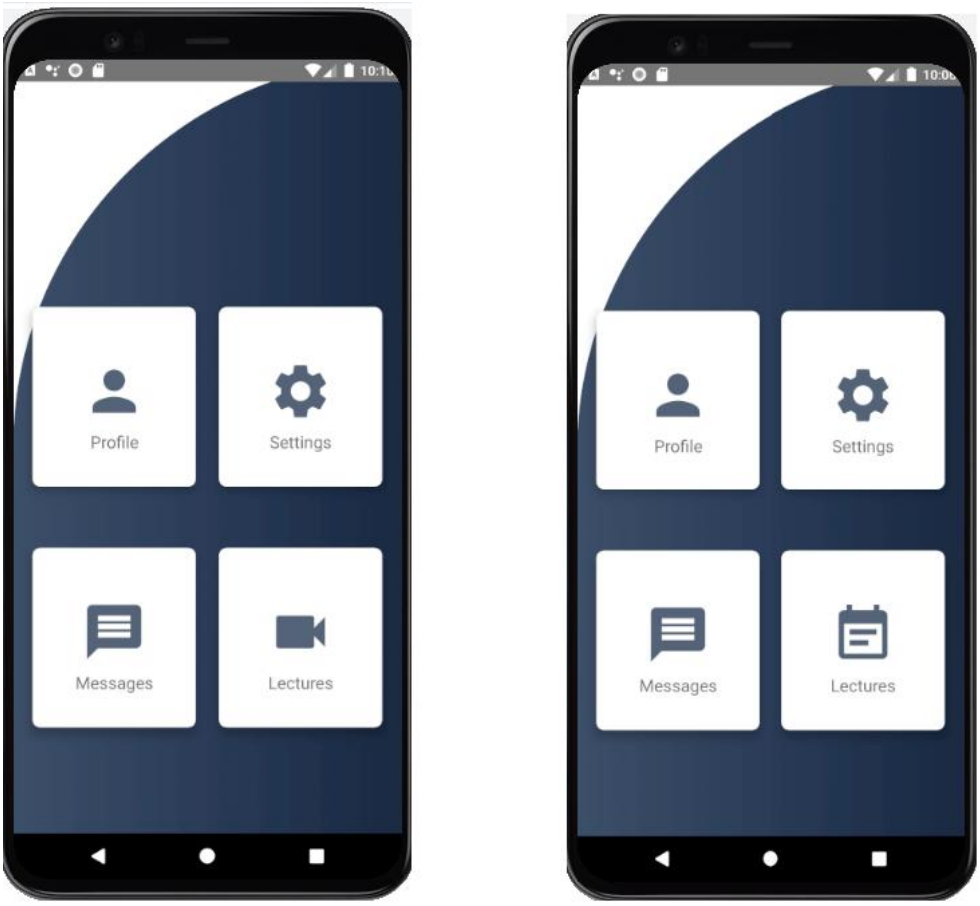
Στη συνέχεια, με το πάτημα του κουμπιού “Log in”, εκτελούνται συγκεκριμένες ενέργειες: αφού εξασφαλιστεί ότι κανένα από τα δύο EditTexts δεν περιέχει κενό αλφαριθμητικό ή αλφαριθμητικό που αποτελείται μόνο από χαρακτήρες κενού διαστήματος, καλείται μια μέθοδος για τη δημιουργία του JSONObject με ορίσματα τα ελεγμένα αλφαριθμητικά των δύο EditTexts και το id του χρήστη, το οποίο λαμβάνεται μέσω ενός αντικειμένου της τάξης SharedPreferences από το xml αρχείο της συσκευής. Επίσης σχηματίζεται και το πλήρες URL του web service στο οποίο θα σταλθεί το request, αναλόγως της μόνιμης ιδιότητας του χρήστη, η οποία είναι επίσης αποθηκευμένη στο xml αρχείο της συσκευής. Η ιδιότητα του χρήστη καθορίζει και το σε ποιον πίνακα της βάσης δεδομένων, αυτόν των καθηγητών ή αυτόν των φοιτητών, θα γίνει η αναζήτηση. Στο εσωτερικό μιας ακόμα μεθόδου, το JSONObject και το URL στέλνονται μέσω ενός αιτήματος POST προς το web service και ορίζονται τα ανώνυμα instances των static interfaces με τον κώδικα που κάνει override τις δύο μεθόδους που αναφέρθηκαν και παραπάνω, για τη διαχείριση της επιτυχημένης ή αποτυχημένης αποστολής των δεδομένων προς το web service.

Η αντίστοιχη μέθοδος της τάξης Controller της Spring Boot εφαρμογής πραγματοποιεί αναζήτηση στη βάση δεδομένων με το id, το email και τον κωδικό πρόσβασης του χρήστη, ο οποίος δίνεται ως είσοδος στον αλγόριθμο κατακερματισμού Argon2 μαζί με το salt του κωδικού της αντίστοιχης γραμμής και η έξοδος του αλγόριθμου συγκρίνεται με τον ήδη αποθηκευμένο κωδικό . Αν δεν βρεθεί χρήστης στον αντίστοιχο πίνακα της βάσης δεδομένων με αυτόν τον συνδυασμό στοιχείων, επιστρέφεται null στην client συσκευή και ο χρήστης ενημερώνεται με το κατάλληλο μήνυμα. Αντιθέτως, αν τα στοιχεία που δόθηκαν είναι σωστά, επιστρέφεται μήνυμα μέσω Toast για την επιτυχή είσοδο του χρήστη στην εφαρμογή.

Τέλος, αν ο χρήστης είχε πατήσει το CheckBox με όνομα “Keep me logged in”, δημιουργείται ένα αντικείμενο της τάξης LocalDateTime με τιμή την ισχύουσα ημερομηνία και ώρα. Στην τιμή αυτή προστίθεται το διάστημα για το οποίο έχει αποφασιστεί οι χρήστες να παραμένουν συνδεδεμένοι στην εφαρμογή, παρακάμπτοντας τη διαδικασία της εισόδου. Το νέο LocalDateTime αντικείμενο που έχει ως τιμή τη νέα ημερομηνία και ώρα αποθηκεύεται στο xml αρχείο της συσκευής μέσω SharedPreferences. Σε κάθε άνοιγμα της εφαρμογής, ο κώδικας του activity Redirect που αναφέρθηκε παραπάνω, ελέγχει την ημερομηνία και ώρα ανοίγματος και, αν προηγούνται της αποθηκευμένης ημερομηνίας και ώρας, ο χρήστης μεταφέρεται κατευθείαν στην οθόνη με το βασικό μενού της εφαρμογής. Η τελευταία γραμμή κώδικα της μεθόδου για την διαχείριση της επιτυχούς εισόδου της τάξης Login πραγματοποιεί και αυτή τη μετάβαση στην οθόνη του βασικού μενού. Η μέθοδος που ενεργοποιείται σε περίπτωση αποτυχημένης προσπάθειας εισόδου, ενημερώνει τον χρήστη μέσω της επίδειξης ενός μηνύματος, αντίστοιχου με το σφάλμα που προέκυψε.

Τέλος, το αντικείμενο της τάξης JsonObjectRequest που δημιουργήθηκε προστίθεται στην ουρά με τα προς αποστολή αιτήματα στο web service.

*5. MainMenu activity*



Το activity αυτό περιλαμβάνει το βασικό μενού του χρήστη. Πατώντας κάποιο από τα τέσσερα διαφορετικά κουμπιά, ο χρήστης μεταφέρεται στην αντίστοιχη νέα οθόνη της εφαρμογής. Αριστερά απεικονίζεται η οθόνη για τους χρήστες με ιδιότητα φοιτητή και δεξιά για αυτούς με ιδιότητα καθηγητή. Το εικονίδιο του κουμπιού με κείμενο “Lectures” καθορίζεται κατά την εκτέλεση της μεθόδου onCreate() της τάξης MainMenu βάσει της τιμής της ιδιότητας του χρήστη, που βρίσκεται αποθηκευμένη στο αντίστοιχο κλειδί του xml αρχείου της συσκευής.

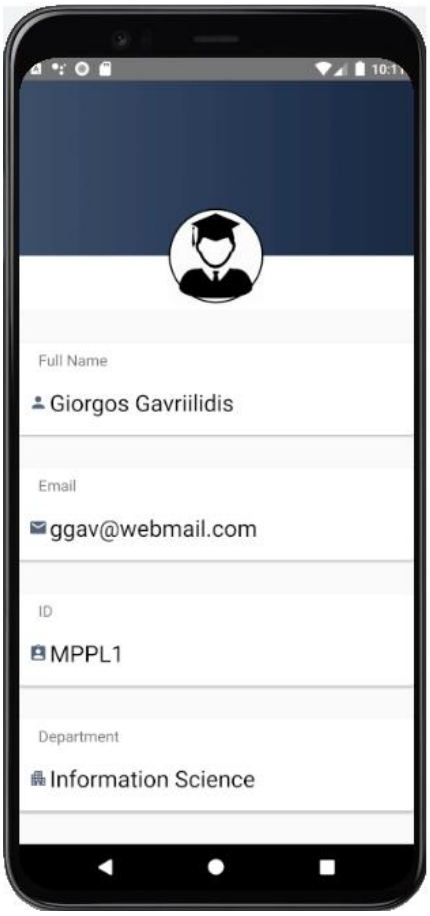
Βάσει αυτής καθορίζεται και η οθόνη που θα εμφανιστεί μετά το πάτημα του κουμπιού “Messages”, καθώς η υλοποίηση της αποστολής και λήψης μηνυμάτων διαφέρει από καθηγητή σε φοιτητή, με τους πρώτους να έχουν τη δυνατότητα αποστολής μηνυμάτων στα τμήματα όπου διδάσκουν και τους δεύτερους αυτή της ανάγνωσης των μηνυμάτων που έχουν σταλθεί σε κάποιο τμήμα μαθήματος στο οποίο ανήκουν. Ο διαχωρισμός αυτός οδήγησε στη δημιουργία δύο διαφορετικών τάξεων που κληρονομούν από την τάξη Activity.

