Εφαρμογή μεθόδων μηχανικής μάθησης για την ανίχνευση κακόβουλου λογισμικού

Διπλωματική Εργασία

Λαμπράκης Δημήτριος
Μεταπτυχιακός Φοιτητής
ΕΔΕΜΜ
ΑΜ 03400059

Επιβλέπουσα:
Ιωάννα Ρουσσάκη
Επίκουρη Καθηγήτρια
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
(Σ.Η.Μ.Μ.Υ.)

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο
Ελλάδα
11 Νοεμβρίου 2021

Εφαρμογή μεθόδων μηχανικής μάθησης για την ανίχνευση κακόβουλου λογισμικού

Διπλωματική Εργασία

Λαμπράκης Δημήτριος
Μεταπτυχιακός Φοιτητής
ΕΔΕΜΜ
ΑΜ 03400059

Επιβλέπουσα:
Ιωάννα Ρουσσάκη
Επίκουρη Καθηγήτρια
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
(Σ.Η.Μ.Μ.Υ.)

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την $11^{η}$ Νοεμβρίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ιωάννα Ρουσσάκη	Μιλτιάδης Αναγνώστου	Συμεών Παπαβασιλείου
Επίκουρη Καθηγήτρια Σ.Η.Μ.Μ.Υ.	Καθηγητής Σ.Η.Μ.Μ.Υ.	Καθηγητής Σ.Η.Μ.Μ.Υ.

Δημήτριος Λαμπράκης
Πτυχιούχος Φυσικός Ε.Κ.Π.Α

Περίληψη

Η χρήση της μηχανικής μάθησης στον τομέα της κυβερνοασφάλειας τα τελευταία χρόνια έχει αυξηθεί κατακόρυφα. Ιδιαίτερα το πεδίο της ανίχνευσης κακόβουλου λογισμικού προσφέρεται εξαιρετικά για την εφαρμογή μεθόδων μηχανικής μάθησης. Η παρούσα εργασία στοχεύει στην ανάπτυξη ενός ανιχνευτή βασισμένου εξ' ολοκλήρου σε τεχνικές βαθιάς μάθησης. Προφανώς, για την σωστή ανάπτυξη ενός ανιχνευτή κακόβουλου λογισμικού, αρχικά, παρουσιάζεται η βασική γνώση λειτουργίας των κακόβουλων λογισμικών, του λειτουργικού συστήματος πάνω στο οποίο εκτελούνται αλλά και των κλασσικών μεθόδων ανίχνευσης. Η λύση ενός ανιχνευτή βασισμένου σε τεχνικές μηχανικής μάθησης έρχεται για να καλύψει κάποια κενά που υπάρχουν από τις κλασσικές προσεγγίσεις με κυριότερο την απαίτηση για χειροκίνητη ανανέωση της βάσης που εμπεριέχει τις πληροφορίες για την ταυτοποίηση των απειλών. Η διεθνής βιβλιογραφία βρίθει από δημοσιεύσεις που αντιμετωπίζουν το συγκεκριμένο πρόβλημα. Η προσπάθεια διαφοροποίησης γίνεται με γνώμονα δύο βασικά σημεία: στον χρόνο που χρειάζεται ο ανιχνευτής για την εκπαίδευση μέχρι να δίνει καλά αποτελέσματα και στην επεξεργασία των δεδομένων. Ο τύπος των δεδομένων που επιλέχθηκε είναι στατικές εικόνες που παράχθηκαν από την δυαδική αναπαράσταση των εκτελέσιμων αρχείων. Τα στατικά χαρακτηριστικά είναι τα πιο ασφαλή να εξαχθούν καθώς δεν απαιτείται δυναμική ανάλυση και επειδή είναι σε μορφή εικόνων υπάρχει η δυνατότητα εκμετάλλευσης πανίσχυρων τεχνητών νευρωνικών δικτύων που ειδικεύονται στην αναγνώριση και κατηγοριοποίηση εικόνων. Κατά την διάρκεια των πειραμάτων πραγματοποιήθηκε η μείωση των διαστάσεων των εικόνων και η εξαγωγή πιο αφηρημένων χαρακτηριστικών πριν την τελική τροφοδοσία στα δίκτυα κατηγοριοποίησης. Η μείωση διαστάσεων και η εξαγωγή έγιναν από ένα δίκτυο autoencoder και τα αποτελέσματα ήταν εκπληκτικά. Με έναν αρκετά μικρό αριθμό συνολικών επαναλήψεων ένα απλό δίκτυο CNN πλησίασε σε ακρίβεια πολλά και περίπλοκα δίκτυα από την διεθνή βιβλιογραφία, ενώ ένα προ εκπαιδευμένο δίκτυο ResNet18 έφτασε την ακρίβεια από δημοσιεύματα της διεθνής βιβλιογραφίας με λίγες επαναλήψεις και βασισμένο εξ' ολοκλήρου σε τεχνικές βαθιάς μάθησης, δεν χρειάστηκε πουθενά χειροκίνητη παρέμβαση. Κλείνοντας, τα πειράματα που εκτελέστηκαν δείχνουν ξεκάθαρα ότι μια λύση βασισμένη σε μεθόδους βαθιάς μηχανικής μάθησης για τον εντοπισμό κακόβουλου λογισμικού είναι μια απολύτως βιώσιμη λύση και προσφέρεται πάρα πολύ για περαιτέρω έρευνα και βελτιστοποίηση.

Λέξεις Κλειδία

Επιστήμη δεδομένων, Μηχανική μάθηση, Βαθιά μάθηση, Κακόβουλο λογισμικό, Ανίχνευση, Στατικά χαρακτηριστικά, Εικόνες, Μείωση διαστάσεων, Εξαγωγή Χαρακτηριστικών, Δίκτυα CNN, Δίκτυα autoencoder

Abstract

The use of machine learning models in the field of cybersecurity has increased drastically over the last few years. More specifically malware detection applications have seen great advancements with the use of machine learning. This thesis focuses on developing a malware detector based completely on deep learning techniques. First of all some basic knowledge on topics regarding the methods of operation of malware executables, the underlying operating system and traditional detection methods is presented. The proposed machine learning detector comes to combat a major flaw that traditional detection methods introduce, the need to manually update the database that holds the information required to detect new malware. Internationally a lot of research papers have been published trying to solve the detection challenge. This thesis tries to differentiate in two crucial points: the time required by the detector to train and yield sufficiently good results and the processing of the data. The malware executables were transformed into static images, static features were chosen because they are the safest to extract since no dynamic analysis is required. Using images powerful artificial neural networks can be deployed in order to detect and classify the images as malware or benign executables. During the testing phase, it became quite clear that a dimensionality reduction and feature extraction technique is mandatory, so an autoencoder net was used and the results were exceptional. With a relatively low number of total iterations a simple CNN was able to approach the performance of many and much more complex solutions proposed by many research papers whereas a pretrained model based on ResNet18 was able to achieve similar results, using only deep learning techniques, nothing was done manually. To conclude, the experiments that were performed clearly show that a machine learning based malware detector is a completely viable solution that offers excellent protection and a great topic for further research and optimization.

Keywords

Data science, Machine learning, Deep learning, Malware, Detection, Static features, Images, Dimensionality reduction, Feature Extraction, CNN networks, Autoencoder networks

Περιεχόμενα

Ευρετήριο Πινάκων

Ευρετήριο Σχημάτων

1 Εισαγωγή

1.1 Πρόλογος

Το κακόβουλο λογισμικό ή malware όπως αναφέρεται στην διεθνή βιβλιογραφία είναι λογισμικό που έχει αναπτυχθεί με στόχο να προκαλέσει κακό σε κάποιους χρήστες - στόχους. Το κακόβουλο λογισμικό συνήθως εκμεταλλεύεται τα προσωπικά αρχεία των χρηστών - στόχων είτε για την εξαγωγή ευαίσθητων προσωπικών δεδομένων είτε για την απόκτηση χρημάτων μέσω της "ομηρίας" ευαίσθητων αρχείων είτε πολλές φορές χωρίς κανέναν λόγο απλά για να προκαλέσει αναστάτωση και ταραχή. Ο όρος "κακόβουλο λογισμικό" είναι αρκετά ευρύς όπως θα δούμε στην συνέχεια και περιλαμβάνει μια πληθώρα διαφορετικών εφαρμογών και τύπων επιθέσεων. Όμως, υπάρχει ένα κοινό σημείο αναφοράς και αυτό είναι ότι όλα τα κακόβουλα λογισμικά έχουν αυτό που ονομάζεται "ωφέλιμο φορτίο". Ακόμα και αν δύο, ή περισσότερα, κακόβουλα λογισμικά είναι φαινομενικά εντελώς διαφορετικά και καταλήγουν στην ίδια επίθεση τότε υπάρχει περίπτωση να μοιράζονται το ίδιο "ωφέλιμο φορτίο".

Τα τελευταία χρόνια έχουν αυξηθεί ραγδαία οι αναφορές σε επιθέσεις από κακόβουλα λογισμικά (viruses, trojans, ransomwares κ.α.) [1]. Από την εμφάνιση του πρώτου κακόβουλου λογισμικού, το 1971 [2], η απάντηση στην αντιμετώπιση του κακόβουλου λογισμικού βασιζόταν κυρίως σε τεχνικές εντοπισμού στοιχείων, ή πιο σωστά υπογραφών, ήδη γνωστού κακόβουλου λογισμικού. Δυστυχώς, κάποιες από αυτές τις επιθέσεις εκμεταλλεύονται ευπάθειες συστημάτων που ονομάζονται 0days. Ευπάθειες για τις οποίες, μέχρι την στιγμή της επίθεσης, δεν έχει υπάρξει κάποια ένδειξη ότι υπάρχουν ή ότι μπορούν να χρησιμοποιηθούν για κακόβουλους σκοπούς. Τα 0days είναι μόνο ένας από τους λόγους που πλέον τα κλασσικά συστήματα προστασίας εναντίων του κακόβουλου λογισμικού αδυνατούν να μας προστατεύουν πλήρως. Ακόμα πιο συνηθισμένες περιπτώσεις αδυναμίας προστασίας από κακόβουλα λογισμικά είναι οι εφαρμογές μεθόδων κρυπτογράφησης και συσκότισης των εκτελέσιμων αρχείων ακριβώς για την αποφυγή κάποιας ταυτοποίησης με ήδη υπάρχον και γνωστό δείγμα κακόβουλου λογισμικού. Από την άλλη πλευρά, την πλευρά της ασφάλειας, θα μπορούσαμε ν' αναφέρουμε ότι βρισκόμαστε στην "χρυσή εποχή" του Deep Learning. Η τεράστια αύξηση των νέων δειγμάτων κακόβουλου λογισμικού καθημερινά, μπορεί να λειτουργήσει με θετικό τρόπο σε μια προσέγγιση μηχανικής μάθησης καθώς υπάρχει υπέρ αρκετός όγκος δεδομένων, και ποιοτικών δεδομένων, που απαιτούνται, συνήθως, για την σωστή εφαρμογή μεθόδων μηχανικής μάθησης. Επομένως, είναι φυσικό επακόλουθο να προσπαθήσουμε να εφαρμόσουμε τεχνικές Deep learning κατηγοριοποίησης στον τομέα της κυβερνοασφάλειας για την έγκαιρη και σωστή κατηγοριοποίηση πιθανού κακόβουλου λογισμικού.

Είναι σαφές, ότι υπάρχει ένας διαρκής αγώνας μεταξύ των ερευνητών ασφαλείας και των προγραμματιστών κακόβουλου λογισμικού καθώς και οι δύο πλευρές αγωνίζονται εναντίον της άλλης πλευράς αλλά και εναντίον του χρόνου για το ποιος θα κρατήσει το προβάδισμα, δεδομένου ότι ο συγκεκριμένος αγώνας μάλλον είναι ατέρμονος. Σε αυτό το σημείο να σημειώσουμε ότι δεν αναφερόμαστε σε περιπτώσεις όπου για παράδειγμα έχει βρεθεί κάποιο κενό ασφαλείας σε επίπεδο υλικού με αποτέλεσμα να μην μπορεί να διορθωθεί με τίποτα ή σε πολύ απλά συστήματα που ανεύρεση κενών ασφαλείας μοιάζει αδύνατη. Αναφερόμαστε στην γενικευμένη κατάσταση που έχει παρατηρηθεί να επαναλαμβάνεται ανά τακτά διαστήματα. Με την παρούσα εργασία, λοιπόν, θα προσπαθήσουμε, στηριζόμενοι σε τεχνικές μηχανικής μάθησης να αναπτύξουμε ένα εργαλείο το οποίο θα είναι σε θέση να βοηθάει τους ερευνητές ασφαλείας να διατηρούν το προβάδισμα εναντίον του κακόβουλου λογισμικού για μεγαλύτερα χρονικά διαστήματα χωρίς να εξαρτώνται πλέον τόσο στενά από προϋπάρχουσα

γνώση παρόμοιων δειγμάτων.

1.2 Δομή της Εργασίας

Έχοντας αναφέρει τα βασικά κίνητρα που μας οδήγησαν στην επιλογή του συγκεκριμένου θέματος στην παρούσα εργασία, είμαστε σε θέση να παρουσιάσουμε την δομή που θα ακολουθήσει η εργασία:

- Το δεύτερο κεφάλαιο αφορά το συναφές υπόβαθρο. Προτού αρχίσει η ενασχόληση με το κύριο θέμα της εργασίας που είναι η μηχανική μάθηση ενάντια του κακόβουλου λογισμικού είναι απαραίτητο να παρουσιαστούν τα θεμελιώδη γνωρίσματα των πεδίων πάνω στα οποία χτίζεται η εργασία. Τα κύρια πεδία που αναφέρονται είναι: το κακόβουλο λογισμικό, η δομή των εκτελέσιμων αρχείων, οι κλασσικοί τρόποι αντιμετώπισης του κακόβουλου λογισμικού και οι βασικές αρχές της μηχανικής μάθησης.

