Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Ηλεκτρικων Βιομηχανικων Διαταξεων και Συστηματων Αποφασεων

Ανίχνευση fake news στα social media με χρήση συνελικτικών δικτύων γράφων

Διπλωματικη Εργασια

του

ΓΕΩΡΓΙΟΥ Θ. ΤΖΟΥΜΑΝΕΚΑ

Επιβλέπων: Δημήτριος Ασκούνης
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Ηλεκτρικων Βιομηχανικων Διαταξεων και Συστηματων Αποφασεων

Ανίχνευση fake news στα social media με χρήση συνελικτικών δικτύων γράφων

Διπλωματικη Εργασια

του

ΓΕΩΡΓΙΟΥ Θ. ΤΖΟΥΜΑΝΕΚΑ

Επιβλέπων: Δημήτριος Ασκούνης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Φεβρουαρίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Δημήτριος Ασκούνης	Ιωάννης Ψαρράς	Ευάγγελος Μαρινάκης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Ηλεκτρικων Βιομηχανικων Διαταξεων και Συστηματων Αποφασεων

(Υπογραφή)

.....
Γεώργιος Τζουμανέκας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η χρήση των μέσων κοινωνικής δικτύωσης είναι ένα αναπόσπαστο κομμάτι της καθημερινότητας στο σύγχρονο κόσμο. Πλατφόρμες όπως το Twitter, το Facebook και το Instagram συγκεντρώνουν καθημερινά εκατομμύρια χρήστες. Το Twitter, ειδικότερα, έχει αναδειχθεί ως μια ισχυρή πλατφόρμα για συνομιλίες σε πραγματικό χρόνο και διάδοση ειδήσεων. Με το όριο 280 χαρακτήρων, το Twitter ενθαρρύνει την κοινή χρήση γρήγορων ενημερώσεων και απόψεων σε μοντέρνα θέματα. Έχει διαδραματίσει σημαντικό ρόλο στη διαμόρφωση του δημόσιου λόγου, στην δημιουργία κοινωνικών κινημάτων και στη σύνδεση ανθρώπων από όλον τον κόσμο. Αυτή η ταχεία ανάπτυξη έχει ενθαρρύνει την εμφάνιση των bots, που είναι αυτοματοποιημένοι λογαριασμοί σχεδιασμένοι να μιμούνται την ανθρώπινη συμπεριφορά. Μπορούν να παραπληροφορήσουν, να προωθήσουν ιδεολογίες και να ενισχύσουν τις ψευδείς ειδήσεις με ανησυχητικό ρυθμό. Όχι μόνο υπονομεύουν την εμπιστοσύνη του κοινού στα μέσα ενημέρωσης και τους θεσμούς, αλλά μπορούν επίσης να έχουν συνέπειες στον πραγματικό κόσμο, από τον επηρεασμό εκλογών έως την πρόκληση κοινωνικής αναστάτωσης. Ο αποτελεσματικός εντοπισμός αυτών των λογαριασμών είναι μια μεγάλη πρόκληση στον σύγχρονο κόσμο και έχει αποτελέσει θέμα πολυάριθμων μελετών.

Σε αυτή τη διπλωματική, παρουσιάζουμε την εργασία μας για ένα αποτελεσματικό μοντέλο ανίχνευσης bot. Χρησιμοποιούμε σχεσιακά συνελικτικά νευρωνικά δίκτυα γράφων που εκμεταλλεύονται τα δεδομένα χρήστη και τεχνικές μετάδοσης μηνυμάτων για να προσδώσουν ετικέτες σε κάθε κόμβο. Χρησιμοποιήσαμε επίσης την τεχνική της Αναζήτησης Αρχιτεκτονικής Νευρωνικών Δικτύων που αναζητά την αρχιτεκτονική με την υψηλότερη ακρίβεια, χρησιμοποιώντας έναν εξελικτικό αλγόριθμο.

Στο τέλος, παρουσιάζουμε τα αποτελέσματα και καταλήγουμε στο συμπέρασμα ότι η μέθοδός μας μπορεί να ανταγωνιστεί προηγούμενα μοντέλα στην ανίχνευση bot, υπογραμμίζοντας τα πλεονεκτήματα της χρήσης και των δύο τεχνικών.

Λέξεις Κλειδιά

Μέσα Κοινωνικής δικτύωσης, Bots, Twitter, Νευρωνικά Δίκτυα Γράφων, Πρωτόκολλο Μετάδοσης Μηνυμάτων, Αναζήτηση Αρχιτεκτονικής

Abstract

Social media usage is at an all-time high in the modern world. Platforms like Twitter, Facebook, and Instagram congregate daily millions of users. Twitter, in particular, has emerged as a powerful platform for real-time conversation and news dissemination. With a concise character limit, Twitter encourages sharing quick updates and opinions on trending topics. It has played an important role in shaping public discourse, mobilizing social movements, and connecting people across the globe. This rapid growth has encouraged the emergence of bots, which are automated programs designed to mimic human behavior. They can spread misinformation, promote ideologies, and amplify fake news stories at an alarming rate. Not only do they undermine public trust in media and institutions, but they can also have real-world consequences, from influencing elections to provoking social disruption. The effective detection of these accounts is a big challenge in the modern world and has attracted numerous studies.

In this thesis, we present our work on an efficient bot detection model. We use relational graph convolutional neural networks that exploit user data and message-passing techniques to assign labels to each node. We also used a technique called Neural Architecture Search that searches for the architecture with the highest accuracy, using an evolutionary algorithm.

In the end, we present the results and conclude that our method can compete with other state-of-the-art models in bot detection, underlining the advantages of using both techniques in this task.

Keywords

Social Media, Bots, Graph Neural Networks, Message Passing Protocol, Neural Architecture Search

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Δημήτρη Ασκούνη για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα πολύ σύγχρονο θέμα. Οφείλω ένα πολύ θερμό ευχαριστώ στους υποψήφιους διδάκτορες Λουκά Ηλία και Μιχάλη Χατζηαναστάση για την συνεχή καθοδήγηση τους και όλες τις πολύτιμες συμβουλές τους, που έκαναν δυνατή την εκπόνηση της παρούσας διπλωματικής.

Κυρίως, θα ήθελα να ευχαριστήσω την οικογένεια μου, και ιδιαίτερα τους γονείς μου και τον αδερφό μου, για την αγάπη τους και την στήριξη τους σε κάθε βήμα που έχω επιλέξει και επιλέγω να κάνω.

Τέλος, θα ήθελα να ευχαριστήσω από καρδιάς τους φίλους μου, τόσο της σχολικής μου ζωής όσο και της φοιτητικής, για όλες τις αμέτρητες όμορφες στιγμές που έχουμε περάσει μαζί και την βοήθειά τους να ξεπερνάω κάθε δυσκολία.

Αθήνα, Φεβρουάριος 2024

Γεώργιος Τζουμανέκας

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εκτεταμένη Περίληψη

1.1 Εισαγωγή

1.1.1 Μέσα κοινωνικής δικτύωσης

Το Six Degrees ιδρύθηκε από τον Andrew Weinreich το 1997 και θεωρείται ο πρώτος ιστότοπος κοινωνικής δικτύωσης [34]. Προσέφερε στους χρήστες τη δυνατότητα να δημιουργούν προφίλ, να κάνουν φίλους και να στέλνουν μηνύματα. Έφτασε στο αποκορύφωμά του στα τέλη της δεκαετίας του 1990, με 3,5 εκατομμύρια χρήστες, αλλά το 2001 αδυνατώντας να δημιουργήσει έσοδα για να διατηρήσει την ανάπτυξή του, σταμάτησε τη λειτουργία του. Παρά την αναπόφευκτη πτώση του, θεωρείται ο πρωτοπόρος που άνοιξε το δρόμο για τις πλατφόρμες κοινωνικών μέσων, όπως τις γνωρίζουμε σήμερα. Εξ ορισμού, τα μέσα κοινωνικής δικτύωσης αναφέρονται σε πλατφόρμες και εφαρμογές που επιτρέπουν στους χρήστες να δημιουργούν, να μοιράζονται και να ανταλλάσσουν περιεχόμενο. Το περιεχόμενο μέσων κοινωνικής δικτύωσης μπορεί να είναι κείμενο, φωτογραφίες, βίντεο, GIF, ηχητικά μηνύματα κ.λπ. Κάποιος μπορεί να χρησιμοποιήσει τα μέσα κοινωνικής δικτύωσης για διάφορους λόγους, από την επικοινωνία με άλλους χρήστες με παρόμοια ενδιαφέροντα έως την ενημέρωση για τρέχοντα γεγονότα παγκοσμίως. Το 2023 εκτιμάται ότι υπάρχουν περίπου 4,9 δισεκατομμύρια χρήστες μέσων κοινωνικής δικτύωσης [35], ποσοστό πάνω από το 60% του παγκόσμιου πληθυσμού σε περισσότερες από 100 πλατφόρμες μέσων κοινωνικής δικτύωσης. Οι πιο γνωστές πλατφόρμες είναι το Facebook, το Instagram, το Twitter, το WhatsApp, το YouTube και το TikTok, συγκεντρώνοντας τη συντριπτική πλειοψηφία των χρηστών των μέσων κοινωνικής δικτύωσης.

Το Twitter είναι μια δημοφιλής πλατφόρμα μέσων κοινωνικής δικτύωσης που ξεκίνησε το 2006 [36] από τους Jack Dorsey, Biz Stone και Evan Williams και έκτοτε έχει εξελιχθεί σε μια παγκόσμια πλατφόρμα με 330 εκατομμύρια ενεργούς χρήστες. Οι χρήστες μπορούν να δημοσιεύουν και να αλληλεπιδρούν με κείμενα, εικόνες, βίντεο και συνδέσμους, που αναφέρονται ως «tweets», καθιστώντας το μια ιδανική πλατφόρμα για την έκφραση σκέψεων, την κοινή χρήση ειδήσεων και τη συμμετοχή σε συζητήσεις. Αρχικά, τα tweets περιορίζονταν στους 140 χαρακτήρες, μέχρι τον Νοέμβριο του 2017 όταν επεκτάθηκαν στους 280 χαρακτήρες.

Το Twitter περιστρέφεται γύρω από την έννοια της ακολούθησης χρηστών. Ένας χρήστης μπορεί να ακολουθήσει λογαριασμούς που ενδιαφέρεται να δει τα tweets τους στο χρονολόγιο

του ("following") και αντιστρόφως έχει "followers" που βλέπουν τα tweets του. Οι χρήστες μπορούν να απαντήσουν και να αναφέρουν άλλους λογαριασμούς με τον χαρακτήρα "@" ακολουθούμενο από το όνομα του λογαριασμού. Το Twitter ενσωματώνει επίσης τη χρήση "hashtags", τα οποία είναι λέξεις-κλειδιά ή φράσεις που προηγούνται του χαρακτήρα "#" και κατηγοριοποιούν τα tweets, καθιστώντας πιο άνετη την αναζήτηση συγκεκριμένων θεμάτων. Οι χρήστες μπορούν επίσης να αναζητήσουν tweets με βάση το περιεχόμενό τους χρησιμοποιώντας λέξεις που περιλαμβάνονται, με τα αποτελέσματα να παρουσιάζονται σε συνδυασμό χρονολογικής σειράς και σχετικότητας.

Το Twitter με αυτές τις δυνατότητες έχει καθιερωθεί ως ένα ισχυρό εργαλείο για ενημερώσεις ειδήσεων σε πραγματικό χρόνο, δημόσιο διάλογο και κοινωνικά κινήματα και συνεχίζει να εξελίσσεται και να βελτιώνει την εμπειρία των χρηστών.

1.1.2 Ψευδείς ειδήσεις και bots

Οι ψευδείς ειδήσεις είναι αναληθείς πληροφορίες, ιστορίες ή φάρσες που δημιουργούνται για να παραπληροφορήσουν ή να εξαπατήσουν τους αναγνώστες [37]. Επιδιώκουν να παραπλανήσουν τους χρήστες προσομοιάζοντας αξιόπιστους ιστότοπους, χρησιμοποιώντας παρόμοια ονόματα και διευθύνσεις ιστού με αξιόπιστους ειδησεογραφικούς οργανισμούς.

Τα bots των μέσων κοινωνικής δικτύωσης είναι λογαριασμοί που αυτοματοποιούν τις αλληλεπιδράσεις σε πλατφόρμες μέσων κοινωνικής δικτύωσης, συχνά μιμούμενοι την ανθρώπινη συμπεριφορά. Μπορούν να προγραμματιστούν για να εκτελούν διάφορες εργασίες, όπως αυτόματη δημοσίευση περιεχομένου, επισήμανση "μου αρέσει", κοινή χρήση, παρακολούθηση ή σχολιασμό αναρτήσεων. Μερικά μπορούν ακόμη και να προγραμματιστούν να συμμετέχουν σε συνομιλίες ώστε να προωθήσουν συγκεκριμένες ατζέντες.

Οι ψευδείς ειδήσεις και τα bots έχουν σημαντικές συνέπειες στον πραγματικό κόσμο. Καθ' όλη τη διάρκεια της πανδημίας του COVID-19, η παραπληροφόρηση σχετικά με τον ιό και τα εμβόλια εξαπλώθηκε γρήγορα στα μέσα κοινωνικής δικτύωσης, οδηγώντας σε πανικό ή και αντίσταση στα μέτρα δημόσιας υγείας. Το 2016 ψευδείς ειδήσεις διαδόθηκαν ευρέως κατά τη διάρκεια της εκστρατείας για τις προεδρικές εκλογές των ΗΠΑ, με στόχο να επηρεάσουν την κοινή γνώμη για να επηρεάσουν τους ψηφοφόρους. Τα bots μπορούν να χρησιμοποιηθούν για να παρενοχλήσουν άτομα ή να στοχεύσουν συγκεκριμένες ομάδες, συμμετέχοντας σε συντονισμένες καμπάνιες. Μπορούν επίσης να δημιουργήσουν και να διαδώσουν περιεχόμενο ανεπιθύμητης αλληλογραφίας, κακόβουλους συνδέσμους και λογισμικό, θέτοντας σε κίνδυνο την ασφάλεια και το απόρρητο των χρηστών.

1.1.3 Συνεισφορά διπλωματικής

Η ανάγκη εντοπισμού των bot για την αναστολή τους είναι αρκετά άμεση, αξιολογώντας τους κινδύνους της ανεξέλεγκτης παρουσίας τους στα μέσα κοινωνικής δικτύωσης. Αυτός είναι ο λόγος που πολλές πλατφόρμες έχουν εφαρμόσει αλγόριθμους για τον εντοπισμό αυτών των λογαριασμών. Ωστόσο, τα bots προσαρμόζουν συνεχώς τις λειτουργίες τους για να προσομοιώσουν γνήσιους χρήστες, καθιστώντας τον εντοπισμό τους δυσκολότερο. Ένα καινοτόμο μοντέλο για τον εντοπισμό ρομπότ που επιλέξαμε να βελτιστοποιήσουμε είναι το BotRGCN [38], που αποτελεί συντομογραφία για το Bot detection with Relational Graph

Convolutional Networks. Θα ξεπεράσουμε τον περιορισμό απόδοσης λόγω της στατικής δομής των νευρωνικών μοντέλων με την τεχνική Αναζήτησης Αρχιτεκτονικής Νευρωνικών Δικτύων (Neural Architecture Search). Για τα πειράματα μας χρησιμοποιήσαμε το σύνολο δεδομένων Twibot-20 [39], που είναι ένα ολοκληρωμένο σύνολο δεδομένων της ¨σφαίρας¨ του Twitter αντιπροσωπευτικό ενός συνόλου πραγματικών χρηστών. Αποτελείται από τέσσερις διαφορετικές περιοχές: την πολιτική, τις επιχειρήσεις, την ψυχαγωγία και τον αθλητισμό.

1.1.4 Δομή διπλωματικής

Στο Κεφάλαιο 1 παρουσιάζεται μια εκτενής περίληψη της διπλωματικής εξ ολοκλήρου στα Ελληνικά ακολουθώντας ίδια δομή.

Στο Κεφάλαιο 2, δίνεται μια σύντομη εισαγωγή σχετικά με την προέλευση των μέσων κοινωνικής δικτύωσης και μια περιγραφή των ζητημάτων που προκαλούνται από τις ψευδείς ειδήσεις και τους λογαριασμούς bot, καθώς και η συνεισφορά και η δομή της τρέχουσας διπλωματικής.

Στο Κεφάλαιο 3, εισάγουμε κάποιες βασικές έννοιες για τη θεωρητική κατανόηση της διπλωματικής. Παρουσιάζονται κάποιοι ορισμοί και επεξηγήσεις γύρω από τα γραφήματα, αλγόριθμους, μοντέλα και εργαλεία μηχανικής μάθησης και, προκειμένου να κατανοηθούν ζωτικά μέρη του πειράματος.

Στο Κεφάλαιο 4, παρουσιάζουμε σχετικές έρευνες στον τομέα της ανίχνευσης των bot και των ψευδών ειδήσεων, καθώς και μελέτες για τη Αναζήτηση Αρχιτεκτονικής Νευρωνικών Δικτύων σε διάφορες περιπτώσεις χρήσης.

Στο Κεφάλαιο 5, δίνουμε κάποιες πληροφορίες για το σύνολο των δεδομένων που χρησιμοποιούμε και εκτελούμε έναν έλεγχο για να διασφαλίσουμε την καταλληλότητα του.

Στο Κεφάλαιο 6, παρουσιάζουμε το μοντέλο μας. Περιγράφουμε την επεξεργασία των δεδομένων, τη χρήση των συνελικτικών δικτύων γράφων και τη διαδικασία της Αναζήτησης Αρχιτεκτονικής.

Στο Κεφάλαιο 7, παρουσιάζουμε τα αποτελέσματα των πειραμάτων μας και τα συγκρίνουμε με άλλες μεθοδολογίες αιχμής. Τέλος, πραγματοποιούμε το ablation study, για να αποδείξουμε την ακεραιότητα της μεθόδου μας.

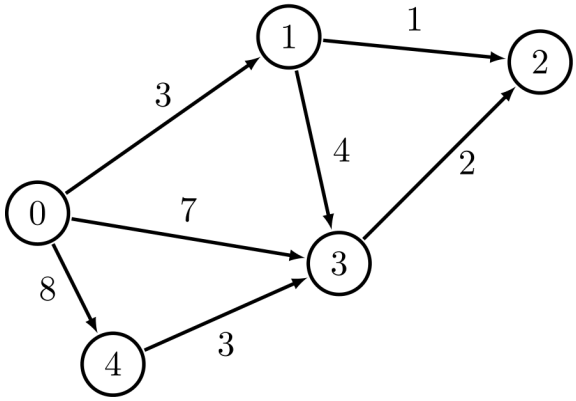
Στο Κεφάλαιο 8, ολοκληρώνουμε αυτή τη μελέτη παρουσιάζοντας τα τελικά συμπεράσματα της εργασίας μας, τους περιορισμούς και τις προτάσεις μας για μελλοντική έρευνα.

1.2 Θεωρητικό μέρος

1.2.1 Γράφοι: δομή και εφαρμογές

Ένας γράφος ορίζεται ως G = V, E όπου V είναι το σύνολο των κορυφών και E είναι το σύνολο των ακμών. Είναι μη γραμμικές δομές δεδομένων που χρησιμοποιούνται για τη μοντελοποίηση σχέσεων μεταξύ στοιχείων. Υπάρχουν πολλοί τύποι γράφων, με τις πιο κοινές παραλλαγές να είναι κατευθυνόμενος ή μη κατευθυνόμενος και με βάρη ή χωρίς. Ένας μη κατευθυνόμενος γράφος είναι τύπος γράφου όπου οι ακμές δεν έχουν συγκεκριμένη κατεύθυνση και οι συνδέσεις μεταξύ των κόμβων είναι αμφίδρομες. Σε έναν κατευθυνόμενο γράφο όλες οι ακμές έχουν μια συγκεκριμένη κατεύθυνση. Ένας γράφος με βάρη είναι

ένας τύπος γράφου που κάθε ακμή έχει μια αριθμητική τιμή που ονομάζεται βάρος. Αυτά τα βάρη μπορούν να αντιπροσωπεύουν μετρήσεις όπως η απόσταση μεταξύ των κόμβων, το κόστος κ.λπ. Αντίθετα σε ένα γράφο χωρίς βάρη όλες οι ακμές έχουν το ίδιο προεπιλεγμένο βάρος ή απλώς κανένα βάρος και μόνο η σύνδεση μεταξύ των κορυφών είναι σημαντική.



Εικόνα 1.1: *Παράδειγμα κατευθυνόμενου γράφου με βάρη. Πηγή [1]*

Οι γράφοι είναι μια ευέλικτη και εξαιρετικά χρήσιμη δομή δεδομένων σε πολλές εφαρμογές του πραγματικού κόσμου. Μερικές από αυτές θα παρουσιαστούν παρακάτω, για να καταλάβουμε τη σημασία της μελέτης τους και τις δυνατότητες στο πρόβλημά μας.

Οι αλγόριθμοι στους γράφους χρησιμοποιούνται ευρέως σε κοινωνικά δίκτυα όπως το Facebook, το Twitter και το ενΛινκεδΙν. Χρησιμοποιούνται για να προτείνουν συνδέσεις στους χρήστες, να βρίσκουν κοινότητες και να προτείνουν περιεχόμενο που θα ενδιέφερε έναν χρήστη.

Πολλά συστήματα συστάσεων, όπως αυτά που χρησιμοποιούνται από το Netflix και το Amazon, χρησιμοποιούν αλγόριθμους με γράφους για την ανάλυση των αλληλεπιδράσεων και των προτιμήσεων των χρηστών. Με βάση παρόμοια συμπεριφορά χρήστη και ομοιότητες αντικειμένων, μπορούν να προτείνουν ταινίες και προϊόντα.

Το PageRank είναι ένας αλγόριθμος βασισμένος σε γράφους που αναπτύχθηκε από την Google και χρησιμοποιείται για την κατάταξη ιστοσελίδων με βάση τη σημασία τους. Η σημασία μιας ιστοσελίδας καθορίζεται από τον αριθμό και την ποιότητα των συνδέσμων που οδηγούν σε αυτήν. Αποτελεί το θεμέλιο της μηχανής αναζήτησης της Google.

Οι αλγόριθμοι με γράφους χρησιμοποιούνται σε εφαρμογές χαρτογράφησης και πλοήγησης GPS για την εύρεση της συντομότερης διαδρομής μεταξύ τοποθεσιών. Μπορούν επίσης να προσαρμοστούν με διάφορους περιορισμούς όπως η κυκλοφορία και οι συνθήκες του δρόμου. Είναι επίσης πολύ χρήσιμα για τον καθορισμό των πιο αποτελεσματικών διαδρομών στα δίκτυα μεταφορών για τη βελτιστοποίηση της ροής αγαθών, δεδομένων, πληροφοριών κ.λπ.

1.2.2 Κλάδοι της μηχανικής μάθησης

Η Μηχανική Εκμάθηση είναι το πεδίο Τεχνητής Νοημοσύνης που επιτρέπει στα συστήματα να μαθαίνουν από δεδομένα, ώστε να μπορούν να κάνουν προβλέψεις μόνο με βάση αυτά. Η Μηχανική Μάθηση είναι επικείμενη σε πολλές σύγχρονες εφαρμογές του πραγματικού

κόσμου, όπως το Διαδίκτυο των πραγμάτων (Internet of Things: IoT), ασφάλεια στον κυβερνοχώρο, τα συστήματα συστάσεων, η υγειονομική περίθαλψη κ.λπ.

Η μάθηση με επίβλεψη περιλαμβάνει τη χρήση δεδομένων με ετικέτα για την εκπαίδευση αλγορίθμων, ώστε να μπορούν έπειτα να ταξινομούν με ακρίβεια τα δεδομένα χωρίς ετικέτα, με βάση μόνο τα χαρακτηριστικά τους. Χωρίζεται σε δύο βασικούς τύπους προβλημάτων: ταξινόμηση και παλινδρόμηση. Στα προβλήματα ταξινόμησης, η έξοδος είναι κατηγορική ή διακριτή. Ορισμένες υλοποιήσεις περιλαμβάνουν: τον εντοπισμό ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου και την ταξινόμηση εικόνων που περιέχουν ένα αντικείμενο. Στα προβλήματα παλινδρόμησης, η έξοδος είναι συνεχής ή αριθμητική. Περιλαμβάνει την πρόβλεψη μιας τιμής ή ποσότητας με βάση τα δεδομένα εισόδου. Περιπτώσεις προβλημάτων παλινδρόμησης περιλαμβάνουν: την πρόβλεψη των τιμών των κατοικιών με βάση διάφορους παράγοντες και την εκτίμηση του όγκου πωλήσεων με βάση τις διαφημιστικές δαπάνες.

Η μάθηση χωρίς επίβλεψη χρησιμοποιεί δεδομένα χωρίς ετικέτα για να κάνει προβλέψεις. Ο κύριος στόχος των αλγορίθμων μάθησης χωρίς επίβλεψη είναι να βρουν ομάδες χαρακτηριστικών που ακολουθούν ένα παρόμοιο μοτίβο ομοιοτήτων και διαφορών.

Η ημι-εποπτευόμενη μάθηση συνδυάζει δεδομένα με ετικέτα και χωρίς ετικέτα κατά τη διάρκεια της εκπαίδευσης. Αυτή η μέθοδος χρησιμοποιεί αρχικά αλγόριθμους εκμάθησης χωρίς επίβλεψη για να ομαδοποιήσει παρόμοια δεδομένα και στη συνέχεια δίνει ετικέτες στα δεδομένα που δεν είχαν προηγουμένως επισημανθεί.

Η ενισχυτική μάθηση είναι μια τεχνική μηχανικής μάθησης όπου ένας πράκτορας βασίζεται στο περιβάλλον για να αναλάβει δράση. Αυτή η τεχνική δεν χρησιμοποιεί δεδομένα με ετικέτα, αλλά χρησιμοποιεί μια προσέγγιση δοκιμής και σφάλματος με μια διαδικασία που βασίζεται στην ανάδραση.

1.2.3 Νευρωνικά Δίκτυα Γράφων (GNNs)

Αρχικά θα κατηγοριοποιήσουμε τα προβλήματα που μπορούμε να επιλύσουμε με την χρήση γράφων ανάλογα με το επίπεδο τους:

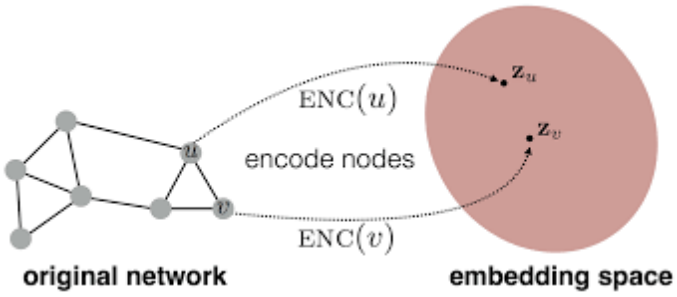
- Ο πρώτος τύπος εργασιών Μηχανικής Μάθησης που μπορούμε να λύσουμε με δομές γραφημάτων είναι οι **εργασίες σε επίπεδο γραφήματος**. Σε αυτές τις εργασίες, στόχος είναι να προβλέψουμε μια ιδιότητα για ολόκληρο το γράφημα. Παραδείγματα που εμπίπτουν σε αυτή την κατηγορία είναι η ταξινόμηση ενός γραφήματος, η παλινδρόμηση γραφήματος, που είναι η πρόβλεψη συνεχών τιμών για ένα γράφημα και η δημιουργία γραφήματος με καθορισμένες ιδιότητες. Ένα παράδειγμα εργασίας σε επίπεδο γραφήματος είναι η Άνίχνευση Κοινοτήτων σε Κοινωνικό Δίκτυο΄. Δοσμένου ενός γραφήματος με κόμβους που αντιπροσωπεύουν τους χρήστες και ακμές που αντιπροσωπεύουν τις σχέσεις μεταξύ των χρηστών, ο στόχος είναι να κατηγοριοποιηθούν οι χρήστες σε κατηγορίες, όπως ¨οικογένεια¨, ¨φίλοι¨, ¨συνάδελφοι¨, κ.λπ., σύμφωνα με παρόμοια μοτίβα στις αλληλεπιδράσεις τους.

- Ο επόμενος τύπος είναι οι **εργασίες σε επίπεδο κόμβου** που ασχολούνται με την πρόβλεψη της ταυτότητας κάθε κόμβου μέσα στο γράφημα. Αυτές οι εργασίες περιλαμβάνουν ταξινόμηση κόμβων, παλινδρόμηση κόμβων, που προβλέπονται συνεχείς

τιμές για τους κόμβους και ομαδοποίηση κόμβων σε συστάδες ή κοινότητες. Ένα πραγματικό παράδειγμα μιας εργασίας σε επίπεδο κόμβου είναι η Ταξινόμηση Εγγράφων σε ένα Δίκτυο Παραπομπών. Κάθε κόμβος σε αυτόν το γράφο αντιπροσωπεύει μια ερευνητική εργασία με κάποια χαρακτηριστικά όπως η περίληψη, οι λέξεις-κλειδιά και οι συγγραφείς, ενώ οι ακμές αντιπροσωπεύουν τις αναφορές μεταξύ των δημοσιεύσεων. Ο στόχος είναι να ταξινομηθεί κάθε ερευνητική εργασία σε κατηγορίες όπως ¨βιολογία¨, ¨επιστήμη υπολογιστών¨, ¨φυσική¨ κ.λπ.

- Ο τελευταίος τύπος είναι οι **εργασίες σε επίπεδο ακμής** που περιλαμβάνουν τη λήψη αποφάσεων στις ακμές, που αντιπροσωπεύουν τις σχέσεις μεταξύ των κόμβων. Τέτοιες εργασίες περιλαμβάνουν την πρόβλεψη συνδέσμων, όπου προβλέπεται εάν υπάρχει μια ακμή μεταξύ δύο κόμβων, την ταξινόμηση ακμών, όπου ετικέτες αντιστοιχίζονται σε ακμές και την παλινδρόμηση ακμών, όπου προβλέπονται συνεχείς τιμές για τις ακμές. Οι προβλέψεις των προβλημάτων μπορεί επομένως να είναι πολύ πιο κατατοπιστικές από τις απλές προβλέψεις συνδέσμων. Για παράδειγμα, μπορεί να είναι λιγότερο διαφωτιστικό ότι δύο χρήστες είναι κοινοί φίλοι από το ότι δύο χρήστες στέλνουν μηνύματα μεταξύ τους τακτικά. Ένα πραγματικό παράδειγμα μιας εργασίας σε επίπεδο ακμής είναι η Πρόβλεψη Φιλίας στα Κοινωνικά Δίκτυα. Οι κόμβοι του γράφου θα μπορούσαν να αντιπροσωπεύουν τους χρήστες και οι ακμές θα μπορούσαν να αντιπροσωπεύουν σχέσεις και αλληλεπιδράσεις, όπως μηνύματα, φιλίες, ανάρτηση κοινών δημοσιεύσεων κ.λπ. Η εργασία είναι να προβλέψουμε εάν δύο χρήστες που δεν είναι ακόμη συνδεδεμένοι θα μπορούσαν να συνδεθούν.

Τα **διανύσματα κόμβων (node embeddings)** είναι διανυσματικές αναπαραστάσεις μικρών διαστάσεων που καταγράφουν τις δομικές και σχεσιακές πληροφορίες των κόμβων σε ένα γράφο. Όπως φαίνεται στο παρακάτω σχήμα, ο στόχος είναι να βρεθεί ένας χώρος ενσωμάτωσης, όπου οι γεωμετρικές αναπαραστάσεις z_u και z_v θα αντιστοιχούν στους κόμβους των κόμβων u και του γραφήματος αντίστοιχα.



Εικόνα 1.2: *Διανύσματα κόμβων (Node embeddings). Πηγή [2]*

Τα διανύσματα χαρακτηριστικών χρησιμοποιούνται σε μια ποικιλία εργασιών μηχανικής μάθησης. Για παράδειγμα, στην Επεξεργασία Φυσικής Γλώσσας χρησιμοποιούνται διανύσματα λέξεων, οι οποίες είναι αναπαραστάσεις λέξεων ως διανύσματα, έτσι ώστε λέξεις με παρόμοια σημασία να βρίσκονται πιο κοντά στον χώρο αναπαράστασης. Με τον ίδιο τρόπο, οι κόμβοι ενός γραφήματος που βρίσκονται κοντά ή ανήκουν στην ίδια γειτονιά, αναμένεται να έχουν παρόμοια διανύσματα κόμβων. Τα διανύσματα κόμβων είναι επίσης πολύ χρήσιμα

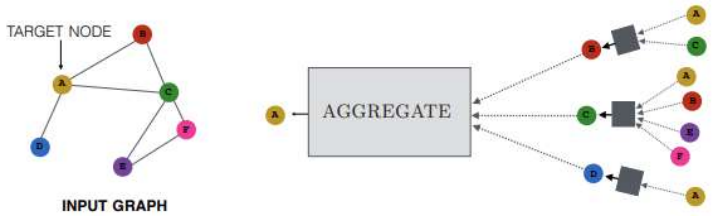
για τη μείωση των διαστάσεων, καθώς η επεξεργασία ακατέργαστων χαρακτηριστικών κόμβων μπορεί να είναι υπολογιστικά ακριβή. Υπάρχουν πολλοί τρόποι δημιουργίας διανυσμάτων κόμβων, ο καθένας με τη δική του προσέγγιση.

Δύο βασικές έννοιες στα νευρωνικά δίκτυα γράφων είναι η **αμεταβλητότητα μετάθεσης (permutation invariance)** και η **ισοδυναμία μετάθεσης (permutation equivariance)**. Η **αμεταβλητότητα μετάθεσης** αναφέρεται στην ιδιότητα ότι το GNN πρέπει να παράγει συνεπή αποτελέσματα ανεξάρτητα από τη σειρά με την οποία επεξεργάζονται οι κόμβοι του γράφου. Τα δεδομένα του γράφου μπορούν να απεικονιστούν με έναν πίνακα γειτνίασης, οπότε η σειρά των κόμβων δεν είναι σταθερή. Η **ισοδυναμία μετάθεσης** σχετίζεται με τον τρόπο με τον οποίο ένα GNN επεξεργάζεται τυχόν μετασχηματισμούς στα δεδομένα. Εάν η είσοδος υφίσταται μετασχηματισμούς, όπως περιστροφές, αλλαγές στο μέγεθος κ.λπ., η έξοδος θα πρέπει να εμφανίζει παρόμοιους μετασχηματισμούς. Ένας επίσημος ορισμός αυτών των εννοιών παρέχεται από τους Meltzer et al.[40]:

Αμεταβλητότητα μετάθεσης: Έστω P_n το σύνολο όλων των έγκυρων πινάκων μετάθεσης τάξης n, τότε μια συνάρτηση f είναι αμετάβλητη στη μετάθεση γραμμών αν $f(X) = f(P_\pi X), \forall X \in \mathbb{R}^{n\times m}, P_\pi \in P_n$ και στη μετάθεση στηλών αν $f(X) = f(XP_\pi^T), \forall X \in \mathbb{R}^{m\times n}, P_\pi \in P_n$.