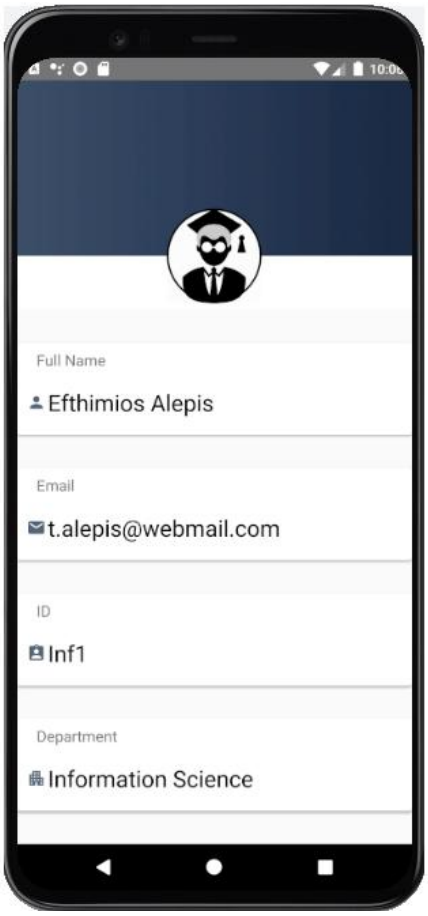
Εκτός των παραπάνω, υπάρχει και υλοποίηση κώδικα για ενημέρωση του φοιτητή με τον αριθμό των μη αναγνωσμένων από αυτόν μηνυμάτων που έοχυν σταλθεί σε κάποιο ή κάποια από τα τμήματα που παρακολουθεί. Η ενημέρωση για ένα μήνυμα που δημιουργείται από τον καθηγητή και αποθηκεύεται στη βάση δεδομένων δεν είναι άμεση, καθώς αυτό θα απαιτούσε την υλοποίηση του πρωτοκόλλου WebSocket για την επίτευξη αμφίδρομης επικοινωνίας μεταξύ ενός client και του web service.

Αντί του παραπάνω, όταν ο φοιτητής επισκεφτεί την οθόνη με τα μηνύματα που έχει λάβει, η λειτουργικότητα της οποίας θα αναλυθεί σε επόμενη υποενότητα, η ίσχύουσα ημερομηνία και ώρα αποθηκεύεται στο xml αρχείο της συσκευής μέσω της τάξης SharedPreferences.

Κατά την εμφάνιση της οθόνης του βασικού μενού, εκτελείται κώδικας που στέλνει ένα αίτημα προς το web service για λήψη μιας λίστας με όλα τα μηνύματα που έχουν σταλθεί σε τμήματα μαθημάτων στα οποία αυτός ανήκει κατά το τρέχον εξάμηνο. Αν η λίστα δεν είναι κενή, συγκρίνεται η ημερομηνία και ώρα αποστολής του κάθε μηνύματος με αυτή της τελευταίας επίσκεψης της οθόνης με τα μηνύματα από τον φοιτητή. Αν αυτή έπεται της τελευταίας του επίσκεψης, αυτό σημαίνει ότι ο φοιτητής δεν έχει διαβάσει το συγκεκριμένο μήνυμα. Για κάθε μη αναγνωσμένο μήνυμα, η τιμή μιας μεταβλητής τύπου ακεραίου αριθμού που λειτουργεί ως μετρητής αυξάνεται κατά ένα. Μετά το τέλος της διαπέρασης της λίστας και αν υπάρχουν μη αναγνωσμένα μηνύματα, εμφανίζεται ένα στρογγυλό κόκκινο TextView στο πάνω δεξιά μέρος του κουμπιού “Messages” με τον αριθμό αυτών, λειτουργώντας ως notification.

*6. Profile activity*



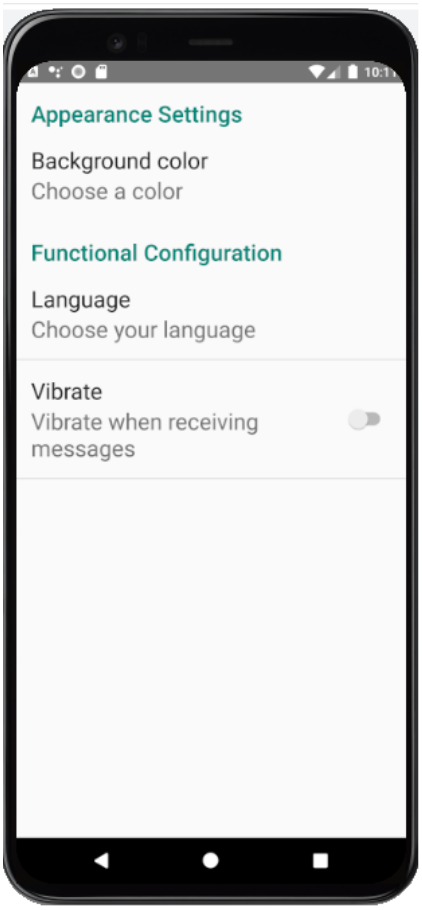



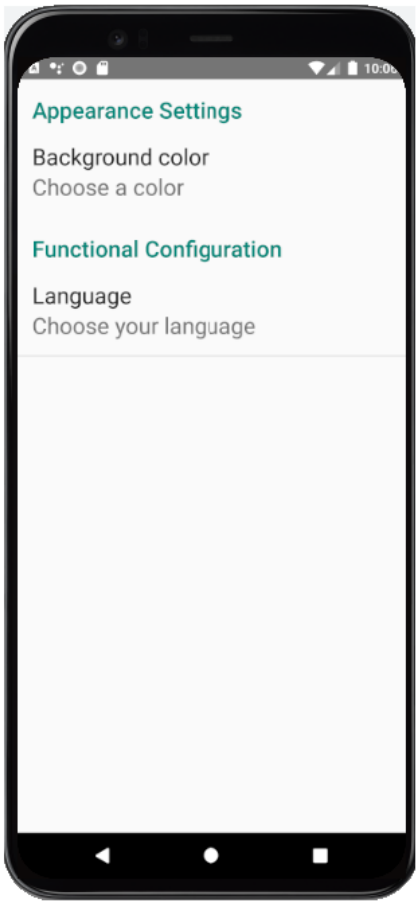

Στην οθόνη αυτή απεικονίζονται οι βασικές πληροφορίες ενός φοιτητή ή καθηγητή που έχει εγγραφεί στην εφαρμογή. Αριστερά απεικονίζεται η οθόνη του φοιτητή και δεξιά αυτή του καθηγητή.

Κατά την εκτέλεση της μεθόδου onCreate() της τάξης Profile, ορίζεται το εικονίδιο που θα εμφανιστεί, βάσει της ιδιότητας του χρήστη. Εξετάζοντας την ιδιότητα αυτή σχηματίζεται και το πλήρες URL του web service στο οποίο θα σταλθεί το αίτημα. Στο URL προστίθεται και το id του χρήστη, το οποίο θα ληφθεί μέσω του annotation @PathVariable από την κατάλληλη μέθοδο της τάξης Controller της Spring Boot εφαρμογής, έτσι ώστε να γίνει μέσω αυτού η αναζήτηση στον αντίστοιχο πίνακα της βάσης δεδομένων.

Τα στοιχεία της γραμμής του πίνακα που έχει ως πρωτεύον κλειδί το συγκεκριμένο id επιστρέφονται ως ένα αντικείμενο της τάξης Student ή της τάξης Professor, οι οποίες έχουν οριστεί σε δύο αρχεία εντός και της client εφαρμογής, εκτός από αυτή του web service, με τις ίδιες ακριβώς ιδιότητες. Μέσω των getters της κατάλληλης τάξης λαμβάνονται οι πληροφορίες, οι οποίες στη συνέχεια απεικονίζονται στα TextViews του Profile activity.

*7.Settings activity*




Πρόκειται για την οθόνη από όπου ο χρήστης επιλέγει τις επιθυμητές ρυθμίσεις για την εφαρμογή. Στα αριστερά απεικονίζεται η οθόνη των ρυθμίσεων του φοιτητή και στα δεξιά αυτή του καθηγητή.

Η εφαρμογή, εκτός από τέσσερα διαφορετικά μεγέθη οθόνης, καθώς και κάθετη και οριζόντια διάταξη αυτής, υποστηρίζει και τρεις διαφορετικές γλώσσες: Ελληνικά, Αγγλικά και Ισπανικά. Ακόμα, ο χρήστης μπορεί να επιλέξει μεταξύ τριών διαφορετικών χρωμάτων απεικόνισης των διάφορων οθονών της εφαρμογής: μπλε, που είναι και το προεπιλεγμένο χρώμα, κίτρινο και κόκκινο.