- Το τρίτο κεφάλαιο αφιερώνεται στην ανασκόπηση της διεθνούς βιβλιογραφίας. Προφανώς, στην "χρυσή εποχή" της βαθιάς μηχανικής μάθησης ένα πεδίο με ευρεία εφαρμογών και μεγάλο όγκο δεδομένων, όπως αυτό της κυβερνοασφάλειας εναντίον του κακόβουλου λογισμικού, δεν θα πέρναγε απαρατήρητο από την διεθνή κοινότητα. Έτσι μια ανασκόπηση των ήδη δοκιμασμένων μεθόδων είναι σε θέση να εμπνεύσει νέες ιδέες για την κάλυψη πιθανών ελλείψεων αλλά και να αποτρέψει από μεθόδους που πλέον είναι ξεπερασμένες ή εξαρχής δοκιμάστηκαν και δεν δίνουν τα αναμενόμενα αποτελέσματα.

- Το τέταρτο κεφάλαιο ορίζει το πρόβλημα το οποίο θα αντιμετωπιστεί στην παρούσα εργασία. Συγκεκριμένα, θέτονται κάποιες απαιτήσεις, οι οποίες προέκυψαν από την μελέτη της διεθνούς βιβλιογραφίας, τις οποίες καλείται η λύση που θα προταθεί να τις καλύπτει ή ακόμα καλύτερα να τις ξεπερνάει. Οι βασικές απαιτήσεις που εισάγονται αφορούν τον χρόνο που θα χρειαστεί η εκάστοτε λύση για να καλύψει την δεύτερη απαίτηση της ακρίβειας.

- Το πέμπτο κεφάλαιο ασχολείται με τα δεδομένα. Ειδικότερα, γίνεται αναφορά στον τρόπο συγκέντρωσης των δεδομένων για την προετοιμασία του "συνόλου δεδομένων". Έπειτα, παρουσιάζεται η τελική μορφή που θα έχουν τα δεδομένα και ο τρόπος μετατροπής τους. Αμέσως μετά αναφέρεται μια βασική τεχνική μείωσης διαστάσεων και εξαγωγής χρήσιμων χαρακτηριστικών, ο autoencoder. Ενώ κλείνοντας το κεφάλαιο γίνεται μια εκτενής αναφορά στις μετρικές αξιολόγησης μιας προτεινόμενης μεθόδου, συναρτήσει των τελικών προβλέψεων επί των δεδομένων.

- Το έκτο κεφάλαιο παρουσιάζει λεπτομερώς τα πειράματα που εκτελέστηκαν με στόχο τον προσδιορισμό της καλύτερης δυνατής λύσης στο συγκεκριμένο πρόβλημα. Στο κεφάλαιο, υπάρχουν πέντε διακριτές υποενότητες με τις τρεις να αναφέρονται σε πειράματα που αποτελούν και πιθανές προτάσεις για την λύση του προβλήματος, μια να παρουσιάζει ένα ενδιάμεσο αλλά ύψιστης σημασία πείραμα εξαγωγής χαρακτηριστικών και μείωσης διαστάσεων και την τελική ενότητα που γίνεται μια καθολική σύγκριση αποτελεσμάτων.

- Το έβδομο κεφάλαιο συνοψίζει τα αποτελέσματα που εξήχθησαν από την παρούσα εργασία. Οριοθετεί το πλαίσιο μέσα στο οποίο μια λύση αντιμετώπισης κακόβουλου λογισμικού μπορεί να θεωρηθεί βιώσιμη ενώ παράλληλα προτείνει και επιπλέον έρευνες

για την περαιτέρω βελτίωση των μεθόδων μηχανικής μάθησης που παρουσιάστηκαν ενάντια στο κακόβουλο λογισμικό.

- Το όγδοο κεφάλαιο αποτελεί την σύνοψη της εργασίας και παρέχει μια περιεκτική απόδοση ολόκληρης της εργασίας, εμπεριέχοντας αναφορές από τα σημαντικότερα αποτελέσματα που εξήχθησαν.

2 Συναφές Υπόβαθρο

2.1 Κακόβουλο Λογισμικό

Ο Sun Tzu στο βιβλίο του "Η τέχνη του πολέμου"[3] έχει αναφέρει ότι "Αν γνωρίζεις τον εαυτό σου και τον εχθρό σου δεν χρειάζεται να φοβάσαι την έκβαση εκατοντάδων μαχών...". Για να μπορέσουμε, λοιπόν, ν' αμυνθούμε αποτελεσματικά στις απειλές στον κυβερνοχώρο, οφείλουμε πρώτα να καταλάβουμε τι είναι αυτό που μας απειλεί.

Οι Grégio, Afonso κ.α.[4] ορίζουν το κακόβουλο λογισμικό ως σύνολο εντολών που τρέχουν σε ένα σύστημα για να κάνει αυθαίρετες ενέργειες για λογαριασμό ενός επιτιθέμενου ή να ενεργήσει με (αυτόματο ή μη) τρόπο τέτοιο ώστε να απειλείται η ασφάλεια του αλωθέντος συστήματος, οι χρήστες του και σχετικά δεδομένα.

Η συντριπτική πλειοψηφία των επιθέσεων που θα συναντήσει κανείς στον χώρο της κυβερνοασφάλειας αφορά επιθέσεις που έγιναν με την βοήθεια κάποιου λογισμικού. Η εμπλοκή του υλικού (hardware) είναι αρκετά σπάνια καθώς απαιτείται φυσική πρόσβαση στα μηχανήματα-θύματα. Βέβαια, αν πραγματοποιηθεί επιτυχώς μια επίθεση με την βοήθεια hardware είναι αρκετά πιο δύσκολο να εντοπιστεί, αν έχει κρυφτεί και ρυθμιστεί κατάλληλα να μην αφήνει πολλά ίχνη, ενώ προσφέρει καλύτερα και πιο σταθερά αποτελέσματα. Όμως, οι επιθέσεις με χρήση λογισμικού δεν απαιτούν φυσική πρόσβαση στα μηχανήματα που θα πληγούν από την επίθεση και είναι πολύ πιο εύκολο να κατασκευαστεί και να διαμοιραστεί το εν λόγω λογισμικό.

Για να υπάρξει επιτυχημένη εισβολή κακόβουλου λογισμικού είναι απαραίτητα τα εξής [5]:

- Μνήμη στην οποία το κακόβουλο λογισμικό θα παραμείνει αδρανές για κάποιο χρονικό διάστημα
- Μνήμη RAM όπου θα φορτωθεί ο προς εκτέλεση κώδικας
- Μια κεντρική μονάδα επεξεργασίας που θα εκτελέσει τον κώδικα
- Ένα λειτουργικό σύστημα

Παρατηρούμε ότι οι δύο τελευταίες απαιτήσεις για μια επιτυχημένη εισβολή είναι αρκετά δεσμευτικές ως προς τον στόχο του κακόβουλου λογισμικού. Δηλαδή, λογισμικά που έχουν φτιαχτεί για να προσβάλουν υπολογιστές με λογισμικό "Windows" προφανώς δεν μπορούν να είναι αποτελεσματικές σε άλλα λογισμικά ή ένα κακόβουλο λογισμικό που έχει αναπτυχθεί σε επεξεργαστή αρχιτεκτονικής "ARM" δεν μπορεί να δουλέψει σε κάποιον επεξεργαστή με αρχιτεκτονική "x64_86".

Όπως και στην βιολογία η αντιμετώπιση μιας μόλυνσης εξαρτάται από τον τύπο της μόλυνσης έτσι και στα κακόβουλα λογισμικά είναι εξαιρετικά σημαντικό να γνωρίζουμε τον τύπο του κακόβουλου λογισμικού για να μπορέσουμε να το αντιμετωπίσουμε αποτελεσματικά. Οι κλάσεις στις οποίες μπορεί κανείς να κατηγοριοποιήσει τα κακόβουλα λογισμικά είναι αρκετές και εξαρτώνται άμεσα από την επιλογή του χαρακτηριστικού βάση του οποίου θα γίνει η κατηγοριοποίηση. Ίσως το πιο συνηθισμένο χαρακτηριστικό που επιλέγεται για να γίνει η κατηγοριοποίηση είναι ο τρόπος διάδοσης ή/και ο σκοπός του κακόβουλου λογισμικού. Για τον τρόπο διάδοσης του κακόβουλου λογισμικού έχουμε τις εξής τρεις κατηγορίες [6]:

- **Ιός ή Virus**: Τα κακόβουλα λογισμικά που χαρακτηρίζονται ως ιοί έχουν πάρει την ονομασία τους από τους βιολογικούς ιούς καθώς μοιράζονται την ίδια λογική

επίθεσης και διάδοσης. Ένας ιός αποτελείται από δύο υπο-προγράμματα. Το πρώτο υπο-πρόγραμμα είναι υπεύθυνο για να μολύνει άλλα, υγιή μέχρι πρότινος, προγράμματα με κώδικα του ιού. Το δεύτερο υπο-πρόγραμμα είναι αυτό που περιέχει στην πραγματικότητα το επιβλαβές φορτίου του ιού και όταν εκτελεστεί θα προκαλέσει ζημιά. Προφανώς, οι ιοί δεν είναι αυτόνομα προγράμματα, καθώς η εκτέλεση αλλά και η διάδοση τους γίνεται όταν εκτελούνται, ακούσια ή μη, τα μολυσμένα προγράμματα "ξενιστές" που έχουν μολυνθεί. Για να εξαπλωθούν περαιτέρω οι ιοί βασίζονται στους χρήστες που θα μεταφέρουν μολυσμένα αρχεία από το ένα σύστημα στο άλλο,

- **Σκουλήκι ή Worm**: Τα σκουλήκια μοιάζουν αρκετά με τους ιούς με την έννοια ότι είναι και αυτά αυτοαναπαραγώμενα. Όμως διαφέρουν σημαντικά στον τρόπο διάδοσής τους. Ένα σκουλήκι για να διαδοθεί στηρίζεται στο δίκτυο μεταξύ των συσκευών, είναι αυτόνομο πρόγραμμα και διαδίδεται εκμεταλλευόμενο ευπάθειες του δικτύου. Φυσικά, τα σκουλήκια μπορούν και αυτά να φέρουν επιβλαβές φορτίο για να προκαλέσουν ζημίες στα συστήματα που προσβάλλουν.

- **Δούρειος Ίππος ή Trojan**: Σε αντίθεση με τα σκουλήκια και τους ιούς τα κακόβουλα προγράμματα που χαρακτηρίζονται ως "Δούρειος Ίππος" δεν διαθέτουν μηχανισμούς αυτοαναπαραγωγής και αυτοδιάδοσης. Αντιθέτως, προσπαθούν να παραπλανήσουν τους χρήστες να τα εκτελέσουν. Στην περίπτωση που ο σκοπός τους επιτευχθεί, έχουν ξεπεράσει, πολύ εύκολα, τις δυσκολίες που έχουν να αντιμετωπίσουν οι ιοί και στα σκουλήκια για την διάδοσή τους, δηλαδή την ανάγκη για κάποιον ξενιστή ή για κάποια ευπάθεια στο δίκτυο.

Προφανώς, οι τρεις κατηγορίες που εξετάσαμε δεν είναι αμοιβαία αποκλειόμενες καθώς αρκετά συχνά παρουσιάζονται κακόβουλα λογισμικά που αποτελούν κάποιον συνδυασμό με σκοπό την ευρύτερη διάδοση (π.χ. ένας ιός που βασίζεται σε κάποιον "Δούρειο Ίππο").

Όπως ήδη αναφέρθηκε η κατηγοριοποίηση ανάλογα με τον τρόπο διάδοσης δεν είναι η μοναδική. Μια άλλη σημαντική κατηγοριοποίηση είναι ο σκοπός του κακόβουλου λογισμικού. Δηλαδή, μετά από μια επιτυχημένη προσβολή ενός συστήματος ποιος είναι ο στόχος του κακόβουλου λογισμικού. Η γνωστή εταιρία προστασίας ηλεκτρονικών συστημάτων McAfee [7] κάνει τις παρακάτω διακρίσεις:

- **Λογισμικό κατασκοπείας (spyware) και Διαφημίσεις (adware)**: Οι δύο τύποι κακόβουλου λογισμικού αποσκοπούν στην συλλογή ευαίσθητων πληροφοριών από τον χρήστη. Το μεν spyware στοχεύει σε πολύ ευαίσθητες πληροφορίες όπως κωδικοί ή αριθμούς λογαριασμών, το δε adware συλλέγει πληροφορίες που αφορούν τις προτιμήσεις του χρήστη για να βγάζει σχετικές pop-up (αναδυόμενες) διαφημίσεις.

- **Botnet**: Τα κακόβουλα λογισμικά τύπου botnet δημιουργούν ένα τεράστιο δίκτυο από εκατοντάδες ή χιλιάδες υπολογιστές που μπορούν να ελεγχθούν απομακρυσμένα από τους επιτιθέμενος και εκτελούν συλλογικά μια από τις ακόλουθες διεργασίες:

  - Αποστολή spam mail
  - Εξόρυξη κρυπτονομισμάτων
  - Διεξαγωγή κατανεμημένης επίθεσης "άρνησης εξυπηρέτησης" (DDoS)
  - Διανομή κακόβουλου λογισμικού για την δημιουργία νέου δικτύου botnet

- **Ransomware**: Τα ransomware στοχεύουν στην απόσπαση λύτρων (ransom) από τους χρήστες που έχουν προσβληθεί. Ένα ransomware μόλις προσβάλει επιτυχώς

ένα σύστημα αρχίζει και κρυπτογραφεί σημαντικά αρχεία του θύματος, ενώ αρκετά συχνά φροντίζει να διαγράφει και τα αντίγραφα ασφαλείας, αν φυσικά τα βρει. Η κρυπτογράφηση έχει πραγματοποιηθεί με κλειδί που γνωρίζει μόνο ο επιτιθέμενος. Έπειτα γίνεται η διαπραγμάτευση με το θύμα για την πώληση του κλειδιού. Δυστυχώς, υπάρχουν πάρα πολλές αναφορές σε περιπτώσεις που ενώ έγινε η διαπραγμάτευση και η πληρωμή των λύτρων το κλειδί αποκρυπτογράφησης δεν στάλθηκε ποτέ και το θύμα έχασε μόνιμα την πρόσβαση στα αρχεία του.