Ισοδυναμία μετάθεσης: Έστω P_n το σύνολο όλων των έγκυρων πινάκων μετάθεσης τάξης n, τότε μια συνάρτηση f είναι αμετάβλητη στη μετάθεση γραμμών αν $P_\pi f(X) = f(P_\pi X), \forall X \in \mathbb{R}^{n\times m}, P_\pi \in P_n$ και στη μετάθεση στηλών αν $f(X)P_\pi^T = f(XP_\pi^T), \forall X \in \mathbb{R}^{m\times n}, P_\pi \in P_n$.

Όπως αναφέρουν οι Gilmer et al. [41], το πλαίσιο μετάδοσης μηνυμάτων (message passing framework) παρουσιάζει ως βασική ιδέα ότι τα διανύσματα κάθε κόμβου πρέπει να δημιουργούνται με βάση τα διανύσματα κόμβων της γειτονιάς του. Θα παρουσιάσουμε μια πιο υψηλού επιπέδου προσέγγιση της μαθηματικής διατύπωσης αυτού του πρωτοκόλλου. Για ένα μη κατευθυνόμενο γράφημα G αντιπροσωπεύουμε τα χαρακτηριστικά κόμβου ως x_v και τα χαρακτηριστικά ακμής ως e_{uv}.



Εικόνα 1.3: *Πλαίσιο μετάδοσης μηνυμάτων (Message Passing Framework). Πηγή [3]*

Όπως φαίνεται στο παραπάνω σχήμα, το διάνυσμα κάθε κόμβου, που συμβολίζεται με N_A για την κορυφή A, επηρεάζεται από τα διανύσματα της γειτονιάς του. Αυτή η διαδικασία μετάδοσης μηνυμάτων επαναλαμβάνεται για σταθερό αριθμό επαναλήψεων ή μέχρι να επιτευχθεί σύγκλιση. Κάθε κόμβος v έχει ένα διάνυσμα κρυφής κατάστασης h_v και στο χρονικό βήμα t η κρυφή κατάσταση h_v^t ενημερώνεται με βάση τα μηνύματα από τη γειτονιά m_v^{t+1}. Τα μηνύματα συγκεντρώνονται από μια συνάρτηση AGGREGATE και στη συνέχεια οι καταστάσεις ενημερώνονται από μια συνάρτηση UPDATE, παράγοντας τη νέα κρυφή κατάσταση h_v^{t+1}. Οι εξισώσεις αυτής της διαδικασίας μπορούν να συνοψιστούν παρακάτω:

$$m_{N(v)}^{k} = AGGREGATE^{(k)}\left(\{h_u^{(k)}, \forall u \in N(v)\}\right) \tag{1.1}$$

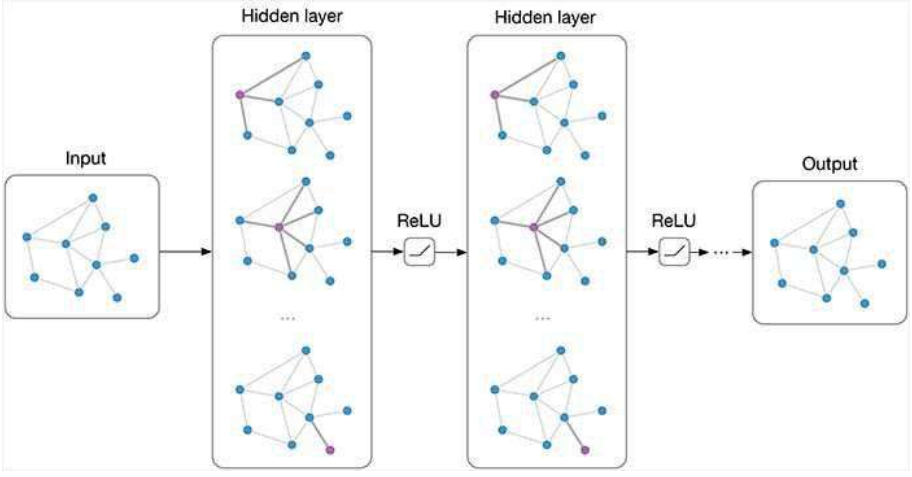
$$h_v^{k+1} = UPDATE^{(k)}(\{h_v^{(k)}, \forall u \in N(v)\}) \tag{1.2}$$

Αφού εκτελέσουμε τα βήματα K της φάσης μετάδοσης μηνυμάτων, μπορούμε να χρησιμοποιήσουμε την κρυφή κατάσταση h_v^K κάθε κόμβου v ως ενσωμάτωση του κόμβου. Η φάση ανάγνωσης είναι το τελευταίο βήμα στη διαδικασία μετάδοσης μηνυμάτων, όπου οι πληροφορίες από όλους τους κόμβους συγκεντρώνονται για να ληφθεί μια συνολική αναπαράσταση σε επίπεδο γραφήματος. Στη φάση ανάγνωσης, υπάρχει ένα χαρακτηριστικό $\widehat{y}$ που υπολογίζεται χρησιμοποιώντας μια συνάρτηση ανάγνωσης READOUT:

$$\widehat{y} = READOUT(h_v^{(k)} | v \in G) \tag{1.3}$$

Μία σημαντική συνθήκη του πλαισίου μετάδοσης μηνυμάτων είναι ότι όλοι οι κόμβοι πρέπει να έχουν κάποια αρχική κρυφή κατάσταση h_v^0 που μπορεί να είναι μια τιμή όπως ο βαθμός ή η κεντρική θέση κάθε κόμβου. Υπάρχουν διάφοροι τύποι GNN με διαφορετικές χρήσεις των συναρτήσεων AGGREGATE, UPDATE, και READOUT.

Τα νευρωνικά δίκτυα γράφων (GNNs) είναι μια κατηγορία νευρωνικών δικτύων που έχουν σχεδιαστεί για να χειρίζονται δεδομένα που αναπαρίστανται με τη μορφή γράφων. Έχουμε ήδη παρουσιάσει τη δομή των γράφων και τις εφαρμογές τους σε προβλήματα του πραγματικού κόσμου, όπως σε κοινωνικά δίκτυα, βιολογικά δεδομένα, συστήματα συστάσεων κ.λπ. Οι οντότητες των δεδομένων αναπαριστώνται ως κόμβοι, ενώ οι σχέσεις μεταξύ αυτών παρουσιάζονται ως ακμές. Τα GNN μπορούν να εκπαιδευτούν αξιοποιώντας τόσο τις πληροφορίες σε επίπεδο κόμβου όσο και από τη δομή του καθολικού γράφου, μέσω μηχανισμών μετάδοσης μηνυμάτων όπως περιγράψαμε προηγουμένως. Κάθε κόμβος συγκεντρώνει πληροφορίες από τους γείτονές του και ενημερώνει τη δική του κατάσταση. Διαφορετικοί τύποι GNNs χρησιμοποιούν διαφορετικές παραλλαγές των συναρτήσεων που χρησιμοποιούνται στο πλαίσιο μετάδοσης μηνυμάτων. Παρακάτω θα παρουσιάσουμε μερικούς από τους πιο χρησιμοποιούμενους τύπους των GNNs.



Εικόνα 1.4: *Νευρωνικό Δίκτυο Γράφου (Graph Neural Network). Πηγή [4]*

- **GCNs**: Τα συνελικτικά δίκτυα γράφων GCNs είναι ένας τύπος GNN που εισήχθη από τους Thomas Kipf και Max Welling [42]. Είναι ιδιαίτερα εμπνευσμένα από τα συνελικτικά νευρωνικά δίκτυα (CNNs), προσαρμοσμένα για την επεξεργασία δεδομένων σε γράφους, αντί για εικόνες. Τα GCN λειτουργούν με βάση την αρχή της μετάδοσης μηνυμάτων. Κάθε κόμβος συγκεντρώνει πληροφορίες από τους γείτονές του λαμβάνοντας ένα σταθμισμένο άθροισμα των διανυσμάτων τους και στη συνέχεια περνά αυτές τις πληροφορίες μέσω μιας συνάρτησης ενεργοποίησης για να ενημερώσει την κατάστασή του. Τα GCNs αποτελούνται συνήθως από πολλαπλά συνελικτικά επίπεδα. Κάθε επίπεδο υπολογίζει τα διανύσματα κόμβων χρησιμοποιώντας πληροφορίες από προοδευτικά πιο απομακρυσμένους κόμβους στο γράφο. Υπάρχουν δύο τύποι GCNs: τα φασματικά και τα χωρικά.

- **GraphSAGE**: Το GraphSAGE (SAmple and aggreGatE) είναι ένας τύπος GNN που προτάθηκε από τους Hamilton et al. [43] ως εναλλακτική λύση στις παραδοσιακές εγγενώς μεταγωγικές μεθόδους για τη δημιουργία διανυσματικών κόμβων. Οι προηγούμενες μέθοδοι χρησιμοποιούσαν συχνά μεθόδους βασισμένους σε παραγοντοποίηση πίνακα και επομένως περιορίζονταν σε ένα σταθερό γράφημα. Το GraphSAGE από την άλλη πλευρά χρησιμοποιεί μια επαγωγική προσέγγιση, ώστε να μπορεί να επεκταθεί και σε άγνωστους κόμβους και συνεπώς σε άγνωστα γραφήματα. Η βασική ιδέα είναι ότι το GraphSAGE δεν εκπαιδεύει ένα ξεχωριστό διάνυσμα ενσωμάτωσης για κάθε κόμβο, αλλά εκπαιδεύει ένα σύνολο συναρτήσεων (AGGREGATE) για να μάθει να συγκεντρώνει πληροφορίες χαρακτηριστικών από την τοπική γειτονιά ενός κόμβου. Το forward pass κομμάτι του αλγορίθμου ακολουθεί τη διαδικασία μετάδοσης μηνυμάτων του γενικού πλαισίου διέλευσης μηνυμάτων σε K επαναλήψεις χρησιμοποιώντας σε κάθε επανάληψη k τις εξισώσεις:

$$m_{N(v)}^{k} = AGGREGATE^{(k)}(\{h_u^{(k-1)}, \forall u \in N(v)\}) \tag{1.4}$$

$$h_v^{k+1} = \sigma(W^k CONCAT(h_v^{(k-1)}, m_{N(v)}^{(k)})) \tag{1.5}$$

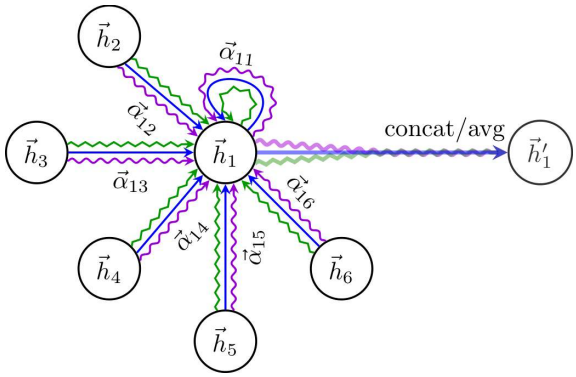
Η συνάρτηση AGGREGATE είναι ιδανικά μια συμμετρική και εκπαιδεύσιμη συνάρτηση. Η δημοσίευση παραπάνω εξετάζει τρεις λειτουργίες, κυρίως έναν mean aggregator, έναν LSTM aggregator και έναν τελεστή max pooling. Η τρέχουσα αναπαράσταση του κόμβου $h_v^{(k-1)}$ με το συγκεντρωτικό διάνυσμα γειτονιάς $h_{N(v)}^{(k-1)}$ συνενώνονται και το προκύπτον ενωμένο διάνυσμα τροφοδοτείται σε ενός πλήρως συνδεδεμένο στρώμα με μια μη γραμμική συνάρτηση ενεργοποίησης σ. Η εργασία παρουσιάζει επίσης μια σχέση του GraphSAGE με το Τεστ Ισομορφισμού Weisfeiler-Lehman.

- **GAT**: Το GAT (Graph Attention Network) είναι ένας τύπος GNN που προτείνεται από τους Veličković et al. [44] ως επέκταση των GCNs. Σε αντίθεση με το GraphSAGE, το GAT επεκτείνει τη μέθοδο AGGREGATE συνδυάζοντάς τη με attention mechanisms που δίνουν ένα attention score στους διάφορους γείτονες του κόμβου, δίνοντάς τους κατά συνέπεια διαφορετική σημασία. Ένα επίπεδο του δικτύου λαμβάνει ως είσοδο

ένα σύνολο χαρακτηριστικών κόμβου $h = \left\{\vec{h_1}, \vec{h_2}, ..., \vec{h_N}\right\}, \vec{h_i} \in R_F$ και παράγει ένα νέο σύνολο χαρακτηριστικών κόμβου $h' = \left\{\vec{h'_1}, \vec{h'_2}, ..., \vec{h'_N}\right\}, \vec{h'_i} \in R_{F'}$. Αρχικά, ένας γραμμικός μετασχηματισμός, παραμετροποιημένος από έναν πίνακα βάρους Ω, εφαρμόζεται σε κάθε κόμβο και στη συνέχεια πραγματοποιείται μηχανισμός self-attention στους κόμβους. Στη συνέχεια, σε κάθε κόμβο, ένας κοινός attention mechanism a απεικονίζει τα ζεύγη κόμβων στους συντελεστές e_{ij}. Οι συντελεστές αυτοί υπολογίζουν τη σημασία των χαρακτηριστικών ϑ που ανήκουν στον κόμβο ι. Αυτή η διαδικασία φαίνεται στο παρακάτω σχήμα. Μαθηματικά, οι συντελεστές προσοχής υπολογίζονται με την ακόλουθη εξίσωση:

$$e_{ij} = a(W\vec{h_i}, W\vec{h_j}) \tag{1.6}$$

Πρακτικά, αυτοί οι συντελεστές προσοχής υπολογίζονται μόνο για τα ζεύγη κόμβων που είναι γειτονικά στο γράφημα. Μια επέκταση αυτού είναι η χρήση multi-head attention, όπου κάθε κεφαλή (head) υπολογίζει ανεξάρτητα τα attention scores και τα αποτελέσματα συνενώνονται, για την παραγωγή των διανυσμάτων χαρακτηριστικών. Μια απεικόνιση αυτής της διαδικασίας φαίνεται στο παρακάτω σχήμα:



Εικόνα 1.5: *Απεικόνιση της multi-head προσέγγισης. Πηγή [5]*

- **GIN**: Το GIN (Graph Isomorphism Network) είναι ένας τύπος GNN που προτείνεται από τους Xu et al. [45], εμπνευσμένο σε μεγάλο βαθμό από την έννοια του ισομορφισμού γράφου, που σημαίνει ότι δύο γράφοι έχουν παρόμοια δομή μετά την αναδιάταξη των κόμβων χωρίς να γίνουν αλλαγές στις συνδέσεις. Η εργασία επικεντρώνεται στο τεστ Weisfeiler-Lehman για τον ισομορφισμό γράφου. Η σημασία του τεστ WL είναι η ενετική ενημέρωση που μπορεί να αναπαραστήσει διαφορετικές γειτονιές με διαφορετικά διανύσματα. Επεκτείνοντας τα GNNs, τα διανύσματα στη γειτονιά ενός κόμβου μπορούν να θεωρηθούν ως ένα πολυσύνολο και η συνάρτηση AGGREGATE εκτελείται σε αυτό το πολυσύνολο. Η αρχιτεκτονική του GNN που προκύπτει στη δημοσίευση [45] είναι εξίσου αποτελεσματική με το τεστ WL στη διάκριση γειτόνων γραφήματος. Το Θεώρημα Καθολικής Προσέγγισης (Universal Approximation Theorem) [46] δηλώνει ότι ένα feedforward νευρωνικό δίκτυο με ένα μόνο κρυφό στρώμα, με επαρκή αριθμό νευρώνων και με δεδομένες κατάλληλες συναρτήσεις ενεργοποίησης, μπορεί να προσεγγίσει οποιαδήποτε συνεχή συνάρτηση με κάποια αυθαίρετη ακρίβεια. Αυτή

η ιδέα οδηγεί στη χρήση πολυστρωματικών perceptrons (MLPs) για τις συναρτήσεις AGGREGATION και READ OUT. Στην δημοσίευση, προτείνεται ότι με το ϵ να είναι μια εκπαιδεύσιμη παράμετρος ή μια βαθμωτή σταθερά, και το GIN ενημερώνει τις αναπαραστάσεις κόμβων χρησιμοποιώντας την εξίσωση:

$$h_v^{(k)} = MLP^{(k)}((1 + \epsilon^{(k)})h_v^{k-1} + \sum_{u \in N(v)} h_u^{k-1}) \tag{1.7}$$

Μετά από σύγκριση με άλλα αξιοσημείωτα μοντέλα που παρουσίασαν χειρότερα αποτελέσματα, το GIN τελικά αποδείχθηκε ότι είναι μια πολύ χρήσιμη παραλλαγή GNN με τη συνάρτηση συνάθροισης γειτονιάς.

1.2.4 Αναζήτηση Αρχιτεκτονικής Νευρωνικών Δικτύων

Τα νευρωνικά δίκτυα γράφων βασίζονται σε μεγάλο βαθμό στην αρχιτεκτονική τους, για να παρέχουν καλά αποτελέσματα. Η εύρεση της κατάλληλης αρχιτεκτονικής μπορεί να είναι εξαιρετικά χρονοβόρα και επομένως μπορεί να είναι και αρκετά περιορισμένη. Αυτός είναι ο λόγος που η αναζήτηση αρχιτεκτονικής (Neural Architecture Search (NAS)) έχει προσελκύσει πολυάριθμες μελέτες, μιας και αυτή η τεχνική μπορεί να κατασκευάσει αρχιτεκτονικές που επιτυγχάνουν ικανοποιητικά αποτελέσματα με μικρή ανθρώπινη παρέμβαση. Σε αυτή την ενότητα, θα παρουσιάσουμε τα τρία βασικά στοιχεία της Αναζήτησης Αρχιτεκτονικής Νευρωνικού Δικτύου: ο χώρος αναζήτησης, η στρατηγική αναζήτησης και η αξιολόγηση απόδοσης.

Ο **χώρος αναζήτησης** αντιπροσωπεύει το σύνολο όλων των πιθανών αρχιτεκτονικών νευρωνικών δικτύων. Μια αρχιτεκτονική ορίζεται από τη λειτουργία που σχετίζεται με κάθε επίπεδο και τις ενδιάμεσες συνδέσεις των επιπέδων. Το μέγεθος του χώρου αναζήτησης καθορίζει το υπολογιστικό κόστος της διαδικασίας αναζήτησης. Επομένως, υπάρχει μια αντιστάθμιση μεταξύ του αριθμού των αρχιτεκτονικών που θέλουμε να δοκιμάσουμε και του κόστους του αλγορίθμου.

- **Μονολιθικά δομημένος χώρος αναζήτησης**: Ο χώρος των μονολιθικά δομημένων νευρωνικών δικτύων [47] είναι η πιο προφανής επιλογή για έναν χώρο αναζήτησης. Αυτά τα μοντέλα κατασκευάζονται με στοίβαξη ενός προκαθορισμένου αριθμού κόμβων. Αυτή η επιλογή θα μπορούσε επίσης να υποστηρίξει δομές κόμβων που συνδέονται με τυχαίες συνδέσεις παράλειψης. Ο ολόκληρος δομημένος χώρος αναζήτησης μπορεί να είναι εύκολος στην εφαρμογή, αλλά είναι πιο ακριβός υπολογιστικά και στερείται φορητότητας.

- **Χώρος αναζήτησης βασισμένος σε δομικές μονάδες**: Αυτή η αναζήτηση νευρικής αρχιτεκτονικής, αντί να αναζητά ολόκληρη την αρχιτεκτονική, αναζητά μοτίβα ή κύτταρα [48]. Αυτό το κύτταρο στοιβάζεται πολλές φορές για να σχηματίσει μια μεγαλύτερη αρχιτεκτονική. Αυτή η μέθοδος έχει μειωμένο χώρο αναζήτησης και επομένως είναι υπολογιστικά λιγότερο ακριβή. Επιπλέον, λύνει το ζήτημα της φορητότητας, καθώς η στοίβαξη περισσότερων ή λιγότερων μονάδων θα μπορούσε να δημιουργήσει αρχιτεκτονικές για άλλες εργασίες.

- **Ιεραρχικός χώρος αναζήτησης**: Οι προηγούμενες προσεγγίσεις αγνοούν το επίπεδο δικτύου. Οι Liu et al. [49] όρισαν μια γενική διατύπωση για μια δομή σε επίπεδο δικτύου με το μοντέλο HierNAS. Αυτό ορίζεται ως ο ιεραρχικός χώρος αναζήτησης, στον οποίο δημιουργείται ένα μπλοκ υψηλότερου επιπέδου με την επαναληπτική ενσωμάτωση μονάδων χαμηλότερου επιπέδου.

- **Χώρος αναζήτησης βάσει μορφισμού**: Αυτός ο χώρος αναζήτησης επιχειρεί να σχεδιάσει νέα νευρωνικά δίκτυα βασισμένα σε ένα υπάρχον δίκτυο χρησιμοποιώντας μετασχηματισμούς μορφισμού μεταξύ των επιπέδων [50]. Για παράδειγμα, οι μετασχηματισμοί μορφισμού βάθους ή πλάτους αντικαθιστούν το αρχικό μοντέλο με ένα αντίστοιχο βαθύτερο ή ευρύτερο μοντέλο. Τα θυγατρικά δίκτυα από τον μορφισμό του δικτύου κληρονομούν όλη τη γνώση των γονικών δικτύων τους. Στη συνέχεια, αυτά τα παιδικά δίκτυα εκπαιδεύονται για μικρό χρονικό διάστημα.

Πολλοί διαφορετικοί **αλγόριθμοι βελτιστοποίησης** έχουν χρησιμοποιηθεί για να βρεθεί η καλύτερη αρχιτεκτονική στον χώρο αναζήτησης. Η επιλογή της μεθόδου βελτιστοποίησης επηρεάζεται σε μεγάλο βαθμό από τον χώρο αναζήτησης και έχει μεγάλη επίδραση στο υπολογιστικό κόστος και τα αποτελέσματα του μοντέλου. Θα κάνουμε μια σύντομη ανασκόπηση στις πιο γνωστές μεθόδους βελτιστοποίησης.

- **Ενισχυτική μάθηση**: Μια αξιοσημείωτη τεχνική βελτιστοποίησης για την αναζήτηση αρχιτεκτονικής είναι η ενισχυτική μάθηση [47]. Ο ελεγκτής προτείνει αρχιτεκτονικές θυγατρικών μοντέλων από τον χώρο αναζήτησης για αξιολόγηση. Ο ελεγκτής είναι συνήθως ένα RNN. Εκπαιδεύουμε και αξιολογούμε την αρχιτεκτονική του δείγματος και αυτή η απόδοση είναι η ανταμοιβή που λαμβάνει ο ελεγκτής. Το σήμα ανταμοιβής δεν είναι διαφοροποιήσιμο, επομένως χρησιμοποιούμε μια μέθοδο διαβάθμισης για να ενημερώσουμε τον ελεγκτή και θέλουμε να μεγιστοποιήσουμε την αναμενόμενη ανταμοιβή:

$$J(\partial_c) = E_{P(a_{1:T};\partial_c)[R]} \tag{1.8}$$

όπου $a_{1:T}$ είναι οι ενέργειες του ελεγκτή, T είναι ο συνολικός αριθμός των tokens, ∂_c είναι οι παράμετροι του ελεγκτή και R είναι η ανταμοιβή. Χρησιμοποιώντας τον κανόνα REINFORCE ως μέθοδο κλίσης, λαμβάνουμε τη διαβάθμιση της αναμενόμενης ανταμοιβής ως:

$$\nabla_{\partial_c} J(\partial_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\partial_c)}[\nabla_{\partial_c} logP(a_t|a_{t-1:1} : \partial_c)R] \tag{1.9}$$

- **Εξελικτικοί αλγόριθμοι**: Οι εξελικτικοί αλγόριθμοι εξελίσσουν έναν πληθυσμό αρχιτεκτονικών με στόχο την εύρεση της βέλτιστης αρχιτεκτονικής. Σε κάθε βήμα της διαδικασίας, μερικές αρχιτεκτονικές δειγματοληπτούνται από τον πληθυσμό για να δημιουργηθούν οι θυγατρικές αρχιτεκτονικές μέσω μεταλλάξεων. Οι μεταλλάξεις αλλάζουν τη δομή της αρχιτεκτονικής, όπως η προσθήκη ή η αφαίρεση ενός επιπέδου ή η αλλαγή των λειτουργιών των επιπέδων. Οι θυγατρικές αρχιτεκτονικές αξιολογούνται και έπειτα προστίθενται στον πληθυσμό. Στο τέλος, αφαιρείται η γηραιότερη

αρχιτεκτονική ή εκείνη με την χειρότερη απόδοση.

- **Μέθοδοι κατάβασης κλίσης**: Οι προηγούμενες στρατηγικές αναζήτησης λειτουργούν σε έναν διακριτό χώρο αναζήτησης. Ένας πρωτοποριακός αλγόριθμος, το DARTS [51], βασίστηκε στη Βελτιστοποίηση κατάβασης κλίσης που αναζητούσε νευρωνικές αρχιτεκτονικές σε έναν χώρο συνεχούς αναζήτησης χρησιμοποιώντας μια συνάρτηση softmax για να χαλαρώσει το διακριτό χώρο αναζήτησης, όπως περιγράφεται παρακάτω:

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{K} \frac{e^{a_{i,j}^k}}{\sum_{l=1}^{K} e^{a_{i,j}^l}} o^k(x) \tag{1.10}$$

όπου $o(x)$ είναι η λειτουργία που εκτελείται στην είσοδο x, $a_{i,j}^k$ είναι το βάρος που εκχωρείται στη λειτουργία o^k μεταξύ ενός ζεύγους κόμβων (i,j) και K είναι ο αριθμός των προκαθορισμένων υποψηφίων ενεργειών. Μετά τη χαλάρωση του χώρου αναζήτησης, το έργο της αναζήτησης αρχιτεκτονικών μετατρέπεται σε βελτιστοποίηση της νευρωνικής αρχιτεκτονικής a και των βαρών ∂ αυτής της νευρωνικής αρχιτεκτονικής, χρησιμοποιώντας το σύνολο επικύρωσης και το σύνολο εκπαίδευσης, αντίστοιχα.

- **Μπεϋζιανή Βελτιστοποίηση**: Η αναζήτηση αρχιτεκτονικής θα μπορούσε να προσεγγιστεί ως πρόβλημα βελτιστοποίησης μαύρου κουτιού, με μια αντικειμενική συνάρτηση f. Η μπεϋζιανή βελτιστοποίηση είναι μια ευρέως διαδεδομένη τεχνική λόγω της ικανότητάς της να δημιουργεί ένα υποκατάστατο μοντέλο για τη μοντελοποίηση της αντικειμενικής συνάρτησης (που συμβολίζεται ως f). Επιλέγει επαναληπτικά αρχιτεκτονικές για αξιολόγηση με βάση την αντικειμενική συνάρτηση, ενημερώνοντας παράλληλα το υποκατάστατο μοντέλο. Μόλις πραγματοποιηθούν αρκετές αξιολογήσεις της f, χρησιμοποιεί τον κανόνα του Bayes για να εξαγάγει την επόμενη τιμή της f. Στο επόμενο βήμα, αξιοποιεί μια συνάρτηση απόκτησης $a(x)$, για να προσδιορίσει το επόμενο σημείο δείγματος $x_t = argmax_x a(x)$ που βελτιστοποιεί τη συνάρτηση αυτή. Αυτή η διαδικασία είναι σημαντικά λιγότερο χρονοβόρα και θα μπορούσε να βελτιώσει την αποτελεσματικότητα της αναζήτησης σε συνεχείς χώρους χαμηλών διαστάσεων.

Κάθε στρατηγική αναζήτησης εξάγει την αρχιτεκτονική που μεγιστοποιεί μια επιθυμητή μετρική απόδοσης, όπως η ακρίβεια στο σύνολο επικύρωσης ή η ακρίβεια στο σύνολο δοκιμής. Κάθε αρχιτεκτονική από ένα μεγάλο χώρο αναζήτησης πρέπει να εκπαιδευτεί πρώτα για να αξιολογηθεί, οπότε αυτή η διαδικασία μπορεί να είναι αρκετά χρονοβόρα και να έχει μη διαχειρίσιμο υπολογιστικό κόστος. Για να γίνει πιο αποτελεσματική η αναζήτηση αρχιτεκτονικής, έχουν γίνει προσπάθειες να μειωθεί η διαδικασία εκπαίδευσης των αρχιτεκτονικών. Για παράδειγμα, ορισμένες προσεγγίσεις εκπαιδεύουν τις αρχιτεκτονικές για λιγότερες εποχές ή σε ένα υποσύνολο δεδομένων. Μια άλλη προσέγγιση είναι να κληρονομηθούν γονικά χαρακτηριστικά στις αρχιτεκτονικές, αντί να εκπαιδεύονται τα μοντέλα από την αρχή. Με αυτές τις τεχνικές το υπολογιστικό κόστος μειώθηκε σημαντικά χωρίς να μειώνεται η απόδοση της αναζήτησης.

1.2.5 Αξιολόγηση μοντέλων ταξινόμησης

Ένα πολύ σημαντικό βήμα για την εξασφάλιση της ποιότητας ενός μοντέλου μηχανικής μάθησης είναι η αξιολόγηση του με βάση διάφορες μετρικές. Μία από τις σημαντικότερες μετρικές αξιολόγησης στα προβλήματα ταξινόμησης είναι ο πίνακας σύγχυσης (confusion matrix). Ένας πίνακας σύγχυσης είναι ένας πίνακας που ενσωματώνει την απόδοση ενός μοντέλου ταξινόμησης σε ένα σύνολο δεδομένων δοκιμής. Οι σειρές του πίνακα αντιπροσωπεύουν τα δεδομένα στην προβλεπόμενη κλάση, ενώ οι στήλες αντιπροσωπεύουν τα δεδομένα στην πραγματική κλάση. Στη δυαδική ταξινόμηση, ο πίνακας θα είναι μεγέθους 2x2. Για την ταξινόμηση σε N κλάσεις, το σχήμα του πίνακα θα είναι NxN.

		Πραγματικές τιμές	
		Ναι	Όχι
Προβλέψεις	Ναι	True Positive	False Positive
	¨Οχι	False Negative	True Negative

Πίνακας 1.1: *Πίνακας σύγχυσης*

Ορίζουμε τις μεταβλητές του πίνακα σύγχυσης ως εξής:

- **True Positive (TP)**: Οι περιπτώσεις στις οποίες έχουμε προβλέψει θετικά και η πρόβλεψη επιβεβαιώνεται.
- **True Negative (TN)**: Οι περιπτώσεις στις οποίες έχουμε προβλέψει αρνητικά και η πρόβλεψη επιβεβαιώνεται.
- **False Positive (FP)**: Πρόκειται για περιπτώσεις στις οποίες έχουμε προβλέψει θετικά και η πρόβλεψη δεν επιβεβαιώνεται.
- **False Negative (FN)**: Πρόκειται για περιπτώσεις στις οποίες έχουμε προβλέψει αρνητικά και η πρόβλεψη δεν επιβεβαιώνεται.

Με βάση αυτές τις μεταβλητές ορίζουμε τις ακόλουθες μετρήσεις:

- **Accuracy**: Υπολογίζεται διαιρώντας τον αριθμό των σωστών προβλέψεων με τον συνολικό αριθμό των προβλέψεων.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.11}$$

  Σε περιπτώσεις ανισορροπίας των δειγμάτων κλάσεων στο σύνολο δεδομένων, ο ταξινομητής θα τείνει να προβλέψει την κλάση με τα περισσότερα δεδομένα, με αποτέλεσμα να μετράται πολύ υψηλή ακρίβεια. Αυτό, ωστόσο, δεν αντικατοπτρίζει μια ικανοποιητική ποιότητα του μοντέλου. Αυτός είναι ο λόγος που χρειαζόμαστε κι άλλες μετρικές.

- **Precision**: Ορίζεται ως το ποσοστό των σωστών θετικών προβλέψεων του ταξινομητή.

$$precision = \frac{TP}{TP + FP} \tag{1.12}$$

- **Recall**: Ορίζεται ως το ποσοστό των θετικών δειγμάτων που έχει προβλεφθεί σωστά από τον ταξινομητή.

$$recall = \frac{TP}{TP + FN} \tag{1.13}$$

- **F1-score**: Συνδυάζει την ακρίβεια και την ανάκληση, για να παρέχει μια ισορροπημένη μέτρηση της απόδοσης του μοντέλου. Συγκεκριμένα, είναι ο αρμονικός μέσος ακρίβειας και ανάκλησης.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{1.14}$$

- **Specificity**: Ορίζεται ως το ποσοστό των αρνητικών δειγμάτων που έχει προβλεφθεί σωστά από τον ταξινομητή.

$$specificity = \frac{TN}{TN + FP} \tag{1.15}$$

- **Cross-Entropy Loss**: Μετρά την ανομοιότητα μεταξύ των προβλεπόμενων ετικετών και των ετικετών πραγματικών κλάσεων. Στη δυαδική ταξινόμηση, η απώλεια διασταυρούμενης εντροπίας υπολογίζεται ως εξής:

$$BinaryCross - EntropyLoss = -(y \log(p) + (1 - y) \log(1 - p)) \tag{1.16}$$

όπου p είναι η προβλεπόμενη πιθανότητα και y ο δείκτης (0 ή 1 σε δυαδική ταξινόμηση)

- **Matthews Correlation Coefficient (MCC)**: Ορίζεται ως η εκτίμηση της συσχέτισης μεταξύ της προβλεπόμενης κλάσης και της κλάσης στην οποία ανήκουν πραγματικά οι χρήστες.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \tag{1.17}$$

1.2.6 Διανύσματα λέξεων

Τα διανύσματα λέξεων είναι μια τεχνική που εμπίπτει στον τομέα της Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing: NLP) που χρησιμοποιείται για την αναπαράσταση λέξεων με αριθμούς βάσει της σημασίας τους και των ενδιάμεσων τους σχέσεων. Αντιστοιχίζουν λέξεις σε πολυδιάστατα διανύσματα με τρόπο ώστε παρόμοιες λέξεις να είναι κοντινά σημεία στον ίδιο διανυσματικό χώρο. Μερικά γνωστά μοντέλα είναι τα Word2Vec, GloVe, BERT και τέλος αυτό που θα χρησιμοποιήσουμε στο πείραμα μας RoBERTa.