Με πάτημα της οθόνης στο σημείο επιλογής γλώσσας ή χρώματος, εμφανίζεται ένα dialog μέσω του οποίου μπορεί να επιλεχθεί η επιθυμητή γλώσσα ή χρώμα. Μια ακόμη επιλογή, που αφορά μόνο τους χρήστες με φοιτητική ιδιότητα, καθορίζει το αν η συσκευή θα δονείται κατά την εμφάνιση της οθόνης του βασικού μενού στην περίπτωση που υπάρχουν μη αναγνωσμένα μηνύματα, ταυτόχρονα με την εμφάνιση του notification με τον αριθμό των μηνυμάτων αυτών.

Η τάξη Settings κληρονομεί από την τάξη PreferenceActivity, που χρησιμοποιείται για την απεικόνιση μιας λίστας ρυθμίσεων στον χρήστη της Android συσκευής. Σε ένα xml αρχείο με τίτλο preferences.xml έχουν οριστεί δύο ListPreferences για την επιλογή της γλώσσας και του χρώματος μέσω λίστας και ένα SwitchPreference για την επιλογή της δόνησης μέσω του διακόπτη. Σε ένα ακόμη αρχείο με τίτλο arrays.xml ορίζονται τα strings που θα εμφανίζονται ως επιλογές για τη γλώσσα και το χρώμα στο αντίστοιχο dialog, καθώς και η τιμή στην οποία θα αντιστοιχεί κάθε μία από τις παραπάνω επιλογές. Με την επιλογή του string από τον χρήστη, η αντίστοιχη τιμή αποθηκεύεται αυτόματα στο xml αρχείο εντός της συσκευής που περιλαμβάνει όλες τις απαραίτητες για την εφαρμογή πληροφορίες. Η οπτική παρουσίαση

κάθε οθόνης της εφαρμογής που προβάλλεται από τον χρήστη καθορίζεται βάσει του αποθηκευμένου χρώματος και της αποθηκευμένης γλώσσας.

*8.MyMessagesStudent activity*


Η συγκεκριμένη οθόνη είναι ορατή μόνο από χρήστες με την ιδιότητα του φοιτητή. Κατά τη δημιουργία του activity αυτού μέσω της μεθόδου onCreate(), η ισχύουσα ημερομηνία και ώρα αποθηκεύεται εντός μιας ιδιότητας της τάξης, ώστε να είναι ορατή από όσες από τις υπόλοιπες μεθόδους της θα χρειαστεί να την χρησιμοποιήσουν. Ταυτόχρονα, η ημερομηνία και ώρα στην οποία ο φοιτητής πρόβαλε την οθόνη την τελευταία φορά που διάβασε τα εισερχόμενα μηνύματα λαμβάνεται από το αρχείο xml εντός της συσκευής μέσω SharedPreferences και αποθηκεύεται σε μια ακόμα ιδιότητα της τάξης.

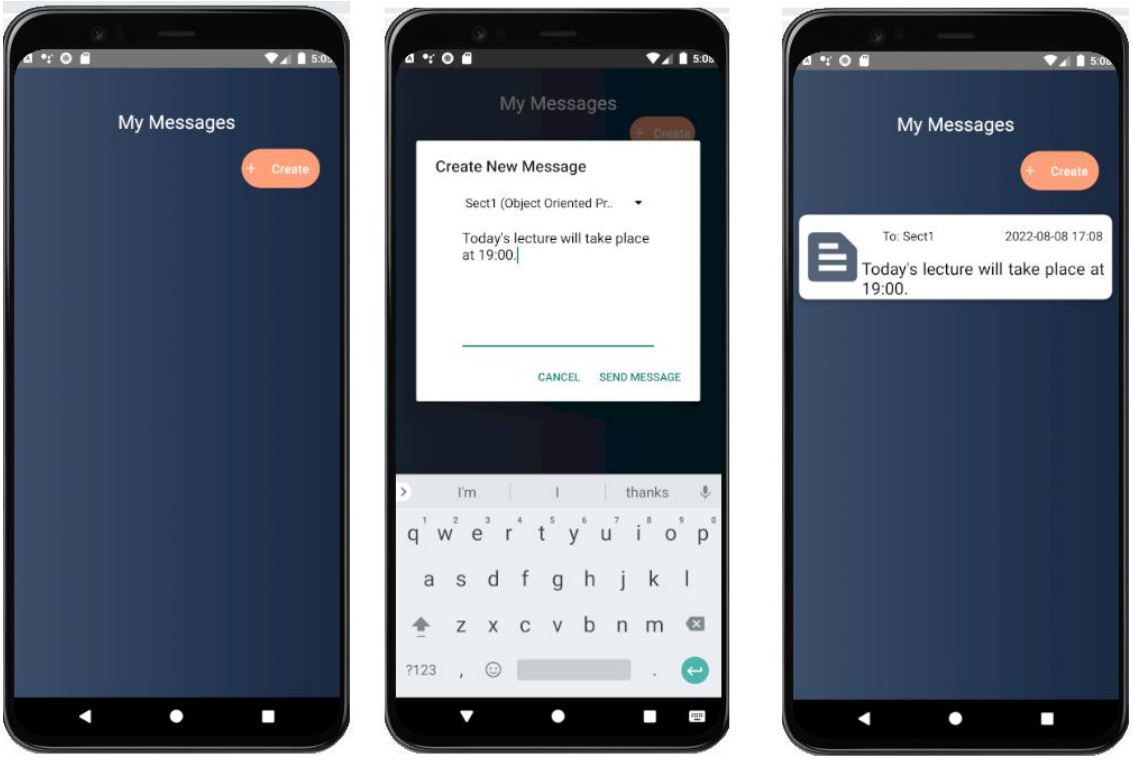
Στη συνέχεια, μέσω μιας μεθόδου, στέλνεται αίτημα στο κατάλληλο URL του web service για λήψη μιας λίστας με όλα τα μηνύματα που έχουν σταλθεί κατά το τρέχον εξάμηνο σε οποιοδήποτε από τα τμήματα μαθημάτων που παρακολουθεί ο φοιτητής, ταξινομημένα με φθίνουσα ημερομηνιακή σειρά, ώστε το πιο πρόσφατο να βρίσκεται στην κορυφή της λίστας. Τα μηνύματα απεικονίζονται σε ένα RecyclerView που συνδέεται με ένα αντικείμενο της τάξης Adapter, μέσω του οποίου το πρώτο θα λάβει τα απαραίτητα δεδομένα με την κατάλληλη μορφοποίηση, ώστε να απεικονιστούν στον χρήστη. Στο εσωτερικό της μεθόδου για την αποστολή αιτήματος, αν η λίστα μηνυμάτων που θα επιστραφεί από το web service δεν

είναι κενή, καλείται ο constructor της τάξης MyMessagesStudentAdapter με ορίσματα, μεταξύ άλλων, τη λίστα με τα μηνύματα και την ιδιότητα με την ημερομηνία και ώρα της τελευταίας προηγούμενης προβολής της οθόνης.

Εντός της κατάλληλης μεθόδου της τάξης MyMessagesStudentAdapter, αντιπαραβάλλεται η παραπάνω ημερομηνία και ώρα με αυτή της αποστολής του κάθε μηνύματος. Αν η δεύτερη έπεται της πρώτης, σημαίνει ότι το προς έλεγχο μήνυμα στάλθηκε μετά την τελευταία προβολή της οθόνης από τον φοιτητή, οπότε το κείμενο όλων των TextViews του απεικονίζεται με έντονη γραφή. Αντιθέτως, αν το μήνυμα έχει ήδη προβληθεί, απεικονίζεται χωρίς bold χαρακτήρες.

Τέλος, στον κώδικα της τάξης γίνεται override και η μέθοδος onStop(), η οποία είναι μια από τις μεθόδους που αφορούν στον αποκαλούμενο ως κύκλο ζωής του activity, όπως και η onCreate() Η onStop() ενεργοποιείται μόλις η οθόνη σταματήσει να βρίσκεται στο προσκήνιο. Στο εσωτερικό της μεθόδου, αποθηκεύεται στο xml αρχείο με τις απαραίτητες πληροφορίες η ημερομηνία και ώρα κατά την οποία προβλήθηκε αυτή την φορά η οθόνη στον χρήστη, αντικαθιστώντας την προηγούμενη. Αυτή θα αποτελέσει το χρονικό σημείο αναφοράς για τον εντοπισμό των μη αναγνωσμένων μηνυμάτων την επόμενη φορά που θα προβληθεί το συγκεκριμένο activity.

*9.MyMessagesProfessor activity*




Μέσω της συγκεκριμένης οθόνης δίνεται η δυνατότητα στον καθηγητή να στείλει μηνύματα προς τα τμήματα στα οποία διδάσκει κατά το τρέχον εξάμηνο. Κατά την αρχική φόρτωση του activity, στέλνεται ένα αίτημα προς το web service για λήψη και εμφάνιση όλων των μηνυμάτων που έχει στείλει ο καθηγητής. Στην αριστερή οθόνη από αυτές που απεικονίζονται παραπάνω η λίστα είναι κενή, αφού ο καθηγητής δεν έχει αποστείλει ακόμη κάποιο μήνυμα.

Αυτό μπορεί να πραγματοποιηθεί με το πάτημα του κουμπιού πάνω δεξιά, για δημιουργία ενός νέου μηνύματος. Τότε εμφανίζεται ένα dialog, ο κώδικας για τη λειτουργικότητα του οποίου βρίσκεται εντός ενός αρχείου που περιέχει την τάξη CreateMessageDialog. Η τάξη αυτή κληρονομεί από την AppCompatDialogFragment και μέσω των μεθόδων της γίνεται η διαχείριση της δημιουργίας ενός μηνύματος.