- **Cryptojacking**: Μια σχετικά καινούρια κατηγορία κακόβουλου λογισμικού. Τα λογισμικά που χαρακτηρίζονται ως Cryptojacking αποσκοπούν στην αξιοποίηση του υπολογιστή του θύματος για την εξόρυξη κρυπτονομισμάτων. Φυσικά το θύμα δεν γνωρίζει τίποτα ενώ ταυτόχρονα πέφτει η απόδοση του συστήματός του και ανεβαίνει το αντίτιμο που πρέπει να πληρώσει για την αυξημένη κατανάλωση ενέργειας. Επιπλέον, έχουν παρατηρηθεί περιπτώσεις όπου τα Cryptojacking λογισμικά δεν χρησιμοποιούνται για να παράγουν κρυπτονομίσματα, αλλά για να κλέψουν τα ήδη υπάρχοντα κρυπτονομίσματα του χρήστη.

Κλείνοντας αυτή την αναφορά στα είδη του κακόβουλου λογισμικού είναι σημαντικό να τονίσουμε ξανά το γεγονός ότι σχεδόν καμία κατηγορία δεν είναι αυτοαποκλειόμενη με κάποια άλλη. Το γεγονός αυτό, μπορεί να δυσχεραίνει τον σκοπό κάποιο λογισμικών προστασίας καθώς η κατηγοριοποίηση δεν είναι ξεκάθαρη και υπάρχει η πιθανότητα ένα κακόβουλο λογισμικό να κατηγοριοποιηθεί ως backdoor από ένα λογισμικό προστασίας και ως trojan από κάποιο άλλο, καθώς μπορεί να εφαρμόζει και τις δύο τεχνικές για να πετύχει τον σκοπό του.

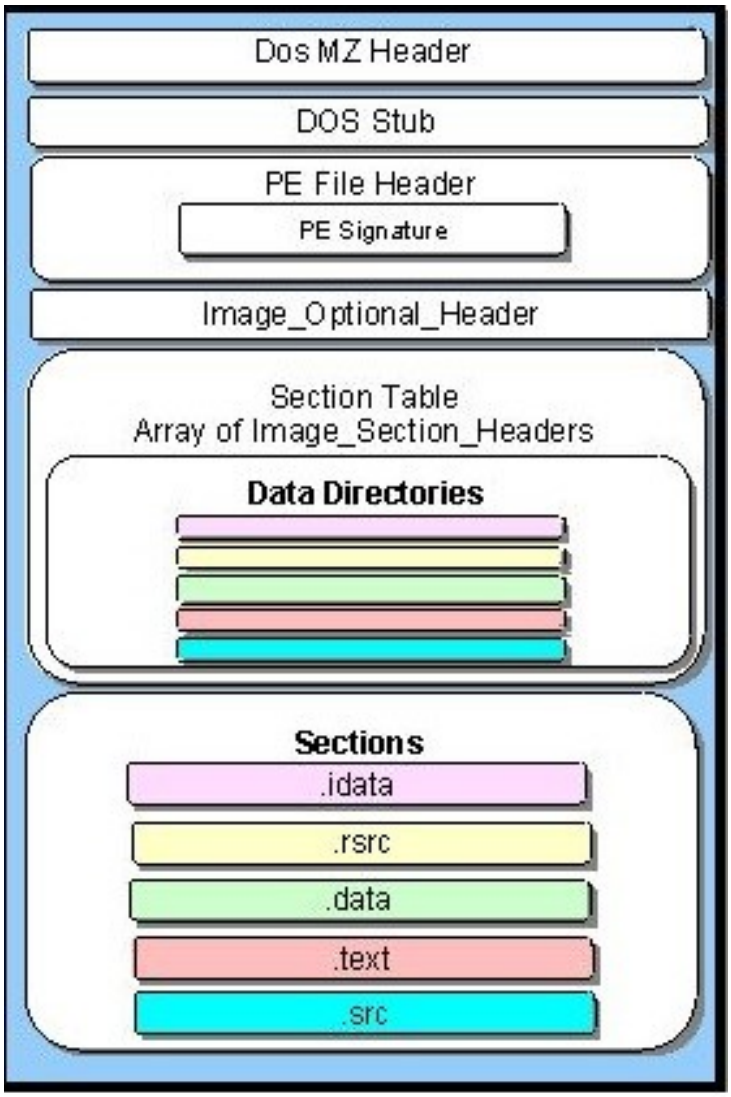
2.2 PE αρχεία

Κατά την περίοδο συγγραφής της παρούσας εργασίας το λογισμικό *Windows* κατέχει το μεγαλύτερο μερίδιο της αγοράς λειτουργικών συστημάτων υπολογιστών. Επομένως, είναι λογικό να εξετάσουμε την πιο κοινή μορφή που εμφανίζονται τα κακόβουλα λογισμικά για το πιο κοινό λειτουργικό σύστημα, τα PE (Portable Executable) [8] αρχεία.

Η μορφή των αρχείων PE χρησιμοποιείται για εκτελέσιμα αρχεία σε 32 και 64 bit Windows πλατφόρμες. Η πιο γνωστή κατάληξη αρχείων που έχουν την μορφή PE είναι η .EXE αλλά δεν είναι η μόνη, για παράδειγμα τα αρχεία με καταλήξεις DLL, SYS και SCR χρησιμοποιούν την δομή PE.

2.2.1 Δομή

Ένα αρχείο PE χωρίζεται σε δύο βασικές και διακριτές περιοχές, οι οποίες με την σειρά τους διαιρούνται σε υπο-περιοχές. Η μια περιοχή ονομάζεται *Header* και η άλλη *Section*. Για την καλύτερη κατανόηση κρίνεται απαραίτητο να παραθέσουμε μια απεικόνιση της δομής PE.



Σχήμα 1: Δομή ενός αρχείου PE
Πηγή: Malware researcher's handbook[9]

2.2.2 DOS Header

Σε κάθε αρχείο PE τα πρώτα 64 byte αφορούν το τμήμα *DOS Header*. Υπάρχει έτσι ώστε το DOS να μπορέσει να αναγνωρίσει το αρχείο σαν ένα εκτελέσιμο και να το εκτελέσει σε περιβάλλον DOS (ξεπερασμένο απλά δεν αφαιρείται γιατί θα έπρεπε να αλλάξει ριζικά η δομή των PE αρχείων).

2.2.3 DOS Stub

Η περιοχή *DOS Stub* μπορεί να είναι ακόμα και ένα πλήρες DOS πρόγραμμα το οποίο συνήθως τυπώνει ένα μήνυμα του τύπου "This program cannot be run in DOS mode." όταν προσπαθεί κάποιος να εκτελέσει το εν λόγω αρχείο σε περιβάλλον DOS.

2.2.4 PE File Header

Όπως όλα τα εκτελέσιμα αρχεία εμπεριέχουν πεδία τα οποία λειτουργούν σαν ευρετήρια για το πως είναι δομημένο στην συνέχεια το εκτελέσιμο, έτσι και τα PE αρχεία έχουν το *File Header* που εμπεριέχει πληροφορίες όπως η θέση και το μέγεθος του κώδικα. Συγκεκριμένα:

- **Signature:** Εμπεριέχει μόνο την υπογραφή έτσι ώστε να αναγνωρίζεται το αρχείο από τον loader των Windows. Συνήθως είναι τα γράμματα PE ακολουθούμενα από δύο μηδενικά (0).
- **NumberOfSections:** Καθορίζει το μέγεθος του *setction table* που έρχεται αμέσως μετά το File Header.
- **SizeOfOptionalHeader:** το μέγεθος του *Optional Header* που για εκτελέσιμα αρχεία είναι υποχρεωτικός. Για αρχεία που λειτουργούν ως βιβλιοθήκες το μέγεθος πρέπει να είναι μηδέν.
- **Characteristics:** Εδώ εμπεριέχονται πεδία που ανάλογα με την τιμή τους καθορίζονται κάποια βασικά χαρακτηριστικά του αρχείου, για παράδειγμα το αν είναι DLL ή όχι.
- **Image_Optional_Header:** Ίσως το πιο σημαντικό πεδίο για ένα εκτελέσιμο αρχείο με μορφή PE. Εμπεριέχει σχεδόν όλες τις βασικές πληροφορίες για το εκτελέσιμο, όπως το αρχικό μέγεθος του stack, το σημείο εκκίνησης του κώδικα του προγράμματος, προτεινόμενη διεύθυνση εκκίνησης, έκδοση λειτουργικού συστήματος κ.ο.κ.

2.2.5 Section Table

Το *Section Table* είναι ένας πίνακας που εμπεριέχει τόσες εγγραφές όσες αναφέρονται στο πεδίο *NumberOfSections* στο *File Header*. Κάθε εγγραφή έχει μέγεθος τουλάχιστον 40 byte. Μερικές από τις πιο σημαντικές εγγραφές είναι οι ακόλουθες:

- Name: Μια συμβολοσειρά αποτελούμενη από 8 byte. Μπορεί να είναι και κενό πεδίο.
- VirtualSize: Το πραγματικό μέγεθος των δεδομένων του τμήματος όταν φορτώνονται στην μνήμη.
- SizeOfRawData: Το μέγεθος των δεδομένων του τμήματος όσο είναι αποθηκευμένα στον δίσκο.

- PointerToRawData: Ένας pointer στα δεδομένα τμήματος. Η τιμή του είναι η διαφορά της διεύθυνσης των δεδομένων τμήματος από την αρχή του αρχείου.
- Characteristics: Τα χαρακτηριστικά του τμήματος.

2.2.6 PE File Section

Σε αυτό το κομμάτι των PE αρχείων εμπεριέχονται όλα τα κύρια συστατικά, όπως ο κώδικας, τα δεδομένα, πόροι και άλλα εκτελέσιμα αρχεία. Κάθε ξεχωριστό τμήμα έχει την δικιά του κεφαλίδα (header) και κύριο μέρος (body). Μια τυπική εφαρμογή για Windows έχει, συνήθως, εννέα (9) προκαθορισμένα τμήματα:

1. **text:** Σε αυτό το τμήμα υπάρχει όλος ο κώδικας του αρχείου.
2. **bss:** Το τμήμα αντιπροσωπεύει τα δεδομένα του προγράμματος που δεν έχουν αρχικοποιηθεί ακόμα.
3. **rdata:** Εδώ βρίσκονται τα δεδομένα που είναι read-only, όπως οι συμβολοσειρές και οι σταθερές.
4. **data:** Το τμήμα αντιπροσωπεύει τα δεδομένα του προγράμματος που έχουν αρχικοποιηθεί.
5. **rsrc:** Όλοι οι πόροι του εκτελέσιμου, π.χ. κάποιες εικόνες που χρησιμοποιεί εσωτερικά το πρόγραμμα.
6. **edata:** Αν το εκτελέσιμο μοιράζει κάποιες από τις συναρτήσεις του με άλλα εκτελέσιμα, αυτό το πεδίο περιέχει τις πληροφορίες για αυτές τις συναρτήσεις.
7. **idata:** Αντιθέτως, αν το εκτελέσιμο παίρνει κάποιες από τις συναρτήσεις του από άλλα εκτελέσιμα οι πληροφορίες για αυτές τις συναρτήσεις τοποθετούνται σε αυτό το πεδίο.
8. **pdata:** Πεδίο που δίνει στον επεξεργαστή πληροφορίες για το πως να χειριστεί κάποια συμβάντα που μπορεί να προκύψουν.
9. **.debug:** Δεδομένα που βοηθούν στην αποσφαλμάτωση του εκάστοτε εκτελέσιμου. Συνήθως, το συγκεκριμένο πεδίο απουσιάζει από εκτελέσιμα που είναι έτοιμα να διατεθούν στο ευρύ κοινό.

Ανάλογα με τον σκοπό της εκάστοτε εφαρμογής κάποια από τα προηγούμενα τμήματα χρησιμοποιούνται, αλλά δεν είναι απαραίτητα να χρησιμοποιηθούν όλα.

2.3 Ανίχνευση κακόβουλου λογισμικού

Σε αυτή την ενότητα θα αναπτύξουμε κάποιες τεχνικές ανίχνευσης κακόβουλου λογισμικού. Βασιζόμενοι κυρίως στο βιβλίο του Peter Szor [10] θα παραθέσουμε τις βασικές μεθόδους εντοπισμού κακόβουλου λογισμικού, αρχίζοντας από τις απλές και καταλήγοντας στις πιο σύνθετες. Προτού προχωρήσουμε στην ονομαστική αναφορά των τεχνικών πρέπει να τονίσουμε ότι δεν είναι αποτελεσματικές όλες οι τεχνικές για όλα τα κακόβουλα λογισμικά. Όμως, καμία τεχνική δεν πρέπει να θεωρείται ως αναποτελεσματική αν δεν εφαρμόζεται πάντα. Η καλύτερη στρατηγική αντιμετώπισης είναι να υπάρχουν όλες οι τεχνικές διαθέσιμες και μια εκ των διαθέσιμων να μπορέσει να αντιμετωπίσει πλήρως το κακόβουλο λογισμικό.

2.3.1 Πρώτης Γενιάς Ανιχνευτές

Οι ανιχνευτές πρώτης γενιάς λειτουργούν πολύ απλά και στηρίζονται στον εντοπισμό συγκεκριμένων και προκαθορισμένων ακολουθιών bytes εντός του κακόβουλου αρχείου. Οι τεχνικές που χρησιμοποιούν, αναφέρονται ως τεχνικές που βασίζονται στις υπογραφές των αρχείων (signature based) καθώς απαιτείται προηγούμενη γνώση των προκαθορισμένων bytes που θα προσπαθήσουν να ταυτοποιήσουν στα κακόβουλα λογισμικά. Η αποτελεσματικότητα τους, εξαρτάται άμεσα από την έγκαιρη ενημέρωση της βάσης που εμπεριέχει τις υποψήφιες ακολουθίες bytes προς ταυτοποίηση στα πιθανώς κακόβουλα λογισμικά.