Το RoBERTa (A Robustly Optimized BERT Pretraining Approach) [52] είναι ένα προηγμένο μοντέλο διανύσματος λέξεων βασισμένο στον προκάτοχό του BERT. Μετά από βελτιστοποιήσεις στη διαδικασία προεκπαίδευσης, μπορεί να βελτιώσει την απόδοση του μοντέλου σε ένα ευρύ φάσμα εργασιών της επεξεργασίας φυσικής γλώσσας. Ένα σημαντικό πλεονέκτημα του RoBERTa είναι ότι εκπαιδεύεται σε σημαντικά μεγαλύτερο σύνολο δεδομένων κειμένου σε σύγκριση με το BERT και κατά συνέπεια εκπαιδεύεται σε ένα πιο ευρύ γλωσσικό φάσμα. Επίσης, αγνοεί την εργασία Πρόβλεψη Επόμενης Πρότασης (Next Sentence Prediction: NSP) που χρησιμοποιείται στο BERT και εστιάζει στην εργασία Μοντέλο Μασκαρισμένης Γλώσσας (Masked Language Model: MLM). Η εκπαιδευτική διαδικασία του RoBERTa περιέχει μία

λεπτομερή εύρεση υπερπαραμέτρων. Το RoBERTa ξεπερνά τα άλλα προηγούμενα μοντέλα σε αρκετές εργασίες επεξεργασίας φυσικής γλώσσας, πιο συγκεκριμένα στις General Language Understanding Evaluation (GLUE), ReAding Comprehension from Examinations (RACE), και Stanford Question Answering Dataset (SQuAD). Η αποτελεσματικότητα καθώς και η ευελιξία του μοντέλου είναι οι λόγοι που χρησιμοποιούμε το συγκεκριμένο μοντέλο διανύσματος λέξεων στο μοντέλο μας.

1.3 Σχετικές έρευνες

Σε αυτό το κεφάλαιο, παρουσιάζουμε μια πλήρη ανασκόπηση των μεθόδων που χρησιμοποιούνται για τον εντοπισμό bot, καθώς και μελέτες για την Αναζήτηση Αρχιτεκτονικής Νευρωνικών Δικτύων σε διάφορες περιπτώσεις χρήσης. Συγκρίνοντας τα αποτελέσματα τους θα επιλέξουμε την ακριβέστερη μέθοδο που στη συνέχεια θα προσπαθήσουμε να βελτιστοποιήσουμε.

Οι Lee et al [53] χρησιμοποίησαν ενισχυτική μάθηση και εκπαίδευσαν έναν ταξινομητή για να διακρίνει μεταξύ γνήσιων χρηστών και ρομπότ με βάση χαρακτηριστικά που εξάγονται από δεδομένα του Twitter, όπως τον αριθμό των ακολούθων, φίλων, τωεετς, χρόνος δημιουργίας λογαριασμού, χρήση διευθύνσεων ΥΡΛ, ηασηταγς, κτλ. Οι ερευνητές εφάρμοσαν διάφορους αλγόριθμους μηχανικής μάθησης, συμπεριλαμβανομένων των μηχανών υποστήριξης διανυσμάτων (SVM), του Naive Bayes και των δέντρων αποφάσεων, για να δημιουργήσουν και να αξιολογήσουν το μοντέλο ανίχνευσης.

Οι Yang et al [54] χρησιμοποίησαν μη επιβλεπόμενη μάθηση για τον εντοπισμό ρομπότ. Συγκεκριμένα, χρησιμοποίησαν χαρακτηριστικά που προέρχονται από τα μεταδεδομένα χρήστη, χρονικά μοτίβα, δομή του δικτύου, ανάλυση συναισθήματος και γλωσσικά στοιχεία. Αξιολόγησαν την ικανότητα του μοντέλου να διακρίνει τα βοτς από τους πραγματικούς χρήστες, υπολογίζοντας την περιοχή κάτω από τη χαρακτηριστική καμπύλη λειτουργίας του δέκτη (AUC-ROC). Ανέφεραν επίσης μετρικές όπως η ακρίβεια, η ανάκληση και το F1-score, για περισσότερες πληροφορίες σχετικά με την απόδοση του μοντέλου. Για να αποκτήσουν το σύνολο δεδομένων οι συγγραφείς χρησιμοποίησαν το Botometer API, ένα εργαλείο που αναπτύχθηκε από την ίδια ερευνητική ομάδα.

Οι Cresci et al [55] εισήγαγαν την τεχνική του Social Fingerprinting για ανίχνευση των bot, μια μέθοδο ψηφιακού DNA για τη μοντελοποίηση των συμπεριφορών των χρηστών κοινωνικών δικτύων. Κάθε χρήστης αναπαρίσταται ως μια ακολουθία χαρακτήρων ανάλογα με τον τύπο και το περιεχόμενο των tweets που δημοσιεύει παρόμοια με μια ακολουθία DNA. Οι ερευνητές επεδίωξαν να βρουν ομοιότητες στις ακολουθίες που θα βοηθήσουν στο διαχωρισμό των δύο κατηγοριών χρηστών. Το μέτρο ομοιότητας ορίζεται ως το μήκος της μεγαλύτερης κοινής υποσυμβολοσειράς (LCS) μεταξύ δύο ακολουθιών. Για ένα σύνολο πραγματικών χρηστών, το μήκος αυτό βρέθηκε να είναι ιδιαίτερα μικρότερο. Με βάση αυτή την ιδέα, οι συγγραφείς ανέπτυξαν δύο τεχνικές, μια βασισμένη στην επιβλεπόμενη μάθηση και μια άλλη με μη επιβλεπόμενη μάθηση χωρίς επίβλεψη για να βρουν ομοιότητες στη συμπεριφορά των λογαριασμών. Χρησιμοποίησαν το σύνολο δεδομένων ̈ρεσγι. [56]

Το Botometer [57] είναι ένα διαδικτυακό πρόγραμμα που αναπτύχθηκε από το Πανεπιστήμιο της Ιντιάνα. Χρησιμοποιεί τεχνικές μηχανικής εκμάθησης για να ταξινομήσει τους

λογαριασμούς σε bots και ανθρώπους, εξετάζοντας μια σειρά από χαρακτηριστικά του προφίλ. Το Botometer διακρίνει τους λογαριασμούς με ένα συνολικό σκορ bot (0-5), μαζί με αρκετές άλλες βαθμολογίες. Όσο υψηλότερη είναι αυτή η συνολική βαθμολογία, τόσο πιο πιθανό είναι αυτός ο λογαριασμός να ανήκει σε ένα bot.

Οι Feng et al [38] πρότειναν το δικό τους μοντέλο για ανίχνευση bots, με όνομα BotRGCN. Το BotRGCN δημιουργεί έναν ετερογενή γράφο από τις σχέσεις ακολούθησης μεταξύ των λογαριασμών και χρησιμοποιεί πληροφορίες, όπως η περιγραφή του χρήστη, τα tweets, τον αριθμό ακολούθων και φίλων και οι πληροφορίες γειτονιάς. Τα πειράματα διεξήχθησαν στο σύνολο δεδομένων Twi-Bot 20, ωστόσο το BotRGCN θα μπορούσε να εκμεταλλευτεί και άλλους τύπους σχέσεων μεταξύ χρηστών εάν υποστηρίζονταν από το σύνολο δεδομένων. Με μία υψηλή ακρίβεια της τάξης του 86,42% ξεπέρασε προηγούμενα μοντέλα, υπογραμμίζοντας τα οφέλη της χρήσης συνελικτικών νευρωνικών δικτύων στην ανίχνευση bot.

Έχοντας επιβεβαιώσει τα οφέλη από τη χρήση γραφημάτων για την ανίχνευση bots, θα μελετήσουμε εφαρμογές της Αναζήτησης Αρχιτεκτονικής Νευρωνικών Δικτύων σε μια προσπθάεια για να επιτύχουμε πιθανές βελτιώσεις.

Οι Nunes et al [58] παρουσίασαν δύο μεθόδους για τη βελτιστοποίηση των νευρωνικών δικτυών γράφων: μία βασισμένη στην ενισχυτική μάθηση και μία βασισμένη σε εξελικτικούς αλγόριθμους. Οι συγγραφείς πειραματίστηκαν με αυτές τις μεθόδους για να αξιολογήσουν εάν παρέχουν καλύτερη ακρίβεια από μια τυχαία αναζήτηση σε όλες τις πιθανές παραμέτρους του δικτύου. Οι εξελικτικοί αλγόριθμοι είναι μέθοδοι που βασίζονται στη θεωρία της εξέλιξης. Συγκεκριμένα, πολλές διαφορετικές παράμετροι στα επίπεδα παράγουν ένα σύνολο από δίκτυα, τα οποία ανταγωνίζονται για να επιτύχουν την καλύτερη ακρίβεια σε ένα σύνολο επικύρωσης (validation set), και να επιλεγούν για να παράξουν μια νέα γενιά απογόνων. Στην ενισχυτική μάθηση, ένα δίκτυο LSTM χρησιμοποιείται ως ελεγκτής για τη δημιουργία αρχιτεκτονικών. Οι συγγραφείς όρισαν δύο περιπτώσεις χώρων αναζήτησης: Macro, όπου οι αρχιτεκτονικές που δημιουργούνται έχουν την ίδια δομή, και Micro, όπου οι αρχιτεκτονικές δεν είναι αυστηρά δομημένες αλλά συνδυάζουν πολλά συνελικτικά σχήματα. Οι συγγραφείς χρησιμοποίησαν επτά διαφορετικά σύνολα δεδομένων για 100 επαναλήψεις σε κάθε χώρο αναζήτησης. Κατέληξαν στο συμπέρασμα ότι και οι δύο μέθοδοι βρήκαν πολύ παρόμοιες αρχιτεκτονικές με της τυχαίας αναζήτησης. Ωστόσο, σημείωσαν ότι οι εξελικτικοί αλγόριθμοι παρήγαγαν ως επί το πλείστον αρχιτεκτονικές που μπορούσαν να χωρέσουν στη GPU, σε αντίθεση με τις άλλες μεθόδους.

Οι Zhang et al. [6] πρότειναν το μοντέλο DFG-NAS, μια καινοτόμο μέθοδο που επιτρέπει την αυτόματη αναζήτηση βαθιών και δυναμικών αρχιτεκτονικών νευρωνικών δικτύων γράφων. Το DFG-NAS δίνει έμφαση στην αναζήτηση σε μακρο-αρχιτεκτονικές, και συγκεκριμένα στο πώς οι λειτουργίες ατομικής διάδοσης (propagation: P) και μετασχηματισμού (transformation: T) υλοποιούνται στο δίκτυο. Το Π σχετίζεται στενά με τη δομή του γραφήματος, ενώ το Τ εστιάζει στον μη γραμμικό μετασχηματισμό στο νευρωνικό δίκτυο. Οι συγγραφείς σημείωσαν ότι οι περισσότερες προηγούμενοι μέθοδοι εφάρμοσαν μετασχηματισμό μετά τη διάδοση (Π-Τ) σε κάθε στρώμα, η οποία είναι μια περιοριστική προσέγγιση. Χρησιμοποίησαν έναν εξελικτικό αλγόριθμο για να βρουν τη βέλτιστη αρχιτεκτονική, η οποία υποστήριζε τέσσερις περιπτώσεις μετάλλαξης: (α) πρόσθεση ενός Π, (β) πρόσθεση ενός Τ, (γ) αντικατάσταση ενός Π με ένα Τ, (δ) αντικατάσταση ενός Τ με ένα Π. Οι συγγραφείς κα-

τέληξαν με μια βελτίωση ακρίβειας έως και 0,9% σε σχέση με χειροκίνητα σχέδια τελευταίας τεχνολογίας, με επιτάχυνση 15,96x σε σχέση με άλλες μεθόδους.

1.4 Σύνολο δεδομένων και περιγραφή του προβλήματος

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε μια ευρεία εξήγηση των ιδιοτήτων του συνόλου δεδομένων που χρησιμοποιήσαμε για το πείραμά μας, καθώς και μια περιγραφή του προβλήματος στο οποίο εστιάζουμε.

1.4.1 Σύνολο δεδομένων TwiBot-20

Διεξάγαμε το πείραμα μας χρησιμοποιώντας το σύνολο δεδομένων TwiBot-20, το οποίο λάβαμε από τους Feng et al. [39]. Αυτό το σύνολο δεδομένων περιλαμβάνει το σύνολο εκπαίδευσης, επικύρωσης, δοκιμής και υποστήριξης. [59]. Το TwiBot-20 έχει κατασκευαστεί με μεθοδολογία αναζήτησης κατά πλάτους (BFS). Παρουσιάζουμε τα χαρακτηριστικά του συνόλου με μια σύντομη περιγραφή για περαιτέρω επεξήγηση:

Χαρακτηριστικό	Περιγραφή
ID	αναγνωριστικό από το Twitter για την αναγνώριση του χρήστη
profile	στοιχεία προφίλ από το Twitter API
tweet	200 πρόσφατα tweets του χρήστη χρήστη
neighbor	20 τυχαίοι ακόλουθοι και λογαριασμοί που ακολουθεί ο χρήστης
domain	τομέας χρήστη (πολιτική, επιχείρηση, ψυχαγωγία, αθλητισμός)
label	ετικέτα του χρήστη (‘1’:bot, ‘0’:άνθρωπος)

Πίνακας 1.2: *Χαρακτηριστικά και περιγραφή του συνόλου δεδομένων TwiBot-20 Dataset*

[ητ!]

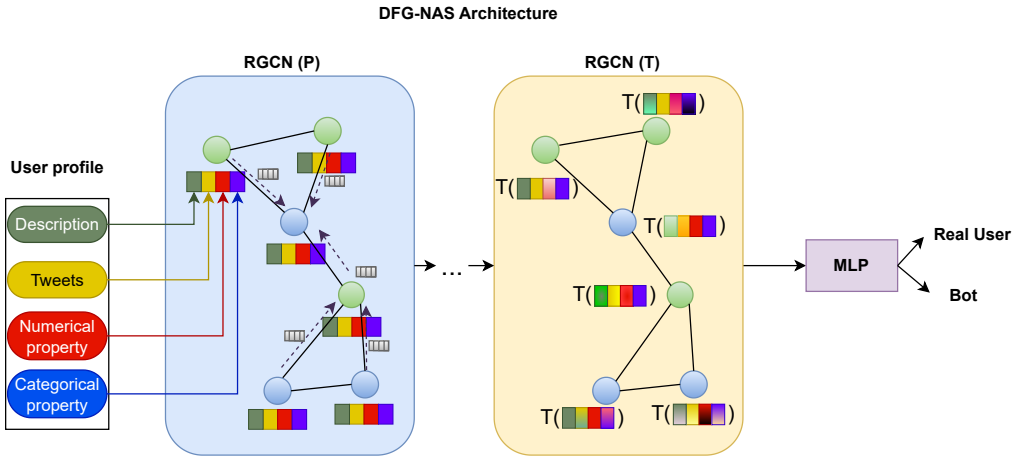
Το Twibot-20 έχει χρησιμοποιηθεί σε πολλές περιπτώσεις εντοπισμού ρομπότ λόγω των σημασιολογικών και γειτονικών ιδιοτήτων χρήστη που μπορεί να απεικονίσει. Στη συνέχεια θα παρουσιάσουμε τον ορισμό του προβλήματος που θέλουμε να λύσουμε.

1.4.2 Περιγραφή του προβλήματος

Έστω B = $b_{i_{i=1}}^{L}$ που αντιπροσωπεύει την περιγραφή του χρήστη με αριθμό λέξεων ίσο με L. Έστω T = $t_{i_{i=1}}^{M}$ τα M tweets του χρήστη και κάθε tweet $t_i = \{w_1^i, ..., w_{Q_i}^i\}$ περιέχει Q_i λέξεις. Έστω $P = P^{num}, P^{cat}$ το σύνολο αριθμητικών και κατηγορικών ιδιοτήτων του χρήστη αντίστοιχα. Τέλος, έστω $N = N^f, N^t$ οι πληροφορίες γειτονιάς του χρήστη, όπου $N^f = N_1^f, ..., N_u^f$ υποδηλώνει τους λογαριασμούς που ακολουθεί ο χρήστης και $N^t = N_1^t, ..., N_u^t$ υποδηλώνει τους λογαριασμούς που ακολουθούν τον χρήστη. Το έργο του εντοπισμού των bots στο Twitter είναι να ξεχωρίσει το σύνολο δεδομένων σε bots και αληθινούς χρήστες με τις πληροφορίες από τα B, T, P και N.

1.5 Πειραματικό μέρος

Ο εντοπισμός των bots είναι κρίσιμος στο σημερινό κόσμο, καθώς ο πολλαπλασιασμός τους αποτελεί σημαντική απειλή για τις διαδικτυακές πλατφόρμες. Για να αναπτύξουμε ένα βελτιωμένο μοντέλο για αυτήν την εργασία, ενσωματώσαμε μια πολλά υποσχόμενη προσέγγιση ανίχνευσης bot με την τεχνική της Αναζήτησης Αρχιτεκτονικής. Αρχικά, θα περιγράψουμε την προεπεξεργασία των μεταδεδομένων χρήστη που χρησιμοποιούνται στο μοντέλο μας. Στη συνέχεια, περιγράφουμε τη χρήση των Σχεσιακών Συνελικτικών Νευρωνικών Δικτύων Γράφων και των δύο διαφορετικών συναρτήσεων στο Πρωτόκολλο Διαβίβασης Μηνυμάτων. Τέλος, εξηγούμε την εφαρμογή του DFG-NAS [6] στην αναζήτηση της καλύτερης μετάθεσης των συναρτήσεων Διάδοσης και Μετασχηματισμού.



Εικόνα 1.6: *Μοντέλο για την ανίχνευση των bots*

1.5.1 Ανίχνευση bots με σχεσιακά συνελικτικά νευρωνικά δίκτυα γράφων

Οι Feng et al. [38] πρότειναν το μοντέλο τους BotRGCN για τον εντοπισμό των bots, αποδεικνύοντας τα πλεονεκτήματα της χρήσης Σχεσιακών Συνελικτικών Δικτύων Γράφων. Εμπνευσμένοι από αυτή τη μέθοδο, χρησιμοποιούμε νευρωνικά δίκτυα γράφων στο μοντέλο μας και χρησιμοποιούμε το σύνολο δεδομένων Twibot-20.

Ακολουθούμε την προεπεξεργασία που εισάγεται για το BotRGCN [38]. Η αναπαράσταση κάθε χρήστη περιλαμβάνει μεταδεδομένα που υποβάλλονται σε προεπεξεργασία ως εξής:

- **Συνολικά**: Η περιγραφή του χρήστη, τα tweets, οι αριθμητικές και οι κατηγορικές ιδιότητες κωδικοποιούνται και συνδέονται για να αντιπροσωπεύσουν τελικά τα χαρακτηριστικά του χρήστη:

$$r = [r_b; r_t; r_p^{num}; r_p^{cat}] \in \mathbb{R}^{D\times 1} \tag{1.18}$$

  όπου D είναι οι διαστάσεις διανυσμάτων του χρήστη. Η επεξεργασία και η αναπαράσταση κάθε διαφορετικού χαρακτηριστικού εξηγείται παρακάτω.

- **Περιγραφή χρήστη**: Οι περιγραφές των χρηστών κωδικοποιούνται με το προεκπαιδευμένο RoBERTa:

$$\bar{b} = RoBERTa(\{b_i\}_{i=1}^{L}), \bar{b} \in \mathbb{R}^{D_s\times 1} \tag{1.19}$$

όπου το $\bar{b}$ υποδηλώνει την αναπαράσταση της περιγραφής του χρήστη και το D_s είναι η διάσταση του RoBERTa. Στη συνέχεια προκύπτουν τα διανύσματα για την περιγραφή του χρήστη ως:

$$r_b = \phi(W_B \cdot \bar{b} + b_B), r_b \in \mathbb{R}^{D/4 \times 1} \tag{1.20}$$

όπου τα W_B και b_B είναι παράμετροι με δυνατότητα εκμάθησης, το ϕ είναι η συνάρτηση ενεργοποίησης και το D είναι η διάσταση των διανυσμάτων.

- **Tweets χρήστη**: Τα tweets του χρήστη κωδικοποιούνται επίσης χρησιμοποιώντας το RoBERTa. Η τελική αναπαράσταση των tweets του χρήστη r_t είναι ο μέσος όρος των αναπαραστάσεων όλων των tweets.

- **Αριθμητικές ιδιότητες χρήστη**: Οι αριθμητικές ιδιότητες του χρήστη υιοθετούνται απευθείας από το API του Twitter χωρίς feature engineering και παρουσιάζονται στον παρακάτω πίνακα. Για αυτές τις πληροφορίες, πραγματοποιείται z-score κανονικοποίηση για να ληφθεί η αναπαράσταση r_p^{num}.

Feature Name	Description
#followers	number of followers
#followings	number of followings
#favorites	number of likes
#statuses	number of statuses
active_days	number of active days
screen_name_length	screen name character count

Πίνακας 1.3: *Αριθμητικές ιδιότητες του χρήστη*

- **Κατηγορικές ιδιότητες χρήστη**: Οι κατηγορικές ιδιότητες του χρήστη κωδικοποιούνται επίσης με τη χρήση MLPs και GNNs, χωρίς feature engineering, όπως και οι αριθμητικές ιδιότητες. Έχουν υιοθετηθεί απευθείας από το API του Twitter και παρουσιάζονται στον παρακάτω πίνακα. Μετά από μια απλή κωδικοποίηση, ενώνονται και μετασχηματίζονται με ένα πλήρως συνδεδεμένο επίπεδο και μία leaky-relu για να λάβουν τελικά την αναπαράστασή τους r_p^{cat}.

1.5.2 Σχεσιακά συνελικτικά νευρωνικά δίκτυα γράφων

Η μέθοδός μας δημιουργεί έναν ετερογενή γράφο από τις σχέσεις ακολούθησης. Οι χρήστες θεωρούνται κόμβοι και οι σχέσεις 'following' και 'followers' αντιπροσωπεύονται ως ακμές που συνδέουν τους κόμβους. Επομένως, οι «ακόλουθοι» του χρήστη αντιπροσωπεύονται διαφορετικά από τους «ακολούθους» του χρήστη. Ο ετερογενής γράφος που κατασκευάζεται μπορεί να αντιπροσωπεύει καλύτερα τις σχέσεις μεταξύ των χρηστών και περισσότερες σχέσεις μεταξύ των χρηστών θα μπορούσαν να ενσωματωθούν στο γράφημα εάν υποστηρίζονται από το σύνολο δεδομένων. Οι χρήστες περιέχουν επίσης τα συνδεδεμένα μεταδεδομένα που περιγράψαμε παραπάνω.

Για να συνδυάσουμε τις αναπαραστάσεις των χρηστών με τις σχέσεις μεταξύ των χρηστών, χρησιμοποιούμε Σχεσιακά συνελικτικά νευρωνικά δίκτυα γράφων (RGCNs). Ο αγωγός διέλευσης μηνυμάτων των RGCNs περιλαμβάνει δύο τύπους ατομικών λειτουργιών: διάδοση

Feature Name	Description
protected	protected or not
geo_enabled	geo-location enabled or not
verified	verified or not
contributors_enabled	enable contributors or not
is_translator	is translator or not
is_translation_enabled	translation or not
profile_background_tile	the background tile
profile_user_background_image	background image or not
has_extended_profile	extended profile or not
default_profile	the default profile
default_profile_image	the default profile image

Πίνακας 1.4: *Κατηγορικές ιδιότητες του χρήστη*

(Π) αναπαραστάσεων των γειτόνων του και εφαρμογή μετασχηματισμού (Τ) στις αναπαραστάσεις. Παρακάτω περιγράφουμε τη διαδικασία πίσω από τις δύο λειτουργίες:

- **Διάδοση (Propagation)**: Η διάδοση περιλαμβάνει συνάθροιση μηνυμάτων από γειτονικούς κόμβους χωρίς μετασχηματισμό χαρακτηριστικών κόμβων. Η μαθηματική έκφραση για το βήμα της διάδοσης είναι η εξής:

$$h_i^{(l+1)} = \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} \tag{1.21}$$

  όπου $h_i^{(l+1)}$ είναι το νέο χαρακτηριστικό κόμβου μετά τη διάδοση, R είναι το σύνολο των σχέσεων, N_i^r είναι οι γείτονες του κόμβου με τη σχέση r, το $c_{i,r}$ είναι μια σταθερά κανονικοποίησης για συγκεκριμένο πρόβλημα που μπορεί είτε να μάθει το δίκτυο είτε να επιλεγεί εκ των προτέρων (για παράδειγμα $c_{i,r} = N_i^r$) και $W_r^{(l)}$ είναι ο πίνακας βάρους για τη σχέση r.

- **Μετασχηματισμός (Transformation)**: Ο μετασχηματισμός πραγματοποιείται σε κάθε κόμβο με βάση τις σχέσεις. Η μαθηματική έκφραση για το βήμα μετασχηματισμού είναι η εξής:

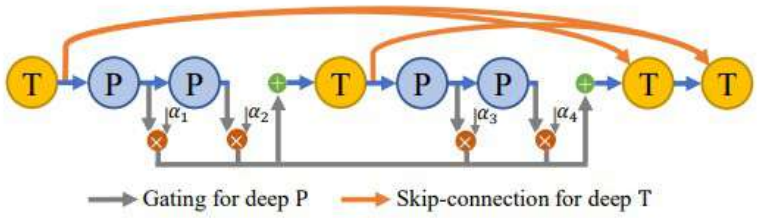
$$h_i^{(l+1)} = W_{root} h_i^{(l)} + \sum_{r \in R} (W_r h_i^{(l)}) \tag{1.22}$$

  όπου $h_i^{(l+1)}$ είναι η νέα δυνατότητα κόμβου μετά τον μετασχηματισμό, W_{root} είναι η μήτρα βάρους εκμάθησης για τον ριζικό κόμβο, W_r είναι η μήτρα βάρους εκμάθησης για τη σχέση r και R είναι το σύνολο των σχέσεων.

Διαχωρίζουμε αυτούς τους δύο τύπους συναρτήσεων αφού συνδυασμοί τους θα δημιουργήσουν τον χώρο αναζήτησης για την Αναζήτηση Αρχιτεκτονικής Νευρωνικού Δικτύου.

1.5.3 Αναζήτηση αρχιτεκτονικής νευρωνικών δικτύων

Η χρήση των νευρωνικών δικτύων γραφικών προσφέρει αναμφισβήτητα πλεονεκτήματα στο έργο της ανίχνευσης bot. Ωστόσο, η μεγιστοποίηση της απόδοσής τους μπορεί να απαιτεί εκτενή μηχανική χαρακτηριστικών. Αυτός είναι ο λόγος για τον οποίο χρησιμοποιούμε την Αναζήτηση Αρχιτεκτονικής, χρησιμοποιώντας το μοντέλο ΔΦΓ-ΝΑΣ [6]. Οι περισσότερες μέθοδοι G-NAS υιοθετούν έναν σταθερό αγωγό διέλευσης μηνυμάτων με δύο τύπους ατομικών λειτουργιών: διαδίδουν (P) αναπαραστάσεις των γειτόνων του κάθε κόμβου και εφαρμόζουν μετασχηματισμό (T) στις αναπαραστάσεις. Επίσης, οι περισσότερες μέθοδοι G-NAS έχουν σταθερό μήκος αγωγού διέλευσης μηνυμάτων, καθώς η απόδοση μειώνεται με περισσότερες λειτουργίες P, το οποίο αναφέρεται ως ζήτημα υπερβολικής εξομάλυνσης. Οι λειτουργίες διάδοσης και μετασχηματισμού αντιστοιχούν στην αύξηση και στον μετριασμό της εξομάλυνσης αντίστοιχα. Αναζητούμε έναν αγωγό διέλευσης μηνυμάτων που αποτελείται από λειτουργίες P και T, χρησιμοποιώντας έναν γενετικό αλγόριθμο. Χρησιμοποιούμε επίσης μηχανισμούς πύλης (gate operation) και παράκαμψης (skip-connection) στις λειτουργίες P και T αντίστοιχα.



Εικόνα 1.7: *Παράδειγμα διαύλου στον χώρο αναζήτησης. Πηγή [6]*

Ο χώρος αναζήτησης περιλαμβάνει συνδυασμούς P-T και τον αριθμό των λειτουργιών P-T. Επιπλέον, προστίθενται συνδέσεις μεταξύ κάθε τύπου. Για μια λειτουργία P ή T σε ένα επίπεδο GNN στο μοντέλο, το $o_v^{(l)}$ είναι η έξοδος του κόμβου v στο επίπεδο l και τα L_P και L_T είναι δύο σύνολα με τους δείκτες στρώματος όλων των πράξεων P και πράξεων T αντίστοιχα. Οι λειτουργίες και οι συνδέσεις των στρωμάτων περιγράφονται παρακάτω.

Συνδέσεις διάδοσης: Ένα επικείμενο πρόβλημα στα GNN είναι η υπερ- ή υπο-εξομάλυνση, η οποία οφείλεται σε πολλές ή πολύ λίγες λειτουργίες διάδοσης. Για να επιτευχθεί κατάλληλη ομαλότητα για διαφορετικούς κόμβους, οι λειτουργίες P ενισχύονται με τον μηχανισμό πύλης (Gate operation). Η έξοδος της l-στής λειτουργίας P είναι η ενσωμάτωση κόμβου που έχει διαδοθεί του $o^{(l-1)}$ εάν η επόμενη πράξη είναι επίσης P. Εάν η επόμενη πράξη είναι T, ένα βάρος που διαδίδεται από όλες τις προηγούμενες λειτουργίες P εφαρμόζεται. Διατυπωτικά:

$$z_v^{(l)} = P(o_v^{(l-1)}) \tag{1.23}$$

$$o_v^{(l)} = \begin{cases} z_v^{(l)} & followed\ by\ P \\ \sum_{i \in L_P, i \leq l} softmax(a_i) z_v^{(i)} & followed\ by\ T \end{cases} \tag{1.24}$$

όπου $a_i = \sigma(s \cdot o_v^i)$ είναι το βάρος για την έξοδο του i-στου επιπέδου του κόμβου v. Το s είναι το εκπαιδεύσιμο διάνυσμα που μοιράζονται όλοι οι κόμβοι και το σ υποδηλώνει τη

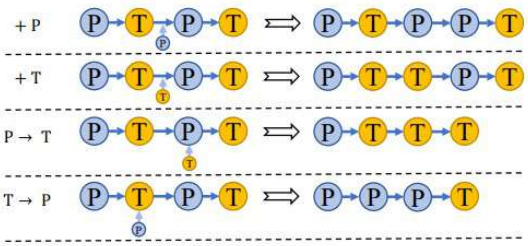
σιγμοειδή συνάρτηση.

Συνδέσεις Μετασχηματισμού: Ένα επικείμενο πρόβλημα με τα GNN είναι η υποβάθμιση του μοντέλου (degradation issue), που προκαλείται από υπερβολικό αριθμό πράξεων μετασχηματισμού και μειώνει την ακρίβεια του μοντέλου. Για να αντιμετωπιστεί αυτό το ζήτημα, χρησιμοποιούνται μηχανισμοί παράλειψης (skip-connection) στις λειτουργίες T. Η είσοδος κάθε πράξης T είναι το άθροισμα της εξόδου του τελευταίου επιπέδου και των εξόδων όλων των προηγούμενων πράξεων T πριν από το τελευταίο επίπεδο. Η είσοδος και η έξοδος της l-στής T λειτουργίας μπορούν να διαμορφωθούν ως εξής:

$$z_v^{(l)} = o_v^{(l-1)} + \sum_{i \in L_T, i < m(l)} o_v^{(i)} \tag{1.25}$$

$$o_v^{(l)} = \sigma(z_v^{(l)} w^{(l)}) \tag{1.26}$$

όπου $m(l)$ είναι ο δείκτης της τελευταίας πράξης T πριν από το l-στο επίπεδο και $W(l)$ είναι η παράμετρος στην l-στη λειτουργία T.



Εικόνα 1.8: *Απεικόνιση των 4 διαφορετικών μεταλλάξεων. Πηγή [6]*

Οι εξελικτικοί αλγόριθμοι είναι μια κατηγορία αλγορίθμων βελτιστοποίησης που εμπνέονται από τη βιολογική εξέλιξη. Για να εξελίξουν τον πληθυσμό των ατόμων εφαρμόζουν μεταλλάξεις. Κάθε αρχιτεκτονική GNN κωδικοποιείται ως ακολουθία λειτουργιών P και T. Τέσσερις διαφορετικές περιπτώσεις μετάλλαξης μπορούν να συμβούν σε μια τυχαία θέση στην ακολουθία:

- **+P**: προσθήκη μιας λειτουργίας διάδοσης
- **+T**: προσθήκη μιας λειτουργίας μετασχηματισμού
- **P→T**: αντικατάσταση μια λειτουργίας διάδοσης με μια μετασχηματισμού
- **T→P**: αντικατάσταση μια λειτουργίας μετασχηματισμού με μια διάδοσης

Αρχικά, k διαφορετικές αρχιτεκτονικές GNN δημιουργούνται τυχαία ως το αρχικό σύνολο πληθυσμού Q και στη συνέχεια αξιολογούνται στο σύνολο επικύρωσης. Στη συνέχεια, m (m < k) άτομα από τον πληθυσμό επιλέγονται τυχαία και αυτό με την καλύτερη απόδοση επιλέγεται ως γονέας A. Η θυγατρική αρχιτεκτονική B δημιουργείται από την τυχαία εφαρμογή μιας από τις τέσσερις μεταλλάξεις. Το B αξιολογείται και προστίθεται στον πληθυσμό και στη συνέχεια αφαιρείται το γηραιότερο άτομο. Αυτή η διαδικασία επαναλαμβάνεται για T γενιές και επιστρέφεται η επιλεγμένη αρχιτεκτονική με την καλύτερη απόδοση. Μπορούμε

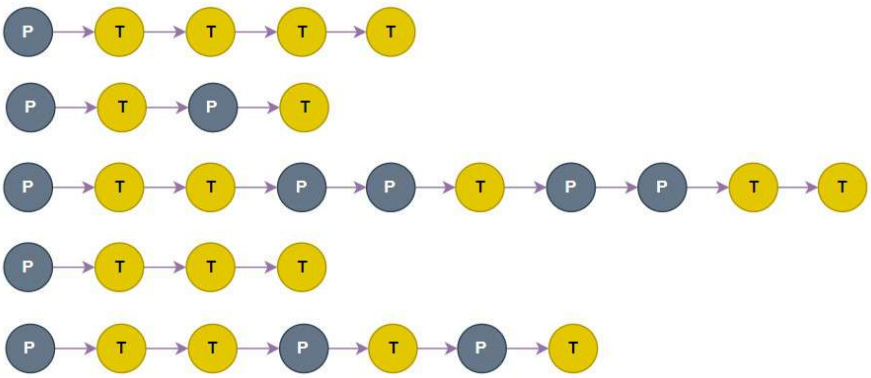
να επιλέξουμε μια αρχιτεκτονική με την υψηλότερη ακρίβεια στο σετ επικύρωσης ή στο σετ εκπαίδευσης ή μια με συνδυασμό και των δύο.