Εντός μίας από αυτές πραγματοποιείται ένα αίτημα προς το web service, μέσω του οποίου στέλνονται προς την client συσκευή τα τμήματα στα οποία διδάσκει ο καθηγητής που την χρησιμοποιεί. Μέσω του Spinner στο πάνω μέρος του dialog, επιλέγεται το επιθυμητό τμήμα και στη συνέχεια πληκτρολογείται το προς αποστολή μήνυμα εντός του EditText. Με το πάτημα της αποστολής του μηνύματος, στέλνεται ένα ακόμη αίτημα προς το web service, που περιέχει το κείμενο του μηνύματος και τις υπόλοιπες απαραίτητες πληροφορίες. Από το JSONObject που αποστέλλεται στην κατάλληλη μέθοδο της τάξης Controller σχηματίζεται εντός της μεθόδου αυτής ένα αντικείμενο της τάξης της Spring Boot εφαρμογής με όνομα Message. Μέσω της τάξης με κατάληξη -Service και του interface με κατάληξη -Repository που αντιστοιχίζονται στη συγκεκριμένη τάξη, πραγματοποιείται η αποθήκευση του νέου μηνύματος στον ομώνυμο πίνακα της βάσης δεδομένων.

Αφού ληφθεί η ειδοποίηση επιτυχούς αποθήκευσης του μηνύματος, γίνεται ένα ακόμα αίτημα προς το web service, για την εκ νέου αποστολή στην client συσκευή των μηνυμάτων που έχουν σταλθεί από τον καθηγητή, τα οποία ταξινομούνται με φθίνουσα ημερομηνιακή σειρά, έτσι ώστε το πιο πρόσφατο να εμφανίζεται πάντοτε στο ανώτατο μέρος της οθόνης.

*10.MyLectures activity*


Στη συγκεκριμένη οθόνη, προβάλλεται η λίστα με την επόμενη διαθέσιμη διάλεξη για κάθε τμήμα διδασκαλίας το οποίο παρακολουθεί ο φοιτητής ή διδάσκει ο καθηγητής, καθώς η οθόνη είναι κοινή και για τους δύο τύπους χρηστών.

Κατά τη δημιουργία του activity μέσω της μεθόδου onCreate(), ελέγχεται η ιδιότητα του χρήστη εξετάζοντας την τιμή του αντίστοιχου κλειδιού στο αρχείο xml που βρίσκεται στον φάκελο shared_preferences εντός της συσκευής. Αναλόγως αυτής, δημιουργείται και το κατάλληλο URL για αποστολή αιτήματος προς το web service, με προσθήκη και του id του χρήστη, το οποίο επίσης λαμβάνεται από το xml αρχείο.

Μετά την αποστολή του αιτήματος από την συσκευή, η Spring Boot εφαρμογή, μέσω μιας σειράς ελέγχων, δημιουργεί και επιστρέφει την λίστα με κάθε επόμενη διάλεξη, ταξινομημένη με αύξουσα ημερομηνιακή σειρά, έτσι ώστε η πλησιέστερη χρονικά διάλεξη να βρίσκεται στην κορυφή της λίστας.

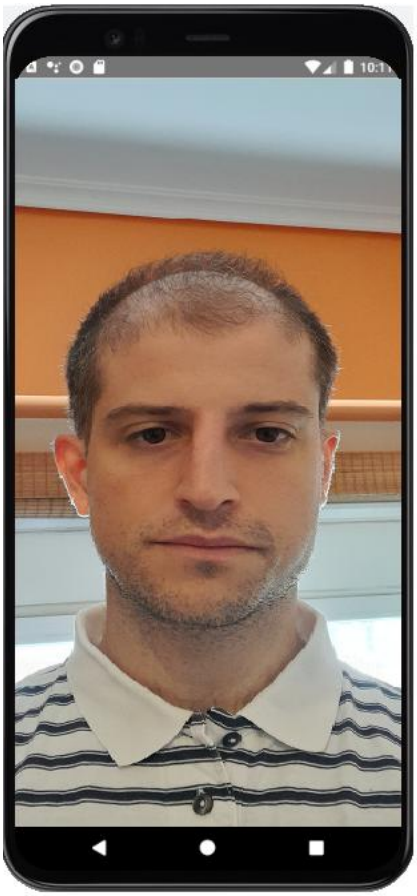
Ο χρήστης μπορεί να πατήσει εντός της περιοχής όπου απεικονίζονται τα στοιχεία της κάθε διάλεξης, έτσι ώστε να μεταβεί στην οθόνη όπου προσομοιώνεται η διαδικασία πραγματοποίησής της μέσω ζωντανής ροής με παράλληλο άνοιγμα της κάμερας της συσκευής, αν δοθεί η απαραίτητη άδεια από τον χρήστη.

Κατά το πάτημα της περιοχής αυτής, πραγματοποιούνται μια σειρά από έλεγχοι: αν η ημερομηνία της διαδικτυακής διάλεξης στην οποία θέλει να μεταβεί ο χρήστης είναι ίδια με την ισχύουσα και η ώρα βρίσκεται μεταξύ δέκα λεπτών πριν από την ώρα έναρξης της διάλεξης και της λήξης της, τότε επιτρέπεται η είσοδος στον χρήστη, ο οποίος μεταφέρεται στην οθόνη της ζωντανής διαδικτυακής διάλεξης.

Αν η παραπάνω συνθήκη δεν είναι αληθής, ελέγχεται αν η ισχύουσα ημερομηνία ή ώρα προηγούνται της αντίστοιχης ημερομηνίας ή ώρας έναρξης της διάλεξης που επιλέχθηκε. Αν κάποια από τις δυο συνθήκες ισχύει, η διάλεξη δεν είναι ακόμα διαθέσιμη στον χρήστη, ο οποίος ενημερώνεται με το ανάλογο μήνυμα.

Αν ούτε η δεύτερη συνθήκη είναι αληθής, τότε ισχύει μια τρίτη περίπτωση, κατά την οποία ο χρήστης μεταφέρθηκε στην οθόνη με τις επόμενες προγραμματισμένες διαλέξεις πριν να λήξει η διάλεξη την οποία θέλει να παρακολουθήσει, αλλά όταν πάτησε στην αντίστοιχη περιοχή για να μεταβεί σε αυτήν, η διάλεξη έχει τελειώσει, καθώς η ώρα λήξης της προηγείται της ισχύουσας ώρας. Και σε αυτή την περίπτωση, γίνεται ενημέρωση του χρήστη της συσκευής με το κατάλληλο μήνυμα μέσω Toast.

*11.LiveLecture activity*



Μέσω της οθόνης αυτής προσομοιώνεται η πραγματοποίηση της διάλεξης μέσω διαδικτύου. Εντός της οθόνης απεικονίζεται η εικόνα που λαμβάνεται από την μπροστινή κάμερα της συσκευής, η οποία περιλαμβάνει το πρόσωπο του χρήστη και, σε περίπτωση που αυτός έχει την ιδιότητα του φοιτητή, πραγματοποιείται η αναγνώριση και η αποστολή των συναισθημάτων του ανά ορισμένο χρονικό διάστημα, όπως θα αναλυθεί και στη συνέχεια.

Κατά την μέθοδο onCreate() της τάξης LiveLecture, λαμβάνεται από το intent του προηγούμενου activity το id της διάλεξης που επιλέχθηκε από τον χρήστη, μέσω της μεθόδου getIntent().getExtras(). Μέσω της μεθόδου putExtra() του αντικειμένου της τάξης Intent στάλθηκαν επίσης

από το προηγούμενο activity και η ώρα έναρξης και λήξης της διάλεξης, τα οποία επίσης λαμβάνονται μέσω του κώδικα της μεθόδου onCreate().

Ακόμα, δημιουργείται μια λίστα με τα strings των permissions που θα χρειαστεί η εφαρμογή, τα οποία έχουν δηλωθεί στο κατάλληλο τμήμα του αρχείου AndroidManifest.xml. Το πρώτο αφορά στην χρήση της κάμερας της συσκευής. Το δεύτερο πρόκειται για ένα custom permission, το οποίο χρειάζεται να δοθεί στην περίπτωση που ο χρήστης είναι φοιτητής, και αφορά στην πραγματοποίηση περιοδικής ανάλυσης συναισθήματος του προσώπου που απεικονίζεται στην κάμερα.

Η λίστα με τα strings των permissions ελέγχεται και, αν κάποιο από αυτά δεν έχει δοθεί, ζητείται η αντίστοιχη άδεια από τον χρήστη με την εμφάνιση του ανάλογου dialog. Μέσω μιας ακόμα μεθόδου η οποία ενεργοποιείται ασύγχρονα, όταν ο χρήστης επιτρέπει ή αρνείται κάθε ένα από τα permissions μέσω της επιλογής allow ή deny στο εμφανισμένο dialog, ακολουθούνται κάποιες ενέργειες ανά περίπτωση: σε περίπτωση άρνησης παραχώρησης του permission για χρήση της κάμερας της συσκευής, ο χρήστης ενημερώνεται με μήνυμα για το ότι αυτό το permission είναι απαραίτητο για το συγκεκριμένο activity και στη συνέχεια γίνεται επιστροφή στην προηγούμενη οθόνη της εφαρμογής. Ως προς το permission για ανάλυση συναισθήματος προσώπου, ελέγχεται αν αυτό έχει δοθεί και, σε αυτήν την περίπτωση ενημερώνεται με την τιμή true μια ιδιότητα της τάξης με τύπο boolean, η οποία εξετάζεται από άλλες μεθόδους της τάξης που παρουσιάζονται παρακάτω.