- **Ανίχνευση αλφαριθμητικών χαρακτήρων - String Scanning**: Η πιο απλή μέθοδος ανίχνευσης στηρίζεται στην απλή ανίχνευση και ταυτοποίηση bytes που ενώ είναι κοινά σε κακόβουλα λογισμικά, συνήθως απουσιάζουν στα κανονικά λογισμικά. Προφανώς, αυτή η τεχνική είναι αρκετά εύκολο να παρακαμφθεί με μικρές αλλαγές στον πηγαίο κώδικα του κακόβουλου προγραμματισμού. Για τον λόγο αυτό η συγκεκριμένη τεχνική πολύ σπάνια χρησιμοποιείται μόνη της για την απόρριψη ενός λογισμικού ως κακόβουλο.

- **Μπαλαντέρ - Wildcards**: Χτίζοντας πάνω στην τεχνική string scanning, προσπαθούμε να την κάνουμε πιο ανθεκτική σε τεχνικές παράκαμψης με την εισαγωγή μπαλαντέρ. Τα μπαλαντέρ επιτρέπουν στο πρόγραμμα ανίχνευσης να πραγματοποιεί πιο "ελαστικές" ταυτοποιήσεις σε γνωστά bytes. Δηλαδή δεν χρειάζεται να είναι παρόν όλη η ακολουθία, αρκεί ένα μέρος της, ή κάποια σημαντικά κομμάτια της.

- **Αναντιστοιχίες - Mismatches**: Επεκτείνοντας και άλλο την τεχνική string scanning, επιτρέπουμε N αναντιστοιχίες σε κάθε αναζήτηση. Για παράδειγμα έστω ότι αναζητούμε το μοτίβο "01 02 03 04" με 2 αναντιστοιχίες. Το μοτίβο "01 BE EF 04" αλλά και το "CA FE 03 04" είναι αποδεκτά στην αναζήτηση που εκτελούμε.

- **Γενική Ανίχνευση - General Detection**: Συνήθως, καμία τεχνική από τις προηγούμενες δεν εφαρμόζεται μόνη της. Αντιθέτως, στην περίπτωση του general detection δημιουργούνται γενικά μοτίβα που περιέχουν και μπαλαντέρ και αναντιστοιχίες με σκοπό την ανίχνευση κακόβουλων λογισμικών που έχουν παραχθεί με τέτοιο τρόπο ώστε να παρακάμπτουν τις προηγούμενες μεμονωμένες τεχνικές.

Οι τεχνικές που αναφέραμε είναι συνήθως αρκετά αποτελεσματικές, με την προϋπόθεση πάντα ότι η βάση με τις ακολουθίες των bytes από τα κακόβουλα λογισμικά είναι ενημερωμένη. Όμως, δεδομένου ότι χρειάζεται να ψαχτεί ολόκληρο το εκτελέσιμο αρχείο για να

εντοπιστούν τα bytes, αν εντοπιστούν, είναι αρκετά αργές. Για τον λόγο αυτό υπάρχουν οι ακόλουθες "βοηθητικές" τεχνικές για να επισπεύσουν την διαδικασία αναζήτησης:

- **Κατακερματισμός - Hashing**: Ο αλγόριθμος κατακερματίζει συγκεκριμένα σημεία του αρχείου. Έπειτα, χρησιμοποιεί γνωστά αποτυπώματα (hash signatures) κακόβουλων λογισμικών για να κάνει τις απαραίτητες συγκρίσεις. Το πλεονέκτημα της μεθόδου προέρχεται από το γεγονός ότι οι συναρτήσεις κατακερματισμού είναι συνήθως αρκετά γρήγορες και πάντα παράγουν μια έξοδο σταθερού μήκους, δηλαδή το αρχείο έχει ταυτιστεί με μια σταθερού μήκους ακολουθία, την υπογραφή του.

- **Σελιδοδείκτες - Bookmarks**: Οι ακολουθίες που αναζητούνται σε ένα εκτελέσιμο ή σε ένα μολυσμένο αρχείο, στην συντριπτική πλειοψηφία των περιπτώσεων, βρίσκονται σε μια συγκεκριμένη απόσταση από την αρχή του αρχείου (offset). Οι σελιδοδείκτες, λοιπόν, είναι ακριβώς αυτά τα "offests". Με τους σελιδοδείκτες η αναζήτηση εκτελείται πιο γρήγορα αλλά και τα αποτελέσματα είναι πιο αξιόπιστα, καθώς αν μια "κακόβουλη" ακολουθία βρίσκεται και στην θέση που αναμένεται να βρίσκεται, τότε σίγουρα έχει εντοπιστεί ένα κακόβουλο αρχείο.

- **Σάρωση Αρχής και Τέλους - "Top-and-Tail" Scanning**: Σχεδόν όμοια με την τεχνική του κατακερματισμού, δεν γίνεται σάρωση σε όλο το αρχείο αλλά μόνο στην αρχή και το τέλος του. Η τεχνική είναι ιδιαίτερα αποτελεσματική σε μολυσμένα αρχεία που έχουν παραχθεί από κακόβουλα λογισμικά που τοποθετούν το επικίνδυνο φορτίο τους στην αρχή ή στο τέλος των αρχείων.

- **Σάρωση σημείου εισαγωγής και σάρωση σταθερού σημείου - "Entry-Point and Fixed-Point" Scanning**: Η συγκεκριμένη τεχνική εκμεταλλεύεται την δομή των εκτελέσιμων αρχείων. Αντί να σαρώνεται ολόκληρο το αρχείο, η σάρωση ξεκινάει από το σημείο όπου το αρχείο έχει τον κώδικα που τρέχει πρώτος και ακολουθεί όλες τις εντολές αλλαγής ροής του κώδικα (jump, return, call κ.α). Στην περίπτωση που η αρχή του κώδικα δεν έχει αρκετές ακολουθίες για να μπορέσει ένα αρχείο να χαρακτηριστεί κακόβουλο ή μη από τους σαρωτές ακολουθείται η μέθοδος του σταθερού σημείου. Η σάρωση του αρχείου γίνεται με αναφορά σε ένα σταθερό σημείο. Για παράδειγμα το σταθερό σημείο μπορεί να είναι η "main" έστω στην διεύθυνση **X** (το σταθερό σημείο) και η σάρωση να ξεκινάει στις διευθύνσεις $\mathbf{X} + \mathbf{M}$ bytes.

2.3.2 Δεύτερης Γενιάς Ανιχνευτές

Στην δεύτερη γενιά ανιχνευτών θα μελετήσουμε τεχνικές οι οποίες θα κάνουν την ανίχνευση κακόβουλων λογισμικών πιο αποτελεσματική.

- **Έξυπνη Σάρωση - Smart Scanning**: Η απλή αναζήτηση ακολουθιών Bytes είναι αρκετά εύκολο να ξεγελαστεί και ν' αποτύχει να εντοπίσει ένα κακόβουλο αρχείο. Για παράδειγμα, μπορούν να εισαχθούν σε κρίσιμα σημεία του κώδικα εντολές "NOP" που ενώ δεν αλλάζουν την ροή του προγράμματος αλλάζουν την τελική ακολουθία από Bytes. Η έξυπνη σάρωση, δεν ψάχνει απλά να ταιριάξει μια ακολουθία από Bytes αλλά λαμβάνει υπόψη της και σε αντιστοιχούν τα Bytes. Στο παράδειγμα που αναφέραμε τα Bytes της εντολής "NOP" θα αγνοηθούν. Η τεχνική είναι ιδιαίτερα αποτελεσματική στον εντοπισμό παραλλαγών κάποιων κακόβουλων αρχείων.

- **Ανίχνευση Πυρήνα - Skeleton Detection**: Η συγκεκριμένη τεχνική εφευρέθηκε από τον *Eugene Kaspersky*[11] πρόεδρο της γνωστής εταιρείας λογισμικού αντιμετώπισης ηλεκτρονικών απειλών *Kaspersky* και αφορά ιούς μακροεντολών[12]. Αντί να γίνει ανάλυση σε κάποια μεμονωμένα στοιχεία του αρχείου μακροεντολής (macro) γίνεται ανάλυση γραμμή προς γραμμή με σκοπό να αγνοηθούν οι μη κρίσιμες εντολές. Το αποτέλεσμα είναι ο πυρήνας (skeleton) του αρχικού Macro, ο οποίος αποτελείται από εντολές που είναι πιθανόν να συναντηθούν σε κάποιο κακόβουλο λογισμικό. Τέλος, το λογισμικό ανίχνευσης χρησιμοποιεί τις πληροφορίες, από τον πυρήνα που δημιούργησε, για να αποφανθεί αν πρόκειται για κακόβουλο λογισμικό.

- **Σχεδόν Ακριβής Ταυτοποίηση - Nearly Exact Identification**: Με αυτή την τεχνική είναι δυνατό να γίνει πιο ακριβής η ανίχνευση κακόβουλων λογισμικών. Για παράδειγμα, στους ανιχνευτές πρώτης γενιάς χρησιμοποιούνταν μια ακολουθία από Bytes για να γίνει η ταυτοποίηση του λογισμικού. Πλέον, θα χρησιμοποιούνται δύο ακολουθίες. Αν υπάρξει ταυτοποίηση και με τις δύο ακολουθίες τότε είναι πολύ πιθανό, σχεδόν σίγουρο, να έχει βρεθεί ακριβώς το κακόβουλο λογισμικό που αναμενόταν. Αν γίνει ταυτοποίηση μόνο της μίας ακολουθίας τότε πιθανόν έχει ανιχνευτεί κάποια παραλλαγή του αναμενόμενου λογισμικού. Εκτός από την δεύτερη ακολουθία Bytes μπορεί να χρησιμοποιηθεί και κάποιο checksum του λογισμικού. Η τεχνική είναι ιδιαίτερα αποτελεσματική στην αναγνώριση κακόβουλων λογισμικών που βασίζονται σε γνωστά κακόβουλα λογισμικά.

- **Ακριβής Ταυτοποίηση - Exact Identification**: Η μοναδική μέθοδος που εγγυάται πλήρη ταυτοποίηση κάποιας παραλλαγής ενός κακόβουλου λογισμικού. Σε αντίθεση με την προηγούμενη τεχνική, εδώ χρησιμοποιούνται όσα checksums κρίνονται απαραίτητα να καλύψουν το κομμάτι που εμπεριέχει τον αμετάβλητο κώδικα του κακόβουλου λογισμικού. Συνήθως, αυτή η τεχνική είναι λίγο πιο αργή σε σχέση με τις προηγούμενες, καθώς αποσκοπεί στην πλήρη αναγνώριση της απειλής.

Ακόμα και η *δεύτερη γενιά* ανιχνευτών βασίζεται στην αναζήτηση και ταυτοποίηση υπογραφών από τα αρχεία. Το γεγονός αυτό καθιστά την αποτελεσματικότητα των συγκεκριμένων τεχνικών άμεσα συνδεδεμένη με το πόσο ενημερωμένη είναι η βάση που εμπεριέχει τις υποψήφιες υπογραφές προς αναζήτηση.

Αλγοριθμικές Μέθοδοι Ανίχνευσης

Αρκετές φορές, οι βασικές μέθοδοι ανίχνευσης που αναφέραμε δεν μπορούν να αντιμετωπίσουν αποτελεσματικά μια ενδεχόμενη απειλή, από ένα κακόβουλο λογισμικό. Για ν'

αποφευχθούν τέτοια φαινόμενα χρειάζεται να εισαχθεί καινούριος κώδικας (διαφορετικός για κάθε μη ανιχνεύσιμη απειλή) στα λογισμικά ανίχνευσης. Προφανώς, μια τέτοια λύση δημιουργεί αρκετά προβλήματα στην καλή και γρήγορη λειτουργία των ανιχνευτών, καθώς συχνά ο κώδικας που εισάγεται καθίσταται σύντομα ξεπερασμένος.

Προσομοίωση Κώδικα

Η συγκεκριμένη τεχνική χρησιμοποιεί μεθόδους δυναμικής ανάλυσης. Η δυναμική ανάλυση, εξετάζει τον κώδικα κατά την διάρκεια της εκτέλεσής του. Προφανώς, για να μην τεθεί σε κίνδυνο η ασφάλεια του συστήματος που αναλαμβάνει την εκτέλεση του κώδικα χρησιμοποιούνται εικονικές μηχανές[13] (virtual machines) με αποτέλεσμα η δράση του λογισμικού να είναι πλήρως περιορισμένη και αποκομμένη από ευαίσθητα δεδομένα. Το αρνητικό της συγκεκριμένης μεθόδου είναι σχετικά προφανές και αφορά τους πόρους που δεσμεύονται από το σύστημα για την προσομοίωση, καθώς κάποιες φορές δεν είναι λίγοι οι απαιτούμενοι πόροι.

Ευρετική Ανάλυση

Ο συνδυασμός στοιχείων από στατική και δυναμική ανάλυση των αρχείων μας οδηγεί στην τεχνική της ευρετικής αναζήτησης. Μερικά παραδείγματα από χαρακτηριστικά που αναζητούνται (για την ύπαρξη του ή μη) είναι η εκτέλεση κώδικα σε περίεργες διευθύνσεις, ύποπτες ανακατευθύνσεις κώδικα, σφάλματα στην επικεφαλίδα του αρχείου και γενικά οτιδήποτε μπορεί να υποδηλώσει ότι το εκτελέσιμο αρχείο έχει προγραμματιστεί με τέτοιον τρόπο ώστε να κρύβεται η πραγματική λειτουργία του. Το μεγάλο πλεονέκτημα είναι η αναγνώριση νέων κακόβουλων λογισμικών καθώς δεν στηρίζεται σε προϋπάρχουσα γνώση, αλλά δεν είναι λίγες οι φορές που απλά λογισμικά χαρακτηρίζονται ως κακόβουλα από την συγκεκριμένη τεχνική ανεβάζοντας αρκετά το πλήθος των ψευδοθετικών αποτελεσμάτων.