1.6 Πειραματική διαδικασία

Το πείραμα εκτελέστηκε στο Google Colab χρησιμοποιώντας GPU T4. Για την αναζήτηση αρχιτεκτονικής, ο αριθμός του πληθυσμού k είναι 15 και ο αριθμός των γενεών T είναι 80. Ο προϋπολογισμός εκπαίδευσης κάθε αρχιτεκτονικής GNN είναι 70 εποχές. Αυτοί οι αριθμοί, αν και περιορισμένοι λόγω των πόρων μας, παρέχουν ένα καλό παράδειγμα της αποτελεσματικότητας του μοντέλου μας. Η εκπαίδευση γίνεται χρησιμοποιώντας τον Adam optimizer με ρυθμό εκμάθησης (learning rate) 0,04. Το κριτήριο είναι η απώλεια διασταυρούμενης εντροπίας (Cross Entropy Loss) και ο συντελεστής κανονικοποίησης (regularization factor) είναι 2e-4. Εφαρμόζεται dropout σε όλα τα διανύσματα χαρακτηριστικών με ρυθμό 0,5 και dropout μεταξύ των διαφορετικών επιπέδων GNN είναι 0,8.

Μετά την εκτέλεση της μεθόδου NAS επεξεργαζόμαστε τα αποτελέσματα και εξετάζουμε τις πέντε αρχιτεκτονικές με την καλύτερη ακρίβεια στο σύνολο επικύρωσης. Κάθε αρχιτεκτονική πλέον εκπαιδεύεται για 100 εποχές στο σύνολο δεδομένων TwiBot-20 [39], για να δημιουργήσει ένα ετερογενές γράφημα 229.580 κόμβων και 227.979 ακμών. Το σύνολο εκπαίδευσης αποτελεί το 70% του συνόλου δεδομένων, το σύνολο επικύρωσης το 20% και το σύνολο δοκιμής το 10%. Η εκπαίδευση πραγματοποιείται επίσης με τη χρήση του Adam optimizer με ρυθμό εκμάθησης 1e-3. Στη συνέχεια, κάθε αρχιτεκτονική δοκιμάζεται στο σύνολο δοκιμής για να ληφθούν τα αποτελέσματα, που θα παρουσιάσουμε παρακάτω.

Κάθε αρχιτεκτονική κατά τη διάρκεια της αναζήτησης αποθηκεύεται με το συνδυασμό P-T, την ακρίβεια στο σύνολο επικύρωσης και την ακρίβεια στο σύνολο δοκιμής. Στην παρακάτω εικόνα απεικονίζονται οι πέντε αρχιτεκτονικές που επιλέγονται από τη μέθοδο NAS.



Εικόνα 1.9: *Κορυφαίες 5 αρχιτεκτονικές απόδοσης. Οι ακρίβειες επικύρωσης από το NAS (από πάνω προς τα κάτω) είναι: 87,01%, 86,99%, 86,95%, 86,89%, 86,82%*

Αυτές οι αρχιτεκτονικές εκπαιδεύονται και δοκιμάζονται από την αρχή στο σύνολο δεδομένων TwiBot-20. Παρουσιάζουμε όλες τις μετρικές που επιτεύχθηκαν για όλες τις αρχιτεκτονικές.

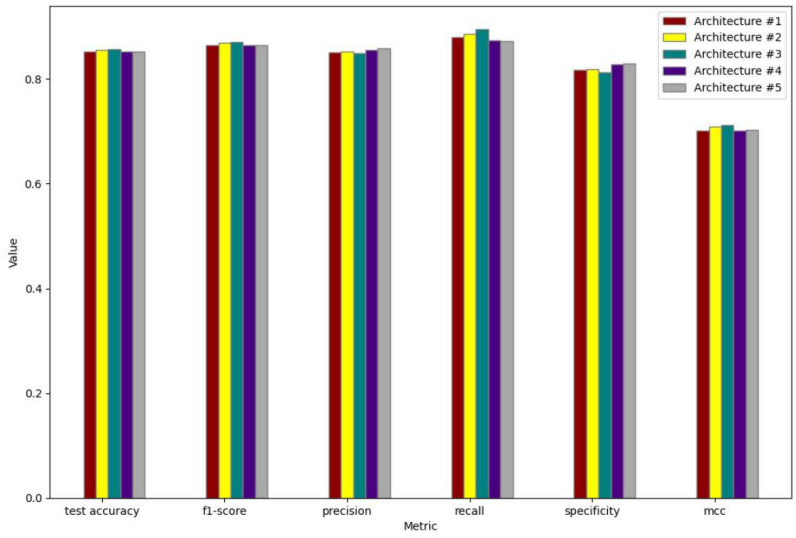
Όλες οι επιλογές επιτυγχάνουν καλές μετρήσεις και παρουσιάζουν πλεονεκτήματα στον εντοπισμό των bot έναντι μεθόδων αιχμής. Αυτά τα αποτελέσματα υπογραμμίζουν τα σημα-

Αρχιτεκτονική	Accuracy	F1-score	Precision	Recall	Specificity	MCC
1η	0.852	0.865	0.851	0.88	0.818	0.702
2η	0.855	0.869	0.853	0.886	0.819	0.709
3η	**0.857**	**0.871**	0.849	**0.895**	0.812	**0.712**
4η	0.852	0.864	0.856	0.873	0.828	0.702
5η	0.852	0.864	**0.858**	0.872	**0.829**	0.703

Πίνακας 1.5: *Επίδοση των αρχιτεκτονικών από την αναζήτηση αρχιτεκτονικής*

ντικά πλεονεκτήματα που προκύπτουν από τη χρήση τεχνικών αναζήτησης αρχιτεκτονικής στον τομέα της ανίχνευσης bot. Επιπλέον, καθιερώνουν την αποτελεσματικότητα της χρήσης των χαρακτηριστικών των χρηστών και των σχέσεων μεταξύ των χρηστών στην ανίχνευση bot.

Μετά από προσεκτικότερη εξέταση των αποτελεσμάτων, η δεύτερη αρχιτεκτονική έχει την υψηλότερη ακρίβεια. Το πέμπτο μοντέλο έχει επίσης το υψηλότερο specificity. Επιπλέον, όλες οι αρχιτεκτονικές έχουν υψηλές μετρικές ακρίβειας, F1-score και MCC. Όποια αρχιτεκτονική κι αν επιλέξουμε θα μπορούσε να ανταγωνιστεί άλλα μοντέλα. Στο εξής θα αναφερόμαστε στη δεύτερη αρχιτεκτονική ως το μοντέλο μας, αφού παρέχει την υψηλότερη ακρίβεια.



Εικόνα 1.10: *Σύγκριση των αρχιτεκτονικών από την αναζήτηση αρχιτεκτονικής*

Στον παρακάτω πίνακα παρουσιάζουμε την απόδοση άλλων μεθόδων στο σύνολο δεδομένων TwiBot-20 σε σύγκριση με του δικού μας. Βλέπουμε ότι το μοντέλο μας επωφελείται από την αναζήτηση της βέλτιστης αρχιτεκτονικής που πραγματοποιήσαμε εκ των προτέρων, καθώς επιτυγχάνει υψηλότερη ακρίβεια, F1-score και MCC από άλλα αξιοσημείωτα προηγούμενα μοντέλα.

1.7 Επίλογος

Το αντικείμενο μελέτης μας ήταν η ανίχνευση bots στο Twitter. Το Twitter είναι μια κοινωνική πλατφόρμα που έχει γνωρίσει ταχεία ανάπτυξη και ως εκ τούτου η παρουσία

Μοντέλο	Accuracy	F1-score	MCC
Lee et al.	0.7456	0.7823	0.4879
Yang et al.	0.8191	0.8546	0.6643
Kuduganta et al.	0.8174	0.7517	0.6710
Wei et al.	0.7126	0.7533	0.4193
Cresci et al.	0.4793	0.1072	0.0839
Miller et al.	0.4801	0.6266	-0.1372
Botometer	0.5584	0.4892	0.1558
SATAR	0.8412	0.8642	0.6863
BotRGCN	0.8462	0.8707	0.7021
δικό μας	**0.8568**	**0.8712**	**0.7116**

Πίνακας 1.6: *Απόδοση των μοντέλων στο σύνολο δεδομένων TwiBot-20*

αυτοματοποιημένων λογαριασμών, γνωστών ως bots, είναι μεγαλύτερη από ποτέ. Στοχεύουν στη διάδοση ψεύτικων πληροφοριών και τη χειραγώγηση των χρηστών, κατακλύζοντας τα χρονολόγια και τα σχόλια των χρηστών. Ενώ τα bots αποκτούν την ικανότητα να μιμούνται καλύτερα την ανθρώπινη συμπεριφορά, το έργο της ανίχνευσής τους γίνεται όλο και πιο δύσκολο.

Εστιάζουμε στη χρήση νευρωνικών δικτύων γράφων (GNNs), τα οποία έχουν σχεδιαστεί για να χειρίζονται δεδομένα με δομή γράφων. Συγκεκριμένα, εστιάζουμε στα συνελικτικά δίκτυα γραφήματος (GCNs), τα οποία περιλαμβάνουν τεχνικές για τη μετάδοση μηνυμάτων μεταξύ των κόμβων για τη λήψη πληροφοριών από ολόκληρο το γράφο. Εμπνεόμαστε από το BotRGCN [38], που κατασκευάζει ένα ετερογενές γράφημα, όπου οι χρήστες αναπαρίστανται με κόμβους και οι σχέσεις ακολούθησης μεταξύ τους με ακμές. Οι χρήστες περιέχουν επίσης πληροφορίες, συμπεριλαμβανομένων των περιγραφών τους, των tweets τους και αριθμητικών και κατηγορικών ιδιοτήτων τους.

Σε μια προσπάθεια να βελτιώσουμε την απόδοση του μοντέλου, εξετάσαμε την υλοποίηση του Graph Neural Architecture Search, μια διαδικασία που επιστρέφει την αρχιτεκτονική του νευρωνικού δικτύου με την υψηλότερη ακρίβεια. Εμπνεόμαστε από το DFG-NAS [6], το οποίο αναζητά την πιο αποτελεσματική μετάθεση πράξεων διάδοσης (P) και μετασχηματισμού (T) που υλοποιούνται στο GNN. Επομένως, ξεπερνάμε τους περιορισμούς μιας σταθερής αρχιτεκτονικής. Με τη χρήση των λειτουργιών Gate και skip-connection, αποφεύγουμε την υπερβολική εξομάλυνση και την υποβάθμιση του μοντέλου που θα μείωναν την ακρίβεια του μοντέλου. Η αναζήτηση ακολουθεί έναν εξελικτικό αλγόριθμο, μια μέθοδο εμπνευσμένη σε μεγάλο βαθμό από τη φυσική εξέλιξη που επιχειρεί να ενισχύσει έναν πληθυσμό μέσω μεταλλάξεων.

Το μοντέλο μας χρησιμοποιεί σχεσιακά GCNs μετά την εκτέλεση αναζήτησης αρχιτεκτονικής, για να βρει την πιο αποτελεσματική διαμόρφωση P-T. Χρησιμοποιήσαμε το σύνολο δεδομένων Twibot-20. Καταλήξαμε ότι η καλύτερη αρχιτεκτονική είναι αυτή με υποδειγματική ακρίβεια εκπαίδευσης και ακρίβεια δοκιμής. Εκπαιδεύσαμε το μοντέλο και επιτύχαμε ακρίβεια 85,7%, ξεπερνώντας άλλα μοντέλα για ανίχνευση βοτ.

Τα αποτελέσματα που προέκυψαν από αυτήν την εργασία υπογραμμίζουν τα πλεονεκτήματα της χρήσης Νευραλ Αρχιτεκτυρε Σεαρχη και Ρελατιοναλ ΓΝ στο έργο του εντοπι-

σμού βοτ. Είναι ιδιαίτερα ικανοποιητικές και ενθαρρυντικές στην κατεύθυνση της περαιτέρω έρευνας.

1.7.1 Μελλοντικές προεκτάσεις

Πρώτον, είναι σημαντικό να αναγνωρίσουμε τους περιορισμούς των πόρων μας. Η χρήση πόρων που θα μπορούσαν να υποστηρίξουν περισσότερες γενιές στην αναζήτηση αρχιτεκτονικής θα μπορούσε να είναι επωφελής για την έρευνά μας, καθώς θα μπορούσε να εξετάσει βαθύτερες αρχιτεκτονικές. Έτσι, ίσως ένα μοντέλο με περισσότερα επίπεδα θα μπορούσε να παρουσιάσει ακόμα καλύτερα αποτελέσματα.

Μια πρώτη επέκταση της τρέχουσας εργασίας θα μπορούσε να είναι η εφαρμογή του μοντέλου σε άλλα σύνολα δεδομένων. Αυτό το μοντέλο θα μπορούσε πιθανώς να προσαρμοστεί ώστε να χρησιμοποιεί περισσότερες πληροφορίες χρήστη. Για παράδειγμα, θα μπορούσαμε να βελτιώσουμε την εκπαίδευση προσθέτοντας πληροφορίες όπως η ζώνη ώρας, ώρα των tweet κ.λπ. Εκτός από τα μεταδεδομένα του χρήστη, θα μπορούσαμε να προσθέσουμε περισσότερες σχέσεις χρηστών, όπως μηνύματα, retweets κ.λπ. Μια άλλη περαιτέρω μελέτη θα μπορούσε να είναι η αναγνώριση των bots σε άλλα μέσα κοινωνικής δικτύωσης, όπως το Facebook ή το Instagram.

Η εφαρμογή του μοντέλου στον πραγματικό κόσμο θα μπορούσε να σημαίνει την προσαρμογή του για την αναγνώριση bots σε πραγματικό χρόνο. Αυτό θα μπορούσε να πραγματοποιηθεί χρησιμοποιώντας μια εφαρμογή που λαμβάνει υπόψιν τα μεταδεδομένα χρήστη κατά τη δημιουργία του λογαριασμού για την αναγνώριση bots. Αυτό θα μπορούσε επίσης να εφαρμοστεί σε σύντομα χρονικά 'παράθυρα' (ημερήσια, ωριαία) για να ληφθούν επίσης υπόψιν οι σχέσεις μεταξύ των χρηστών.

Κεφάλαιο 2

Introduction

2.1 The World of Social Media

Founded by Andrew Weinreich in 1997, Six Degrees is considered to be the first social networking site [34]. Named after the "six degrees of separation" theory, which suggests that two individuals are connected through a chain of no more than six intermediate connections, Six Degrees offered users the ability to create profiles, make friends, and send messages. It reached its peak in the late 1990s, attracting 3.5 million users but eventually shut down in 2001, unable to generate revenues to sustain its growth. Despite the unavoidable fall, it is now considered the trailblazer that paved the way for social media platforms, as we know them today

By definition, social media refers to platforms and applications that enable users to create, share, and exchange content within virtual communities. Social media content can be text, photos, videos, GIFs, audio, etc. Somebody can use social media for various reasons, from communicating with others with similar interests to getting informed about current worldwide events.

The existence of social media in our day-to-day lives is more prevalent than ever. As of 2023, there are roughly 4.9 billion social media users [35], a percentage that is more than 60% of the world's population and more than 100 social media platforms. The average user approximately spends an average of 2 hours and 31 minutes daily on social media. The most notable platforms are Facebook, Instagram, Twitter, WhatsApp, YouTube, and TikTok, congregating the vast majority of social media users.

2.2 Twitter

Twitter is a popular social media platform that was launched in 2006 [36] by Jack Dorsey, Biz Stone, and Evan Williams, and has since grown into a global platform with 330 million active users. Users can post and interact with texts, images, videos, and links, referred to as "tweets", making it a great platform for expressing thoughts, sharing news, and engaging in discussions. At first, tweets were limited to 140 characters, until November 2017 when they were extended to 280 characters.

Twitter revolves around the concept of "following" other users. A user can follow accounts they are interested in seeing their tweets in their timeline ("following") and

conversely has "followers" that see their tweets. Users can reply and mention other accounts with the character "@" followed by the account's name. Twitter also incorporates the use of "hashtags", which are keywords or phrases preceded by the "#" character, and categorizes tweets, making it more comfortable to search specific topics. Users can also search tweets based on their content using words that are included, with the results presented in a combination of chronological and relativity order. In addition, Twitter offers features such as Twitter Moments, which curates collections of tweets around specific events or topics, and Twitter Spaces, which allows users to host live audio conversations.

Twitter taking advantage of its features has been established as a powerful tool for real-time news updates, public discourse, and social movements, and continues to evolve and enhance user experience.

2.3 Fake News and Bots

Fake news is false information, stories, or hoaxes created to misinform or deceive readers [37]. Fake news can even mislead people by looking like trusted websites using similar names and web addresses to respected news organizations.

Social media bots are accounts that automate interactions on social media platforms, often mimicking human behavior. These bots can be programmed to execute various tasks, such as automatically publishing content, liking, sharing, following, or commenting on posts. Some can even be programmed to engage in conversations to promote specific agendas.

Fake news and bots have had significant real-world consequences in several cases. Throughout the COVID-19 pandemic, misinformation about the virus and the vaccines spread rapidly on social media. This led to mob panic, confusion, promotion of dangerous medications, and even resistance to public health measures, jeopardizing public health. In 2016 fake news stories spread widely during the U.S. presidential election campaign. False narratives and fabricated stories aimed to influence public opinion to sway voters. Fake news spread has also resulted in the manipulation of the stock markets and artificial fluctuations in financial markets. Bots can be used to harass individuals or target specific groups, engaging in coordinated campaigns and flooding comment sections. This leads to the creation of a hostile environment for online users. Bots can also generate and spread spam content, malicious links and malware, compromising the security and privacy of social media users.

2.4 Purpose of this Thesis

The need to detect bot accounts to shut them down is quite immediate, assessing the hazards of their uncontrollable presence on social media. This is why many platforms have implemented algorithms to detect that type of accounts. However, bots can adapt their functions to simulate genuine users, making their detection a challenging task. There have been several methods that can perform with high accuracy, leveraging properties like followers and following count, verification marks, the content of tweets, etc.

One innovative model for bot detection that we chose to optimize is BotRGCN [38], which is short for Bot detection with Relational Graph Convolutional Networks. State-of-the-art models are limited due to their fixed structures. We will overcome this performance restriction with the Neural Architecture Search technique, aiming to achieve higher accuracies than other models and pave a new way to bot detection. Briefly, the models and the dataset we used which will be described in detail in the following chapters:

- We utilized the Twibot-20 Dataset [39] for our experiments. This is a comprehensive sample of the Twittersphere and is representative of the present age of Twitter bots and real users. It consists of four different territories: politics, business, entertainment, and sports. Each user has semantics, property, and neighborhood information. These qualities make it an optimal dataset to build the GNNs and study the classification method of the accounts to bots and real users.

- We handle the challenge of the community aspects of Twitter space, by constructing a heterogeneous graph from follower-following relationships and then applying relational graph convolutional networks. We follow BotRGCN's preprocessing and leverage users' semantical and property information and neighborhood information to augment its ability to detect disguised bots, outperforming other competitive models.

- Neural Architecture Search (NAS) for Graph Neural Networks (GNNs) is an innovative approach that can leverage random search, supervised learning or evolutionary algorithms to discover optimal network architectures. NAS explores a vast space of possible network configurations, intelligently mutating and recombining architectures to uncover those with superior performance. Evolutionary algorithms incorporate principles inspired by natural evolution, where only the fittest architectures survive and reproduce, leading to efficient and specialized GNN models.

2.5 Thesis Structure

In Chapter 1, an extended summary of the entire thesis is presented entirely in Greek following the same structure.

In Chapter 2, a brief introduction about the origins of social media was given, and a description of the issues caused by fake news and bot accounts, as well as the purpose and structure of the current thesis.

In Chapter 3, we introduce some background information on the theoretical parts of the thesis. Definitions and explanations about graphs, machine learning algorithms and models and word embeddings are presented in order to understand vital parts of the experiment.

In Chapter 4, we present related work done in the field of bot and fake news detection, as well as studies for Neural Architecture Search in a plethora of use cases.

In Chapter 5, we give some insight of the dataset that we use in the experiments and perform a validation check to ensure its adequacy.

In Chapter 6, we present our model. We describe the processing of the data, the use of Graph Convolutional Networks in bot detection and the process of Neural Architecture Search.

In Chapter 7, we present the results and compare them to other state-of-the-art methods we have presented before. Finally, we perform an ablation study, to prove the integrity of our method.

In Chapter 8, we complete this thesis presenting the final conclusions of our work, the limitations and suggestions for future research.

Κεφάλαιο 3

Theoretical Background

In this Chapter, we will introduce some basic principles of graphs, Machine Learning, and word embeddings that we used during the process of the experiment and for further understanding. Specifically, section 3.1 presents the basic terms used in graphs and some fundamental algorithms as well as real-world use cases. Section 3.2 presents the different branches of Machine Learning. Section 3.3 analyzes the most common traditional machine learning algorithms. In section 3.4 there is an introduction to some of the most well-known artificial neural networks for a variety of applications. Their architecture, capabilities, and limitations are summarized. In Section 3.5, we focus on Graph Neural Networks, presenting some key concepts, their types, and applications. Section 3.5 presents some key ideas on Neural Architecture Search (NAS). Section 3.6 discusses evaluation methods for a neural network model and gives an introduction to the principles of cross validation. At last section 3.7 presents some common word embedding models, including the one we will make use of RoBERTa.

3.1 Graph Structure and Algorithms

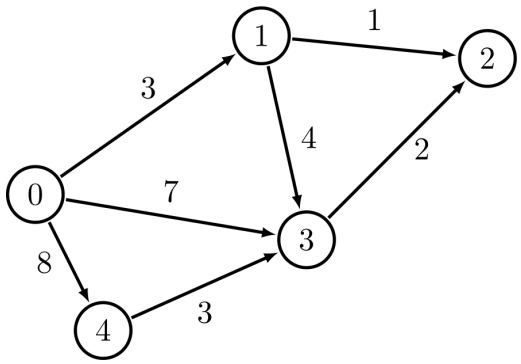
3.1.1 Graph Structure

A graph is defined as G = V, E where V is the set of vertices and E is the set of edges (connections between vertices). They are non-linear data structures that are used to model relationships or connections between elements. There are many types of graphs, with the most common variants being directed or undirected and weighted or unweighted. An undirected graph is a type of graph where the edges have no specific direction, meaning the connections between nodes are bidirectional. A directed graph is a type of graph where all the edges have a specific direction. A weighted graph is a type of graph where each edge has an associated numerical value called a weight. These weights can represent metrics as the distance between nodes, cost, time, etc. An unweighted graph is a type of graph where all edges have the same default weight or simply no weight associated with them and only the connection between vertices is significant.

Some basic terminologies used in graph theory:

- The total number of edges occurring to a vertex in a graph is called degree.

- The in-degree of a node in a directed graph is the number of incoming edges directed towards that node.
- A path is a set of alternating vertices and edges, where the vertices are connected by the edges.
- If the path starts and finishes on the same vertex, it is known as a cycle.
- A spanning subgraph that is also a tree is known as a spanning tree.
- A connected component is the unconnected graph's most connected subgraph.
- A bridge is an edge that if removed would seperate the graph to two unconnected graphs.
- A forest is a graph without a cycle.

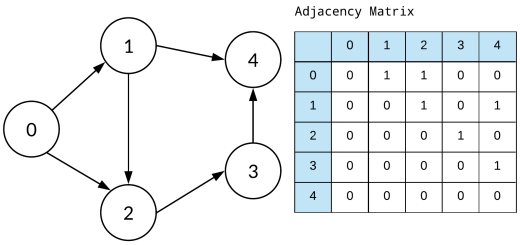


Εικόνα 3.1: *Example of directed and weighted graph. Source [1]*

3.1.2 Representations for graphs

The two most common ways to represent a graph are with an adjacency matrix and an adjacency list.

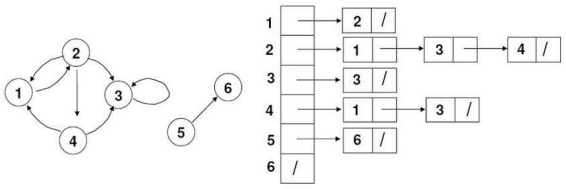
An adjacency matrix is a 2D array where the rows and columns represent the nodes in a graph, and the value of each cell indicates whether there is an edge between the corresponding nodes. In weighted graphs the value in the cells is usually the weight of each edge, while in unweighted graphs it is a boolean value (0 or 1). If there is an edge, the value is typically 1. If there is no edge, the value is usually 0. In a directed graph the value of a cell is 1 if there is an edge from source to destination. An example of a graph representation with an adjacency matrix is shown below:



	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Εικόνα 3.2: *Graph to adjacency matrix. Source [7]*

An adjacency list is a collection of lists that represent the connections between nodes in a graph. Each node is associated with a list of its neighboring nodes (adjacent nodes). It is a very efficient way to represent sparse graphs. In directed graphs, each node is considered the source and associated with a list of its destinations. An example of a graph representation with an adjacency list is shown below:



Εικόνα 3.3: *Graph to adjacency list. Source [8]*

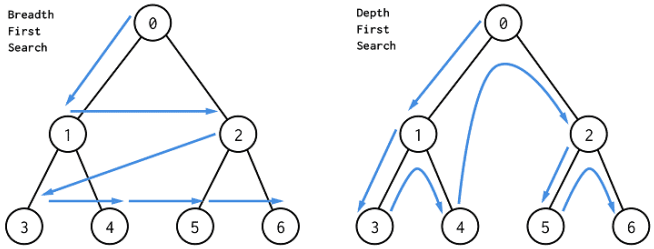
Adjacency lists are more memory efficient, due to their dynamic size, but are slower for edge detection, so they are not suitable for dense graphs. On the other hand, adjacency matrices have a fixed size and are less memory efficient, but are more effective in edge detection.

3.1.3 Graph Algorithms

There are plenty significant algorithms implemented in graphs in order to solve real world use cases. Some of the most well known algorithms are presented below.

BFS (Breadth-First Search): It is a graph traversal algorithm that explores all the vertices in a graph level by level, starting from a given source vertex. It uses a queue structure to keep track of the visited vertices to ensure that nodes closer to the source are visited before those farther away. BFS can be utilized in unweighted graphs to find the shortest path, as it explores level by level.

DFS (Breadth-First Search): It is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It uses a stack structure to keep track of the visited vertices and their order so it can explore as deep as possible on each branch before backtracking to explore other paths. DFS can be used for cycle detection, as it detects back edges (edges connecting a node to an ancestor).



Εικόνα 3.4: *BFS and DFS algorithms traversal. Source [9]*

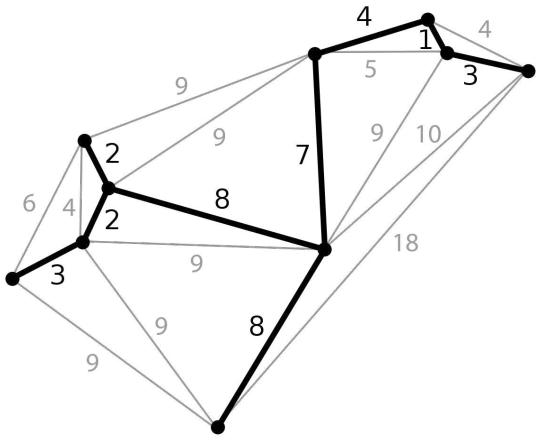
Dijkstra's Algorithm: It is a greedy algorithm that finds the shortest path from a single source vertex to all other vertices in a weighted graph with non-negative edge weights. It initializes the distances from the source to all the other nodes to infinity and then updates them using the minimum distance of all the unvisited notes at each iteration. It terminates when the paths cease to update.

Bellman-Ford Algorithm: It is a dynamic programming algorithm that is used to find the shortest path from a single source vertex to all other vertices in a weighted graph. Bellman-Ford can handle graphs with negative edge weights, unlike Dijkstra's algorithm. The algorithm iterates through all edges for a number of iterations (equal to the number of vertices minus one) and updates the distances to nodes considering all the possible paths. Bellman-Ford can also be used to detect negative cycles in a weighted graph.

Kruskal's Algorithm: It is a greedy algorithm used to find the minimum spanning tree (MST) in a weighted, undirected graph, meaning a tree with all the vertices of the graph and the minimum total weight. The algorithm works by repeatedly selecting the edges with the least weight that don't form a circle. When all the vertices are included the MST is formed. The time complexity of the algorithm is O(ElogV).

Prim's Algorithm: It is another greedy algorithm that finds the minimum spanning tree (MST) in a weighted, undirected graph. It starts with an random node and keeps growing the MST by selecting the shortest edge that connects the existing MST to a new vertex, ensuring that the tree remains connected. The time complexity of the algorithm is also O(ElogV).

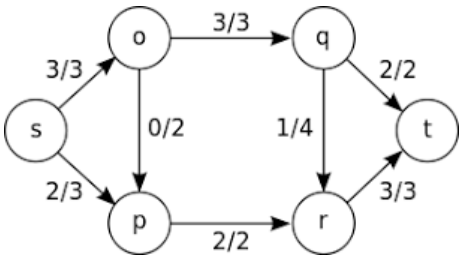
Boruvka's Algorithm: It is a greedy algorithm that similarly finds the MST in a weighted, undirected graph. It works by iteratively growing the MST in multiple phases, where in each phase it selects the cheapest edge for each connected component of the graph. The algorithm continues until the entire graph becomes a single connected component. The time complexity of the algorithm is also O(ElogV).



Εικόνα 3.5: *Minimum Spanning Tree. Source [10]*

Floyd-Warshall Algorithm: It is a dynamic programming algorithm used to find the shortest paths between all pairs of vertices in a weighted graph. The algorithm maintains a matrix where each entry represents the shortest distance between two vertices. It efficiently computes the shortest distances, considering all possible intermediate vertices in each iteration. It works for both directed and undirected graphs. It can also handle graphs with negative weights, but can not be used in graphs with negative cycles.

Ford-Fulkerson algorithm: The Ford-Fulkerson algorithm is a widely used algorithm to solve the maximum flow problem in a flow network. The maximum flow problem involves determining the maximum amount of flow that can be sent from a source vertex to a sink vertex, including the capacity constraints on the edges. The algorithm repeatedly searches for augmenting paths from the source to the sink and increases the flow along these paths until no other augmenting paths can be found. In practice the Edmonds-Karp algorithm, a specific variant of Ford-Fulkerson using BFS for finding augmenting paths, is often preferred due to its better time complexity.



Εικόνα 3.6: *Example of graph with flow and capacity. Source [11]*

Tarjan's algorithm: It is a graph traversal and analysis algorithm used to find strongly connected components in a directed graph. A strongly connected component is a subgraph where there is a path between any pair of nodes. The algorithm uses a depth-first search (DFS) approach to explore the graph and a stack to keep track of the visited nodes. As it traverses the graph, it adds nodes to the stack and assigns DFS order numbers and low-link values. When a complete strongly connected component is found, its nodes are popped from the stack.

3.1.4 Graph Applications

Graphs are a flexible and extremely useful data structure for many real-world applications. Some of them will be presented below, to fathom the importance of their study and the capabilities in our problem.

Graph algorithms are widely used in social networks like Facebook, Twitter, and LinkedIn. They are used to suggest connections (named as mutual friends) to the users, identify communities, and recommend content that would interest a user.

Many recommendation systems, like those used by Netflix and Amazon, use graph algorithms to analyze user interactions and preferences. Based on similar user behavior and item resemblances, they can suggest movies and products.

PageRank is a graph-based algorithm developed by Google, used to rank web pages based on their importance. The importance of a webpage is determined by the number and quality of the links pointing to it. It constitutes the foundation of Google's search engine.

Graph algorithms are used in mapping and GPS navigation applications to find the shortest or fastest routes between locations. They can also adapt to various constraints like traffic and road conditions. They are also very useful in determining the most efficient routes in transportation and logistics networks and help in optimizing the flow of goods, data, information, etc.

Graph algorithms can also be employed to analyze biological data, such as protein-protein interaction networks, molecular structures, DNA sequences, etc. They help visualize biological data and further understand complex biological structures and interactions.

Graph-based image segmentation algorithms use a graph representation of pixels to group them into regions based on similarities. These segments can be used in applications such as image analysis and object detection.

Game developers use graph algorithms to model game worlds, and design AI decision-making systems. Simple graph algorithms can also be used for path-finding in mazes.

3.2 Branches of Machine Learning

Machine Learning is a field of Artificial intelligence that enables systems to learn from data so they can make predictions only based on it. In order to do this, they observe the data to uncover patterns. Machine Learning is imminent in many real-world applications, such as the Internet of Things (IoT), fraud detection and cybersecurity, Natural Language Processing (NLP), Recommendation Systems, healthcare, etc. They mainly fall into four main categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

3.2.1 Supervised Learning

Supervised learning involves using labeled data to train algorithms, so we can accurately classify unlabeled data, according only to their features. Supervised machine

learning is divided into two types of problems: classification and regression. In classification problems, the output is categorical or discrete. It involves assigning the input data to predefined classes. Some implementations include: identifying spam emails and classifying images that contain an object. In regression problems, the output is continuous or numerical. It involves predicting a value or quantity based on input data. Cases of regression problems include: predicting house prices based on a variety of factors and estimating the sales volume based on advertising expenditure.

3.2.2 Unsupervised Learning

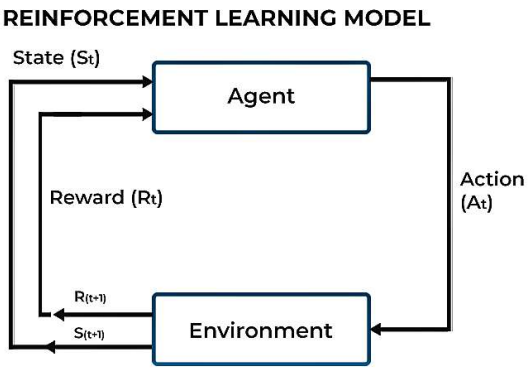
Unsupervised learning uses unlabeled data to make predictions. The main goal of unsupervised learning algorithms is to find groups of features that follow a similar pattern of similarities and differences. Applications of unsupervised learning include customer segmentation to recommend products, outlier detection in data, forecasting models, etc.

3.2.3 Semi-Supervised Learning

Semi-supervised learning combines labeled and unlabeled data during training. This method first uses unsupervised learning algorithms to group similar data and then gives labels to the previously unlabeled data. This approach is useful when obtaining labeled data can be either challenging or simpler. Semi-supervised learning can be used in the fields of medical image analysis, sentiment analysis in text, speech recognition, etc.

3.2.4 Reinforcement Learning

Reinforcement learning is a machine learning technique where an agent relies on environmental feedback to take action. This technique does not make any use of labeled data but uses a trial-and-error approach with a feedback-based process. The model learns from experience to improve performance over time. Implementations of reinforcement learning include intelligent robotics, personalized plans, gamification, etc.