Επιστρέφοντας στη μέθοδο onCreate(), μετά τον έλεγχο των permissions, «φορτώνεται» σε ένα αρχείο, με τη χρήση αντικειμένων των τάξεων InputStream και FileOutputStream, ένα μοντέλο αναγνώρισης προσώπων εντός μιας εικόνας. Το μοντέλο αυτό ανήκει στη βιβλιοθήκη OpenCV, ή οποία έχει προστεθεί στα dependencies της εφαρμογής και βρίσκεται σε ένα αρχείο εντός του φακέλου με όνομα raw, με τίτλο *haarcascade_frontalface_alt.xml*. Το απόλυτο μονοπάτι του αρχείου με το μοντέλο δίνεται ως όρισμα σε ένα νέο αντικείμενο της τάξης CascadeClassifier, που αποθηκεύεται επίσης σε μια ιδιότητα της τάξης με τον ίδιον τύπο.

Στον φάκελο raw που αναφέρθηκε παραπάνω έχει αποθηκευτεί με τίτλο *emotion_recognition_model.tflite* και το συμπιεσμένο αρχείο με το εκπαιδευμένο μοντέλο του συνελικτικού νευρωνικού δικτύου, το οποίο θα πραγματοποιήσει την περιοδική ανάλυση του απεικονιζόμενου συναισθήματος. Το μοντέλο αυτό «φορτώνεται» σε μια ιδιότητα της τάξης LiveLecture που είναι αντικείμενο της τάξης Interpreter της βιβλιοθήκης Tensorflow Lite, η οποία έχει επίσης προστεθεί στο αρχείο με τα dependencies της εφαρμογής.

Ακόμα, δημιουργείται ένα νέο αντικείμενο της τάξης Handler, το οποίο αποθηκεύεται σε μια ιδιότητα της τάξης και καλείται μια άλλη μέθοδος, με όνομα startRepeatingFerTask(). Με τον κώδικα στο εσωτερικό της, δημιουργείται ένα αντικείμενο του interface Runnable, το οποίο πρόκειται για ένα νέο thread που θα τρέχει στο παρασκήνιο. Εντός της μεθόδου του με τίτλο run(), η οποία γίνεται override, τίθεται μια ιδιότητα τύπου boolean με όνομα isTimeForFer της τάξης LiveLecture με τιμή true και ορίζεται ότι το thread θα ξανατρέξει σε δέκα δευτερόλεπτα.

Ακόμα, ενεργοποιείται η διαδικασία εμφάνισης και ανάλυσης των frames που λαμβάνονται από την μπροστινή κάμερα της συσκευής μέσω ενός PreviewView, με την κλήση μιας μεθόδου με τίτλο bindPreview(), η οποία περιλαμβάνει τη βασική λειτουργικότητα αυτού του activity. Εντός αυτής, ορίζεται ένα αντικείμενο της τάξης ImageAnalysis.Analyzer με override της μεθόδου analyze(): στην αρχή της, λαμβάνεται το frame που έχει φτάσει αυτή την στιγμή μέσω της ανοιχτής μπροστινής κάμερας με τη μορφή αντικειμένου της τάξης Image και δίνεται ως όρισμα σε μια μέθοδο που το μετατρέπει σε αντικείμενο της τάξης Mat, που ανήκει στη βιβλιοθήκη OpenCV. Επίσης, αποθηκεύεται η ισχύουσα ώρα σε μια μεταβλητή και έπειτα ο κώδικας μεταβαίνει σε μια σειρά από if statements.

Μέσω του πρώτου από αυτά ελέγχεται αν η ιδιότητα της τάξης με όνομα isTimeForFer, η οποία αναφέρθηκε προηγουμένως, έχει τιμή true. Η μέθοδος analyze() τρέχει χωρίς διακοπή, λαμβάνοντας συνεχώς νέα frames, ωστόσο η συγκεκριμένη ιδιότητα αποκτά την τιμή true κάθε δέκα δευτερόλεπτα, έτσι ώστε η πρόβλεψη για το συναίσθημα που απεικονίζεται στο πρόσωπο του χρήστη να στέλνεται ανά αυτό το ορισμένο χρονικό διάστημα στη βάση δεδομένων μέσω του web service.

Μέσω του δευτέρου if statement ελέγχεται αν έχει δοθεί η άδεια αναγνώρισης συναισθημάτων προσώπου από τον χρήστη, ενώ μέσω των επόμενων δύο εξετάζεται αν η ισχύουσα ώρα βρίσκεται εντός των χρονικών πλαισίων της διάλεξης, μέσω των ιδιοτήτων startTime και endTime, στις οποίες αποθηκεύτηκε η αντίστοιχη πληροφορία από την διάλεξη που επιλέχθηκε στην προηγούμενη οθόνη.

Αν όλες οι παραπάνω συνθήκες είναι αληθείς, καλείται μια μέθοδος με όνομα recognizeEmotion() και αμέσως μετά η ιδιότητα isTimeForFer της τάξης λαμβάνει τιμή false, ώστε να μην τρέξει ξανά η συγκεκριμένη μέθοδος πριν αυτή τεθεί ως true μέσω του thread που θα τρέξει στο παρασκήνιο μετά από δέκα δευτερόλεπτα.

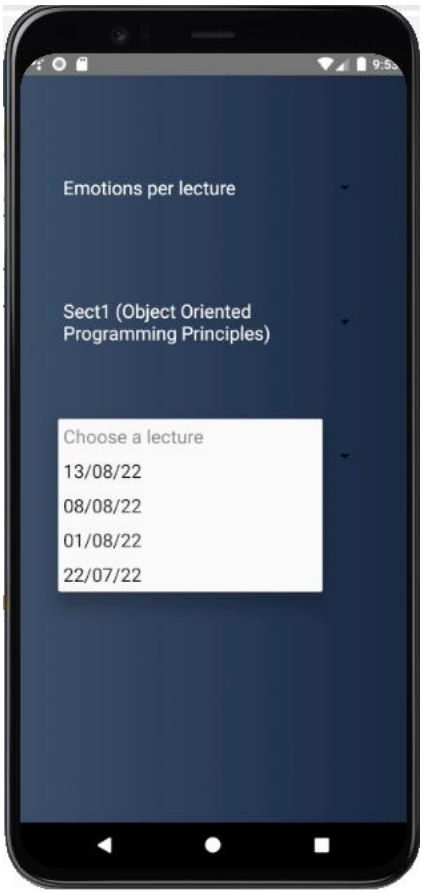
Στο εσωτερικό της μεθόδου recognizeEmotion(), η εικόνα του frame που λήφθηκε από την μπροστινή κάμερα και που της δίνεται ως όρισμα με τη μορφή Mat, μετατρέπεται από έγχρωμη σε ασπρόμαυρη, καθώς το συνελικτικό νευρωνικό δίκτυο εκπαιδεύτηκε με ασπρόμαυρες φωτογραφίες ως εισόδους, όπως αναφέρθηκε και στην αντίστοιχη ενότητα της διπλωματικής εργασίας.

Στη συνέχεια, μέσω του αντικειμένου της τάξης CascadeClassifier, εντοπίζονται όλα τα τετράγωνα με πρόσωπα που απεικονίζονται εντός της εικόνας, αν υπάρχουν τέτοια στο εσωτερικό της, και αποθηκεύονται σε μια λίστα. Από αυτή, αν υπάρχουν εντός της περισσότερα από ένα πρόσωπα, λαμβάνεται μόνο το πρώτο τετράγωνο προσώπου, καθώς αυτό θα πρόκειται κατά πάσα πιθανότητα για το πρόσωπο του φοιτητή που παρακολουθεί τη διάλεξη, ενώ τα υπόλοιπα θα ανήκουν πιθανώς σε άτομα που συγκατοικούν με αυτόν και που θα βρίσκονται στο παρασκήνιο. Το τμήμα με το τετράγωνο που περιλαμβάνει το πρόσωπο αποκόπτεται από την αρχική εικόνα, σχηματίζοντας μια νέα, η οποία δίνεται ως όρισμα στη μέθοδο doInference(), που πραγματοποιεί την πρόβλεψη για το συναίσθημα που απεικονίζεται στο πρόσωπο του χρήστη.