Sandbox

Το sandbox αποτελεί την εξέλιξη της τεχνικής *Προσομοίωσης Κώδικα*, εισάγοντας πιο περίπλοκες μεθόδους προσομοίωσης. Πλέον, ολόκληρο το σύστημα προσομοιώνεται και τα κακόβουλα λογισμικά μπορούν να προσπελάσουν αντίγραφα πραγματικών πόρων του συστήματος. Δεδομένου ότι οι πόροι είναι απλά αντίγραφα του πραγματικού συστήματος η πιθανή ζημιά που μπορεί να προκαλέσει το κακόβουλο λογισμικό είναι αρκετά περιορισμένη. Το τεράστιο πλεονέκτημα αυτής της μεθόδου είναι ότι το λογισμικό αναλύεται πλήρως δυναμικά και μπορεί να κριθεί για το αν είναι κακόβουλο ή όχι απευθείας από τις αλλαγές που έχει επιφέρει στο προσημειωμένο σύστημα. Προφανώς και αυτή η μέθοδος έχει και αρνητικά στοιχεία. Αρχικά, οι απαιτήσεις σε πόρους συστήματος αυξάνονται δραματικά σε σχέση με άλλες μεθόδους. Έπειτα, τα κακόβουλα λογισμικά έχουν εξελιχτεί και κάποια είναι σε θέση να αναγνωρίζουν ότι εκτελούνται μέσα σε προσημειωμένο περιβάλλον και έτσι να αποκρύπτουν την πραγματική τους λειτουργία. Τέλος, το χειρότερο ίσως μειονέκτημα αυτής της μεθόδου, είναι ότι και οι τεχνικές προσομοίωσης έχουν ευπάθειες. Δηλαδή, αν το κακόβουλο λογισμικό έχει προγραμματιστεί ώστε όχι μόνο να αναγνωρίζει το προσημειωμένο περιβάλλον, αλλά να εκμεταλλεύεται και ευπάθειες που αφορούν την προσομοίωση τότε η δράση του μπορεί να επεκταθεί και στο πραγματικό σύστημα με πολύ ανεπιθύμητα αποτελέσματα.

Έπειτα από την σύντομη αναφορά που κάναμε στις τεχνικές εντοπισμού κακόβουλου λογισμικού μπορούμε να εξάγουμε το συμπέρασμα ότι δεν είναι μια τυποποιημένη διαδικασία και ότι σε διαφορετικές περιπτώσεις δουλεύουν διαφορετικές τεχνικές. Τα λογισμικά ανίχνευσης-αντιμετώπισης πρέπει να αναλάβουν το δύσκολο έργο να εφαρμόζουν τις κατάλληλες τεχνικές για να είναι το σύστημα προστατευμένο, χωρίς όμως να καταναλώνουν όλους τους διαθέσιμους πόρους μόνο για τον συγκεκριμένο σκοπό.

2.4 Μηχανική Μάθηση

Δεδομένου ότι στο υπόλοιπο της παρούσας εργασίας θα αναφερόμαστε διαρκώς σε τεχνικές και εφαρμογές που αφορούν τον τομέα της μηχανικής μάθησης κρίνεται απαραίτητο να γίνει μια σύντομη παρουσίαση των εν λόγω τεχνικών.

Αρχικά, πρέπει να διασαφηνιστεί ο όρος "Μηχανική Μάθηση". Αρκετές φορές, δημιουργείται η σύγχυση ότι ο όρος "Μηχανική Μάθηση" είναι ταυτόσημος με τον όρο "Τεχνητή Νοημοσύνη". Η πραγματικότητα είναι ότι η μηχανική μάθηση είναι κλάδος της τεχνητής νοημοσύνης. Η μηχανική μάθηση περιλαμβάνει αλγόριθμους οι οποίοι εκπαιδεύονται και πραγματοποιούν προβλέψεις σε ένα σύνολο δεδομένων. Ο Tom Mitchell [14] έδωσε τον ακόλουθο πιο επίσημο ορισμό: "Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία E ως προς μια κλάση εργασιών T και ένα μέτρο επίδοσης P, αν η επίδοσή του σε εργασίες της κλάσης T, όπως αποτιμάται από το μέτρο P, βελτιώνεται με την εμπειρία E". Από τον ορισμό προκύπτει ότι οι αλγόριθμοι της μηχανικής μάθησης δεν είναι σχεδιασμένοι να λύνουν κάποιο συγκεκριμένο πρόβλημα. Αντιθέτως, αν το πρόβλημα είναι καλώς ορισμένο, μέσω των T, P και E, ο αλγόριθμος μπορεί να μάθει στα συγκεκριμένα δεδομένα, χωρίς απαραίτητα να εγγυάται και την επίλυση του προβλήματος.

Μετά τον καθορισμό του προβλήματος πρέπει να γίνει η επιλογή των αλγορίθμων που θα εφαρμοστούν για την επίλυση του. Το τελικό σύνολο που θα δώσει τα αποτελέσματα, ονομάζεται μοντέλο και διαμορφώνεται πάντα σύμφωνα με τις ανάγκες του κάθε προβλήματος. Οι αλγόριθμοι της μηχανικής μάθησης χωρίζονται στις εξής βασικές κατηγορίες[15]:

- **Supervised Learning:** Στην περίπτωση του supervised learning (επιβλεπόμενη μάθηση) ο αλγόριθμος πρέπει να μάθει, από ένα σετ εισόδων-εξόδων, ν' αντιστοιχεί μια είσοδο στην σωστή έξοδο. Ο όρος "επιβλεπόμενη" προέρχεται από το γεγονός ότι ο αλγόριθμος έχει την πληροφορία για το ποια είναι η σωστή έξοδος σε κάθε είσοδο κατά την διάρκεια της εκπαίδευσης.

- **Unsupervised Learning:** Σε αντίθεση με την επιβλεπόμενη μάθηση, δεν υπάρχει κάποια πληροφορία για την έξοδο. Αντ' αυτού ο αλγόριθμος έχει στην διάθεσή του μόνο τα δεδομένα εισόδου και ψάχνει να βρει επαναλαμβανόμενα μοτίβα στον χώρο που υπάρχουν τα δεδομένα εισόδου. Η πιο συνηθισμένη εφαρμογή μη-επιβλεπόμενης μάθησης είναι η συσταδοποίηση όπου τα δεδομένα εισόδου ομαδοποιούνται ανάλογα με κάποια χαρακτηριστικά.

- **Reinforcment Learning:** Σε κάποια προβλήματα η έξοδος είναι μια ακολουθία από ενέργειες. Μια μεμονωμένη ενέργεια δεν είναι σημαντική, μας ενδιαφέρει μόνο η συνολική ακολουθία από ενέργειες, η οποία ακολουθία προκύπτει από μια *πολιτική*. Μια ενδιάμεση ενέργεια δεν μπορεί να χαρακτηριστεί καλή, είναι όσο "καλή" όσο η πολιτική που την περιέχει. Σε τέτοιες περιπτώσεις το μοντέλο μηχανικής μάθησης θα πρέπει να είναι σε θέση να εκτιμάει την καταλληλότητα διαφόρων πολιτικών και να επιλέγει την βέλτιστη. Συνήθως η εκτίμηση του μοντέλου βασίζεται στην μέθοδο ανταμοιβής - τιμωρίας. Δηλαδή, εκτελούνται οι ενδιάμεσες ενέργειες και ανάλογα με το αποτέλεσμα υπάρχει η αντίστοιχη ανταμοιβή ή τιμωρία. Το σύνολο (ανταμοιβών/τιμωριών) σε κάθε ενδιάμεση κατάσταση μπορεί να οδηγήσει το μοντέλο στην εξαγωγή βέλτιστης πολιτικής.

Στην παρούσα εργασία θα ασχοληθούμε κυρίως με την επιβλεπόμενη μάθηση καθώς ο τελικός στόχος είναι να αναγνωρίζουμε κακόβουλα λογισμικά αποτελεσματικά με μεθόδους μηχανικής μάθησης. Συνεπώς, έχουμε προκαθορισμένες εξόδους, το εκάστοτε δείγμα

εισόδου ή είναι ή δεν είναι κακόβουλο λογισμικό. Κάποια από τα πιο γνωστά μοντέλα επιβλεπόμενης μάθησης που θα ασχοληθούμε είναι τα ακόλουθα:

- **Decision Trees:** Ένα decision tree (δέντρο απόφασης) είναι μια ιεραρχημένη δομή δεδομένων στην οποία εφαρμόζεται η στρατηγική διαίρει και βασίλευε. Αποτελούν ένα μη-παραμετρικό μοντέλο, με αποτέλεσμα να έχουν υψηλή απόδοση και σχετικά μικρό χρόνο εκπαίδευσης. Το μεγάλο πλεονέκτημα του συγκεκριμένου μοντέλου είναι ότι μπορεί να μετασχηματιστεί σε ένα σύνολο απλών κανόνων που είναι εύκολοι στην κατανόηση.

- **Random Forest:** Η μέθοδος Random Forest μπορεί να θεωρηθεί μια επέκταση των Decision Trees. Κατά την διάρκεια της εκπαίδευσης δημιουργούνται πολλά Decision Trees και το τελικό αποτέλεσμα, σε περίπτωση προβλήματος κατηγοριοποίησης, είναι η κλάση που έχει επιλεγεί από τα περισσότερα decision trees. Η συγκεκριμένη μέθοδος διορθώνει το βασικό πρόβλημα των Decision Trees να υπερεκπαιδεύονται στα δεδομένα εκπαίδευσης με αποτέλεσμα να γενικεύουν εξαιρετικά δύσκολα.

- **Bayesian Learning:** Οι μέθοδοι που ανήκουν στην συγκεκριμένη κατηγορία εφαρμόζονται σε περιπτώσεις όπου τα δεδομένα φαίνεται να ακολουθούν μια πιθανοκρατική κατανομή. Είναι ιδιαίτερα αποτελεσματική τεχνική σε εφαρμογές ταξινόμησης κειμένων. Το μειονέκτημα της μεθόδου είναι το γεγονός ότι απαιτεί να υπάρχει προϋπάρχουσα γνώση για την κατανομή και τα δεδομένα, ενώ αρκετές φορές είναι υπολογιστικά αδύνατο να βρεθεί επακριβώς η καλύτερη υπόθεση που ταιριάζει στην εκάστοτε εφαρμογή.

- **SVMs:** Τα Support-Vector machines, είναι γραμμικά μοντέλα που χρησιμοποιούνται για την επίλυση προβλημάτων κατηγοριοποίησης και παλινδρόμησης. Η βασική ιδέα λειτουργίας των SVM είναι η κατασκευή μιας διαχωριστικής (υπερ)επιφάνειας για τον διαχωρισμό των δεδομένων σε κλάσεις. Χρησιμοποιώντας μαθηματικούς μετασχηματισμούς τα SVM είναι ικανά να εκτελέσουν και μη-γραμμικές κατηγοριοποιήσεις με το να προβάλλουν τα δεδομένα εισόδου σε χώρους υψηλότερων διαστάσεων.

- **Artificial Neural Networks:** Η μελέτη του ανθρώπινου εγκεφάλου οδήγησε κατά μια έννοια στην συγκεκριμένη τεχνική. Έχοντας, πλέον αρκετή γνώση για την μορφολογία και την δομή του εγκεφάλου, η ιδέα είναι να δημιουργηθούν μοντέλα που προσομοιώνουν την μορφολογία και την λειτουργία του εγκεφάλου. Τα μοντέλα με βάση τα νευρωνικά δίκτυα είναι αποτελεσματικά σε μια πληθώρα προβλημάτων, συμπεριλαμβανομένου και του προβλήματος που θα αντιμετωπίσουμε στην συνέχεια της παρούσας εργασίας. Το μειονέκτημα τους είναι ότι χρειάζεται εξαιρετικά πολύς χρόνος για την εκπαίδευσή τους και το γεγονός ότι μπορούν αρκετά εύκολα να υπερπροσαρμοστούν στα δεδομένα με αποτέλεσμα να μην μπορούν να δώσουν αξιόπιστα αποτελέσματα σε νέα δεδομένα.

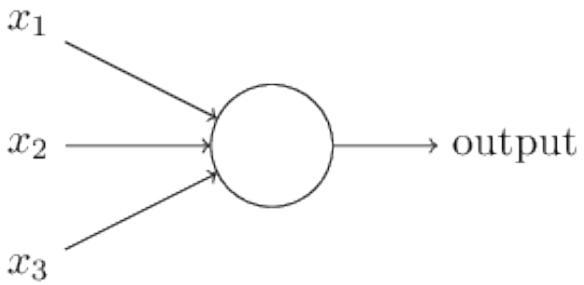
Το πεδίο της μηχανικής μάθησης είναι εξαιρετικά μεγάλο και ιδίως τα τελευταία χρόνια έχει γνωρίσει τεράστια ανάπτυξη. Παρ'ολ'αυτά, ο στόχος της μηχανικής μάθησης είναι, σχεδόν πάντα, η γενίκευση μιας διαδικασίας από υπάρχοντα δεδομένα. Στην σημερινή εποχή κάτι τέτοιο φαντάζει αρκετά εύκολο, καθώς ο όγκος των δεδομένων έχει αυξηθεί ραγδαία τον τελευταίο καιρό. Όμως δεν αρκεί ο τεράστιος όγκος, πρέπει να ελέγχεται η ποιότητα, η πιστότητα και το περιεχόμενο πληροφορίας που απαιτείται στην εκάστοτε εφαρμογή.

2.4.1 Τεχνητά Νευρωνικά Δίκτυα: Σε βάθος

Το μεγαλύτερο κομμάτι της εργασίας στην συνέχεια θα αναφέρεται σε τεχνητά νευρωνικά δίκτυα. Αξίζει, λοιπόν, να αφιερώσουμε μια ξεχωριστή υποενότητα για την περαιτέρω μελέτη και κατανόηση της λειτουργίας των τεχνητών νευρωνικών δικτύων.

Η εργασία των McCulloch και Pitts έδωσε το έναυσμα για την δημιουργία των νευρώνων τύπου **Perceptron** ή **Αντίληπτρο**. Στις σύγχρονες εφαρμογές δικτύων δεν χρησιμοποιούνται, γενικά, τέτοιο είδους νευρώνες αλλά η μελέτη τους θα μας βοηθήσει για να καταλάβουμε τον τρόπο λειτουργίας των νέων μοντέλων νευρών που θα συναντήσουμε.

Οι νευρώνες τύπου **Perceptron** παίρνουν κάποιες δυαδικές εισόδους και επιστρέφουν μία δυαδική έξοδο (0 ή 1).