Εικόνα 3.7: *Reinforcement Learning Model. Source [12]*

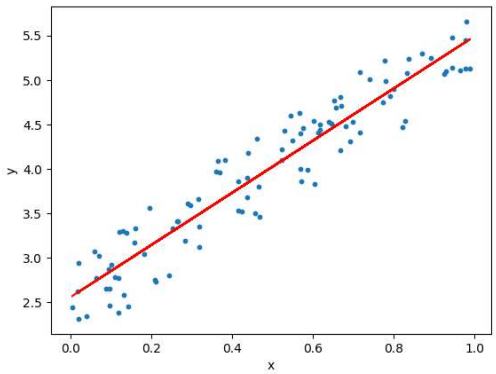
3.3 Machine Learning Algorithms

3.3.1 Linear Regression

Linear regression is a supervised learning algorithm in machine learning, used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable and the one you are using for the prediction is called the independent variable. The goal is to find the optimal line, that minimizes the difference between the predicted values and the actual values of the output variable. This line is determined by estimating the coefficients (slope and intercept) that define the relationship between the input features and the output variable. In simple linear regression, there is only one input feature, and the relationship between the feature and the output variable is represented by a straight line. The equation for simple linear regression can be written as:

$$y = m * x + b \tag{3.1}$$

where y is the predicted value, x is the input feature, m is the slope or coefficient that represents the relationship between x and y and b is the y-intercept. In multiple linear regression, there are multiple input features, and the relationship between the features and the output variable is represented by a hyperplane. The process of finding the line involves estimating the coefficients that minimize a cost function, for example the sum of squared errors.



Εικόνα 3.8: *Linear Regression. Source [13]*

3.3.2 Logistic Regression

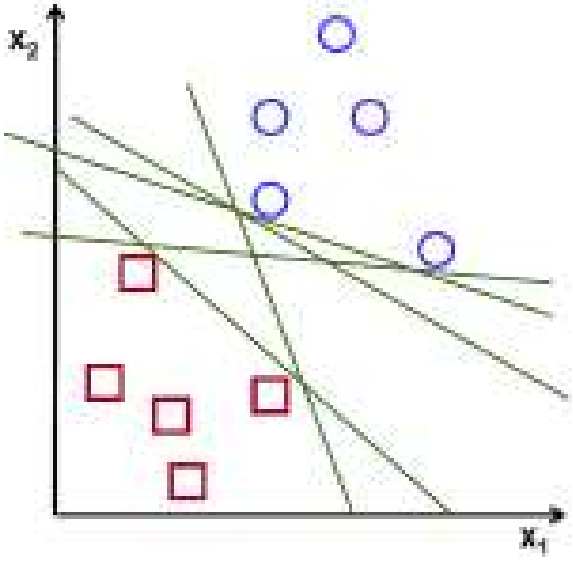
Logistic regression is used in classification problems, with the simplest form being binary sorting, where the output y takes two distinct values (usually 0 or 1). Then, the hypothesis h(x) can be expressed by the following sigmoid function:

$$h(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

The hypothesis output h(x) equals the probability that a sample of the data set belongs to a particular class. It can also be extended to handle multi-class classification problems and does not produce a linear output

3.3.3 Support Vector Machine

Support vector machines are a part of supervised learning algorithms and are based on the graphical representation of the various elements and their separation into classes. SVMs are constructed with the idea of finding an optimal hyperplane that separates classes, by maximizing the margin between the data points. This hyperplane is known as the maximum-margin hyperplane. The closest data of the two classes in terms of the hyperplane are called support vectors. For example, while the two classes are separated by all the lines, none of them maximizes the margin between them.



Εικόνα 3.9: *Example of linear separation of classes. Source [14]*

We can consider the training set of n points x_i, y_i where i=1,..,n and y_i is a binary value (-1,1) for the class of x_i. We want to find a hyperplane which can map the points into higher dimensional space and the two different classes of points can be divided with the maximum margin. In linear cases, the hyperplane can be written as:

$$\mathbf{x}_i\mathbf{w} + b = 0 \tag{3.3}$$

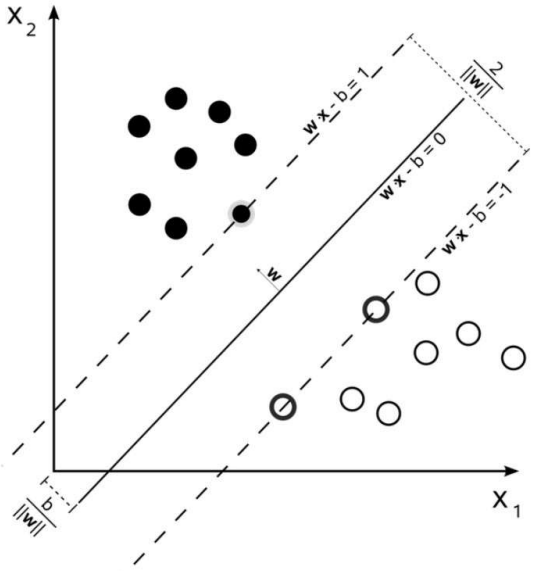
We want to find two parallel hyperplanes that can also separate the data and we want their distance to be as large as possible. We describe them as:

$$\mathbf{x}_i\mathbf{w} + b = +1 \tag{3.4}$$

$$\mathbf{x}_i\mathbf{w} + b = -1 \tag{3.5}$$

We can prove that the distance between these two hyperplanes is $\frac{\|w\|^2}{2}$. The figure below shows the support vectors, the separation margin, the hyperplanes satisfying equations

(3.4) & (3.5) and the maximum margin hyperplane.



Εικόνα 3.10: *Support vector machines. Source [15]*

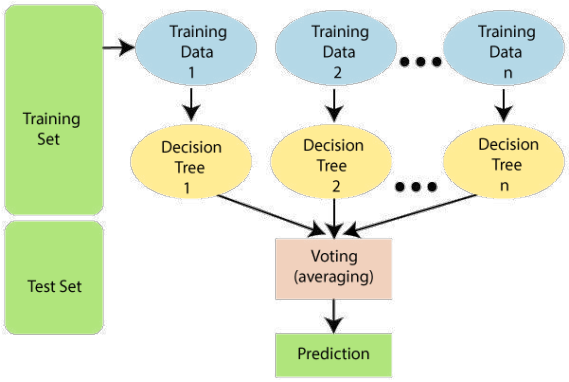
In cases where the training data are not separated linearly, there are some transformations, called kernels, where the non-separable data are separated in more than 2 dimensions.

3.3.4 Decision Trees

Decision trees are a popular and intuitive supervised machine learning algorithm used for both classification and regression tasks. They represent a flowchart-like structure where each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents the outcome or prediction. In a decision tree, the goal is to split the data into homogeneous subsets based on the values of the input features, ultimately creating decision rules for predicting the target variable. The more important a feature is in predicting the class, the higher it is in the tree. Common algorithms for constructing decision trees include ID3 (Iterative Dichotomiser 3), C4.5, CART (Classification and Regression Trees), and Random Forests.

3.3.5 Random Forests

Random Forests consist of a large number of Decision Trees and belong to the family of ensemble methods, a technique that combines machine learning algorithms for better results. In classification problems, each Decision Tree, individually, predicts the class to which the specific training data belongs, and in the end the class, which was predicted the most, is the class of our final prediction. In regression problems, the output is the average of the predictions of all the individual trees. It is required that the predictions and thus the errors of the individual Decision Trees have a low correlation with each other. This can be achieved by using the bagging (Bootstrap Aggregating) technique.



Εικόνα 3.11: *Random Forests. Source [16]*

3.3.6 Naive Bayes

Naive Bayes classifier is a popular supervised machine learning algorithm used for classification tasks. It is based on the probabilistic principle of Bayes' theorem and assumes that the features are conditionally independent given the class label. If $X = (x1, x2, .., xn)$ is a vector of features. The probability that the vector X belongs to one of the classes C_i, where i=1,...,k is equal to:

$$p(C_i|x) = \frac{p(x|C_i)\,p(C_i)}{p(x)} \tag{3.6}$$

According to the Naive Bayes' theorem, the Naive Bayes classifier calculates for the feature vector X the probabilities of belonging to each of the classes and finally classifies it into that class for which the probability is greater.

A very common variant of this classifier is Gaussian Naive Bayes. In this case, we assume that the data follow a normal distribution. For each feature x_i with mean μ_y and variance σ_y^2, and class y, the probability density function is assumed to be Gaussian, that is:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{3.7}$$

3.3.7 k-Nearest Neighbors

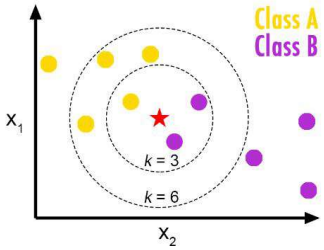
The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm used for both classification and regression tasks. A point in the feature space is classified into the class that is the most common among its k nearest neighbors, the criterion being a distance function. Most common functions are the Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \tag{3.8}$$

and the Manhattan distance:

$$d(M, P) \equiv |M_x - P_x| + |M_y - P_y| \tag{3.9}$$

The choice of the distance metric and the value of k can significantly impact the performance of the KNN algorithm. Larger values of k provide a smoother decision boundary but may introduce more bias, while smaller values of k can lead to a more flexible decision boundary but may be more sensitive to noise in the data. In binary classification problems, it is preferable to choose an odd value for k, in order to avoid a tie. For example for k=3 in the following image the point (indicated by the star) would be classified in class B, while for k=6 it would be classified in class A.



Εικόνα 3.12: *kNN algorithm with k=3 and k=6. Source [17]*

KNN is often used in applications such as recommender systems, image recognition, anomaly detection etc. It is particularly useful when there is a large amount of training data and the decision boundary is non-linear or irregular.

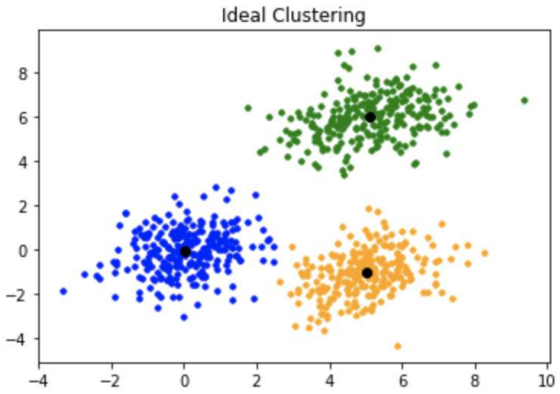
3.3.8 k-Means

The k-means algorithm is an unsupervised machine learning algorithm used for clustering, which is the process of grouping similar instances together based on their feature similarity. It is a simple and widely used algorithm for partitioning a dataset into k distinct clusters that works as follows:

1. Choose the number of clusters k, called cluster centroids.
2. Assign each instance to the nearest centroid based on a distance metric
3. Recalculate the centroid of each cluster by taking the mean of all instances assigned to that cluster.
4. Alternate between assigning instances to the nearest centroid and updating the centroids until the centroids no longer change significantly or a maximum number of iterations is reached.

The random centroid initialization is very important, as the algorithm could have different solutions based on the initial conditions. To reduce errors due to wrong initializations, the algorithm is often run multiple times with different initializations, and the best solution in terms of the minimized WCSS (Within-Cluster Sum of Square) is selected.

K-Means also requires specifying the number of clusters in advance, which can be challenging if the optimal number is not known. It also assumes that the clusters are more spherical and have similar variance, making it less effective for datasets with irregularly shaped or overlapping clusters. However, it is a computationally efficient algorithm and has applications in various domains, including image segmentation, customer segmentation, anomaly detection, and document clustering.

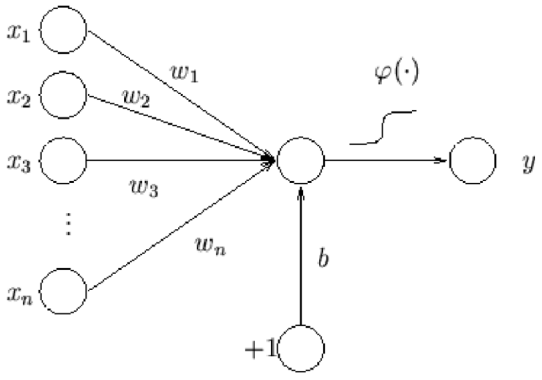


Εικόνα 3.13: *Ideal example of k-Means algorithm with k=3. Source [18]*

3.4 Artificial Neural Networks

3.4.1 Single Layer Perceptron

The single-layer perceptron (SLP) is the simplest form of an artificial neural network and a fundamental concept in the field of neural network research. It is a type of feedforward neural network with no hidden layers, consisting of just one layer of neurons that directly connect the input to the output.



Εικόνα 3.14: *Single Layer Perceptron. Source [19]*

Each individual feature of the input **x** is multiplied by a numerical weight from **w**. The weight indicates how important each feature is. After the inputs are multiplied by their respective weights, they are summed together with a numerical bias value b and

the output is passed through an activation function ϕ. This is the final output and the equation is:

$$y = \sigma(\mathbf{w}\mathbf{x} + b) \tag{3.10}$$

where,

- y: the output of the perceptron
- σ: the activation function
- w: represents the weight vector, containing the weights associated with each input feature
- x: the input vector, representing the input features
- b: the bias term, which allows shifting the decision boundary

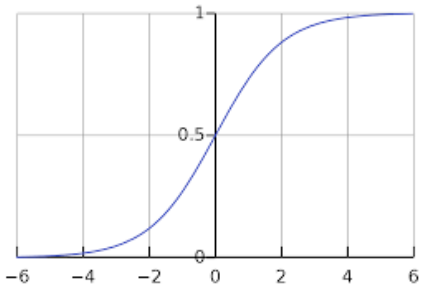
3.4.2 Activation Functions

Activation functions are an essential component of neural networks used to introduce non-linearity into the network. They operate on the output of a neuron or a layer and determine whether the neuron should be activated or not. Some of the most common activation functions are:

- **Sigmoid**: The sigmoid activation function compresses the input into the range (0, 1).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.11}$$

The graph is given in the following figure:

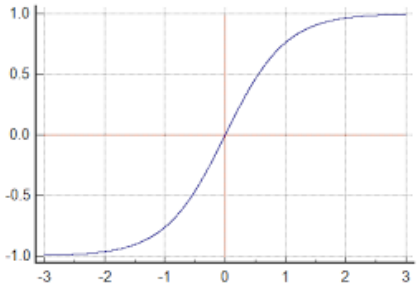


Εικόνα 3.15: *Sigmoid Activation Function. Source [20]*

- **Tanh (Hyperbolic Tangent)**: Tanh is similar to the sigmoid function but compresses the input into the range (-1, 1).

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.12}$$

The graph is given in the following figure:

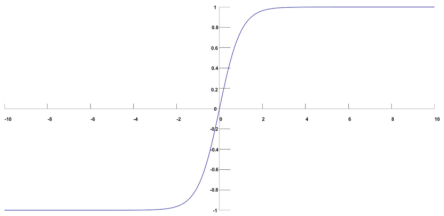


Εικόνα 3.16: *Tanh Activation Function. Source [21]*

- **Softmax**: The softmax activation function is commonly used in the output layer for multiclass classification tasks. It transforms the raw output values into a probability distribution, where each output represents the probability of the input belonging to a particular class. Softmax is defined as:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{3.13}$$

where z_i is the raw output of the i-th class, and N is the total number of classes. The graph is given in the following figure:

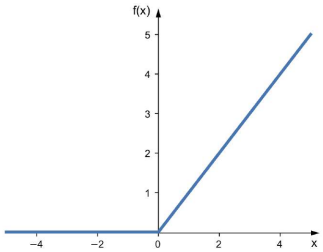


Εικόνα 3.17: *Softmax Activation Function. Source [22]*

- **ReLU (Rectified Linear Unit)**: It is one of the most popular activation functions used in deep learning. It returns the input value if it is positive and zero otherwise. Mathematically, ReLU is defined as:

$$Relu(z) = max(0, z) \tag{3.14}$$

The graph is given in the following figure:

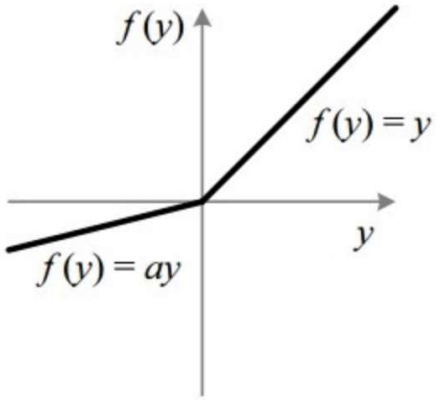


Εικόνα 3.18: *ReLU Activation Function. Source [23]*

- **Leaky ReLU (LReLU)**: It is a variation of the ReLU activation function that addresses the "dying ReLU" problem. In the standard ReLU, when the input is negative, the output becomes zero, and the neuron effectively becomes inactive. The Leaky ReLU introduces a small slope for negative inputs, which allows a small, non-zero output for negative values. The mathematical definition of Leaky ReLU is:

$$LReLU(z) = max(az, z) \tag{3.15}$$

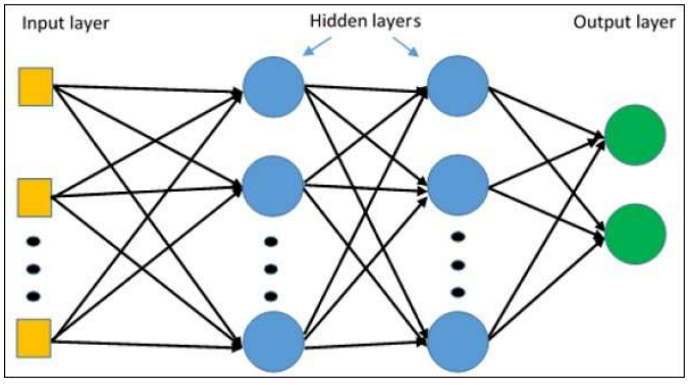
The graph is given in the following figure:



Εικόνα 3.19: *Leaky ReLU Activation Function*

3.4.3 Multilayer Perceptron (MLP)

The Multi-Layer Perceptron is an extension of single-layer perceptron, with at least one hidden layer between the input layer and the output layer. What distinguishes the hidden layers from the output layer, is that the output of these neurons serve as an input for the next layers (hidden or output layer). Each layer consists of numerous neurons connected to the neurons of the next layer through weights, similar to the single-layer perceptron. The number of neurons in the output layer is equal to the number of labels we wish to classify the data. The common structure of an MLP with two hidden layers is shown in the figure below:

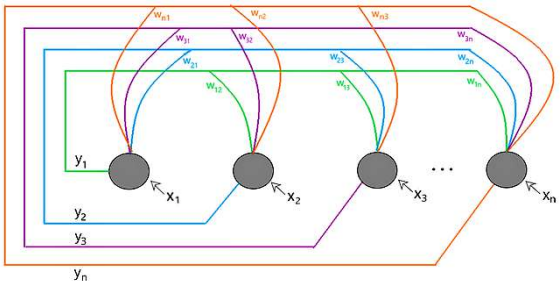


Εικόνα 3.20: *Multilayer Perceptron (MLP). Source [24]*

While MLPs have been instrumental in advancing the field of neural networks and remain an essential building block in deep learning, more complex architectures like convolutional neural networks (CNNs) for images, recurrent neural networks (RNNs) for sequential data, and transformers for natural language processing have become more predominant for many specific tasks.

3.4.4 Hopfield neural networks

Hopfield neural networks are a type of recurrent artificial neural network proposed by John Hopfield [60]. Hopfield networks consist of a single layer of neurons with each neuron connecting to every other neuron in the network. These connections are symmetric and can be represented in a connection matrix. At each iteration of the training, the network is presented with an input pattern or initial state. Each neuron receives an input based on its connection weights and the states of other neurons. The input is summed, and the neuron's activation function is applied to determine the new state of the neuron. Additionally at each iteration, the network tries to minimize its energy function. The network updates the neuron states iteratively until the energy reaches a minimum, which corresponds to a stable state and the convergence of the model. One of the key features of Hopfield networks is their ability to function as an associative memory, meaning they can recall and converge to stored patters when presented with a partial or noisy version of a learned pattern.



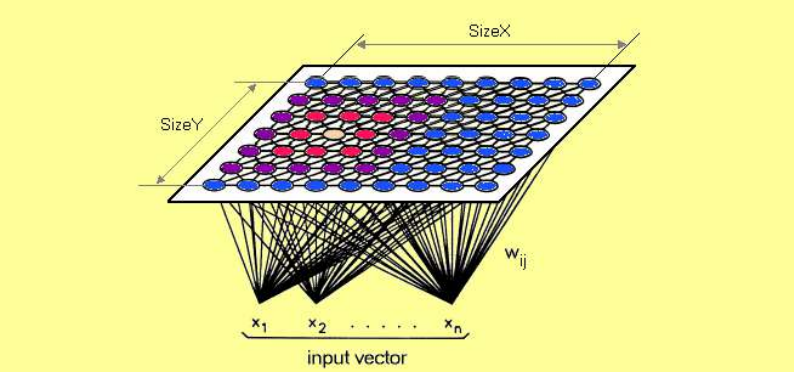
Εικόνα 3.21: *Hopfield Neural Network. Source [25]*

While Hopfield networks were a significant development in the history of neural networks, they have been largely overshadowed by more advanced architectures in deep learning as CNNs and RNNs. However, besides their historical importance, they remain an interesting area of study in neural network theory.

3.4.5 Self-Organizing Maps (SOM)

Self-Organizing Maps (SOMs), also known as Kohonen maps after their inventor Teuvo Kohonen [61], are a type of artificial neural network for unsupervised learning. The main idea behind Self-Organizing Maps is to map high-dimensional input data onto a lower-dimensional grid or lattice, typically in 2D or 3D space. Each node or neuron in the grid represents a weight vector with the same dimensionality as the input data. At first, the weight vectors are initialized to random values. The training consists of two main parts:

competition and cooperation. In the phase of the competition, for each input data point, the Euclidean distance with the weight vectors is computed and the closest is named the Best Matching Unit (BMU). In the phase of the cooperation the weights of the BMU are adjusted, so the neighbor moves closer to the input data point, subsequently organizing the neurons based on similarity in smaller neighborhoods. The training process ends with convergence after all the data points are iterated and the neurons have settled to their final positions.

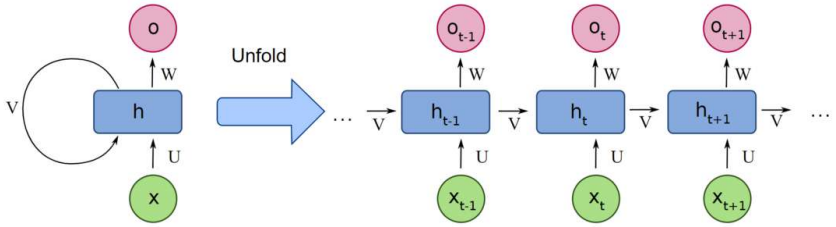


Εικόνα 3.22: *Self Organizing Map (SOM). Source [26]*

SOMs are powerful tools for data visualization and clustering, as they can capture the topological relationships in the input, considering that nearby neurons in the grid tend to represent similar data points. They have been successfully applied in various domains, including pattern recognition, feature extraction, image processing, outliar detection and data clustering.

3.4.6 Recurrent Neural Networks (RNN)

Recurrent Neural Networks are designed to handle sequential data, where each element depends on the previous ones. Applications include natural language processing and visual information resulting from movement. In RNNs, the output of the previous step is fed as input to the current one contrary to traditional neural networks, where all pairs of inputs and outputs are independent of each other. However, in cases where we are asked to predict the next word in a sentence, the previous words are paramount, so the model must have some sort of memory. RNNs manage to solve this problem with the help of their internal hidden state, which holds information about the sequence. The folded image shows the recursive operation of the RNN. The RNN receives the elements of the sequence one after the other and updates its internal or hidden state. Another way of representing RNNs is by "unfolding" the network in time, as shown in the right part of the image. In this case, x_0 is initially taken as input, the output h_0 is produced, which in turn is given as input along with x_1 in the next step. Similarly for the next timesteps and that is how the neural network remembers the content of the sequence.



Εικόνα 3.23: *Recurrent Neural Network. Source [27]*

The general form of the equations of a recurrent network is as follows: The general form of the equations of a recurrent network is as follows:

$$h_t = \phi(Wx_t + Uh_{t-1} + b) \tag{3.16}$$

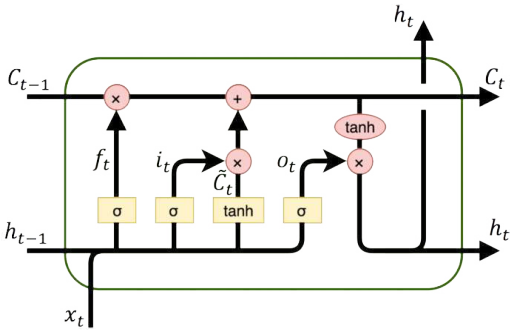
where,

- h_t: the hidden representation at timestep t
- x_t: the element vector of the sequence at timestep t
- *phi*: a non-linear activation function
- W: the array of parameters, which affect the input x_t
- U: the table of parameters, which affect the output of the network in the previous timestep
- b: a polarization vector

In theory, recurrent neural networks (RNNs) are a great choice for information in long sequences. However, in practice, they are limited to shorter lengths due to the vanishing or exploding gradient problem. Long Short Term Memory networks (LSTM) are presented as a solution to that.

3.4.7 Long Short-Term Memory (LSTM) Networks

LSTM, which stands for Long Short-Term Memory, is a type of recurrent neural network (RNN) architecture designed to handle the vanishing gradient problem, a common issue in traditional RNNs. LSTMs were introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997 [62] and have since become a fundamental building block in various deep learning applications, especially in natural language processing, speech recognition, and time series analysis. What sets them apart from simple recurrent neural networks is the architecture of their hidden layer, commonly referred to as an LSTM cell. An LSTM cell has three ports: the forget port, the input port, and the output port. The LSTM cell is depicted in the image below:



Εικόνα 3.24: *Long Short Term Memory Cell. Source [28]*

The first step is to decide what information to erase from memory. This decision is made in the **forget gate**. It takes as input the current input x_t and the output of the previous timestep h_{t-1} and produces a number between 0 and 1 (sigmoid activation function). This output is multiplied by each number of the C_{t-1} vector of the previous state, thus setting which information will be "forgotten".

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.17}$$

The next step is to decide what new information to store in memory. First, through the **input gate** it is decided which values will be updated. The input gate accepts as input the current input x_t and the output of the previous timestep h_{t-1}. Then, the current input x_t and the output of the previous timestep h_{t-1} are passed through a single-layer neural network with hyperbolic tangent activation function, which produces the new candidate cell state values $\tilde{C}_t$, to be added to memory.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.18}$$

$$\tilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.19}$$

In this step, we update the old memory c_{t-1} to the new memory c_t. Specifically, we multiply the forget gate with the values of the old memory. We also add the term it $i_t \odot \tilde{c}_t$. These are the new candidate values, scaled by a decided update value to the current state.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.20}$$

The output is based on a filtered version of the memory state. First, the current input x_t and the output of the previous timestep h_{t-1} are passed through a single-level neural network with a sigmoid activation function to decide which parts of the memory state should participate in the final output. Then, the memory state, is compressed via the tanh function and multiplied by the **output gate**, thus deciding which parts of that state should participate in the final output.

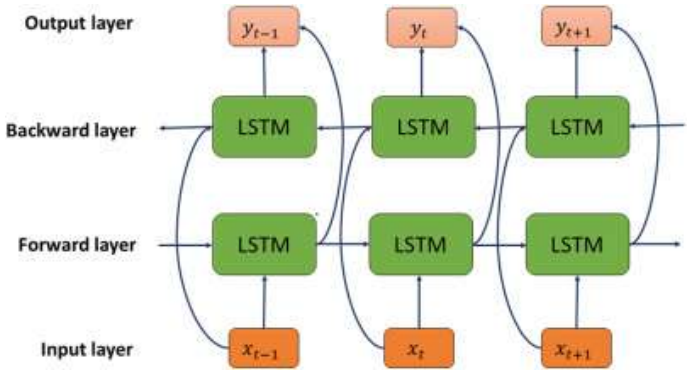
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.21}$$

$$h_t = o_t \odot tanh c_t \tag{3.22}$$

3.4.8 Bidirectional LSTMs

Bidirectional LSTMs (BiLSTMs) are an extension of the traditional LSTM architecture that enables the model to capture information from both past and future time steps in a sequential input sequence, in order to improve the model's performance in sequential classification problems. Unlike standard LSTMs that process the input sequence in a unidirectional manner, BiLSTMs process the sequence in both forward and backward directions simultaneously. For example, in order to predict a missing word in a sentence - sequence, we need to look at the words both before and after the word we are looking for, so that we understand the content of the sentence.

In the figure below, one LSTM processes the sequence from left to right, while the second LSTM processes it from right to left. At each time t, a hidden right-handed LSTM with hidden state $\overrightarrow{h}$ takes as input the previous hidden state $\overrightarrow{h_{t-1}}$ and the input x_t at the current time t. Additionally, a hidden left-handed LSTM with hidden state $\overleftarrow{h}$ takes as input x_t at the current time t and also the future hidden state $\overrightarrow{h_{t+1}}$



Εικόνα 3.25: *Bidirectional LSTM. Source [29]*

While BiLSTMs are effective in many cases, they also come with a higher computational cost compared to unidirectional LSTMs since they process the input sequence twice.

3.4.9 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is a type of artificial neural network and specifically a type of recurrent neural network (RNN). They were proposed [63] as a more computationally efficient alternative to traditional LSTM networks. The key idea behind GRUs is to have fewer gating mechanisms than LSTMs. GRUs are also designed to handle sequential data, using their gates. The primary gates are the reset gate, which determines the part of information that should be forgotten, and the update gate, which controls how much of the previous hidden state should be passed to the current time step. The architecture of GRUs allows them to capture long-range dependencies in sequential data effectively while reducing the vanishing gradient problem, which is present in standard RNNs. They are an effective tool for handling sequential data, such as machine translation, natural language processing, speech recognition, and time series analysis.

3.4.10 Convolutional Neural Networks (CNN)

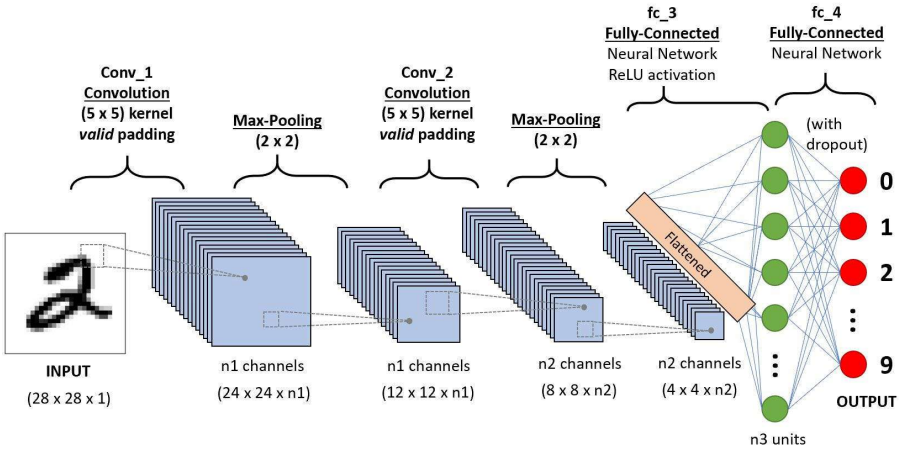
Convolutional Neural Networks (CNNs) [64], while not limited, are a good neural network for visual data processing, such as images and videos. CNNs are widely used in computer vision tasks and have shown remarkable success in image recognition, object detection, and other related tasks. The main components of CNNs are:

The fundamental building block of a CNN is the convolutional layer. It applies a set of learnable filters to the input image, which convolves over an image to produce the feature maps. Each filter is suitable for detecting specific patterns, such as edges or textures.

After the convolution, an activation function is applied element-wise to introduce non-linearity into the network. This allows CNNs to learn complex representations from the input data.

Pooling layers are used to reduce the spatial dimensions of the feature maps and mitigate the computational complexity. A commonly used pooling operation is max-pooling, where the maximum value of a region with defined dimensions is kept, discarding the rest.

After several convolutional and pooling layers, the feature maps are flattened and passed through one or more fully connected layers similar to the ones in traditional neural networks. These layers help make the final decisions, for example in a classification problem.



Εικόνα 3.26: *Convolutional Neural Network. Source [30]*

CNNs can be computationally expensive, memory and time consuming and can fall into overfitting. However they have been at the forefront of recent breakthroughs in computer vision, such as image classification tasks like recognizing objects in images, semantic segmentation, object detection, and more.

3.5 Graph Neural Networks (GNNs)

3.5.1 Types of Tasks on Graphs

The first type of Machine Learning tasks on graph structures are **graph-level tasks**. In these tasks, the goal is to predict a property for the entire graph. Examples that fall into this category are graph classification, where a label for a graph has to be predicted, graph regression, where the goal is to predict continuous values for a graph, and graph generation, where a graph with defined properties has to be constructed. An instance of a graph-level task is Social Network Community Detection. Given a graph with the nodes representing users and the edges representing the relationships between users (friendships, followers, etc.). The task is to categorize the users into classes, such as "family", "friends", and "colleagues", by similar patterns in the users' behaviors and interactions. This task is applicable in social network analysis since the segregation in different communities can help targeted advertising, content recommendation, and understanding social dynamics.

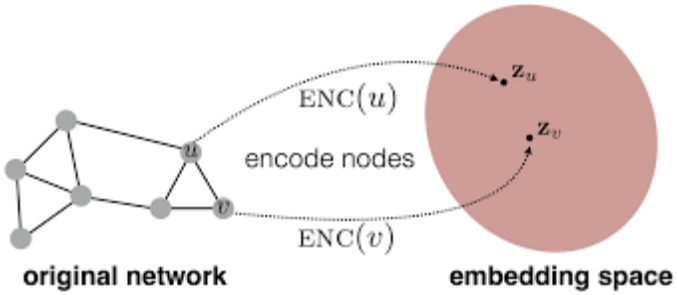
The next type is **node-level tasks** that are concerned with predicting the identity of each node within the graph. These tasks include node classification, where labels are predicted for nodes, node regression, where continuous values are predicted for the nodes, and node clustering, where nodes are assigned to clusters or communities. One real-world example of a node-level task is document classification in a citation network. Each node in this graph represents a research paper with attributes like the abstract, keywords, authors, etc., while the edges represent citations. The task is to classify each research paper into different categories such as "biology," "computer science," "physics," etc. Node classification in a citation network has practical applications in academia and research and it can help in understanding the distribution of topics and finding similar papers.

The last type is **edge-level tasks** that involve making decisions on the edges representing the relationships between the nodes. Such tasks include link prediction, where it is predicted whether an edge between two nodes exists, edge classification, where labels are assigned to edges and edge regression, where continuous values for edges are predicted. The prediction of these problems can thus be far more informative than simple link predictions. For example, it is less illuminating that two users are mutual friends than that two users message each other regularly. One real-world example of an edge-level task is friendship prediction in social networks. The nodes of the graph could represent users and the edges could represent relationships, like messages, friendships, posting shared media, etc. The task is to predict whether two users that are not yet connected should be.