Η μέθοδος αυτή καλεί την μέθοδο .run() της ιδιότητας της τάξης που είναι αντικείμενο της τάξης Interpreter, δίνοντας ως όρισμα την εικόνα και μια άδεια λίστα με πέντε θέσεις. Μέσω της .run(), η εικόνα δίνεται ως είσοδος στο συμπιεσμένο μοντέλο του συνελικτικού νευρωνικού δικτύου, το οποίο γεμίζει την άδεια λίστα με τις διαδοχικές προβλέψεις του για το κατά πόσο ανήκει το συναίσθημα που απεικονίζεται στην εικόνα σε κάθε μία από τις πέντε κατηγορίες συναισθημάτων που αυτό έχει εκπαιδευτεί να αναγνωρίζει. Η λίστα αυτή εξετάζεται και λαμβάνεται η θέση του στοιχείου με τη μεγαλύτερη τιμή, που αντιπροσωπεύει την πρόβλεψη του αλγόριθμου, καθώς και η τιμή αυτή, που αντιπροσωπεύει τη σιγουριά του για την πρόβλεψη αυτή. Αυτές οι δύο τιμές επιστρέφονται ως μια λίστα δεκαδικών με δύο θέσεις στη μέθοδο recognizeEmotion(). Το αλφαριθμητικό του συναισθήματος που αντιστοιχεί στη θέση του πίνακα με τη μεγαλύτερη σιγουριά πρόβλεψης επιλέγεται ως εξής: ως μια από τις ιδιότητες της τάξης LiveLecture έχει οριστεί μια λίστα από πέντε strings. Κάθε ένα από αυτά περιλαμβάνει τη λέξη για ένα από τα πέντε βασικά συναισθήματα και βρίσκεται στην ίδια θέση εντός του πίνακα αυτού στην οποία βρίσκεται και η πρόβλεψη του μοντέλου για το αν η είσοδος απεικονίζει το συγκεκριμένο συναίσθημα στον πίνακα με τις προβλέψεις που δίνεται ως έξοδος από αυτό. Λαμβάνοντας λοιπόν τη θέση του στοιχείου με τη μεγαλύτερη τιμή, αυτή χρησιμοποιείται ως δείκτης για την εύρεση του κατάλληλου string του συναισθήματος εντός του προκαθορισμένου πίνακα με τα αλφαριθμητικά.

Στην τελευταία γραμμή της μεθόδου recognizeEmotion(), καλείται η μέθοδος της εφαρμογής με τίτλο setEmotion() και όρισμα το string “not found” και τον αριθμό 0.0 σε περίπτωση που δεν εντοπίστηκε πρόσωπο εντός της εικόνας. Αν εντοπίστηκε πρόσωπο και ακολουθήθηκε η διαδικασία που περιγράφηκε παραπάνω, δίνονται ως ορίσματα το string που αντιπροσωπεύει το συναίσθημα που εντοπίστηκε και η δεκαδική τιμή που αντιπροσωπεύει τη σιγουριά του αλγόριθμου. Η μέθοδος setEmotion() δημιουργεί ένα JSONObject με τα δύο παραπάνω στοιχεία, καθώς και την ισχύουσα ώρα, το id του φοιτητή που λήφθηκε από τα shared_preferences και το id της διάλεξης που λήφθηκε από την προηγούμενη οθόνη της εφαρμογής. Το JSONObject αυτό αποστέλλεται στο web service, ώστε, μέσω της κατάλληλης μεθόδου της τάξης Controller, να αποθηκευτεί στη βάση δεδομένων εντός του πίνακα attends, που περιλαμβάνει τις προβλέψεις για τα συναισθήματα που απεικονίζονται στις οθόνες των Android συσκευών όσων φοιτητών έδωσαν την άδεια αναγνώρισης συναισθήματος στην εφαρμογή. Η διαδικασία που περιγράφηκε επαναλαμβάνεται ανά δέκα δευτερόλεπτα, όσο ο χρήστης με ιδιότητα φοιτητή βρίσκεται εντός της συγκεκριμένης οθόνης, μέχρι την ώρα λήξης της συγκεκριμένης διάλεξης.

*12.LecturesStatistics activity*




Στην οθόνη του βασικού μενού που παρουσιάστηκε σε προηγούμενη ενότητα, επισημάνθηκε ότι το εικονίδιο του CardView με όνομα Lectures είναι διαφορετικό μεταξύ ενός χρήστη με φοιτητική ιδιότητα και ενός άλλου με ιδιότητα καθηγητή. Αυτό συμβαίνει διότι ο καθηγητής, πατώντας πάνω σε αυτό το CardView, μεταφέρεται σε μια άλλη οθόνη, με όνομα MyLecturesMenuProfessor. Στην οθόνη αυτή, μέσω δύο ακόμα CardViews, του δίνονται ισάριθμες επιλογές: η μία είναι να μεταβεί στο activity MyLectures, από όπου επιλέγοντας την προγραμματισμένη διάλεξη την οποία πρόκειται να πραγματοποιήσει, μεταφέρεται στην οθόνη LiveLecture με άνοιγμα της μπροστινής κάμερας, ώστε να προσομοιωθεί η διαδικασία της διδασκαλίας του μαθήματος εξ αποστάσεως. Η άλλη επιλογή αφορά στην απεικόνιση από τον καθηγητή στατιστικών που αφορούν σε συναισθήματα μαθητών τα οποία στάλθηκαν στη βάση δεδομένων στα πλαίσια ήδη πραγματοποιημένων διαλέξεων. Για την απεικόνισή τους, μεταφέρεται στο activity με όνομα LecturesStatistics.

Κατά την πρώτη εμφάνιση της οθόνης του, το μόνο ορατό στοιχείο είναι ένα Spinner, μέσω του οποίου ο καθηγητής επιλέγει τι είδους στατιστικά θέλει να δει, μεταξύ δύο κατηγοριών: α)κατανομή των πέντε βασικών συναισθημάτων εντός μιας συγκεκριμένης ολοκληρωμένης διάλεξης ή β)εμφάνιση του ποσοστού ενός συγκεκριμένου συναισθήματος επί του συνόλου των συναισθημάτων από την πρώτη μέχρι την τελευταία ολοκληρωμένη διάλεξη.

Κατά την εκτέλεση της μεθόδου onCreate(), το πρώτο Spinner γεμίζει με τα strings, σε κάθε ένα από τα οποία αναγράφεται και ο αντίστοιχος τύπος στατιστικών που θα προβληθούν ανά περίπτωση. Για το παραπάνω Spinner ορίζεται και ένα νέο αντικείμενο του interface AdapterView.OnItemSelectedListener, που κάνει override τη μέθοδο onItemSelected(), η οποία ενεργοποιείται κατά την επιλογή από τον χρήστη ενός στοιχείου του Spinner. Εντός της μεθόδου, ορίζεται ότι σε περίπτωση επιλογής από τον χρήστη κάποιου από τα στοιχεία του Spinner, θα κληθεί μια επόμενη μέθοδος.

Εντός της μεθόδου αυτής, προστίθεται το id του χρήστη με ιδιότητα καθηγητή στο URL που έχει δημιουργηθεί για αποστολή προς το web service και στη συνέχεια πραγματοποιείται ένα αίτημα προς την κατάλληλη μέθοδο της τάξης Controller της Spring Boot εφαρμογής. Εντός αυτής, δημιουργείται μια λίστα από αντικείμενα μιας τάξης της εφαρμογής αυτής με όνομα OnlineLectureWithCourseSection, κάθε ένα από τα οποία έχει την απαραίτητη πληροφορία για το αντίστοιχο τμήμα στο οποίο διδάσκει ο καθηγητής στο τρέχον εξάμηνο, και επιστρέφεται προς την client συσκευή. Αν η λίστα είναι άδεια, κάτι που θα σημαίνει ότι ο καθηγητής δεν διδάσκει κάποιο μάθημα κατά το τρέχον εξάμηνο, γίνεται η ενημέρωση με το ανάλογο μήνυμα μέσω Toast. Σε αντίθετη περίπτωση, εμφανίζεται στην οθόνη ένα δεύτερο Spinner, που θα περιέχει τα συγκεκριμένα τμήματα στα οποία εκείνος διδάσκει και καλείται μια ακόμη μέθοδος.

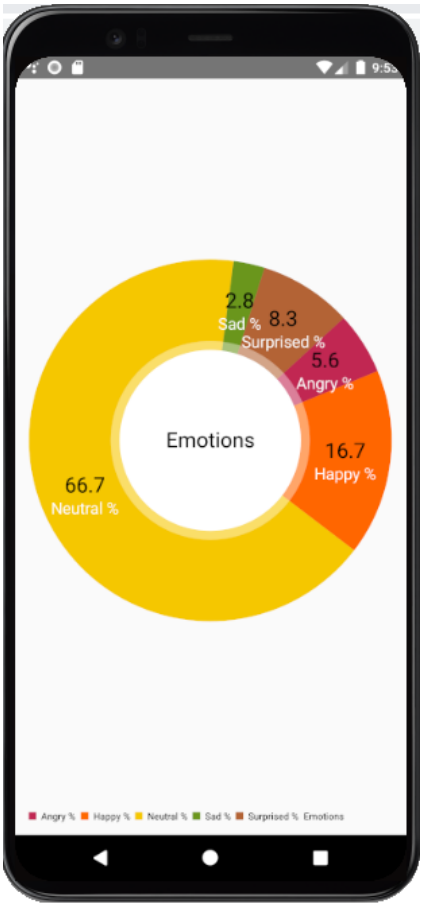
Με τον κώδικα αυτής της μεθόδου, λαμβάνονται από την παραπάνω λίστα μέσω των κατάλληλων getters το id του τμήματος και ο τίτλος του σχετιζόμενου μαθήματος. Αυτό είναι εφικτό καθώς η τάξη OnlineLectureWithCourseSection έχει δημιουργηθεί εντός και της Android εφαρμογής, με τις ίδιες ακριβώς ιδιότητες, getters και setters όπως και στην αντίστοιχη του Spring Boot. Με αυτόν τον τρόπο, η λίστα των αντικειμένων που φτάνουν σε μορφή JSON από το web service μετατρέπονται σε μια λίστα με αντικείμενα ίδιου τύπου με αυτά της λίστας που δημιουργήθηκε από την Spring Boot εφαρμογή πριν μετατραπεί σε μορφή JSON για την αποστολή της. Τα id και ο τίτλος συνενώνονται ώστε να σχηματιστεί ένα string που θα εμφανιστεί ως κείμενο του συγκεκριμένου στοιχείου του Spinner επιλογής τμήματος. Και για αυτόν τον Spinner ορίζεται ότι με την επιλογή ενός στοιχείου του, καλείται μια επιπλέον μέθοδος, με όρισμα το αντικείμενο της τάξης OnlineLectureWithCourseSection που επιλέχθηκε μέσω του παραπάνω Spinner.