Σχήμα 2: Παράδειγμα νευρώνα τύπου Perceptron με τρεις εισόδους

Ο επιστήμονας Frank Rosenblatt ήταν ο πρώτος που πρότεινε τον τρόπο υπολογισμού της εξόδου του νευρώνα από τις εκάστοτε εισόδους του. Εισήγαγε τα λεγόμενα βάρη $w_1, w_2, ... \in \mathcal{R}$ το καθένα από τα οποία εκφράζει την σημαντικότητα της συγκεκριμένης εισόδου για την έξοδο (το w_1 αφορά τη είσοδο x_1 κλπ). Η τελική έξοδος, 0 ή 1, εξαρτάται αν το άθροισμα $\sum_i w_i x_i$ είναι μεγαλύτερο από μια τιμή κατωφλίου ή όχι. Η τιμή κατωφλίου είναι μια παράμετρος του κάθε νευρώνα, δεν εξαρτάται από τις εισόδους και ονομάζεται **bias**. Συνοψίζοντας η έξοδος ενός νευρώνα είναι η εξής:

$$\text{Έξοδος} \quad = \quad \begin{cases} 0 & \text{αν } \sum_i w_i x_i \leq \text{ bias} \\ 1 & \text{αν } \sum_i w_i x_i > \text{ bias} \end{cases} \tag{1}$$

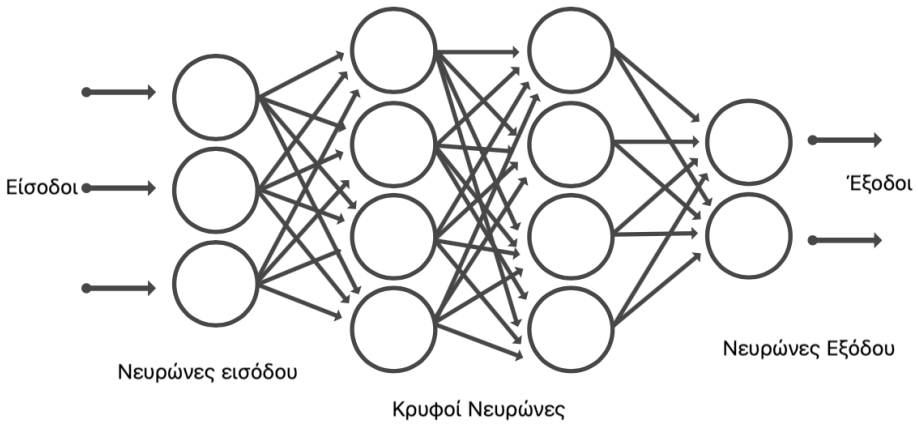
Αυτό είναι το βασικό μαθηματικό μοντέλο για την λειτουργία ενός αντιλήπτρου. Προφανώς οι ικανότητες λήψης απόφασης ενός και μόνο αντιλήπτρου είναι εξαιρετικά περιορισμένες, για τον λόγο αυτό χρησιμοποιούμε πολλά αντίληπτρα συνδεδεμένα μεταξύ τους για να σχηματίσουν ένα δίκτυο.

Το δίκτυο θα αποτελείται από:

- Τους νευρώνες εισόδου, οι είσοδοι των οποίων θα δέχονται τα δεδομένα. Η συστοιχία αυτή ονομάζεται **input layer** και ο αριθμός των νευρών εξαρτάται από τον αριθμό των δεδομένων.

- Τους ενδιάμεσους κρυφούς νευρώνες, οι είσοδοι των οποίων θα είναι η έξοδος της προηγούμενης συστοιχίας, οι συστοιχίες αυτές ονομάζονται **hidden layers** και ο αριθμός τόσο των layers όσο και των νευρώνων ποικίλλει από εφαρμογή σε εφαρμογή.

- Το τελευταίο layer κρυφών νευρώνων θα παρέχει την είσοδο για τους νευρώνες εξόδου (**output layer)** των οποίων τα αποτελέσματα είναι αυτά που θα αξιοποιήσουμε.



Σχήμα 3: Παράδειγμα νευρωνικού δικτύου με 3 inputs, 2 hidden layers με 4 νευρώνες ανα hidden layer, και 2 outputs.

Το σημαντικό πλέον είναι να εκπαιδευτεί το δίκτυο και να μπορέσουμε να το αξιοποιήσουμε στην εφαρμογή που το προορίζουμε. Εκπαίδευση δικτύου σημαίνει κατάλληλη ρύθμιση των 2 παραμέτρων στο δίκτυο, το βάρος κάθε σύνδεσης και το bias του κάθε νευρώνα. Η εκπαίδευση επιτυγχάνεται μέσω διαφόρων εκπαιδευτικών αλγόριθμων που θα μελετήσουμε στην συνέχεια. Προς το παρόν θα εξάγουμε τον πρώτο αλγόριθμο εκπαίδευσης με διάφορες απαιτήσεις και λογικά βήματα.

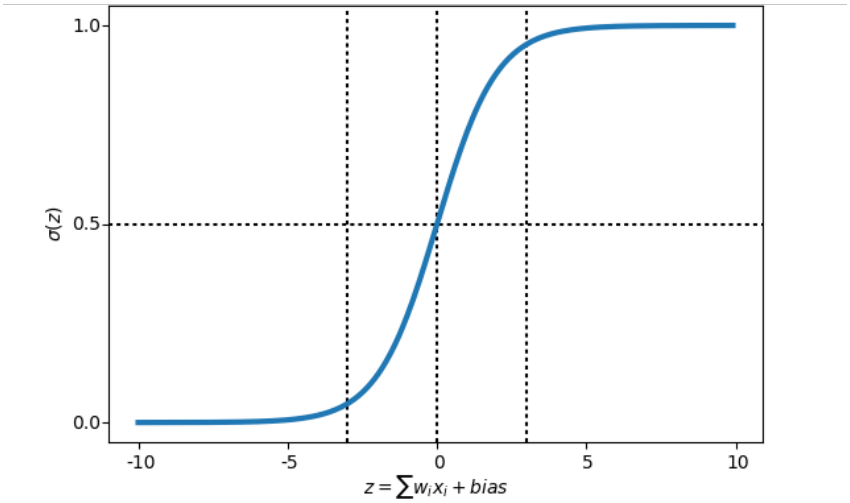
Η πρώτη απαίτηση είναι η οσοδήποτε μικρή αλλαγή σε μία παράμετρο (είτε βάρος είτε bias) να έχει μια οσοδήποτε μικρή επίπτωση στην έξοδο. Δυστυχώς οι νευρώνες τύπου Perceptron δεν ικανοποιούν αυτή την ανάγκη, καθώς σε αρκετές περιπτώσεις μια μικρή αλλαγή μιας παραμέτρου μπορεί να αλλάξει το τελικό αποτέλεσμα εξ' ολοκλήρου, για παράδειγμα από 0 να γίνει 1. Το πρόβλημα είναι ότι, δεν υπάρχουν ενδιάμεσες καταστάσεις ούτε στην είσοδο ούτε στην έξοδο του νευρώνα για τον λόγο αυτό εισάγουμε και χρησιμοποιούμε τους σιγμοειδείς νευρώνες.

Οι σιγμοειδείς νευρώνες έχουν $x_1, x_2, x_3, ..$ εισόδους με $x_i \in [0, 1]$. Τα βάρη και το bias δεν αλλάζουν σε σχέση με τους Perceptrons. Η έξοδος του νευρώνα, όμως, δεν είναι πλέον δυαδική αλλά οποιαδήποτε τιμή στο εύρος [0,1] σύμφωνα με την συνάρτηση:

$$\sigma(z)^{[1]} = \frac{1}{1+e^{-z}} \quad \mu\epsilon \quad z = \sum_i w_i x_i + bias \tag{2}$$

Οι ομοιότητες με τους Perceptrons εμφανίζονται στις ακραίες περιπτώσεις του ορίσματος z. Για $z \to \infty$ η $\sigma(z) \approx 1$ ενώ για $z \to -\infty$ η $\sigma(z) \approx 0$, συνεπώς μόνο για τις ενδιάμεσες τιμές η σιγμοειδής συνάρτηση δίνει διαφορετικά αποτελέσματα από τους Perceptrons.

[1]Η σιγμοειδής συνάρτηση αναφέρεται και ως λογιστική συνάρτηση κατ' επέκταση οι νευρώνες ονομάζονται και λογιστικοί νευρώνες



Σχήμα 4: Γραφική παράσταση της σιγμοειδούς συνάρτησης

Ο καλύτερος τρόπος για να μπορέσουμε να παρουσιάσουμε την διαδικασία εκπαίδευσης είναι ίσως μέσα από την πλέον γνωστή εφαρμογή τεχνητών νευρωνικών δικτύων, την αναγνώριση χειρόγραφων ψηφίων. Συγκεκριμένα, ένα δίκτυο το οποίο τροφοδοτείται με ασπρόμαυρες εικόνες χειρόγραφων ψηφίων, προσπαθεί να τις κατηγοριοποιήσει σε ένα από τα 10 ψηφία (0...9). Το δίκτυο θα είναι δίκτυο εμπρόσθιας τροφοδότησης, δηλαδή δεν υπάρχουν κλειστοί βρόγχοι μεταξύ των νευρώνων, η πληροφορία προχωράει προς τα μέσα στο δίκτυο και ποτέ προς τα πίσω.
Οι εικόνες με τις οποίες θα τροφοδοτείται το δίκτυο είναι 28x28=784 pixel από το MNIST dataset, άρα χρειαζόμαστε 784 νευρώνες εισόδου. Οι νευρώνες εξόδου θα είναι προφανώς 10, έχουμε 10 πιθανές κατηγορίες. Ο αριθμός των νευρώνων στο hidden layer θα είναι κάτι με το οποίο θα πειραματιστούμε αλλά για αρχή θα τοποθετήσουμε 15 κρυφούς νευρώνες.
Ας περάσουμε στην εκπαίδευση του δικτύου. Για να μπορέσουμε να σκεφτούμε πως θα εκπαιδεύσουμε ένα δίκτυο πρέπει πρώτα να σκεφτούμε τι περιμένουμε από το δίκτυο στην έξοδο. Έστω, ότι η εικόνα που του τροφοδοτήσαμε σαν είσοδο x απεικονίζει το ψηφίο "7", η επιθυμητή έξοδος αν δούμε τους νευρώνες εξόδου σαν διάνυσμα στήλης θα είναι $y(x) = (0, 0, 0, 0, 0, 0, 0, 1, 0, 0)^T$, δηλαδή στην ιδανική περίπτωση μόνο ο νευρώνας που αντιστοιχεί στην κατηγοριοποίηση "7" θα έδινε μη-μηδενική έξοδο. Προφανώς κάτι τέτοιο συμβαίνει εξαιρετικά σπάνια και για τον λόγο αυτό πρέπει να ορίσουμε έναν τρόπο με τον οποίο θα ποσοτικοποιήσουμε το λάθος που κάνει το δίκτυο στην κατηγοριοποίηση.
Ορίζουμε την συνάρτηση κόστους

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2. \tag{3}$$

όπου το w είναι η συλλογή όλων των βαρών, b όλα τα biases, n ο αριθμός των συνολικών αντικειμένων εκπαίδευσης, y(x) η πραγματική έξοδος του δικτύου, a η ιδανική έξοδος ενώ η άθροιση πραγματοποιείται σε όλες τις εισόδους. Η συγκεκριμένη συνάρτηση κόστους ονομάζεται τετραγωνική συνάρτηση κόστους ή μέσο τετραγωνικό σφάλμα (MSE). Από την μορφή της συνάρτησης παρατηρούμε ότι είναι μη-αρνητική και όταν η πραγματική έξοδος πάει

να ταυτιστεί με την ιδανική γίνεται πολύ μικρή. Άρα η εκπαίδευση γίνεται σωστά όταν για κατάλληλες συλλογές w και b η συνάρτηση κόστους τείνει στο 0. Από μαθηματική πλευρά αυτό που ζητάμε είναι η ελαχιστοποίηση της συνάρτησης κόστους μέσω των ορισμάτων w και b. Δηλαδή, αυτό που καλούμαστε να κάνουμε είναι να επιλέξουμε κατάλληλα διανύσματα w και b που μας οδηγούν στο ελάχιστο της συνάρτησης κόστους. Είναι γνωστό ότι η βαθμίδα μιας συνάρτησης δίνει διάνυσμα με κατεύθυνση μέγιστης-ταχύτερης αύξησης της συνάρτησης, συνεπώς αν κινηθούμε αντίθετα θα πηγαίνουμε στο ελάχιστο της συνάρτησης. Ορίζοντας μια παράμετρο μάθησης $\boldsymbol{\eta}$, τα νέα βάρη και biases υπολογίζονται σύμφωνα με τις σχέσεις:

$$\begin{aligned} w_k \to w'_k &= w_k - \eta \frac{\partial C}{\partial w_k} \\ b_l \to b'_l &= b_l - \eta \frac{\partial C}{\partial b_l} \end{aligned} \tag{4}$$

Εφαρμόζοντας διαδοχικά την (4) βρισκόμαστε στο μονοπάτι το οποίο ελαχιστοποιεί την συνάρτηση κόστους, με άλλα λόγια αυτός είναι ένας κανόνας με τον οποίο μπορούμε να εκπαιδεύουμε δίκτυα. Από θεωρητική σκοπιά όλα είναι σωστά στην εφαρμογή όμως η βαθμίδα ∇C, για μεγάλο αριθμό δειγμάτων εκπαίδευσης, κάτι που γενικά είναι επιθυμητό, μπορεί να γίνει πολύ χρονοβόρα στον υπολογισμό της και η μάθηση του δικτύου να καθυστερήσει αρκετά ή να είναι τόσο χρονοβόρα που η εκπαίδευση είναι πρακτικά αδύνατη σε λογικό χρόνο. Έτσι χρησιμοποιούμε την τεχνική της *Στοχαστικής καθόδου βαθμίδας*. Η κεντρική ιδέα είναι ότι ο μεγάλος αριθμός δειγμάτων χωρίζεται με τυχαίο τρόπο σε μικρότερα σύνολα δειγμάτων, τα λεγόμενα *mini-batches* και για κάθε mini-batch υπολογίζουμε "τον μέσο όρο της βαθμίδας", $\nabla C = \frac{1}{n}\sum_x \nabla C_x$, όπου $C_x = \frac{\|y(x)-a\|^2}{2}$ σε μικρό αριθμό δειγμάτων και έτσι το δίκτυο αρχίζει την εκπαίδευση αρκετά πιο νωρίς, αφού δεν χρειάζεται να περιμένει να γίνει ένα ολόκληρο πέρασμα σε όλα τα δεδομένα από την στιγμή που από το πρώτο mini-batch υπολογίζουμε την βαθμίδα στην συνάρτηση κόστους.