3.5.2 Node Embeddings

Node embeddings are low-dimensional vector representations that capture the structural and relational information of nodes in a graph. As shown in the figure below the goal is to find an embedding space, where the geometric representations z_u and z_v correspond

to nodes the nodes u and v of the graph respectively.



Εικόνα 3.27: *Node embeddings. Source [2]*

Embeddings are used in a variety of machine-learning tasks. For instance in Natual Language Processing word embeddings are used, which are representations of words as low-dimensional vectors, so that words with similar meanings are closer in the embedding space. In the same way, nodes of a graph that are close or belong to the same neighborhood, are expected to have similar node embeddings. This information can be used in various tasks, such as Link Prediction, Node Classification, and Community Detection. Node embeddings are also very useful for dimensionality reduction since working on raw node attributes can be computationally expensive. There are many ways to generate node embeddings, each with its own approach. Some common methods are matrix factorization or random walk-based methods and even GNNs like GCN and GraphSAGE.

3.5.3 Permutation Invariance and Equivariance

Permutation invariance and equivariance are two key concepts in Graph Neural Networks. **Permutation invariance** refers to the property that a GNN should produce consistent results regardless of the order in which the nodes of the graph are processed. Graph data can be depicted using an adjacency matrix, where the node ordering is not fixed. It is crucial that the model's behavior remains invariant to these changes. **Permutation equivariance** relates to how a GNN processes data transformations. If the input undergoes transformations, such as rotations, size changes, etc., the output should display similar transformations. A formal definition of these concepts is provided by Meltzer et al. [40].

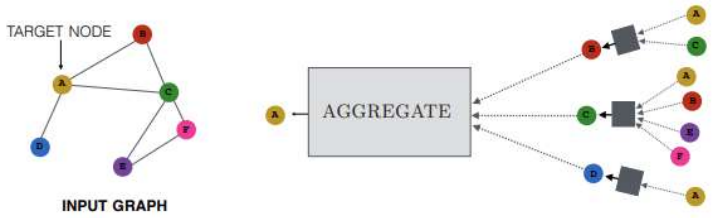
Permutation Invariance: Let P_n be the set of all valid permutation matrices of order n, then a function f is invariant to row permutation iff $f(X) = f(P_\pi X), \forall X \in \mathbb{R}^{n\times m}, P_\pi \in P_n$ and to column permutation iff $f(X) = f(XP_\pi^T), \forall X \in \mathbb{R}^{m\times n}, P_\pi \in P_n$.

Permutation Equivarince: Let P_n be the set of all valid permutation matrices of order n, then a function f is invariant to row permutation iff $P_\pi f(X) = f(P_\pi X), \forall X \in \mathbb{R}^{n\times m}, P_\pi \in P_n$ and to column permutation iff $f(X)P_\pi^T = f(XP_\pi^T), \forall X \in \mathbb{R}^{m\times n}, P_\pi \in P_n$.

3.5.4 The Message Passing Protocol

As stated by Gilmer et al. [41], the message passing framework presents the key idea that the embedding of each node should be generated based on the embeddings

of its neighborhood. We will present a more high-level approach to the mathematical formulation of this protocol. For an undirected graph G we represent the node features as x_v and edge features as e_{uv}.



Εικόνα 3.28: *Message Passing Framework. Source [3]*

As shown in the figure above each node's embedding, denoted by N_A for vertice A, is impacted by the embeddings of its neighborhood. This message passing process is repeated for a fixed number of iterations or until convergence. Each node v has a hidden state vector h_v and at time step t the hidden state h_v^t is updated based on the messages from the neighborhood m_v^{t+1}. The messages are aggregated by an AGGREGATE function, and then the states are updated by an UPDATE function, producing the new hidden state h_v^{t+1}. The equations of this process can be summarized below:

$$m_{N(v)}^{k} = AGGREGATE^{(k)}\left(\{h_u^{(k)}, \forall u \in N(v)\}\right) \tag{3.23}$$

$$h_v^{k+1} = UPDATE^{(k)}(\{h_v^{(k)}, \forall u \in N(v)\}) \tag{3.24}$$

After running the K steps of the message-passing phase, we can use the hidden state h_v^K of each node v as the node's embedding. The readout phase is the final step in the message-passing process, where information from all nodes is aggregated to obtain a global graph-level representation. In the readout phase, there is a feature $\widehat{y}$ computed using a readout function READOUT:

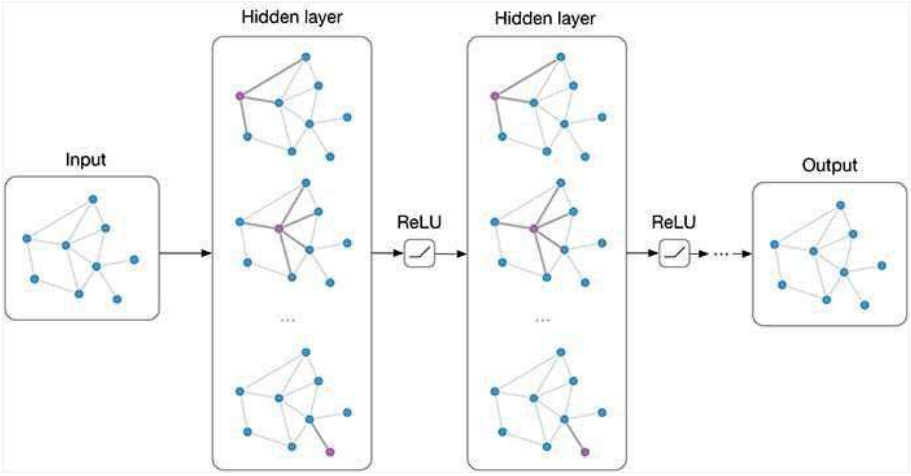
$$\widehat{y} = READOUT(h_v^{(k)} | v \in G) \tag{3.25}$$

An important part of the message-passing framework is that all the nodes must have some initial hidden state h_v^0 which can be a value such as the degree or the centrality of each node. There are various GNN types with different usages of the functions AGGREGATE, UPDATE, and READOUT with learned differentiable functions.

3.5.5 Types of Graph Neural Networks

Graph Neural Networks (GNNs) are a class of neural networks designed to handle data represented in graphs. We have already presented the structure of graphs and their applications in real-world problems, like social networks, biological data, recommendation systems, etc. Data elements are represented as nodes, while the relationships between these elements are represented as edges. GNNs can learn expressive representations by

leveraging both the node-level information and the global graph structure, through the mechanisms of message passing that we have described. Each node aggregates information from its neighbors and updates its own representation. Different types of GNNs use different variations for the functions used in the message passing framework. Below we will present some of the well known types of GNNs



Εικόνα 3.29: *Graph Neural Network. Source [4]*

- **GCNs**: Graph Convolutional Networks (GCNs) are a type of GNNs introduced by Thomas Kipf and Max Welling [42]. They are highly inspired by CNNs adapted to processing data in a graph structure, instead of images. GCNs operate on the principle of message passing. Each node aggregates information from its neighbors by taking a weighted sum of their embeddings and then passes that information through an activation function to update its embedding. GCNs typically consist of multiple graph convolutional layers. Each layer computes the node embeddings by using information from progressively more distant nodes in the graph. In the paper, GCNs are used for semi-supervised learning, where leveraging a small amount of labeled data they can make predictions on the entire graph. There are two types of GCNs: spectral, which transform the graph into a spectral domain for convolution operations, and spatial, which directly aggregate information from neighbors in the original spatial domain.

- **GraphSAGE**: GraphSAGE (SAmple and aggreGatE) is a type of GNN proposed by Hamilton et al. [43] as an alternative to traditional inherently transductive methods for generating node embeddings. State-of-the-art methods often use matrix-factorization-based objectives and therefore are limited to a fixed graph. GraphSAGE on the other hand uses an inductive approach, so it can be extended to unseen nodes, and subsequently to unseen graphs. The key idea is that GraphSAGE does not train a distinct embedding vector for each node, but trains a set of aggregator functions to learn to aggregate feature information from a node's local neighborhood. The forward pass of the algorithm follows the message-passing process of the general Message Passing Framework in K iterations using in every

iteration k the equations:

$$m_{N(v)}^{k} = AGGREGATE^{(k)}(\{h_u^{(k-1)}, \forall u \in N(v)\}) \tag{3.26}$$

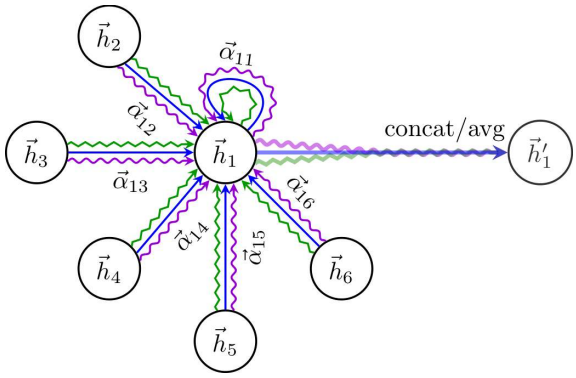
$$h_v^{k+1} = \sigma(W^k CONCAT(h_v^{(k-1)}, m_{N(v)}^{(k)})) \tag{3.27}$$

The AGGREGATE function is ideally a symmetric and trainable function that maintains high representational capacity. The paper examines three functions, notably a mean aggregator, an LSTM aggregator, and a max pooling operator. The node's current representation $h_v^{(k-1)}$ with the aggregated neighborhood vector $h_{N(v)}^{(k-1)}$ are concatenated and the resulting concatenated vector is fed through a fully connected layer with non-linear activation function σ. The paper also presents a relation of GraphSAGE to the Weisfeiler-Lehman Isomorphism Test, also known as "naive vertex refinement". GraphSAGE is a continuous approximation to the WL test, and hence the theoretical context for its algorithm design is provided so that each node's local neighborhood is efficaciously represented.

- **GAT**: GAT (Graph Attention Network) is a type of GNN proposed by Veličković et al. [44] as an extension to graph convolutional neural networks. Contrary to GraphSAGE, GAT extends the AGGREGATE method combining it with attention mechanisms that give an attention score to the different neighbors, consequently giving them different importance. A single layer of the network takes as input a set of node features $h = \left\{\vec{h_1}, \vec{h_2}, ..., \vec{h_N}\right\}, \vec{h_i} \in R_F$ and produces a new set of node features $h' = \left\{\vec{h_1'}, \vec{h_2'}, ..., \vec{h_N'}\right\} \vec{h_i'} \in R_{F'}$. At first, a linear transformation, parametrized by a weight matrix W, is applied to every node and then self-attention is performed on the nodes. Then on each node, a shared attention mechanism a maps feature pairs into attention coefficients e_{ij}. The attention coefficients compute the importance of the j features that belong to node i. Mathematically, attention coefficients are computed with the following equation:

$$e_{ij} = a(W\vec{h_i}, W\vec{h_j}) \tag{3.28}$$

Practically, these attention coefficients are computed only for the pairs of nodes that are neighbors in the graph. An extension to that is the usage of multi-head attention, where each head independently computes attention scores and the results are concatenated or averaged, to produce feature vectors. An illustration of this process is shown in the figure below:



Εικόνα 3.30: *Illustration of multi-head attention. Source [5]*

- **GIN**: GIN (Graph Isomorphism Network) is a GNN type proposed by Xu et al. [45], heavily inspired by the concept of graph isomorphism, meaning that two graphs have a similar structure after rearranging the nodes without alterations in the connections. The paper focuses on the Weisfeiler-Lehman test on graph isomorphism. The importance of the WL test is its injective aggregation update that can map different neighborhoods to different feature vectors. Extended to GNNs, the embeddings on a node's neighborhood can be seen as a multiset and the aggregation function performs on that multiset. The resulting architecture of the GNN in the paper is as effective as the WL test in distinguishing separate graph neighbors. GNNs with injective aggregation and readout functions are proved to be as expressive as the WL test. The Universal Approximation Theorem [46] states that a feedforward neural network with a single hidden layer, with a sufficient number of neurons, given appropriate activation functions can approximate any continuous function to arbitrary accuracy within a bounded domain. This idea leads to the usage of multi-layer perceptrons (MLPs) for the AGGREGATION and READ OUT functions. In the paper, it is suggested that with ϵ being a trainable parameter or a fixed scalar, GIN updates node representations using the equation:

$$h_v^{(k)} = MLP^{(k)}((1 + \epsilon^{(k)})h_v^{k-1} + \sum_{u \in N(v)} h_u^{k-1}) \tag{3.29}$$

  After comparing to other notable models with worse results, GIN finally proved to be an exceptional GNN using the neighborhood aggregation function.

3.6 Neural Architecture Search

All the deep learning models presented in the previous sections, such as convolutional, recurrent, and graph neural networks, rely heavily on their architecture, to provide good results. Architecture engineering can be extremely time-consuming if done manually and therefore can be fairly limited. This is the reason Neural Architecture Search (NAS) has attracted numerous studies since this technique can construct architectures that achieve state-of-the-art accuracies with little human intervention. In this section, we will present

the three basic components of Neural Architecture Search: search space, search strategy, and performance evaluation.

3.6.1 Search Space

The search space represents the set of all possible neural network architectures. An architecture is defined by the operation associated with every node or layer and their in-between connections. The size of the search space determines the computational cost of the search process. Therefore, there is a trade-off between the number of architectures we want to test and the cost of the algorithm.

- **Entire-structured search space**: The space of monolithically structured neural networks [47] is the most obvious choice for a search space. These models are constructed by stacking a predetermined number of nodes, where each node represents a layer with a function. This could also support structures of nodes connected with random skip connections, which is a more complex and promising approach. The entire-structured search space may be easy to implement but has some limitations. This choice can be quite computationally expensive and lacks portability, meaning that a model built on a small dataset may not be suitable for a larger dataset.

- **Cell-based search space**: This neural architecture search, instead of searching for the whole architecture, searches for motifs or cells [48]. This cell is stacked multiple times to form a larger architecture. The input and output of the cell could have the same dimensions or the output could have reduced dimensions. This method has a reduced search space and hence is computationally less expensive. Moreover, this approach solves the portability issue of searching for the whole architecture. Simply stacking more or less cells can produce architectures for other tasks.

- **Hierarchical search space**: Most of the cell-based methods follow a two-level hierarchy. An inner level, which selects the operation and connection for each node, and an outer level, which handles changes in spatial decisions. However, these approaches ignore the network level. Liu et al. [49] defined a general formulation for a network-level structure with their model HierNAS. This is defined as the hierarchical search space, in which a higher-level block is created by iteratively integrating lower-level units. This method can describe more types of unit structures with more flexible topologies.

- **Morphism-based search space**: This search space attempts to design new neural networks based on an existing network using morphism transformations between the layers [50]. For instance, morphism transformations of depth or width replace the original model with a corresponding deeper or wider model. The child networks from network morphism inherit all the knowledge of their parent networks. Then these child networks are trained for a short amount of time. This search space handles arbitrary non-linear activation functions and can perform depth, width, and kernel size transformations in a single operation. This option of search space

can remarkably speed up the training process and achieve great results based on the introductory paper.

3.6.2 Optimization methods

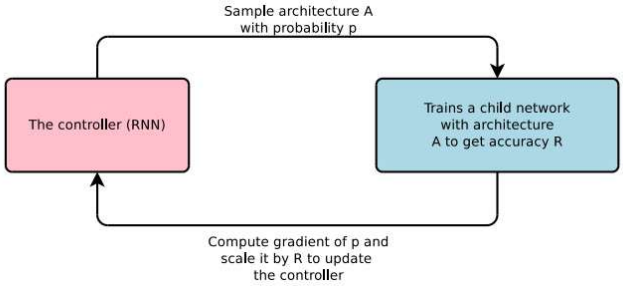
Μανψ διφφερεντ οπτιμιζατιον αλγοριτημς ηας βεεν υσεδ ιν ορδερ το φινδ τηε βεστ αρχηιτεχτυρε ιν τηε σεαρχη σπαχε. Τηε σελεχτιον οφ τηε οπτιμιζατιον μετηοδ ις ηιγηλψ ινφλυενχεδ φρομ τηε σεαρχη σπαχε ανδ ηας βιγ εφφεχτ ον τηε μοδελ'ς χομπυτατιοναλ χοστ ανδ ρεσυλτς. Ωε ωιλλ γιε α βριεφ ρειεω ον τηε μοστ ωελλ-κνοων οπτιμιζατιον μετηοδς.

- **Reinforcement Learning**: One notable optimization technique for neural architecture search is reinforcement learning [47]. The controller proposes child model architectures from the search space for evaluation. The controller is usually a RNN, that outputs a sequence of tokens specifying the network architecture. We train and evaluate the sampled architecture and this performance is the reward the controller receives. The reward signal is non-differentiable so we use a policy gradient method to update the controller and we want to maximize the expected reward:

$$J(\partial_c) = E_{P(a_{1:T};\partial_c)[R]} \tag{3.30}$$

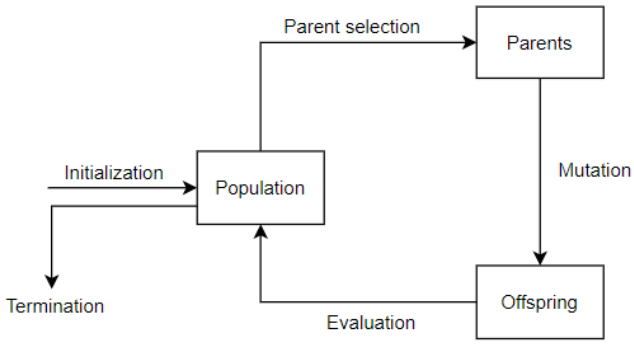
where $a_{1:T}$ are the actions of the controller, T is the total number of tokens, ∂_c are the parameters of the controller, and R is the reward. Using the REINFORCE rule as the policy gradient method, we obtain the gradient of the expected reward as:

$$\nabla_{\partial_c} J(\partial_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\partial_c)}[\nabla_{\partial_c} logP(a_t|a_{t-1:1} : \partial_c)R] \tag{3.31}$$



Εικόνα 3.31: *Overview of reinforcement learning in NAS. Source [31]*

- **Evolutionary algorithms**: Evolutionary methods evolve a population of architectures aiming to find an optimal network. During every step of the process, some architectures are sampled from the population to generate the child architectures through mutations. Mutations alter the network structure, like adding or removing a layer or changing the operations of the layers. The child architectures are evaluated and added to the population. There are many parent selection methods, such as tournament selection or the multi-objective Pareto method. In the end, the worst or oldest model is removed.



Εικόνα 3.32: *Overview of an evolutionary algorithm*

- **Gradient-based methods**: The previous search strategies work in a discrete search space. A pioneering algorithm, namely DARTS [51], was based on Gradient Descent Optimization that searched for neural architectures in a continuous search space using a softmax function to relax the discrete search space, as described below:

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{K} \frac{e^{a_{i,j}^k}}{\sum_{l=1}^{K} e^{a_{i,j}^l}} o^k(x) \tag{3.32}$$

  where $o(x)$ is the operation performed on input x, $a_{i,j}^k$ is the weight assigned to operation o^k between a pair of nodes (i,j) and K is the number of predefined candidate actions. After the relaxation of the search space, the task of searching for architectures turns into optimizing the neural architecture a and the weights ∂ of that neural architecture alternately. Specifically, a and ∂ are optimized using the validation and the training set, respectively.

- **Bayesian optimization**: Neural architecture search could be approached as a black box optimization problem, with an objective function f. Bayesian optimization is a widely adopted technique in this context due to its ability to create a surrogate model for modeling the objective function (denoted as f). It iteratively selects architectures to evaluate based on the objective function, updating the surrogate model as it goes. Once several evaluations of f have been performed, Bayesian optimization employs Bayes' rule to derive the subsequent f. In the next step, it leverages an acquisition function $a(x)$, to determine the next sample point $x_t = argmax_x a(x)$ that optimizes the acquisition function. This process is significantly less time-consuming and could improve efficiency. However, an important constraint is that conventional Bayesian optimization methods are most effective in low-dimensional continuous spaces, making direct application to discrete architecture search spaces challenging.

3.6.3 Performance Evaluation

Every search strategy in Neural Architecture Search outputs the architecture that

maximizes a desired performance metric, such as accuracy on the validation set or accuracy on the test set. Each architecture has to be trained first to be evaluated. Since most search spaces contain a large amount of different architectures, this process can be quite time-consuming and have an unmanageable computational cost. To make neural architecture search more efficient there have been efforts to reduce the training procedure of the architectures. For instance, some approaches train the architectures for fewer epochs or on a subset of the data. Another approach is to inherit parent features into the architectures, instead of training the models from scratch. These techniques made NAS more feasible to implement in various tasks, as the computational cost was significantly reduced without compromising the performance of the process.

3.7 Evaluation of Classification Models

It is important to evaluate the efficiency of the classification model, a procedure that can differ for every use case. We will present the most common evaluation metrics.

3.7.1 Confusion matrix

A confusion matrix is a matrix that encapsulates the performance of a classification model on a set of test data. The rows of the table represent the cases in the predicted class, while the columns represent the cases in the actual class. For binary classification, the matrix will be of a 2x2 table. For multi-class classification to N classes, the matrix shape will be NxN.

		Actual	
		Yes	No
Predicted	Yes	True Positive	False Positive
	No	False Negative	True Negative

Πίνακας 3.1: *Confusion Matrix*

We define the confusion matrix variables as follows:

- **True Positive (TP)**: The cases in which we have predicted positively and the prediction is confirmed.
- **True Negative (TN)**: The cases in which we have predicted negatively and the prediction is confirmed.
- **False Positive (FP)**: These are cases in which we have predicted positively and the prediction is not confirmed.
- **False Negative (FN)**: These are cases in which we have predicted negatively and the prediction is not confirmed.

Based on these variables we define the following metrics:

- **Accuracy**: It is computed by dividing the number of correct predictions by the total number of predictions.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.33}$$

  In cases of imbalance of the class samples in the dataset, the classifier will tend to predict the class with the most data, resulting in very high accuracy. That, however, does not reflect a satisfactory accuracy of the model. This is the reason that we also need the following metrics.

- **Precision**: It is defined as the percentage of correct positive predictions of the classifier.

$$precision = \frac{TP}{TP + FP} \tag{3.34}$$

- **Recall**: It is defined as the percentage of positive samples correctly predicted by the classifier.

$$recall = \frac{TP}{TP + FN} \tag{3.35}$$

- **F1-score**: It combines precision and recall, to provide a balanced measure of the model's performance. Specifically, it is the harmonic mean of precision and recall.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{3.36}$$

- **Specificity**: It is defined as the percentage of negative samples correctly predicted by the classifier.

$$specificity = \frac{TN}{TN + FP} \tag{3.37}$$

- **Cross-Entropy Loss**: It measures the dissimilarity between the predicted probabilities and the true class labels. In binary classification, the cross-entropy loss is calculated as follows:

$$BinaryCross - EntropyLoss = -(y \log(p) + (1 - y) \log(1 - p)) \tag{3.38}$$

  where p is the predicted probability, and y is the indicator (0 or 1 in binary classification)

- **Matthews Correlation Coefficient (MCC)**: It is defined as the estimate of the correlation between the class predicted and the class the users actually belong to.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \tag{3.39}$$

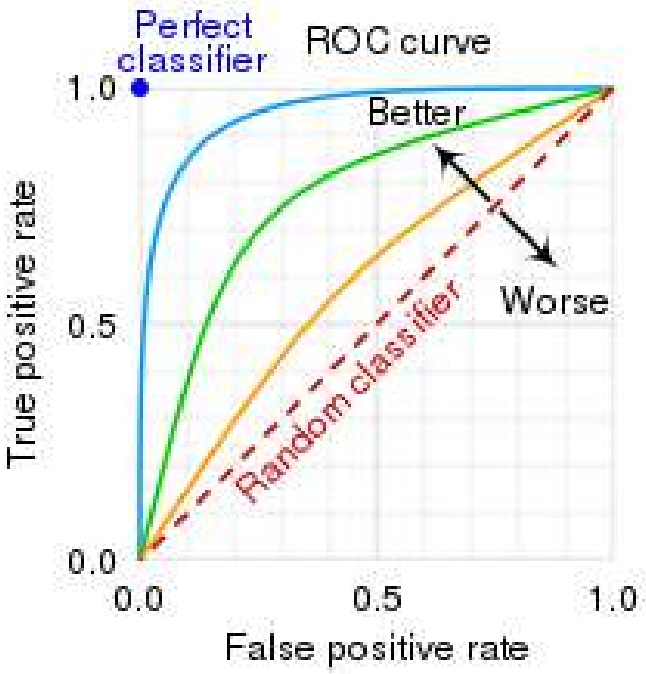
- **True Positive Rate**: It is defined as the percentage of positive samples that are correctly identified by the model.

$$TPR = \frac{TP}{TP + FN} \tag{3.40}$$

- **False Positive Rate**: It is defined as the percentage of negative samples that are incorrectly classified as positive by the model.

$$FPR = \frac{FP}{FP + TN} \tag{3.41}$$

- **ROC Curve and AUC**: The ROC curve plots the true positive rate against the false positive rate at various threshold settings. The AUC provides a single value that represents the overall performance of the model. Higher AUC values indicate better discrimination between classes.



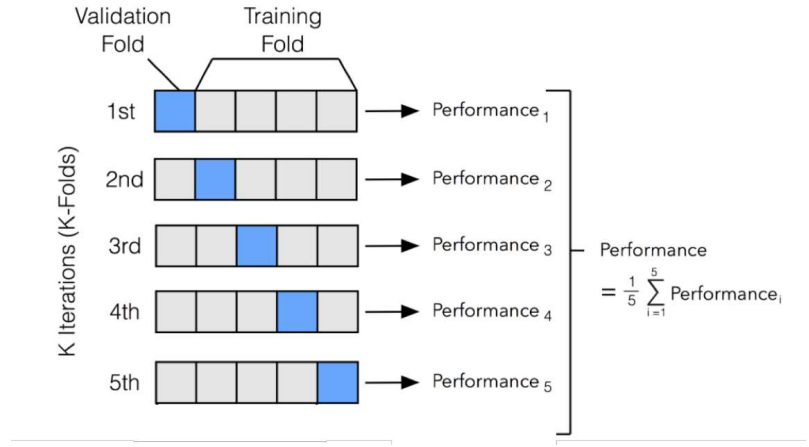
Εικ'ονα 3.33: *ROC Curve and AUC. Source [32]*

3.7.2 Cross Validation

To evaluate the model we can split the dataset into training and test set. However, this simplified method can be insufficient when we have a small amount of data and may lead to overfitting of the model. This issue can be resolved by using k-fold cross-validation, a method that is analyzed below:

1. Divide the dataset into k partitions.
2. Use the k-1 segments as the training set and evaluate the model using the k-th segment. Compute and save the desired evaluation metric.

3. Repeat steps 1 & 2, until every k-partition has been used as a test set.
4. Find the average of all the k computed evaluation metrics. This is the final evaluation metric of the model.



Εικόνα 3.34: *5-fold cross-validation. Source [33]*

While there is no rule for the choice of k the most used values are 5 or 10. Larger values for k mean that the difference in size between the training set and the resampling subsets gets smaller and thus the bias becomes smaller.

There are also commonly used variations of k-fold cross-validation:

- **Stratified**: The data in the folds may be split using a criterion that could ensure a proportional representation of the different classes in each fold. This is particularly useful for imbalanced datasets to prevent underepresantations of classes in the folds.
- **Leave-One-Out Cross-Validation (LOOCV)**: k in this case is set to the number of instances. In each iteration, a single element is used as the validation set. An extreme variation that ensures the least biased estimate of a model's performance, but can be computationally expensive.
- **Repeated**: K-fold cross-validation is repeated multiple times with a variation of data partition. The final result is the average of the metrics from the entire process. Tthis variation intends to eliminate the dependency of the result from a specific random partition.
- **Nested**: In this variation k-fold cross-validation is also performed within each fold of the initial partition of cross-validation. This variation is used often to perform hyperparameter tuning during the model evaluation.

3.8 Word Embeddings

Word embeddings are a technique falling into Natural Language Processing (NLP) that represents words numerically based on their meaning and in-between relationships. They

map words to multi-dimensional vectors in a way that similar words are nearby points in the same vector space. Some well-known word embeddings that we will analyze for further understanding are Word2Vec, GloVe, BERT, and RoBERTa, which we will use in our experiment.

3.8.1 Word2Vec

There are two main use cases of Word2Vec [65]. It can be used to predict the context of a word based on its neighboring words despite their sequence (Continuous bag-of-words) or to predict neighboring words given a target word (Continuous skip-gram). The architecture consists of an input and output layer, separated by a projection layer that learns the word embeddings. The training part of the algorithm aims to make the model better at predicting words in each case, a process that can be extremely time-consuming with a high number of activation functions and words. Word2Vec speeds this procedure with Negative Sampling, which selects a random subset of words to omit from the possible predictions. After training, each word is represented as a vector and similar words map to geometrically close vectors. Also Word2Vec can capture semantic relationships in words, like word analogies, finally making it a widely used tool in NLP.

3.8.2 GloVe

GloVe, short for "Global Vectors for Word Representation" [66] is a method for learning word embeddings that captures both syntactic and semantic relationships between words based on their co-occurrence statistics. Co-occurrence statistics contain valuable information about word relationships since words that appear frequently together are likely to share semantic context. GloVe uses an objective function based on these probabilities to capture both contextual and semantic information. It can also perform word analogies. GloVe has an advantage over traditional pointwise mutual information (PMI) methods, as it directly captures the underlying linear relationships between words and performs with less computational complexity. GloVe presents great efficiency and scalability as presented and is considered a powerful tool in NLP and suitable for large-scale text corpora.

3.8.3 BERT

The BERT model [67], which stands for Bidirectional Encoder Representations from Transformers, is a state-of-the-art deep learning architecture for various Natural Language Processing tasks. Unlike previous unidirectional models, BERT introduces bidirectional context by training a transformer model to predict masked words (Masked Language Model) in a sentence. BERT is pre-trained on a large corpus of text data using two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM involves masking some words of a sentence and training the model to predict the masked words from their context. NSP involves predicting if two sentences appear consecutively in the original text. BERT utilizes a transformer architecture with self-attention mechanisms and feedforward neural networks. There are two variants of BERT with different

performances: $BERT_{BASE}$ with 110 million parameters and $BERT_{LARGE}$ with 340 million parameters. The pre-trained BERT models presented advantages over previous work in NLP tasks over multiple benchmark tests, proving the effectiveness of bidirectional context.

3.8.4 RoBERTa

RoBERTa (A Robustly Optimized BERT Pretraining Approach) [52] is an advanced word embedding model built upon its predecessor BERT. After optimizations in the pretraining process, it can enhance the model's performance on a wide range of NLP tasks. A major advantage of RoBERTa is that is trained on a significantly larger corpus of text data compared to BERT and consequently is trained on a more diverse linguistic landscape. It also ignores the Next Sentence Prediction (NSP) task used in BERT and focuses on the Masked Language Model (MLM) task. The training process of RoBERTa contains detailed hyperparameter tuning for parameters such as batch size, learning rates, etc. RoBERTa outperforms other previous models in several NLP tasks, more particularly in General Language Understanding Evaluation (GLUE), ReAding Comprehension from Examinations (RACE), and Stanford Question Answering Dataset (SQuAD). The model's effectiveness as well as versatility is undeniable and that is a fundamental reason why our model utilizes it for the word embeddings.

Κεφάλαιο 4

Related Work

In this Chapter, we present a complete review of the methods used to detect bot accounts, as well as studies for Neural Architecture Search in different use cases. We will evaluate the metrics and compare the results achieved with different datasets. That will guide us in choosing the most accurate method and then endeavor to optimize it.

Lee et al. [53] employed several machine-learning techniques for bot detection. Specifically, they utilized a supervised learning approach and trained a classifier to distinguish between genuine users and bots based on features extracted from the Twitter data. The features used in their analysis included account-based features (e.g., the number of followers, friends, tweets), temporal features (e.g., time of account creation, tweet frequency), and content-based features (e.g., usage of URLs, hashtags). To train the classifier, they used labeled data, where accounts were manually classified as genuine or bot accounts. The researchers then applied various machine learning algorithms, including Support Vector Machines (SVM), Naive Bayes, and decision trees, to build and evaluate the bot detection model. The dataset consisted of content polluters extracted by the social honeypot and legitimate users sampled from Twitter, information collected over a period of seven months. Combining different elements, the researchers reported an overall accuracy of 95% for the bot detection system they developed.

Yang et al. [54] used a combination of unsupervised and supervised learning methods for bot detection. Specifically, the authors utilized features derived from user metadata, temporal patterns, network structure, sentiment analysis, and linguistic cues. These features were fed into a machine learning pipeline, that reduced dimensionality and included classification algorithms. They evaluated the model's ability to distinguish bots from real users, by calculating the area under the receiver operating characteristic curve (AUC-ROC). They also reported precision, recall, and F1-score, for more insight into the model's performance. The authors leveraged the Botometer API, a tool developed by the same research group, to obtain the dataset. They also proved the model's scalability by including other datasets in the training process and got different AUC scores (up to 99%) in different cases.

Kuduganta et al. [68] attempted to categorize Twitter users into humans and bots. Initially, they used user- and tweet-based features. They used classifiers, such as Logistic Regression, SGD Classifier, Random Forest, AdaBoost, etc. They then proposed a new deep learning approach, using only the user's tweets and some metadata features

implemented. This architecture includes a tokenizer, GloVE embedding layer, LSTM, and Dense layers. The authors used a dataset with a minimal amount of features, that consisted of the Cresci dataset and input from collaborators. To deal with the problem of dataset unbalance they used synthetic data generation with SMOTOMEK and SMOTENN. They achieved an AUC/ROC score of 96%.

Cai et al. [69] proposed their model (BeDM) that involved deep neural networks in bot detection. They employed convolutional neural networks (CNNs) and LSTM, using only the tweet semantics, such as the frequency and the type of publications. They used a public dataset collected with the honeypot method [70]. After experimenting with the parameters of the neural network, they achieved a precision of 88%.

Wei et al. [71] used only users' tweets with no context of prior knowledge or assumption about user profiles, friendship networks, or historical behaviour. They proposed a recurrent neural network (RNN) model that used word embeddings to encode tweets, a three-layer Bidirectional LSTM (BiLSTM), and a softmax layer at the binary output. The authors used real-world datasets for their experiments and using the Cresci Dataset [56] they achieved an accuracy of 97,6%.

Dickerson et al. [72] suggested the innovation of sentiment analysis with their model SentiBot. The information obtained and leveraged for each user is divided into four categories: (a) tweet syntax, (b) tweet semantics (c) user behaviour, and (d) user neighborhood. The authors used six high-level classifiers, including support vector machines (SVM) for classification, Gaussian naive Bayes, AdaBoost, gradient boosting, random forests, and extremely randomized trees. Preprocessing of the data is done through dimensionality reduction techniques (PCA). The authors used a large dataset relating to the 2014 Indian election, collected from July 15, 2013 to March 24, and achieved an accuracy of at least 90%.