Εντός της μεθόδου, λαμβάνονται μέσω getters οι τιμές των τεσσάρων ιδιοτήτων του αντικειμένου που δόθηκε ως όρισμα και οι οποίες αντιστοιχούν στις στήλες του πίνακα course_section, αποτελώντας το σύνθετο κλειδί του. Με αυτές ως τιμές, σχηματίζεται ένα αντικείμενο της τάξης JSONArray, το οποίο στέλνεται προς το web service ώστε να δημιουργηθεί μια λίστα με όλες οι ήδη πραγματοποιημένες διαλέξεις που αφορούν στο συγκεκριμένο μάθημα, καθώς και το πλήθος εμφάνισης του κάθε συναισθήματος στα πλαίσια της κάθε μιας από αυτές. Η παραπάνω λίστα επιστρέφεται στην client συσκευή και, αφού εξεταστεί ποιο είδος στατιστικών επιλέχθηκε να προβληθεί μέσω του πρώτου Spinner, εμφανίζεται ένα τρίτο Spinner.

Στην περίπτωση που επιλέχθηκε η εμφάνιση της κατανομής συναισθημάτων ανά διάλεξη, το τελευταίο Spinner γεμίζει με τις ημερομηνίες των ήδη πραγματοποιημένων διαλέξεων, ώστε ο καθηγητής να επιλέξει κάποια συγκεκριμένη και να προβάλει τα σχετικά στατιστικά. Αν επιλέχθηκε η προβολή της εξέλιξης ενός συγκεκριμένου συναισθήματος, το Spinner γεμίζει με τα strings που αντιστοιχούν στα πέντε συναισθήματα τα οποία αναγνωρίζονται από την εφαρμογή.

Όταν ο χρήστης επιλέξει και το στοιχείο που επιθυμεί στο τρίτο Spinner, εμφανίζεται ένα κουμπί για την προβολή των επιθυμητών στατιστικών. Με το πάτημά του, εξετάζεται πάλι ο τύπος στατιστικών που επιλέχθηκε στο πρώτο Spinner και πραγματοποιείται η μετάβαση στην κατάλληλη οθόνη.

*13. StatisticsPieChart activity*

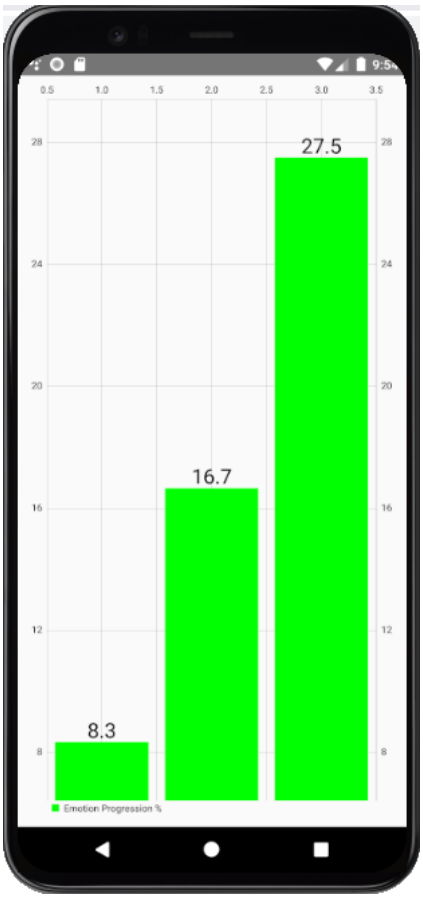



Στο εσωτερικό αυτής της οθόνης απεικονίζεται η κατανομή των συναισθημάτων στα πλαίσια της διάλεξης που επιλέχθηκε στην προηγούμενη οθόνη. Μέσω του κώδικα του activity LecturesStatistics, κατά τη μετάβαση στο συγκεκριμένο activity, με χρήση ενός αντικειμένου της τάξης Intent, δίνεται ως επιπλέον στοιχείο μέσω της μεθόδου με τίτλο .putExtra(), το αντικείμενο της τάξης OnlineLectureWithEmotionCount που επιλέχθηκε στον αντίστοιχο Spinner. Η τάξη αυτή έχει δημιουργηθεί τόσο στη SpringBoot εφαρμογή όσο και σε αυτήν που τρέχει στην Android συσκευή και διαθέτει ιδιότητες που αντιπροσωπεύουν όλες τις απαραίτητες πληροφορίες για την ποσοτικοποίηση των συναισθημάτων που προέκυψαν από όλους τους φοιτητές στα πλαίσια μιας διάλεξης.

Εντός της μεθόδου onCreate() του τρέχοντος activity, καλείται μια ακόμη μέθοδος, ή οποία λαμβάνει αυτό το αντικείμενο μέσω της μεθόδου getIntent().getSerializableExtra() και, μέσω των κατάλληλων getters, παίρνει και στη συνέχεια προσθέτει την ποσότητα όλων των συναισθημάτων που εμφανίστηκαν κατά τη συγκεκριμένη διάλεξη, αποθηκεύοντας το αποτέλεσμα σε μια ιδιότητα της τάξης.

Στη συνέχεια, μέσω του αντίστοιχου getter, λαμβάνεται η ποσότητα κάθε συγκεκριμένου συναισθήματος στα πλαίσια της διάλεξης και, αφού πολλαπλασιαστεί επί τον αριθμό 100, διαιρείται από τον αριθμό που αντιπροσωπεύει την ποσότητα όλων των συναισθημάτων συνολικά. Με αυτόν τον τρόπο βρίσκεται η ποσότητα επί τοις εκατό του συγκεκριμένου συναισθήματος εντός της συγκεκριμένης διάλεξης. Η διαδικασία ακολουθείται και για τα πέντε διαφορετικά συναισθήματα, τα ποσοστά των οποίων επί τοις εκατό μαζί με το αλφαριθμητικό που περιγράφει το συναίσθημα προστίθενται σε ένα γράφημα με μορφή πίτας, μέσω του οποίου απεικονίζονται στον καθηγητή.

*14. StatisticsBarChart activity*




Σε αυτό το activity απεικονίζεται η εξέλιξη του συναισθήματος που επιλέχθηκε μέσω του τρίτου Spinner της προηγούμενης οθόνης στο πλαίσιο των ήδη πραγματοποιημένων διαλέξεων του τμήματος διδασκαλίας το οποίο επιλέχθηκε μέσω του δεύτερου Spinner αυτής.

Το συναίσθημα του οποίου ο καθηγητής επέλεξε να δει την εξέλιξη καθώς και η λίστα με όλα τα αντικείμενα της τάξης OnlineLectureWithEmotionCount που διαθέτουν την πληροφορία για τα συναισθήματα κάθε πραγματοποιημένης διάλεξης του τμήματος διδασκαλίας που επιλέχθηκε, περνιούνται μέσω του κώδικα της τάξης LecturesStatistics ως παράμετροι κατά την μετάβαση στην συγκεκριμένη οθόνη, με τον τρόπο που περιγράφηκε κατά την ανάλυση του κώδικα του StatisticsPieChart activity.

Εντός της μεθόδου onCreate() της τάξης StatisticsBarChart, εξετάζεται διαδοχικά κάθε αντικείμενο της λίστας, που αντιστοιχεί σε μια πραγματοποιημένη διάλεξη. Το πλήθος του κάθε επιμέρους συναισθήματος προστίθενται σε αυτό των υπολοίπων και το τελικό αποτέλεσμα αποθηκεύεται σε μια τοπική μεταβλητή. Στη συνέχεια εξετάζεται ποιο συναίσθημα επιλέχθηκε για προβολή στην οθόνη LecturesStatistics και, βάσει αυτού, λαμβάνεται μέσω του αντίστοιχου getter το πλήθος των φορών τις οποίες εμφανίστηκε στην διάλεξη αυτή. Η τιμή που λήφθηκε μέσω του getter πολλαπλασιάζεται επί τον αριθμό 100 και διαιρείται από την ποσότητα όλων των συναισθημάτων εντός της διάλεξης. Οι επί μέρους ποσότητες επί τοις εκατό που προκύπτουν από την πραγματοποίηση των παραπάνω υπολογισμών στα αντικείμενα της λίστας προστίθεται σε ένα γράφημα ράβδων, το οποίο απεικονίζεται στην οθόνη της κινητής συσκευής του καθηγητή.

# 6 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στα πλαίσια αυτής της διπλωματικής εργασίας υλοποιήθηκε εξ ολοκλήρου με λογισμικό ανοικτού κώδικα μια εφαρμογή που θα μπορούσε να απευθύνεται στους φοιτητές και τους καθηγητές οποιοδήποτε πανεπιστημιακού ιδρύματος, για την πραγματοποίηση διαδικτυακών διαλέξεων.