Έχοντα υπολογίσει την βαθμίδα της συνάρτησης κόστους στην έξοδο του δικτύου, σκοπός είναι να βρούμε έναν εύκολο τρόπο να συσχετίσουμε αυτή την βαθμίδα στο τέλος του δικτύου με όλες τις ενδιάμεσες παραγώγους για να μπορέσουμε να εφαρμόσουμε τους κανόνες μάθησης για τα νέα βάρη και biases, να μπορέσουμε δηλαδή να "γυρίσουμε" προς τα πίσω το σφάλμα για να βελτιώσουμε το δίκτυο, εξού και το όνομα του αλγόριθμου, που θα αξιοποιήσουμε, "Backpropagation". Πριν παρουσιάσουμε την λογική του αλγόριθμου είναι χρήσιμο να δούμε πως μπορούμε να υπολογίσουμε την έξοδο των δικτύων με χρήση πινάκων και γραμμικής άλγεβρας. Θα χρησιμοποιήσουμε την γραφή w^l_{jk} για να δηλώσουμε το βάρος που συνδέει τον k^{th} νευρώνα στο $(l-1)^{th}$ layer στον j^{th} νευρώνα στο l^{th} layer. Με αυτό τον συμβολισμό η έξοδος του j νευρώνα στο l layer υπολογίζεται με την σχέση

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right), \tag{5}$$

όπου το άθροισμα γίνεται ως προς του νευρώνες του προηγούμενου $(l-1)$ layer. Προφανώς, άμα θεωρήσουμε ότι ο δείκτης j δείχνει την γραμμή ενός πίνακα με στοιχεία τις εξόδους, το άθροισμα μπορεί να ερμηνευτεί σαν ένα εσωτερικό γινόμενο, έτσι η παραπάνω σχέση παίρνει την εξής πολύ συμπαγή και όμορφη μορφή

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{6}$$

. Το όρισμα της συνάρτησης ενεργοποίησης υπολογίζεται πάντα και έτσι μπορούμε να το ονομάσουμε z και σε μορφή στοιχείων ορίζεται ως $z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j$. Για την

παρουσίαση του αλγόριθμου χρειαζόμαστε μια τελευταία ποσότητα που θα την ονομάσουμε σφάλμα του νευρώνα j στο layer l, από την ονομασία και μόνο αναμένουμε να έχει σχέση με την συνάρτηση κόστους και πράγματι η ποσότητα ορίζεται ως

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}. \tag{7}$$

Σκοπός του Backpropagation είναι να συνδέσουμε το σφάλμα του νευρώνα σε κάθε layer με τις ποσότητες που μας ενδιαφέρουν, τις μερικές παραγώγους της συνάρτησης κόστους ως προς τα βάρη και τα biases με στόχο την εκπαίδευση του δικτύου. Ο αλγόριθμος Backpropagation απαρτίζεται από 4 βασικές εξισώσεις, η απόδειξη του ειδικά σε μορφή στοιχείων και όχι πινάκων είναι τετριμμένη χρήση του κανόνα αλυσίδας γι' αυτό και παραλείπεται. Αντί των αποδείξεων θα προσπαθήσουμε να δώσουμε ένα διαισθητικό νόημα στις εξισώσεις και να δούμε πόσο απλά προκύπτει ολόκληρη η θεώρηση της χρήσης του Backpropagation για την εκπαίδευση του δικτύου.

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{BP3}$$

$$\frac{\partial C}{\partial w_{j\,k}^l} = a_k^{l-1} \delta_j^l \tag{BP4}$$

Σχήμα 5: Οι εξισώσεις του αλγόριθμου Backpropagation
Πηγή: Neural Networks and Deep Learning [16]

Η εξίσωση $BP1$ προφανώς με χρήση του κανόνα της αλυσίδας μας δίνει το σφάλμα στους νευρώνες εξόδου, να σημειώσουμε εδώ ότι το σύμβολο $\odot$ είναι το γινόμενο Hadamard όπως αυτό ορίζεται από την γραμμική άλγεβρα, π.χ.:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 * 3 \\ 2 * 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}. \tag{8}$$

Συνεπώς, η πρώτη εξίσωση δεν μας δίνει κάποιο καινούριο στοιχείο για τον αλγόριθμο απλά υπολογίζουμε το σφάλμα του νευρώνα εξόδου με χρήση του κανόνα της αλυσιδωτής παραγώγισης. Το επόμενο βήμα του αλγόριθμου είναι να μπορέσουμε να διαδώσουμε προς τα πίσω αυτό το σφάλμα και στους εσωτερικούς κρυφούς νευρώνες με χρήση του κανόνα της αλυσίδας μπορούμε να δημιουργήσουμε την εξής σχέση:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{9}$$

$$= \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \tag{10}$$

$$= \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1}, \tag{11}$$

συνδέσαμε δηλαδή το σφάλμα στον layer που βρισκόμαστε με αυτό στον επόμενο. Αν προχωρήσουμε τις πράξεις θα καταλήξουμε στην δεύτερη σχέση του αλγόριθμου. Στην πραγματικότητα το δύσκολο κομμάτι έχει τελειώσει καθώς από την στιγμή που θα έχουμε το σφάλμα για κάθε layer-νευρώνα οι σχέσεις $BP3$ και $BP4$ δηλώνουν ότι το σφάλμα είναι ίσο με την παράγωγο της συνάρτησης κόστους ως προς τα biases, ενώ το σφάλμα επί την έξοδο του k νευρώνα στον $l-1$ layer είναι ίσο με την παράγωγο της συνάρτησης κόστους ως προς το βάρος w^l_{jk}. Τελικά, υπολογίσαμε τις ποσότητες που μας ενδιαφέρουν για την εκπαίδευση του δικτύου μέσω της ενδιάμεσης ποσότητας "σφάλμα του νευρώνα" για κάθε layer-νευρώνα, αυτό σημαίνει ότι πλέον έχουμε όλα τα εργαλεία για να μπορέσουμε να εκπαιδεύσουμε και να αξιοποιήσουμε ένα τεχνητό νευρωνικό δίκτυο.

2.4.2 Συνελικτικά νευρωνικά δίκτυα

Στην συνέχεια θα παρουσιάσουμε τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks - CNN). Μια ειδική διάταξη νευρών που χρησιμοποιείται κυρίως σε προβλήματα όρασης υπολογιστών και αναγνώρισης προτύπων. Το όφελός τους στην παρούσα εργασία θα φανεί στην συνέχεια, αλλά είναι σίγουρο ότι θα κατέχουν πρωταγωνιστικό ρόλο.

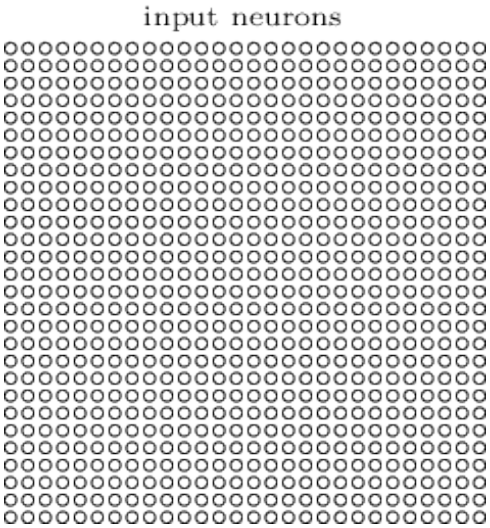
Ακριβώς όπως και τα τεχνητά νευρωνικά δίκτυα εμπνεύστηκαν από την βιολογία έτσι και τα συνελικτικά δίκτυα έχουν τις ρίζες τους στην βιολογία. Συγκεκριμένα, μελέτες του οπτικού φλοιού της γάτας εμφάνισαν μια περίπλοκη διάταξη κυττάρων η οποία επιβάλει συνδέσεις μεταξύ νευρωνικών κυττάρων που είναι ευαίσθητα σε συγκεκριμένα υπο-φάσματα του φωτός, τα οποία λειτουργούν ως τοπικά φίλτρα σε αυτά τα υπο-φάσματα. Με την βοήθεια των εν λόγω "φίλτρων" η γάτα είναι σε θέση να εκμεταλλευτεί την μεγάλη χωρική συσχέτιση που δημιουργεί από τα υπο-φάσματα για την καλύτερη αντίληψη του περιβάλλοντα χώρου. Συνεπώς και τα συνελικτικά νευρωνικά δίκτυα δημιουργήθηκαν για να μπορούν να εκμεταλλεύονται τις χωρικές συσχετίσεις που υπάρχουν στις εικόνες.

Για την κατανόηση των συνελικτικών δικτύων πρέπει να περιγράψουμε τρεις βασικές δομικές διαφορές:

- Local receptive fields
- Κοινά βάρη
- Pooling

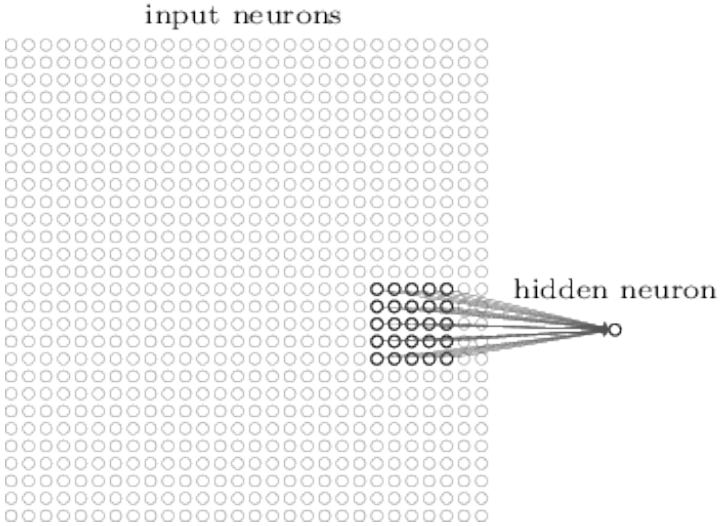
Ο καλύτερος τρόπος για να εξηγήσουμε τα επιμέρους δομικά στοιχεία είναι να δανειστούμε κάποιες εικόνες και νοήματα από το εξαιρετικό διαδικτυακό βιβλίο του Michael Nielsen [16].

Local receptive fields: Ο καλύτερος τρόπος να παρουσιάζουμε την είσοδο ενός δικτύου είναι ένα πλέγμα από pixel.



Σχήμα 6: Πλέγμα εισόδου ενός δικτύου CNN - pixels
Πηγή: Neural Networks and Deep Learning [16]

Πλέον, το δίκτυο δεν συνδέει κάθε είσοδο-pixel με το επόμενο κρυφό επίπεδο αλλά συνδέει μικρές, τοπικά εντοπισμένες περιοχές τις εικόνας με το επόμενο επίπεδο. Για παράδειγμα είναι δυνατόν κάθε νευρώνας στο πρώτο κρυφό επίπεδο να συνδέεται με μια περιοχή 5×5, δηλαδή 25 pixel.



Σχήμα 7: Σύνδεση περιοχής 25 pixel με το πρώτο κρυφό επίπεδο σε ένα δίκτυο CNN
Πηγή: Neural Networks and Deep Learning [16]

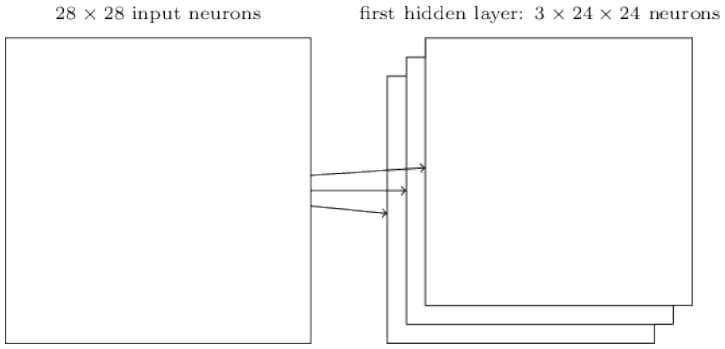
Η περιοχή των 25 pixel ονομάζεται *Local receptive field* για τον εκάστοτε κρυφό νευρώνα. Κάθε σύνδεση έχει ένα ξεχωριστό βάρος και ο κρυφός νευρώνας έχει ένα καθολικό bias. Δεν θα ήταν λάθος να σκεφτεί κανείς ότι ο κάθε κρυφός νευρώνας μαθαίνει να αναλύει το "δικό" local receptive field.