Miller et al. [73] approached bot detection as an anomaly detection problem, whereas previous methods identified it as a classification task. They extracted 107 features from a user's tweet and property information and adapted two stream clustering algorithms, StreamKM++ and DenStream, to facilitate spam detection. Bot users are conceived as abnormal outliers. Combining the two algorithms the system was able to identify 100% of the spammers, while incorrectly detecting 2.2% of the normal users as spammers.

Cresci et al. [55] introduced the Social Fingerprinting technique for bot detection, a Digital DNA method for modeling the behaviours of social network users. Each user is represented as a sequence of characters depending on the type and content of the tweets they publish, simulating a DNA sequence. The authors try to find similarities in the sequences that will help split the two categories of users, humans and bots. The similarity measure is defined as the length of the longest Longest Common Substring (LCS) between two sequences. Afterwards, the LCS problem is extended to M users. For a set of real users, the length of LCS was found to be particularly small. This led the authors to the conclusion that longer sequences than the LCS of the whole dataset were more possibly bot accounts. Based on this idea, the authors developed two techniques, one based on supervised learning and another on unsupervised learning to find similarities in the behaviour of accounts. They used the aforementioned Cresci dataset. [56]

Botometer [57] is a web-based program developed by Indiana University. It leverages more than 1,000 features to classify Twitter accounts as bots and humans. These features include friends, social network structure, user meta-data, temporal activity, content features, and sentiment analysis. Botometer distinguishes the accounts by an overall bot score (0-5), along with several other scores. The higher that score is, the more likely this account belongs to a bot.

Yang et al. [74] presented a study of people's interaction with AI countermeasures and used Botometer as a baseline. They designed scores to evaluate how bot detection methods met general opinion's expectations and underlined the importance of the dataset in supervised methods. As a future reference, they referred to ways bot detection methods could improve by taking into consideration features such as language and device metadata, timezone differences, and content deletion patterns

Feng et al. [75] suggested SATAR, a self-supervised representation learning framework for Twitter users, and applied it to bot detection. In particular, SATAR leverages the user's semantics, property, and neighborhood information. Meanwhile, it adapts by pre-training on a massive number of self-supervised users and parameter fine-tuning on detailed bot detection scenarios. The authors proposed two alternative models: $SATAR_{FC}$ and $SATAR_{FT}$. The experiments were performed in three datasets: Twi-Bot20 [39], Cresci-17 [56], and PAN-19, to prove the model's adaptability. The model outperformed previous models of bot detection, with $SATAR_{FT}$ achieving higher accuracy (reaching an accuracy of 98% at the Cresci-17 dataset).

Alothali et al. [76] presented their framework Bot-MGAT for the task of bot detection, which stands for bot multi-view graph attention network. Considering the ever-changing bot behaviour, the authors noted the inability of other techniques to adapt to other datasets due to a lack of recently updated labeled data. They proposed a framework based on the multi-view graph attention mechanism using transfer learning (TL). The framework also benefited from semi-supervised learning, using both labeled and unlabeled data. The authors used the Twi-Bot20 [39] due to its graph structure, extracting 18 features for the training, and two other datasets for the testing. They discovered that Bot-MGAT with TL outperformed other methods with an accuracy score of 97.8%.

Feng et al. [77] introduced the aspect of heterogeneity of relations and influence among users in the Twittersphere for the task of bot detection. They proposed a bot detection framework that leverages a heterogeneous information network with users as nodes and diversified relations as edges. Then they aggregated messages across users and operated heterogeneity-aware Twitter bot detection. The experiments they conducted using the Twi-Bot20 dataset [39] provided the data set structure that the model required. The authors discovered that a graph-based heterogeneity-aware method for bot detection could outperform other methods and achieved an accuracy of 86%.

Feng et al. [38] proposed their model for bot detection BotRGCN, which is short for Bot detection with Relational Graph Convolutional Networks. BotRGCN builds a heterogeneous graph out of the following relationships and uses information, such as the user's description, tweets, numerical and categorical property set, and neighborhood information. The experiments were conducted on the Twi-Bot20 dataset, but BotRGCN could

exploit other types of relations if supported by the dataset. It achieved an accuracy of 86.42% and outperformed other models, underlining the avails of using RGCNs in bot detection.

Now that we have established the benefits of using graph-based approaches to bot detection, we will study the implementation of NAS to achieve possible improvements.

Zhou et al. [78] proposed the automated graph neural networks (Auto-GNN) framework. The authors noticed the inability of previous NAS algorithms to be applied to the GNN search problem, due to differences in the search space and instability when parameters change. Auto-GNN looks for the best GNN architecture possible in a predetermined search space. The search space is divided into the following six classes of actions: hidden dimension, attention function, attention head, aggregate function, combine function, and activation function. For efficiency reasons, the authors designed a conservative explorer to maintain the best neural architecture found during the search. The authors also implemented constrained parameter sharing, adapted to the heterogeneous GNN architecture. Two approaches were proposed for the experiments: transductive, where the unlabeled data used for validation and testing are accessible during training, and inductive, where the graph structure and node features on the validation and testing sets are unknown during the training process. Three datasets: Cora, Citeseer, and Pubmed [79] were used for transductive learning, and PPI [80] was used for inductive learning. The results of the studies demonstrate that Auto-GNN manages to discover neural architectures with performances that surpass handcrafted models and models derived from other search techniques.

Nunes et al. [58] presented two NAS methods for optimizing GNNs: one based on reinforcement learning and one based on evolutionary algorithms. The authors experimented with these methods to evaluate if they provide better accuracy on the validation set than a random search in all possible parameters of the GNN. Evolutionary algorithms are methods based on the theory of evolution. Specifically, many different parameters in the layers produce a set of GNNs, which will compete to achieve the best accuracy in a validation set, to be the schema that will produce a new offspring. In reinforcement learning, an LSTM is used as a controller to generate architectures, while the training can adapt along with the accuracies achieved at the validation set. Random search includes an iteration through all the possible actions in a layer. The authors defined two cases of search spaces: Macro, where the architectures generated have the same structure, and Micro, where the architectures are not rigidly structured but combine several convolutional schemas. The authors used seven different datasets for 100 iterations within each search space. They concluded that EA and RL found very similar architectures to the ones found by a random search, which is a significantly simple technique. However, they noted that EA produced mostly architectures that could fit in the GPU, while the other methods assembled oversized architectures in up to 80% of the cases.

Gao et al. [81] proposed a Graph Neural Architecture Search method (GraphNAS) to implement an automatic search of the best graph neural architecture based on reinforcement learning. The search space covers sampling functions, aggregation functions, and gated functions. GraphNas also uses more efficient parameter-sharing techniques

than other contiguous models for CNNs and RNNs. The datasets used for transductive learning were Cora, Citeseer, Pubmed, and PPI for inductive learning. The authors also used random search as a baseline for their experiment, noting that it provides satisfactory results despite the implied simplicity of this method. After training 1000 different architectures, the five best ones were used for the testing. Eventually, architectures by GraphNas surpassed human-invented ones or those produced by other models or random searches.

Zhao et al. [82] proposed the SNAG framework (Simplified Neural Architecture Search for Graph neural networks). The researchers found a deficiency in the designed search space of models like GraphNAS and Auto-GNN. Contrary to the message-passing frameworks of other models, the suggested framework had two key components: Node aggregators, which focused on neighborhood features, and Layer aggregators, which focused on the range of the neighborhood used. The search space algorithm was a variant of Reinforcement Learning that adopted the weight-sharing mechanism (SNAGWS). Transductive learning (on the Cora, CiteSeer, and PubMed datasets) and inductive learning (on the PPI dataset) were used for the experiments, which proved the efficiency of the framework compared to the aforementioned models.

Jiang et al. [83] adapted the method of neural architecture search to the design and development of GNNs for molecular property prediction. The authors designed neural networks for message-passing (MPNNs) between nodes. The purpose was to predict molecular properties of small molecules. To find an optimal MPNN from the user-defined search space, they used regularized evolution (RE) from the DeepHyper package, which applies mutation to existing population models to uncover the final best model. The authors experimented on three quantum mechanics and three physical chemistry datasets from the MoleculeNet benchmark. They concluded that RE surpassed results achieved by Random Search of the number of high-performance architectures on validation and test loss.

Zhao et al. [84] proposed their framework, which tries to Search to Aggregate NEighborhood (SANE). The search space has similarities with the search space from the SNAG framework, with Node and Layer aggregators. However, the authors presented a novel differentiable search algorithm, while other models used reinforcement learning. The experiments on the same datasets as the SNAG experiments proved that the effectiveness and efficiency of SANE are superior to those of existing GNN models and other NAS techniques.

Cai et al. [85] proposed Graph Neural Architecture Search (GNAS) with a novel-designed search space and a gradient-based search approach. To build the search space the authors designed a three-level Graph Neural Architecture Paradigm (GAP) with a tree-topology computation procedure and two types of fine-grained atomic operations (feature filtering and neighbor aggregation) from message-passing. In this way, they combined the features of each node and the neighborhood semantics. The experiments were conducted on more complex datasets, such as chemistry-based ZINC, for mathematical modeling PATTERN and CLUSTER and computer vision datasets CIFAR10 and MNIST. The main focus of the survey was the optimization of the search space and the message-passing

depth, and GNAS was able to surpass all human-made GNNs.

Li et al. [86] proposed a novel dynamic one-shot search space for multi-branch neural architectures of GNNs. The dynamic nature of the search space offers a larger capacity. The architectures with lower importance weights are removed periodically from the population, while the operation candidates are unique to every edge of the computational graph, so the model isn't inferior to searches in a larger predefined search space. The authors performed both supervised and unsupervised techniques for the training part. They used the Cora, Citeseer, and Pubmed datasets and compared the architectures found by their model to manually crafted ones, Micro NAS ones, and Macro NAS ones. They discovered that their model outperformed other methods, while also providing a very important up to 10 times speedup.

Peng et al. [87] implemented a NAS approach to human action recognition from skeleton movements. The search space was enlarged with diverse spatial-temporal graph modules while constructing higher-order connections between nodes using Chebyshev polynomial approximation. The search algorithm used is an evolutionary adaptation with a high sampling efficiency, denoted CEIM (Cross-Entropy method with ImportanceMixing). The experiments were conducted on two large datasets NTU RGB+D and Kenitics-Skeleton, that contained human actions. The authors concluded that the architectures produced by their model outperformed state-of-the-art approaches.

Zhang et al. [6] proposed DFG-NAS, an innovative method that allows for automatic search of very deep and adaptable GNN architectures. DFG-NAS emphasizes the search on macro-architectures, and specifically on how atomic propagation (P) and transformation (T) operations are implemented into the GNN. P is closely related to the graph structure, while T focuses on the non-linear transformation in the neural network. The authors noted that most GNAS methods applied transformation after propagation (P-T) in each layer, which is a fixed limiting approach. In addition, they also adopted gating and skip-connection mechanisms to support deeper GNN pipelines. They used an evolutionary algorithm to find the optimal architecture, which supported four cases of mutation: (a) add a P, (b) add a T, (c) replace a P with a T, (d) replace a T with a P. The experiments were conducted on three citation graphs (Cora, Citeseer, and PubMed), and one large OGB graph. The authors concluded with an accuracy improvement of up to 0.9% over state-of-the-art manual designs, with a speedup of 15.96x over other G-NAS methods.

Κεφάλαιο 5

Dataset and Problem Statement

In this Chapter, we present a broad explanation of the properties of the dataset that we used to carry out our experiment as well as the problem we are focusing on.

5.1 TwiBot-20 Dataset

We conducted the experiment using the TwiBot20 dataset, which we obtained from Feng et al. [39]. This dataset includes the training, validation, test and support set. [59]. TwiBot-20 is constructed with a breadth-first search (BFS) methodology. The first two levels of the BFS expansion are the train, dev, and test sets, whereas the third and outermost tier is the support set. The support test doesn't require the neighbor information, because it is included in the train, dev, and test sets. That is the reason the neighbor column in support.json is "null". The support set could enable semi-supervised learning. We present the attributes of the sets with a short description for further explanation:

Attribute	Description
ID	ID from Twitter to identify the user
profile	profile information from Twitter API
tweet	200 recent tweets of the user
neighbor	20 random followers and followings of the user
domain	domain of the user (politics, business, entertainment, sports)
label	label of the user ('1': bot, '0':human)

Πίνακας 5.1: *Attributes and description of TwiBot-20 Dataset*

This dataset has been used in multiple cases of bot detection due to its user semantic and neigborhood properties. Next we will present the definition of the problem we want to solve.

5.2 Problem statement

Let B = $b_{i_{i=1}}^{L}$ represent the user's description with L words. Let T = $t_{i_{i=1}}^{M}$ be the user's M tweets and each tweet $t_i = \{w_1^i, ..., w_{Q_i}^i\}$ contains Q_i words. Let $P = P^{num}, P^{cat}$ be the user's numerical and categorical property set. Finally Let $N = N^f, N^t$ be the user's neighborhood information, where $N^f = N_1^f, ..., N_u^f$ indicates user's following accounts and $N^t = N_1^t, ..., N_u^t$

indicates user's follower accounts. The task of Twitter bot detection is to identify the bots from the users utilizing the information from B, T, P, and N.

5.3 Experimentation with the dataset

Using the Python Pandas libray [88] we will present some processing of the dataset we conducted in the train, test and validation set for further understanding. We will find the null values to check the validity of our data and the frequency of attribute values to ensure that the dataset is optimal for our experiment.

5.3.1 ID

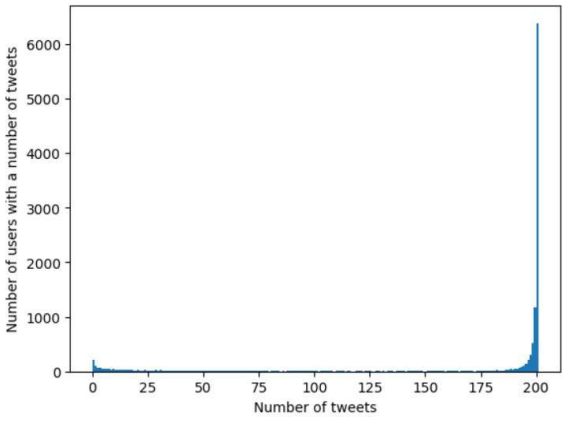
We merge all the sets and confirm that every user ID is unique. This was expected, as these IDs are from Twitter.

5.3.2 profile

Profile data are obtained from the Twitter API. We confirm that no user has null values in that attribute. There are 38 values in profile data, which no user has more or less.

5.3.3 tweet

The dataset is projected to have 200 tweets per user. We find the frequency of the number of tweets per user and present the results in a histogram:

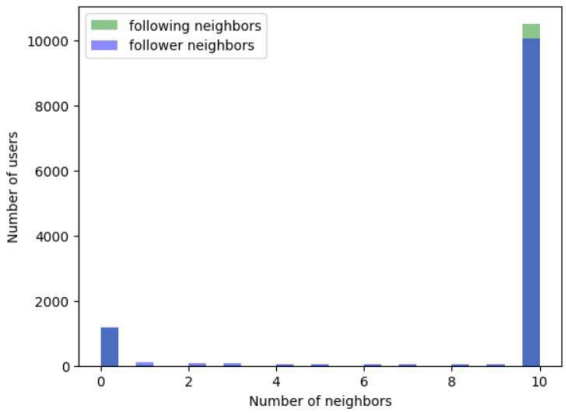


Εικόνα 5.1: *Tweets per user*

We deduce that even if not all the values are 200, the vast majority of the numbers of tweets per user are near 200. It is also noted that 80 users have no tweets, which constitutes 0.67% of the dataset.

5.3.4 neighbor

The dataset is projected to have 20 neighbors per user. We find the frequency of the number of following and follower neighbors per user and present the results in a histogram:

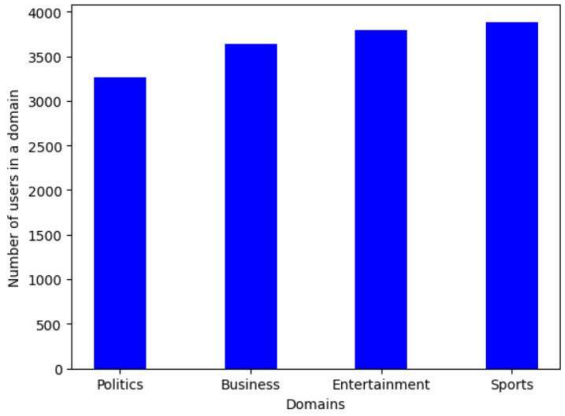


Εικόνα 5.2: *Number of neighbors per user*

The majority of users have ten followers and ten following accounts, as predicted. However there are 1087 accounts with no neighbors, which constitutes 9.19% of the dataset.

5.3.5 domain

We find the number of users that include each of the four domains (politics, business, entertainment, and sports). We intend to test whether the domains are balanced, so we present the results in a bar plot that compares the number of users in each value:

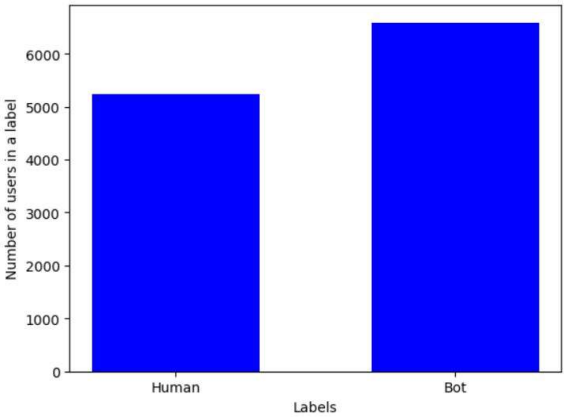


Εικόνα 5.3: *Number of users in each domain*

We conclude that the domains are well-balanced. Small disparities are expected since some domains will interest more users.

5.3.6 label

We find the number of bots and humans in the dataset to check if they are balanced. We present the results in a bar plot:



Εικόνα 5.4: *Number of users in each label*

In the dataset, there are 5237 humans and 6589 bots from all 11826 discrete users. That indicates humans are 44.28% and bots are 55.72% of all users. We deduce that there is not an extreme difference between those values, so the dataset is considered balanced.

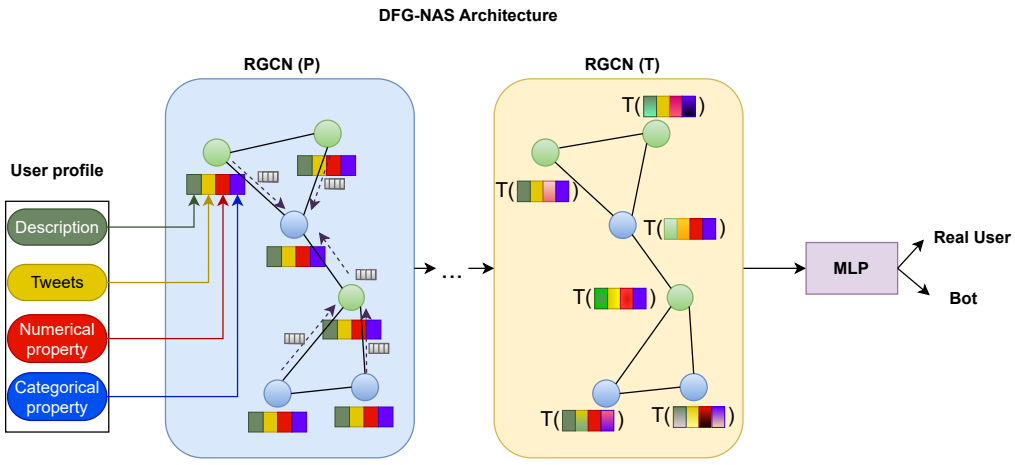
Κεφάλαιο 6

Model

In this Chapter, after a thorough analysis of the theoretical background, related work in Bot Detection and the technique of Neural Architecture Search, and the dataset we will use, we will present a detailed analysis of our model.

6.1 Model Structure

Bot detection is critical in today's digital landscape, as the proliferation of automated bots poses significant threats to online platforms. To develop an improved model for this task, we incorporated a promising bot detection approach with the technique of Neural Architecture Search. First, we will describe the preprocessing of the user metadata used in our model. Next, we introduce the use of Relational Graph Convolutional Neural Networks and the two different functions in the Message Passing Protocol. Last, we explain the application of DFG-NAS [6] in searching for the best permutation of Propagation and Transformation functions.



Εικόνα 6.1: *Model used for Bot detection*

6.1.1 Data Preprocessing

Feng et al. [38] proposed their model for bot detection BotRGCN, proving the advantages of using Relational Graph Convolutional Networks in the task of bot detection. Inspired by this method we employ Graph Neural Networks in our model and make use of the Twibot-20 dataset.

We follow the preprocessing introduced for BotRGCN [38]. Each user's representation includes metadata that are preprocessed as follows:

- **Overall**: User's description, tweets, numerical and categorical properties are encoded and concatenated to finally represent user features:

$$r = [r_b; r_t; r_p^{num}; r_p^{cat}] \in \mathbb{R}^{D\times 1} \tag{6.1}$$

  where D is the user embedding dimension. Each different feature's procession and representation are explained below. Later we will prove that the model's performance is attributed to all these features and not only to the heterogeneous graph.

- **User description**: The user descriptions are encoded with pre-trained RoBERTa:

$$\bar{b} = RoBERTa(\{b_i\}_{i=1}^{L}), \bar{b} \in \mathbb{R}^{D_s\times 1} \tag{6.2}$$

  where $\bar{b}$ denotes the representation of user description and D_s is the RoBERTa embedding dimension. Then the vectors for the user's description are derived:

$$r_b = \phi(W_B \cdot \bar{b} + b_B), r_b \in \mathbb{R}^{D/4\times 1} \tag{6.3}$$

  where W_B and b_B are learnable parameters, ϕ is the activation function and D is the embedding dimension.

- **User tweets**: The user tweets are also encoded using RoBERTa. The final representation of the user's tweets r_t is the average of the representations of all tweets.

- **User numerical properties**: The user's numerical properties are adopted straight from the Twitter API without feature engineering and presented in the table below. For this information z-score normalization is conducted to get the representation r_p^{num} with a fully connected layer.

Feature Name	Description
#followers	number of followers
#followings	number of followings
#favorites	number of likes
#statuses	number of statuses
active_days	number of active days
screen_name_length	screen name character count

Πίνακας 6.1: *User Numerical Properties*

- **User categorical properties**: The user's categorical properties are also encoded with MLPs and GNNs, without feature engineering, just as the numerical properties. They are adopted straight from the Twitter API and presented in the table below. After one-hot encoding, they are concatenated and transformed with a fully connected layer and leaky-relu to get their representation r_p^{cat}.

Feature Name	Description
protected	protected or not
geo_enabled	geo-location enabled or not
verified	verified or not
contributors_enabled	enable contributors or not
is_translator	is translator or not
is_translation_enabled	translation or not
profile_background_tile	the background tile
profile_user_background_image	background image or not
has_extended_profile	extended profile or not
default_profile	the default profile
default_profile_image	the default profile image

Πίνακας 6.2: *User Categorical Properties*

6.1.2 Relational Graph Convolutional Neural Networks

Our method builds a heterogeneous graph out of the following relationships. Users are considered nodes and the 'following' and 'followers' relations are represented as edges connecting the nodes. The user's 'followers' are therefore represented differently than the user's 'following'. The heterogeneous graph that is constructed can represent better the relations between users and more relations between the users could be integrated into the graph if supported by the dataset. The users also contain the concatenated metadata that we described below.

To combine the users' representations with the relationships between users we make use of RGCNs. The message-passing pipeline of RGCNs includes two types of atomic operations: propagating (P) representations of its neighbors and applying transformation (T) to the representations. Below we describe the process behind the two functions:

- **Propagation**: Propagation includes message aggregation from neighbour nodes without explicit node feature transformation. The mathematical expression for the

propagation step is as follows:

$$h_i^{(l+1)} = \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} \tag{6.4}$$

where $h_i^{(l+1)}$ is the new node feature after propagation, R is the set of relations, N_i^r are the neighbors of the node with relation r, $c_{i,r}$ is a problem-specific normalization constant that can either be learned or chosen in advance (for example $c_{i,r} = N_i^r$) and $W_r^{(l)}$ is the learnable weight matrix for relation r.

- **Transformation**: Transformation occurs on each node based on the relations. The mathematical expression for the transformation step is as follows:

$$h_i^{(l+1)} = W_{root} h_i^{(l)} + \sum_{r \in R} (W_r h_i^{(l)}) \tag{6.5}$$

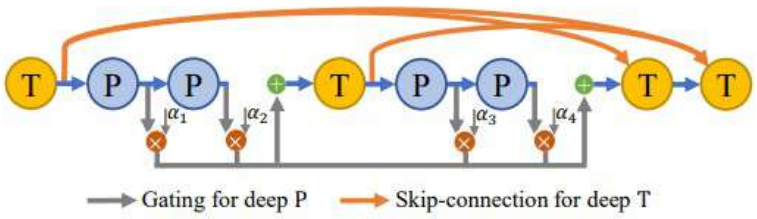
where $h_i^{(l+1)}$ is the new node feature after transformation, W_{root} is the learnable weight matrix for the root node, W_r is the learnable weight matrix for relation r and R is the set of relations.

We segregate these two types of functions since combinations of them will construct the search space for the Neural Architecture Search.

6.1.3 Deep and Flexible Graph Neural Architecture Search

The use of Graph Neural Networks offers undeniable advantages in the task of bot detection. However, maximizing their performance may require extensive feature engineering. This is why we employ Graph Neural Architecture Search, using the model DFG-NAS [6]. Thus, we search for the permutation of Propagation and Transformation steps that achieves the highest accuracy. Most G-NAS methods adopt a fixed message-passing pipeline to organize two types of atomic operations, specifically propagating (P) representations of its neighbors and applying transformation (T) to the representations. This pipeline could be a tight entanglement ($P-T-P-T$) or one with a certain degree of entanglement ($P-T-T-T$, $T-P-P-P$). Also, most G-NAS methods have a fixed pipeline length since the performance decreases with too many P operations as the layers become deeper, which is referred to as the over-smoothing issue. Propagation and transformation operations correspond to enforcing and mitigating the effect of smoothing respectively. Our model searches for flexible pipelines of P and T operations, using a genetic algorithm. It also makes use of gating and skip-connection mechanisms in the P and T operations, respectively.

The search space includes P-T combinations and the number of P-T operations. Additionally, connections among each type are added. For a single P or T operation in one GNN layer in the model, $o_v^{(l)}$ is the output of node v in the l-th layer, and L_P and L_T are two sets with the layer indices of all P operations and T operations respectively. The operations and the layer connections are described below.



Εικόνα 6.2: *Pipeline example in the search space. Source [6]*

Propagation connections: An imminent problem in GNNs is over-smoothing or under-smoothing which is due to too many or too few propagation operations. To achieve suitable smoothness for different nodes, P operations are amplified with a gating mechanism. The output of the l-th P operation is the propagated node embedding of $o^{(l-1)}$ if the next operation is also P. If the next operation is T, a node-adaptive combination weight for the node embeddings propagated by all the previous P operations is assigned. Formulatively:

$$z_v^{(l)} = P(o_v^{(l-1)}) \tag{6.6}$$

$$o_v^{(l)} = \begin{cases} z_v^{(l)} & \textit{followed by P} \\ \sum_{i \in L_P, i \leq l} softmax(a_i) z_v^{(i)} & \textit{followed by T} \end{cases} \tag{6.7}$$

where $a_i = \sigma(s \cdot o_v^i)$ is the weight for the i-th layer output of node v. s is the trainable vector shared by all nodes, and σ denotes the Sigmoid function. The Softmax function is adopted to scale the sum of gating scores to 1.

Transformation connections: An imminent issue with GNNs is the model degradation issue, caused by a hyperbolic amount of transformation operations, decreasing the model's accuracy. To mitigate this issue, skip-connection mechanisms are used in T operations. The input of each T operation is the sum of the output of the last layer and the outputs of all previous T operations before the last layer. The input and output of the l-th T operation can be formulated as:

$$z_v^{(l)} = o_v^{(l-1)} + \sum_{i \in L_T, i < m(l)} o_v^{(i)} \tag{6.8}$$

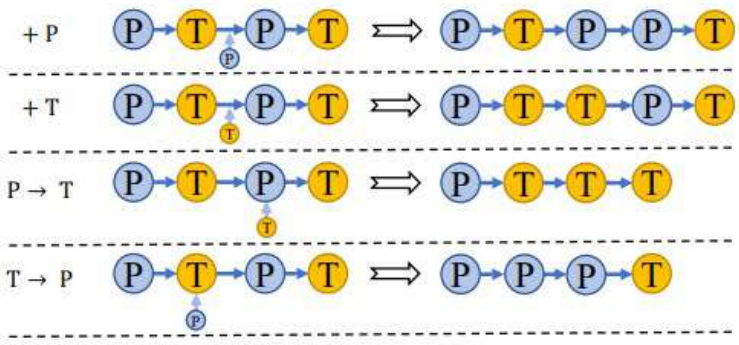
$$o_v^{(l)} = \sigma(z_v^{(l)} w^{(l)}) \tag{6.9}$$

where $m(l)$ is the index of the last T operation before the l-th layer, and $W(l)$ is the learnable parameter in the l-th T operation.

Evolutionary algorithms are a class of optimization algorithms inspired by biological evolution. To evolve the population of individuals they apply mutations. Each GNN architecture is encoded as a sequence of P and T operations. Four different cases of mutation can be enforced:

- **+P**: add a propagation operation
- **+T**: add a transformation operation
- **P→T**: replace a propagation operation with a transformation one
- **T→P**: replace a transformation operation with a propagation one

These mutations can happen at a random position in the sequence. Each pipeline is considered a chromosome and the above mutations simulate nature's mutations.



Eik'ona 6.3: *Depiction of the 4 different mutations. Source [6]*

The pseudocode of the evolutionary algorithm implemented in the search space is described below. At first, k different GNN architectures are randomly generated as the initial population set Q and then they are evaluated on the validation set. Next, m (m < k) individuals from the population are randomly sampled, and the one with the best validation performance is selected as the parent A. The child architecture B is generated by a random pick of one of the four mutations. B is evaluated and added to the population and then the oldest individual is removed. This process is repeated for T generations and finally, we can return the architecture with the best performance.

Αλγοριθμος 6.1: *Searching algorithm*

Initialize the population set Q with k individuals
Evaluate the architectures in Q
for $1 \leq t \leq T$ **do**
    Randomly sample m individuals from Q
    Select from these individuals the one with the best performance as parent A
    Mutate A with a random mutation design and get child B
    Evaluate B and add it to Q
    Remove the oldest individual of Q
end for
Return the architecture with best performance

Κεφάλαιο 7

Results and Ablation Study

In this Chapter, we will present our experiments after the analysis of our model structure. Specifically, section 7.1 presents the settings of our experiments, a brief portrayal of the baseline models we are comparing to ours, and finally our results. Section 7.2 finally presents the ablation study, to prove the integrity of our model.

7.1 Experiments and results

7.1.1 Experiment settings

The experiment was run on Google Colab using Nvidia's T4 GPUs. For the architecture search, the number of the population set k is 15, and the number of maximum generation time T is 80. The training budget of each GNN architecture is 70 epochs. These numbers although limited due to our resources, provide a great example of the efficiency of our model. More intricate architectures that we tested do not provide better results and the number of epochs is sufficient to have a good idea of each architecture's accuracy. The training is done using Adam optimizer with a learning rate of 0.04. The criterion is Cross Entropy Loss and the regularization factor is 2e-4. Dropout to all feature vectors with a rate of 0.5 is applied and the dropout between different GNN layers is 0.8.

After running the NAS method we process the results and examine the five architectures with the best accuracy in the validation set. Each architecture is now trained with 100 epochs on the TwiBot-20 dataset [39], to make a heterogeneous graph of 229,580 nodes and 227,979 edges. The train set is 70% of the dataset, the validation set is 20% and the test set is 10%. The training is also performed using the Adam optimizer with a learning rate of 1e-3. Then each architecture is tested in the test set to get the results, that we will present below.

Each experiment is run 5 times to avoid outliers in our results.

7.1.2 Baselines

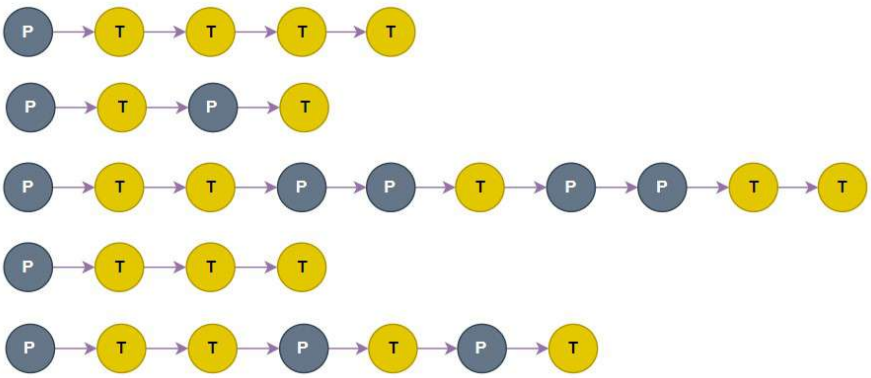
We will compare our results with other state-of-the-art bot detection methods:

- *Lee et al.* used a random forest classifier with several account-based, temporal, and content-based features (e.g. tweet frequency, use of hashtags, etc.)

- *Yang et al.* used a random forest classifier that derived user metadata, temporal patterns, network structure, sentiment analysis, and linguistic cues.
- *Kuduganta et al.* constructed an architecture that used a dataset with a minimal amount of user- and tweet-based features.
- *Wei et al.* used an RNN model with word embeddings, three-layer BiLSTM, and a softmax layer. They only made use of each user's tweets for bot detection.
- *Cresci et al.* represented users as strings based on the type and content of their tweets. They identified bots with similar behaviour after analyzing the longest common substrings. item *Miller et al.* contracted 107 features from the users' tweet and property information and performed two stream clustering algorithms for anomaly detection, to distinguish bots from real users.
- *Botometer* is a web-based program that uses a wide range of features, like a user's social network structure, temporal activity, and language, to provide an overall score of likelihood that this account is a bot.
- *SATAR* conducts self-supervised learning with fine-tuning, using a user's semantics, property, and neighborhood information.
- *BotRGCN* constructs a heterogeneous graph out of the following relationships and uses information, such as the user's description, tweets, and numerical and categorical property set.

7.1.3 Results

Each architecture during the search is saved with its P-T configuration, accuracy in the validation set, and accuracy in the test set. In the following image the five architectures that are chosen from the NAS method are depicted.