Μέσω αυτής, αναδείχθηκε η δυνατότητα υλοποίησης και ένταξης custom λύσεων σχετικών με τη μηχανική μάθηση σε μια Android εφαρμογή, με την χρήση της βιβλιοθήκης Tensorflow για την εκπαίδευση ενός μοντέλου συνελικτικού νευρωνικού δικτύου και της βιβλιοθήκης Tensorflow Lite για την αξιοποίησή του σε συμπιεσμένη μορφή εντός της κινητής συσκευής. Επίσης, διερευνήθηκε περαιτέρω η λειτουργικότητα της διασύνδεσης μεταξύ ενός Android client και του Spring Boot web service, με τη δημιουργία τμημάτων κώδικα για την υλοποίηση κάποιων εξειδικευμένων σεναρίων λήψης και αποστολής δεδομένων, όπως και αυτή της διασύνδεσης μεταξύ του Spring Boot και μιας βάσης δεδομένων PostgreSQL.

Η προσθήκη της δυνατότητας περιοδικής ανάλυσης των συναισθημάτων των μαθητών κατά τη διάρκεια κάθε διάλεξης αποδείχθηκε κομβικής σημασίας για την εξασφάλιση της ποιότητας των διαλέξεων στην εξέλιξη του κάθε διδακτικού εξάμηνου. Διαπιστώθηκε ότι με την χρήση των διαγραμμάτων που δημιουργήθηκαν μέσω του κώδικα της διπλωματικής εργασίας μπορεί να εξαχθεί ένα σχεδόν άμεσο, οπτικοποιημένο συμπέρασμα για την απήχηση των διαλέξεων στους φοιτητές, τόσο μεμονωμένα, όσο και διαχρονικά, καθώς εξελίσσεται το εξάμηνο σπουδών. Αυτή η γνώση εκ μέρους του καθηγητή θα οδηγήσει πιθανότατα σε προσαρμογές του τρόπου ή και του περιεχομένου διδασκαλίας, οδηγώντας σε αύξηση της ποιότητας των διαδικτυακών διαλέξεων.

# 7 ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Για την εκπαίδευση του μοντέλου συνελικτικού νευρωνικού δικτύου το οποίο χρησιμοποιήθηκε για την αναγνώριση του συναισθήματος στα πρόσωπα των φοιτητών-χρηστών της εφαρμογής, έγινε χρήση ενός σετ δεδομένων μεγέθους 43.000 αρχείων. Στο μέλλον, θα μπορούσε να γίνει επανεκπαίδευση του μοντέλου με χρήση ενός μεγαλύτερου σετ δεδομένων, έτσι ώστε να αυξηθεί η ικανότητα αναγνώρισης των συναισθημάτων εντός της εικόνας ακόμη και αν δεν εκφράζονται με μεγάλη ένταση, και ταυτόχρονα να βελτιωθεί η γενίκευση του μοντέλου σε νέες φωτογραφίες.

Μια ακόμη επέκταση που θα μπορούσε να πραγματοποιηθεί είναι η αλλαγή των κατηγοριών ταξινόμησης του περιεχομένου της εικόνας. Στην παρούσα διπλωματική εργασία, οι κατηγορίες αφορούν σε πέντε από τα επτά βασικά συναισθήματα, εξαιρουμένων αυτών της αηδίας και του φόβου. Θα ήταν ενδιαφέρον η έκφραση του προσώπου εντός της εικόνας να ταξινομηθεί βάσει δύο παραμέτρων, της έντασης και της αποδοχής που την χαρακτηρίζουν. Από την διαδικασία αυτή θα προέκυπταν τέσσερεις ευρύτερες ομάδες, κάθε μια από τις οποίες θα περιλάμβανε τρεις κατηγορίες, βάσει του συνδυασμού των δύο παραπάνω παραμέτρων: α) αρνητική αποδοχή και υψηλή ένταση, που θα περιλάμβανε την ένταση, τον εκνευρισμό και τον θυμό, β) αρνητική αποδοχή και χαμηλή ένταση, που θα περιλάμβανε την πλήξη, την κούραση και τη θλίψη, γ) θετική αποδοχή και υψηλή ένταση, που θα περιλάμβανε την χαρά, την ευχαρίστηση και τον ενθουσιασμό, δ) θετική αποδοχή και χαμηλή ένταση, που θα περιλάμβανε την ηρεμία, τη χαλάρωση και την ικανοποίηση. Μέσω του διαχωρισμού των φωτογραφιών του σετ εκπαίδευσης βάσει των παραπάνω δώδεκα κατηγοριών και την εκπαίδευση του μοντέλου στην ταξινόμηση της φωτογραφίας εισόδου σε κάποια από αυτές, ο καθηγητής θα μπορούσε να αποκομίσει ακόμα πιο εξειδικευμένη πληροφορία για την απήχηση των διαδικτυακών διαλέξεών του στους φοιτητές.

Όσον αφορά στην αλληλεπίδραση του κώδικα της Android εφαρμογής με το web service, θα μπορούσε να προστεθεί η λειτουργικότητα για αμφίδρομη επικοινωνία μεταξύ του Spring Boot και των client συσκευών, μέσω της υλοποίησης του πρωτοκόλλου WebSocket. Κάτι τέτοιο θα καθιστούσε δυνατή την άμεση ειδοποίηση των φοιτητών για ένα νέο μήνυμα που στάλθηκε από κάποιον καθηγητή αλλά και την ολοκληρωμένη υλοποίηση μιας διαδικτυακής διάλεξης σε ζωντανή ροή.

# 8 ΒΙΒΛΙΟΓΡΑΦΙΑ

*A basic Introduction to Tensorflow Lite.* Ανακτήθηκε από: https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292

An *end-to-end open source machine learning platform.* Ανακτήθηκε από: https://www.tensorflow.org

*Android Studio*. Ανακτήθηκε από: https://developer.android.com/studio

Ciresan, D., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J.(2011). *Flexible, High Performance Convolutional Neural Networks for Image Classification*. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. 2, 1237–1242. Ανακτήθηκε από: https://people.idsia.ch/~juergen/ijcai2011.pdf

Collobert, R., & Weston, J. (2008). *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*. Proceedings of the 25th International Conference on Machine Learning, 160–167. doi:10.1145/1390156.1390177

*Deploy machine learning models on mobile and edge devices*. Ανακτήθηκε από: https://www.tensorflow.org/lite

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O'Reilly Media.

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009). *A Novel Connectionist System for Improved Unconstrained Handwriting Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5),855–868. doi: 10.1109/TPAMI.2008.137

Hamed, H. A., & Elnaz, J. H. (2017). *Guide to convolutional neural networks : a practical application to traffic-sign detection and classification*. Cham: Springer International Publishing. doi: 10.1007/978-3-319-57550-6

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks.* NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems. *Volume 1*, 1, 1097–1105. Ανακτήθηκε από: https://dl.acm.org/doi/10.5555/2999134.2999257

Lawrence, S., Giles, C. L., Tsoi, Ah Chung Tsoi, & Back, A. D. (1997). *Face Recognition: A Convolutional Neural Network Approach*. IEEE Transactions on Neural Networks, 8 (1), 98–113. doi: 10.1109/72.554195

LeCun, Y. *LeNet-5, convolutional neural networks.* Ανακτήθηκε από: http://yann.lecun.com/exdb/lenet/

LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). *Deep learning*. Nature, 521 (7553), 436–444. doi: 10.1038/nature14539

Matiz, S. & Barner, K. E. (2019). *Inductive conformal predictor for convolutional neural networks: Applications to active learning for image classification*. Pattern Recognition, 90, 172–182. doi: https://doi.org/10.1016/j.patcog.2019.01.035

Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). *Subject independent facial expression recognition with robust face detection using a convolutional neural network*. Neural Networks, 16 (5), 555–559. doi:10.1016/S0893-6080(03)00115-1

Mouton, C., Myburgh, J. C., & Davel, M. H. (2020). *Stride and Translation Invariance in CNNs*. Artificial Intelligence Research. Communications in Computer and Information Science. Cham: Springer International Publishing, 1342, 267–281. doi: 10.1007/978-3-030-66151-9_17

Mouton, C., Myburgh, J. C., & Davel, M. H. (2020). *Tracking Translation Invariance in CNNs*. Artificial Intelligence Research. Communications in Computer and Information Science. Cham:Springer International Publishing, 1342, 282–295. doi: 10.1007/978-3-030-66151-9_18

Ng, Y. A. (2011). *Machine Learning*. Coursera.

*PostgreSQL: The World's Most Advanced Open Source Relational Database*. Ανακτήθηκε από: https://www.postgresql.org/

Rosenblatt, F. (1958). *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, 65(6), 386–408. doi: 10.1037/h0042519

Rumelhart, D.E., & McClelland, J. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge: MIT Press. Ανακτήθηκε από: https://archive.org/details/paralleldistribu00rume

Scherer, D., Müller, A. C., & Behnke, S. (2010). *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition* . Artificial Neural Networks (ICANN), 20th International Conference, 92–101. Ανακτήθηκε από: http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf

*Spring Boot*. Ανακτήθηκε από: https://spring.io/projects/spring-boot

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.(2014). *Dropout: A Simple Way to Prevent Neural Networks from overfitting* . Journal of Machine Learning Research, 15 (1), 1929–1958. Ανακτήθηκε από: http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). *Going deeper with convolutions*. IEEE Conference on Computer Vision and Pattern Recognition, 1–9. doi: 10.1109/CVPR.2015.7298594

VanderBilt University (2018). *Building Cloud Services with the Java Spring Framework.* Coursera.