Εικόνα 7.1: *Top-5 performing architectures. NAS validation accuracies (from up to down) are: 87.01%, 86.99%, 86.95%, 86.89%, 86.82%*

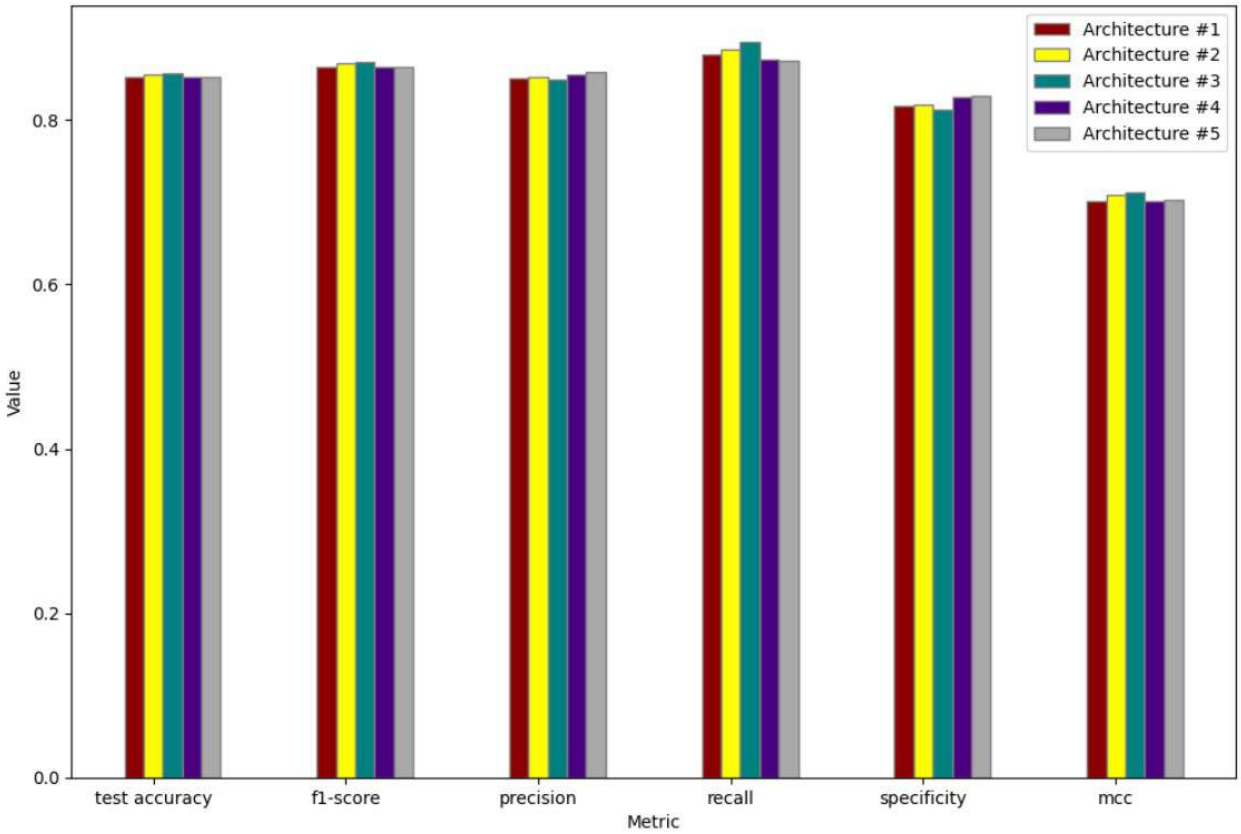
These architectures are trained and tested from scratch in TwiBot-20 dataset. We present all the metrics attained of all the architectures.

Model	Accuracy	F1-score	Precision	Recall	Specificity	MCC
1st Architecture	0.852	0.865	0.851	0.88	0.818	0.702
2nd Architecture	0.855	0.869	0.853	0.886	0.819	0.709
3rd Architecture	**0.857**	**0.871**	0.849	**0.895**	0.812	**0.712**
4th Architecture	0.852	0.864	0.856	0.873	0.828	0.702
5th Architecture	0.852	0.864	**0.858**	0.872	**0.829**	0.703

Πίνακας 7.1: *Performance of the architectures from architecture search*

All selections achieve good metrics and present advantages in bot detection over state-of-the-art methods. These results underscore the significant advantages that emerge from employing architecture search techniques in the realm of bot detection. Moreover, they establish the efficiency of utilizing user features and relationships between users in bot detection.

Upon closer examination of the results, the third architecture has the highest accuracy. The fifth model has the highest specificity. Moreover, all the architectures have high metrics of accuracy, F1-score and MCC. Whichever architecture we choose could compete with state-of-the-art models. From now on we will refer to the third architecture as our model, since it provides the highest accuracy.



Εικόνα 7.2: *Comparison of the architecture performances from NAS*

In the table below we present the performance of other methods on the TwiBot-20 dataset compared to ours. We see that our model benefits from the search for the fittest architecture that we performed beforehand, as it achieves higher accuracy, F1-score and MCC than other state-of-the-art models.

Model	Accuracy	F1-score	MCC
Lee et al.	0.7456	0.7823	0.4879
Yang et al.	0.8191	0.8546	0.6643
Kuduganta et al.	0.8174	0.7517	0.6710
Wei et al.	0.7126	0.7533	0.4193
Cresci et al.	0.4793	0.1072	0.0839
Miller et al.	0.4801	0.6266	-0.1372
Botometer	0.5584	0.4892	0.1558
SATAR	0.8412	0.8642	0.6863
BotRGCN	0.8462	0.8707	0.7021
ours	**0.8568**	**0.8712**	**0.7116**

Πίνακας 7.2: *Performance of models on the TwiBot-20 dataset*

7.2 Ablation Study

To prove the effectiveness and the integrity of our model we will perform an ablation study on the basic ideas: the user's features used for the training, the Gate operation, and the skip-connection operation.

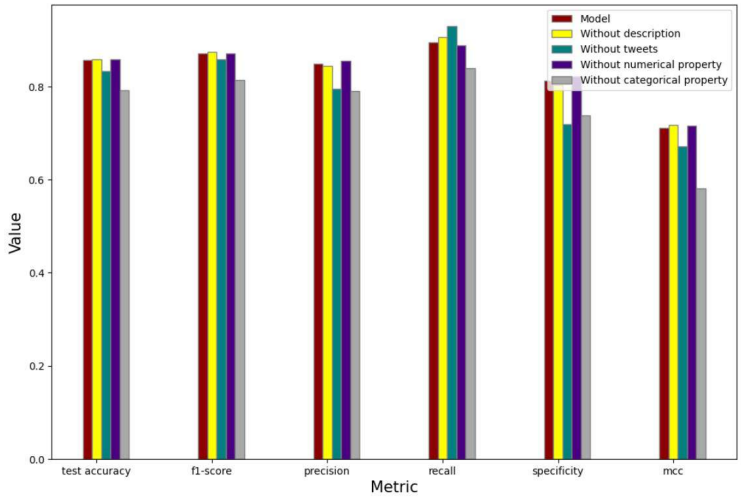
7.2.1 User's features

To prove that using multi-modal information is vital to our model performance we will conduct an ablation study to train the architecture we found produced the best results with reduced features. We will reduce one feature at a time for the first part.

Model	Accuracy	F1-score	Precision	Recall	Specificity	MCC
Ours	0.857	0.871	0.849	0.895	0.812	0.712
w/o description	**0.859**	**0.875**	0.845	0.906	0.804	**0.718**
w/o tweets	0.833	0.858	0.796	**0.93**	0.719	0.671
w/o numerical	**0.859**	0.872	**0.856**	0.889	**0.823**	0.716
w/o categorical	0.792	0.814	0.791	0.84	0.738	0.582

Πίνακας 7.3: *Training model without one feature*

We conclude that all the features are crucial to the model's performance to an extent. Notably, training without descriptions has a higher and accuracy and F1-score than the original model. Also, training without tweets has a higher recall value. These remarks are important to consider for future research, but training the model with all the features provides higher accuracy and makes it more adaptable to other datasets. For further understanding we will train the model using only one feature at a time, to investigate their importance separately.

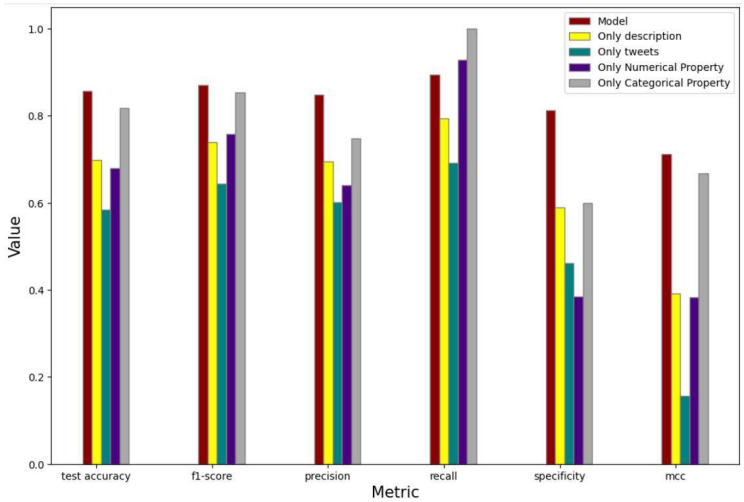


Εικόνα 7.3: *Training model without one feature*

Model	Accuracy	F1-score	Precision	Recall	Specificity	MCC
Ours	**0.857**	**0.871**	**0.849**	0.895	**0.812**	**0.712**
only description	0.699	0.74	0.695	0.793	0.589	0.392
only tweets	0.585	0.643	0.602	0.691	0.461	0.157
only numerical	0.679	0.758	0.641	0.929	0.385	0.383
only categorical	0.817	0.853	0.747	**1.000**	0.6	0.667

Πίνακας 7.4: *Training model with only one feature*

Obviously, the model trained with all the features has the best accuracy. From the results, we deduce that the categorical property is the feature that contributes the most to the model's sufficient accuracy. This ablation study proves that all features are advantageous for training our model to perform well in the task of bot detection. However, they do not contribute equally, and more studies to enhance the quality of the datasets could benefit future studies of bot detection.



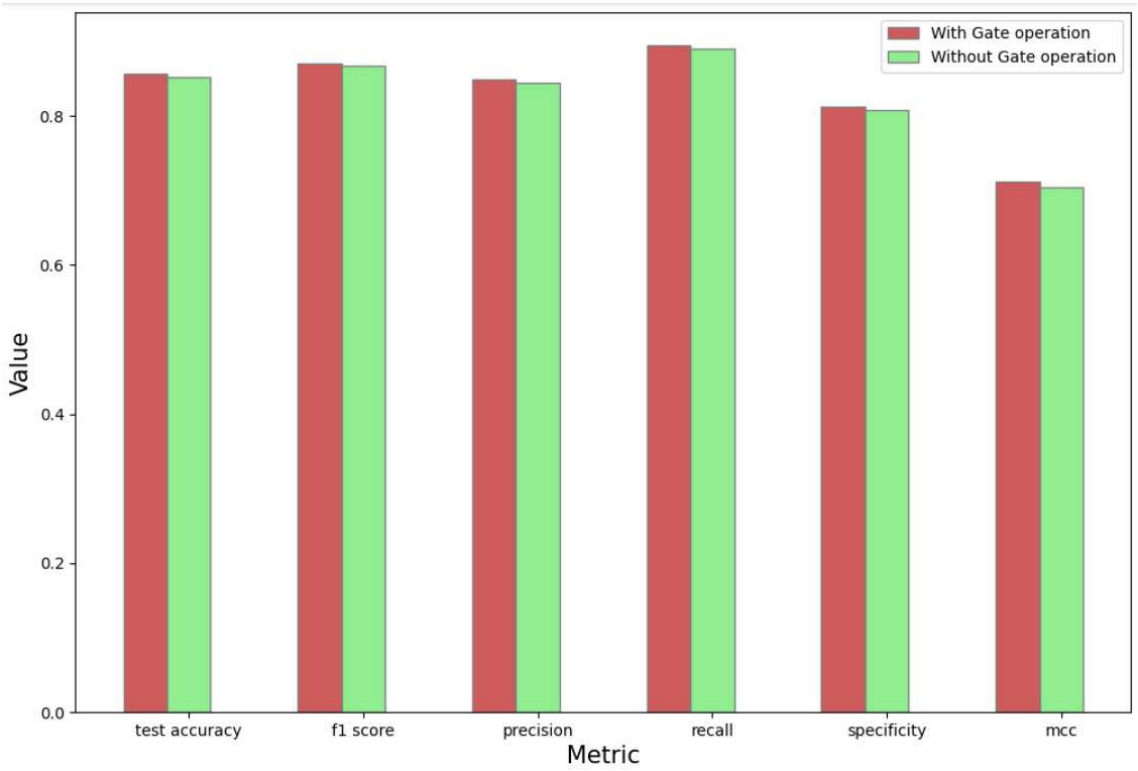
Εικόνα 7.4: *Training model with only one feature*

7.2.2 Gate operation

Model	Accuracy	F1-score	Precision	Recall	Specificity	MCC
With Gate	**0.857**	**0.871**	**0.849**	**0.895**	**0.812**	**0.712**
Without Gate	0.853	0.867	0.845	0.891	0.808	0.704

Πίνακας 7.5: *Ablation study on Gate operation*

We compare the architecture that results from the architecture search with a Gate operation and without a Gate operation. We see that the architecture without the gate has a reduced accuracy by 0.5% compared to the model's and a reduced F1-score by 0.46%. The gating mechanism dynamically aggregates the information from all the propagation steps and manages to control the smoothness of different nodes. Without it, the T operations take as input only the last output of the P steps. This is the reason the model underperforms without the Gate operation in the P functions, as it may suffer from over-smoothing. The architectures that are examined during this search have more T steps and shallower propagation processes, failing to obtain information from nodes during message passing as successfully as the original model. This ablation study proves the importance of the Gate operation in the P functions during our architecture search.



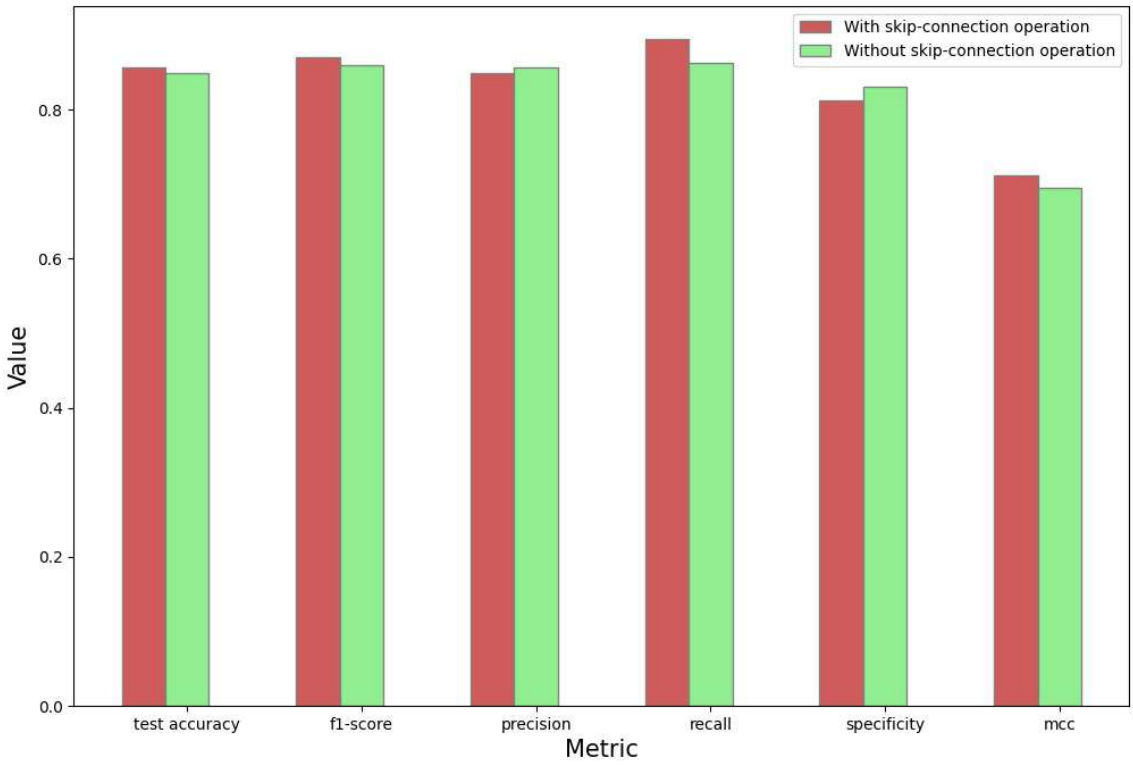
Εικόνα 7.5: *Plot for ablation study on Gate operation*

7.2.3 Skip-connection operation

Model	Accuracy	F1-score	Precision	Recall	Specificity	MCC
With skip	**0.857**	**0.871**	0.849	**0.895**	0.812	**0.712**
Without skip	0.849	0.86	**0.857**	0.863	**0.831**	0.695

Πίνακας 7.6: *Ablation study on skip-connection operation*

We compare the architecture that results from the architecture search with a skip-connection operation and without a skip-connection operation. We see that the architecture without the gate has a reduced accuracy by 0.93% compared to the model's and a reduced F1-score by 1.2%. Without the skip-connection operation, the input of the T steps is only the output of the last step. This may lead to the degradation of the model as the transformation functions can increase. The processing of the messages from nodes is not as effective and the accuracy declines. This ablation study proves the importance of the skip-connection operation in the T functions during our architecture search.



Εικόνα 7.6: *Plot for ablation study on skip-connection operation*

Κεφάλαιο 8

Conclusion

In this Chapter, we summarize the results of this thesis and discuss some directions for future research on the task of social media bot detection.

8.1 General Conclusion

In this thesis, we studied the task of detecting bots on Twitter. Twitter is a social platform that has experienced rapid growth and thus the presence of automated accounts, known as bots, is more pervasive than ever. Bots aim to spread fake information and manipulate users, by flooding users' timelines and comment sections. While bots attain the capability to better imitate human behavior the task of detecting them becomes more and more challenging. Bot detection is a subject that has been the topic of many studies in recent years and many different methods have been proposed.

We focus on the use of Graph Neural Networks, which are designed to handle data in a graph structure. Specifically, we focus on Graph Convolutional Networks, which include techniques for message passing between nodes to get information from the entire graph. We are inspired by the work of BotRGCN [38] and build a heterogeneous graph, where the users are represented with nodes and the following relationships between them with edges. The users also contain metadata information, including their descriptions, tweets, and numerical and categorical properties.

In an effort to enhance the performance of the model, we looked into the implementation of Graph Neural Architecture Search, a process that returns the architecture with the highest accuracy. We are inspired by the work of DFG-NAS [6], which searches for the most effective permutation of propagation (P) and transformation (T) operations that are implemented into the GNN. Therefore, we overcome the limitations of fixed architectures. With the utilization of the Gate operation and skip-connection operation, we overcome over-smoothing and model degradation that would reduce the model's accuracy. The search follows an evolutionary algorithm, a method heavily inspired by natural evolution that attempts to enhance a population through mutations.

Our model makes use of relational GCNs after performing architecture search, to find the most efficient P-T configuration. We used the Twibot-20 dataset, which is an optimal dataset that includes many types of user information and following relations. We conclude that the five architectures with the highest validation accuracy are quite

efficient in our task and compete with other models. Meanwhile, the one with the highest accuracy achieves a test accuracy of 85,7%, surpassing other state-of-the-art models for bot detection.

8.2 Future Work

The results obtained from this work underline the advantages of using Neural Architecture Search and Relational GCNs in the task of bot detection. They are particularly satisfactory and encouraging in the direction of further research.

First, it is important to recognize the limitations of our resources. Having the resources that could support more generations on the architecture search could be beneficial for our research, as it could examine deeper architectures. Thus, perhaps a model with more layers could present even better results.

A first expansion of the current work could be applying the model to other datasets. This model could probably be adapted to utilize more user information. For example, we could enhance the training by adding information like timezone, time of tweets, etc. In addition to user metadata we could add more user relations, such as messages, retweets, etc. Using other datasets will improve the adaptability of the model and could present better results.

Applying the model to the real world could mean adapting it to identify bots in real time. This could be performed using an application that takes into account user metadata during the creation of the account. This could also be implemented in short periods (daily, hourly) to also take into account relations.

An extension of our model in real-world cases would benefit from the use of dynamic graphs. Dynamic graphs are a type of graph data structure in which the structure of the graph changes over time. In a dynamic graph, edges and nodes can be added or removed at different time points, reflecting the evolution of relationships or connections between entities. This could be particularly useful in social networks where new users are added constantly and the relationships between the users alter.

Another further study could be investigating bots in other social media, like Facebook or Instagram. User metadata is different in other social platforms and relations between users can vary. Perhaps state-of-the-art methods in bot detection in other platforms could benefit from Neural Architecture Search.

Βιβλιογραφία

[1] *Weighted graph*. https://hyperskill.org/learn/step/5645.

[2] *Node Representation Learning*. https://snap-stanford.github.io/cs224w-notes/machine-learning-with-networks/node-representation-learning.

[3] *Graph Neural Networks Series | Part 4 |The GNNs, Message Passing & Over-smoothing*. https://medium.com/the-modern-scientist/graph-neural-networks-series-part-4-the-gnns-message-passing-over-smoothing-e77ffee523cc.

[4] *Graph Neural Networks - An overview*. https://theaisummer.com/Graph_Neural_Networks/.

[5] Petar Veličković; Guillem Cucurull; Arantxa Casanova; Adriana Romero; Pietro Liò; Yoshua Bengio. *Graph Attention Networks*. *arXiv:1710.10903*, 2017.

[6] Zhang Wentao; Lin Zheyu; Shen Yu; Li Yang; Yang Zhi; Cui Bin. *Deep and Flexible Graph Neural Architecture Search*. *Proceedings of the 39th International Conference on Machine Learning*, 162:26362–26374, 2022.

[7] *Computer Representation of Graphs*. https://www.cs.mtsu.edu/~xyang/3080/adjacencyMatrix.html.

[8] Cabral Frederico; Osthoff Carla; Nardes Rafael; Ramos Daniel. *Massive Parallelism With Gpus for Centrality Ranking in Complex Networks*. *International Journal of Computer Science and Information Technology*, 6:21–37, 2014.

[9] *WHAT IS THE DIFFERENCE BETWEEN BFS AND DFS ALGORITHMS*. https://www.freelancinggig.com/blog/2019/02/06/what-is-the-difference-between-bfs-and-dfs-algorithms/.

[10] *Minimum spanning tree*. https://en.wikipedia.org/wiki/Minimum_spanning_tree.

[11] *Algorithms Overview*. https://www.valueflo.ws/algorithms/overview/.

[12] *What Is Reinforcement Learning? Working, Algorithms, and Uses*. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-reinforcement-learning/.

[13] *An Insight to Linear Regression in Machine learning*. https://medium.com/analytics-vidhya/an-insight-to-linear-regression-in-machine-learning-a9a1585ae0b2.

[14] *Support Vector Machine and it's Mathematical Implementation*. https://medium.com/@akshay26072000/support-vector-machine-and-its-mathematical-implementation-b7a6f2071422.

[15] Irina Matveeva. *Predictive Coding in E-discovery and the NexLP Story Engine*. *Legal Informatics*, σελίδα 315–334, 2021.

[16] *Random Forest Algorithm*. https://www.javatpoint.com/machine-learning-random-forest-algorithm.

[17] *Pros and Cons of K-Nearest Neighbors*. https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/.

[18] *10 Common Statistical Models You Should Know*. https://algogene.com/community/post/111.

[19] *Multilayer perceptrons*. https://users.ics.aalto.fi/ahonkela/dippa/node41.html.

[20] *Sigmoid function*. https://en.wikipedia.org/wiki/Sigmoid_function.

[21] *L10: Neural networks for NLP*. https://princeton-nlp.github.io/cos484/lectures/lec10.pdf.

[22] *The Purpose Of Activation Functions*. https://khouloud-alkhammassi.medium.com/the-purpose-of-activation-functions-e49a7c2fb97d.

[23] *Why do Neural Networks Need an Activation Function?* https://towardsdatascience.com/why-do-neural-networks-need-an-activation-function-3a5f6a5f00a.

[24] *TensorFlow - Multi-Layer Perceptron Learning*. https://www.tutorialspoint.com/tensorflow/tensorflow_multi_layer_perceptron_learning.htm.

[25] *Hopfield Neural Network*. https://www.geeksforgeeks.org/hopfield-neural-network/.

[26] *Self Organizing Maps*. https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065.

[27] *A Brief Overview of Recurrent Neural Networks (RNN)*. https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/.

[28] Varsamopoulos Savvas; Bertels Koen; Almudever Carmen. *Designing neural network based decoders for surface codes*. *arXiv:1811.12456*, 2018.

[29] Anurag Kulshrestha; Venkataraghavan Krishnaswamy; Mayank Sharma. *Bayesian BILSTM approach for tourism demand forecasting*. *Annals of Tourism Research*, 83, 2020.

[30] *Convolutional Neural Networks*. https://paperswithcode.com/methods/category/convolutional-neural-networks.

[31] *The Fundamentals of Neural Architecture Search (NAS)*. https://pub.towardsai.net/the-fundamentals-of-neural-architecture-search-nas-9bb25c0b75e2.

[32] *Receiver operating characteristic*. https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[33] *model_selection*. http://ethen8181.github.io/machine-learning/model_selection/model_selection.html.

[34] *The Evolution of Social Media: From Six Degrees to Facebook and Beyond*. https://www.linkedin.com/pulse/evolution-social-media-from-six-degrees-facebook-beyond-iyer/. Ημερομηνία πρόσβασης: 03-03-2023.

[35] *HOW MANY PEOPLE USE SOCIAL MEDIA IN 2023?* https://www.oberlo.com/statistics/how-many-people-use-social-media.

[36] *Twitter*. https://www.britannica.com/topic/Twitter. Ημερομηνία πρόσβασης: 31-03-2023.

[37] *What are 'bots' and how can they spread fake news?* https://www.bbc.co.uk/bitesize/articles/zjhg47h.

[38] Shangbin Feng; Herun Wan; Ningnan Wang; Minnan Luo. *BotRGCN: Twitter Bot Detection with Relational Graph Convolutional Networks*. *arXiv:2106.13092*, 2021.

[39] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li και Minnan Luo. *TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark*. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[40] Peter Meltzer; Marcelo Daniel Gutierrez Mallea; Peter J. Bentley. *PiNet: A Permutation Invariant Graph Neural Network for Graph Classification*. *arXiv:1905.03046*, 2019.

[41] Justin Gilmer; Samuel S. Schoenholz; Patrick F. Riley; Oriol Vinyals; George E. Dahl. *Neural Message Passing for Quantum Chemistry*. *arXiv:1704.01212*, 2017.

[42] Thomas N. Kipf; Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. *arXiv:1609.02907*, 2016.

[43] William L. Hamilton; Rex Ying; Jure Leskovec. *Inductive Representation Learning on Large Graphs*. *arXiv:1706.02216*, 2017.

[44] Petar Veličković; Guillem Cucurull; Arantxa Casanova; Adriana Romero; Pietro Liò; Yoshua Bengio. *Graph Attention Networks*. *arXiv:1710.10903*, 2017.

[45] Keyulu Xu; Weihua Hu; Jure Leskovec; Stefanie Jegelka. *How Powerful are Graph Neural Networks?* *arXiv:1810.00826*, 2018.

[46] Kurt Hornik. *Approximation capabilities of multilayer feedforward networks*. *Neural Networks*, 4(2):251–257, 1991.

[47] Barret Zoph; Quoc V. Le. *Neural Architecture Search with Reinforcement Learning*. *arXiv:1611.01578*, 2016.

[48] Barret Zoph; Vijay Vasudevan; Jonathon Shlens; Quoc V. Le. *Learning Transferable Architectures for Scalable Image Recognition*. *arXiv:1707.07012*, 2017.

[49] Chenxi Liu; Liang Chieh Chen; Florian Schroff; Hartwig Adam; Wei Hua; Alan Yuille; Li Fei-Fei. *Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation*. *arXiv:1901.02985*, 2019.

[50] Tao Wei; Changhu Wang; Chang Wen Chen. *Modularized Morphing of Neural Networks*. *arXiv:1701.03281*, 2017.

[51] Hanxiao Liu; Karen Simonyan; Yiming Yang. *DARTS: Differentiable Architecture Search*. *arXiv:1806.09055*, 2018.

[52] Liu Yinhan; Ott Myle; Goyal Naman; Du Jingfei; Joshi Mandar; Chen Danqi; Levy Omer; Lewis Mike; Zettlemoyer Luke και Stoyanov Veselin. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. *arXiv:1907.11692*, 2019.

[53] Lee K.; Eoff B.; Caverlee J. *Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter.* *Proceedings of the International AAAI Conference on Web and Social Media*, 5(1):185–192, 2011.

[54] Yang K. C.; Varol O.; Hui P. M.; Menczer F. *Scalable and Generalizable Social Bot Detection through Data Selection.* *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(1):1096–1103, 2020.

[55] S. Cresci; R. Di Pietro; M. Petrocchi; A. Spognardi; και M. Tesconi. *Social fingerprinting: detection of spambot groups through dna-inspired behavioral modeling,*. *IEEE Transactions on Dependable and Secure Computing*, 15(4):561–576, 2017.

[56] S. Cresci; R. Di Pietro; M. Petrocchi; A. Spognardi; και M.Tesconi. *The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race,*. *Proceedings of the 26th international conference on world wide web companion,*, σελίδα 963–972, 2017.

[57] *Botometer | RAND*. https://www.rand.org/research/projects/truth-decay/fighting-disinformation/search/items/botometer.html.

[58] Matheus Nunes και Gisele L. Pappa. *Neural Architecture Search in Graph Neural Networks*. *arXiv:2008.00077*, 2020.

[59] *BunsenFeng/TwiBot-20*. https://github.com/BunsenFeng/TwiBot-20.

[60] John J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 79(8):2554–2558, 1982.

[61] Kohonen T. *Self-organized formation of topologically correct feature maps*. *Biol. Cybern*, 43:59–69, 1982.

[62] S. Hochreiter; J. Schmidhuber. *LONG SHORT-TERM MEMORY*. *Neural Computation*, 9(8):1735–1780, 1997.

[63] K. Cho; B. Merriënboer; C. Gulcehre; D. Bahdanau; F. Bougares; H. Schwenk; Y. Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1724–1734, 2014.

[64] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng και Jun Zhou. *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022.

[65] *word2vec*. https://www.tensorflow.org/text/tutorials/word2vec.

[66] Jeffrey Pennington, Richard Socher και Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. *Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1532–1543, 2014.

[67] Jacob Devlin; Ming Wei Chang; Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. *arXiv:1810.04805*, 2018.

[68] S. Kudugunta και E. Ferrara. *Deep neural networks for bot detection*. *Information Sciences*, 467:312–322, 2018.

[69] Cai; Chiyu, li; Linjing και Zengi; Daniel. *Behavior enhanced deep bot detection in social media*. *IEEE International Conference on Intelligence and Security Informatics (ISI)*, σελίδες 128–130, 2017.

[70] F. Morstatter; L. Wu; T. H. Nazer; K. M. Carley; και H. Liu. *A new approach to bot detection: striking the balance between precision and recall,*. *Advances in Social Networks Analysis and Mining (ASONAM),2016 IEEE/ACM International Conference on. IEEE,*, σελίδες 533–540, 2016.

[71] F. Wei και U. T. Nguyen. *Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings*. *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), IEEE*, σελίδα 101–109, 2019.

[72] J. P. Dickerson; V. Kagan και V. Subrahmanian. *Using sentiment to detect bots on twitter: Are humans more opinionated than bots?* *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), IEEE*, σελίδα 620–627, 2014.

[73] Zachary Miller; Brian Dickinson; William Deitrick; Wei Hu και Alex Hai Wang. *Twitter spammer detection using data stream clustering*. *Information Sciences 260 (2014)*, σελίδα 64–73, 2014.

[74] Yang K.C.; Varol O.; Davis C.A.; Ferrara E.; Flammini A.; Menczer F. *Arming the public with artificial intelligence to counter social bots.* *Hum. Behav. Emerg. Technol.*, 1:48–61, 2019.

[75] S. Feng; H. Wan; N. Wang; J. Li; και M. Luo. *Satar: A self-supervised approach to twitter account representation learning and its application in bot detection.* *arXiv preprint arXiv:2106.13089*, 2021.

[76] Alothali E.; Salih M.; Hayawi K.; Alashwal H. *Bot-MGAT: A Transfer Learning Model Based on a Multi-View Graph Attention Network to Detect Social Bots.* *Applied Sciences.*, 12(16), 2022.

[77] Shangbin Feng; Zhaoxuan Tan; Rui Li; Minnan Luo. *Heterogeneity-aware Twitter Bot Detection with Relational Graph Transformers*. *arXiv:2109.02927*, 2021.

[78] Kaixiong Zhou; Qingquan Song; Xiao Huang; Xia Hu. *Auto-GNN: Neural Architecture Search of Graph Neural Networks*. *arXiv:1909.03184*, 2019.

[79] Prithviraj Sen; Galileo Namata; Mustafa Bilgic; Lise Getoor; Brian Galligher; και Tina Eliassi-Rad. *Collective classification in network data*. *AI magazine*, 29(3):93–93, 2008.

[80] Marinka Zitnik και Jure Leskovec. *Predicting multicellular function through multi-layer tissue networks*. *Bioinformatics*, 33(14):190–198, 2017.

[81] Yang Gao; Hong Yang; Peng Zhang; Chuan Zhou; Yue Hu. *GraphNAS: Graph Neural Architecture Search with Reinforcement Learning*. *arXiv:1904.09981*, 2019.

[82] Huan Zhao; Lanning Wei; Quanming Yao. *Simplifying Architecture Search for Graph Neural Network*. *arXiv:2008.11652*, 2020.

[83] S. Jiang και P. Balaprakash. *Graph Neural Network Architecture Search for Molecular Property Prediction*. *2020 IEEE International Conference on Big Data (Big Data)*, σελίδες 1346–1353, 2020.

[84] Huan Zhao; Quanming Yao; Weiwei Tu. *Search to aggregate neighborhood for graph neural network*. *arXiv:2104.06608*, 2021.

[85] Cai Shaofei; Li Liang; Deng Jincan; Zhang Beichen; Zha Zheng Jun; Su Li; Huang Qingming. *Rethinking Graph Neural Architecture Search from Message-passing*. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 6653–6662, 2021.

[86] Li Y.; Wen Z.; Wang Y.; Xu C. *One-shot Graph Neural Architecture Search with Dynamic Search Space.* *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8510–8517, 2021.

[87] Wei Peng; Xiaopeng Hong; Haoyu Chen; Guoying Zhao. *Learning Graph Convolutional Network for Skeleton-based Human Action Recognition by Neural Searching*. *arXiv:1911.04131*, 2019.

[88] *pandas*. https://pandas.pydata.org.