Κοινά βάρη: Αναφέρθηκε ήδη ότι ο κάθε κρυφός νευρώνας έχει τα δικά του βάρη, στο συγκεκριμένο παράδειγμα 5×5 βάρη. Όμως, εισάγεται επιπλέον και ο περιορισμός ότι

όλοι οι νευρώνες στο συγκεκριμένο κρυφό επίπεδο μοιράζονται τα συγκεκριμένα 5×5 βάρη:

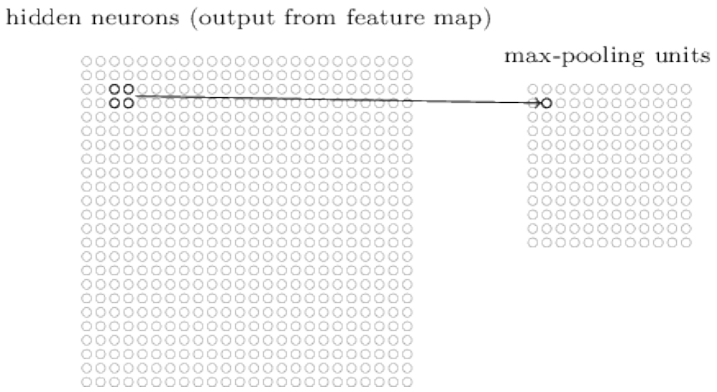
$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right). \tag{12}$$

Όπου, σ είναι η συνάρτηση ενεργοποίησης, η λογιστική για παράδειγμα και $\mathbf{b}$ το κοινό bias. Με αυτή την δομή είναι ξεκάθαρο ότι το κάθε κρυφό επίπεδο εκπαιδεύεται να εντοπίζει ένα διαφορετικό χαρακτηριστικό (feature) στην εικόνα εισόδου. Με άλλα λόγια τα CNN δίκτυα είναι εξαιρετικά στο να διαχειρίζονται αντικείμενα που είναι χωρικά αναλλοίωτα, όπως οι εικόνες: μια εικόνα που απεικονίζει μια γάτα, δείχνει ακόμα μια γάτα ακόμα και αν μεταφερθεί σε άλλο σημείο. Προφανώς, επειδή κάθε κρυφό επίπεδο μαθαίνει ένα χαρακτηριστικό σε κάθε κρυφό επίπεδο υπάρχουν αρκετές στρώσεις για να εντοπίζουν διαφορετικά χαρακτηριστικά. Οι στρώσεις ονομάζονται feature maps.



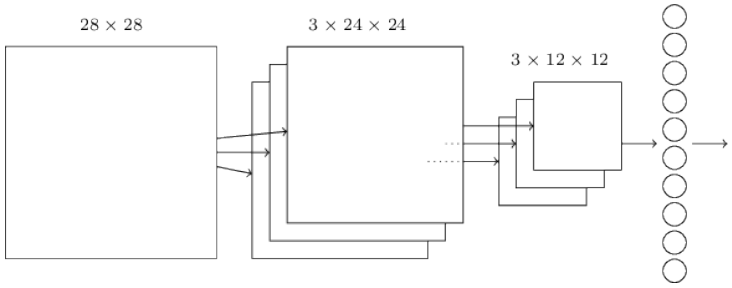
Σχήμα 8: Στρώσεις feature map στο πρώτο κρυφό επίπεδο ενός δικτύου CNN
Πηγή: Neural Networks and Deep Learning [16]

Pooling layers: Εκτός από τα συνελικτικά επίπεδα που είδαμε πριν, τα CNN έχουν και layers που εκτελούν *pooling* διαδικασίες. Οι pooling layers μπαίνουν αμέσως μετά τους συνελικτικούς και χρησιμεύουν στην μείωση της πολυπλοκότητας του δικτύου. Υπάρχουν διάφοροι pooling layers (max, average, κλπ) αλλά η λογική τους είναι ακριβώς η ίδια. Έχουν σαν είσοδο την έξοδο από ένα feature map και μια περιοχή, για παράδειγμα 2×2 την συρρικνώνουν σε ένα pixel. Έτσι μια έξοδος από feature map 24×24 καταλήγει σε 12×12 με pooling 2×2



Σχήμα 9: Γραφική αναπαράσταση του 2×2 pooling
Πηγή: Neural Networks and Deep Learning [16]

Μετά από τα συνελικτικά και pooling επίπεδα ακολουθεί ένα επίπεδο "κανονικού" τεχνητού νευρωνικού δικτύου που λειτουργεί ως ο τελικός ταξινομητής. Δηλαδή, μια άλλη οπτική των νευρωνικών δικτύων είναι ότι λειτουργούν σαν μηχανισμοί μετασχηματισμού των εικόνων για να εξάγουν χαρακτηριστικά που θα βοηθήσουν τον τελικό ταξινομητή να πετύχει καλύτερη ακρίβεια.



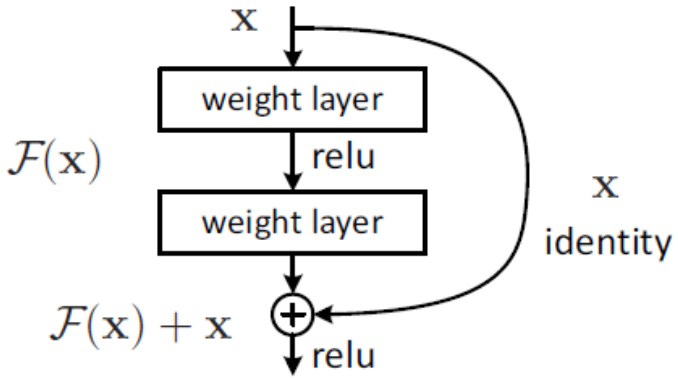
Σχήμα 10: Η γενική δομή ενός CNN δικτύου
Πηγή: Neural Networks and Deep Learning [16]

Στην συνέχεια της παρούσας εργασίας θα ασχοληθούμε εκτενώς με τα CNN δίκτυα. Για τον λόγο αυτό αξίζει να αναφερθούμε σε ένα ιδιαίτερο δίκτυο που θα αξιοποιήσουμε.

2.4.3 ResNet

Το Residual Neural Network ή ResNet αναπτύχθηκε από τους Kaiming He et al. [17] και κατέλαβε την πρώτη θέση στον διαγωνισμό ILSVRC 2015 μειώνοντας το top-5 error, η κατηγορία της εικόνας εισόδου από το ImageNet [18] να βρίσκεται στις 5 πρώτες προβλέψεις του δικτύου, στο 3,6%. Το πρόβλημα που αντιμετώπιζαν τα συμβατικά CNN ήταν ότι η συνεχής αύξηση του βάθους, η αύξηση των κρυφών επιπέδων, δεν συνοδευόταν και από αύξηση των επιδόσεων. Οι σχεδιαστές των ResNets μέσω της εισαγωγής των Residual Blocks κατέστησαν εφικτή την σημαντική αύξηση του βάθους των δικτύων πετυχαίνοντας ταυτόχρονα την καλύτερη επίδοση που είχε καταγραφεί μέχρι εκείνη τη στιγμή στο διαγωνισμό.

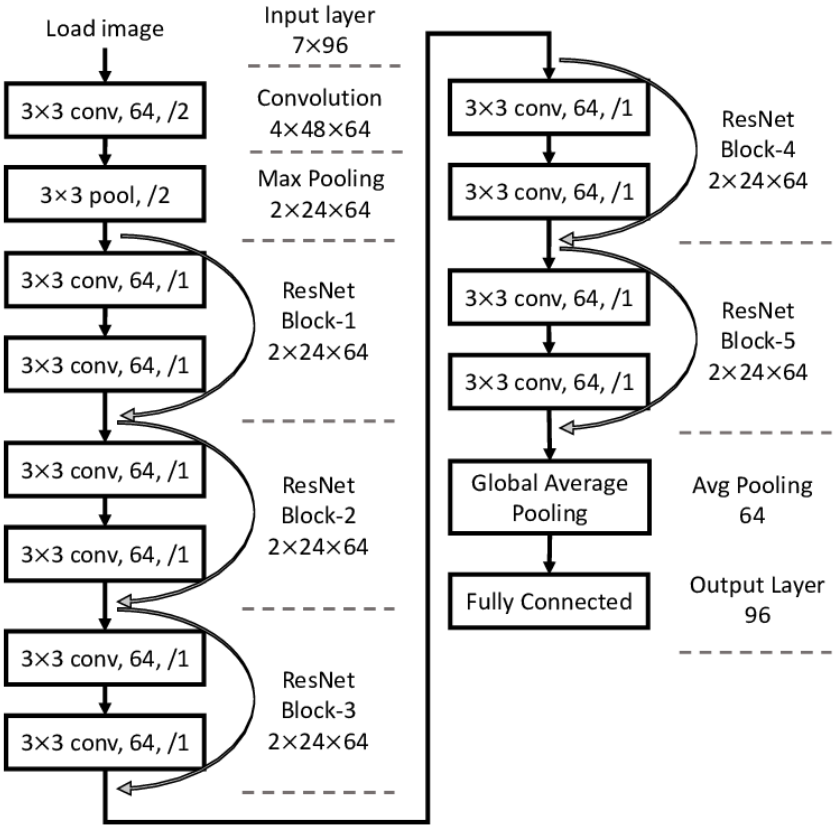
Το residual block, ο πυρήνας του ResNet, ανατρέπει τη συμβατική ροή επεξεργασίας των CNN κατά την οποία το κάθε επίπεδο λαμβάνει είσοδο x και παράγει έξοδο $F(x)$. Συγκεκριμένα, κάθε residual block διαθέτει μια σύνδεση που παρακάμπτει τα παραμετρικά επίπεδα που εφαρμόζουν την συνάρτηση $F(x)$ στην είσοδο x. Η σύνδεση αυτή ονομάζεται "skip identity connection". Έτσι σε καθε residual block λαμβάνουμε έξοδο $F(x) + x$.



Σχήμα 11: Η δομή ενός Residual Block
Πηγή: Residual CNNs for Image Classification Tasks [19]

Με τον τρόπο αυτό ακόμα και αν κάποιος layer εμφανίσει κάποιο πρόβλημα στην εκπαίδευση, υπάρχει η καθαρή σύνδεση με τους προηγούμενους για να μεταφερθεί από εκεί η πληροφορία.

Υπάρχουν πολλές εκδοχές των ResNets με ειδοποιός διαφορά τον αριθμό των επιπέδων τους (ResNet50, ResNet101, ResNet152 κ.ά.).



Σχήμα 12: Γραφική αναπαράσταση του ResNet12
Πηγή: Short-Term Load Forecasting based on ResNet and LSTM [20]

3 Επισκόπηση Συναφούς Βιβλιογραφίας

3.1 Προσεγγίσεις Μηχανικής Μάθησης

Όπως αναφέρθηκε προηγουμένως, οι επιθέσεις κακόβουλου λογισμικού αυξάνονται συνεχώς και ωθούν τους αναλυτές ασφάλειας να αναπτύξουν καλύτερες, πιο έξυπνες και ταχύτερες μεθόδους για την αντιμετώπιση Παραδοσιακά, οι λύσεις AV (Anti-Virus) χρησιμοποιούν έναν συνδυασμό μεθόδων που περιγράφονται στο πρώτο κεφάλαιο. Αυτές οι μέθοδοι απαιτούν την ανάλυση του κακόβουλου λογισμικού πριν από την επίθεση, ώστε η βάση δεδομένων γνώσης να περιέχει όλες τις απαραίτητες πληροφορίες για την αποφυγή οποιουδήποτε επιβλαβούς αποτελέσματος. Κατά τη διάρκεια της τελευταίας δεκαετίας, όμως, οι εφαρμογές μηχανικής μάθησης τείνουν να υπερτερούν των κλασικών ομολόγων τους σε πολλούς τομείς, συμπεριλαμβανομένης της ασφάλειας στον κυβερνοχώρο. Δεδομένου του συνεχώς αυξανόμενου αντίκτυπου των εργαλείων που υποστηρίζουν την τεχνητή νοημοσύνη στην ασφάλεια στον κυβερνοχώρο, μια βιβλιογραφική ανασκόπηση και σύγκριση των τελευταίων προσεγγίσεων κρίνεται απαραίτητη για την περαιτέρω κατανόηση και βελτίωση των αναπτυσσόμενων μεθόδων.

Για να διευκολυνθεί η ανασκόπηση, αρκετά δημοσιεύματα επιλέχθηκαν από την εξαιρετική συγκεντρωτική εργασία των Gibert et al. (2020) [21] οι οποίοι έχουν συγκεντρώσει τα δημοφιλέστερα δημοσιεύματα και τα παρουσιάζουν/συγκρίνουν. Στην εργασία μας θα εμπλουτίσουμε αυτή την παρουσίαση/σύγκριση με τρία σημεία που θα παίξουν κυρίαρχο ρόλο στο πειραματικό μέρος: τις τεχνικές μείωσης χαρακτηριστικών, την αναλογία των δειγμάτων $\frac{benign}{total}\%$ και την μέθοδο με την καλύτερη επίδοση. Όμως όλα τα δημοσιεύματα εξετάστηκαν με βάση την απόδοση, την ημερομηνία δημοσίευσης και την ευκολία αναπαραγωγής τους. Βέβαια, οι εξελίξεις είναι ραγδαίες, με αποτέλεσμα η εργασία των Gibert et al. (2020) [21] παρότι δημοσιεύτηκε πριν από ένα χρόνο να μην εμπεριέχει νέα δημοσιεύματα. Για τον λόγο αυτό, θα προσπαθήσουμε σε κάθε υποενότητα να την εμπλουτίζουμε και με πιο πρόσφατες δημοσιεύσεις. Συνολικά θα συγκρίνουμε τις επιλεγμένες δημοσιεύσεις στα ακόλουθα βασικά σημεία:

- Χρησιμοποιούμενη μέθοδος μηχανικής μάθησης
- Επιλεγμένο Dataset
- Μέγεθος δεδομένων
- Τύπος χαρακτηριστικών (feature)
- Μέθοδος μείωσης διαστάσεων/όγκου (εάν χρησιμοποιείται)
- Αναλογία δειγμάτων
- Στόχος (μεταβλητή εξόδου)

Οι επιλεγμένες εργασίες θα παρουσιαστούν ομαδοποιημένες ανά τεχνικές και ταξινομημένες κατά ημερομηνία δημοσίευσης. Αυτή η διάρθρωση, ελπίζουμε, θα καταστήσει σαφή την εξέλιξη που λαμβάνει χώρα στον τομέα, καθώς προχωρά η έρευνα στο πεδίο της μηχανικής μάθησης.

Πριν ξεκινήσουμε τη βιβλιογραφική ανασκόπηση, είναι προς όφελός μας να παρουσιάσουμε τα στάδια μιας εφαρμογής μηχανικής μάθησης που ισχύει για τις περισσότερες, αν όχι για όλες, τις περιπτώσεις. Είναι μια επαναληπτική διαδικασία που περιλαμβάνει τη

συλλογή διαθέσιμων δεδομένων, τον καθαρισμό και την προετοιμασία των δεδομένων, τη δημιουργία μοντέλων, επικύρωση των μοντέλων και διάθεση προς χρήση. Η χρήση της μηχανικής εκμάθησης για την αντιμετώπιση κακόβουλου λογισμικού απαιτεί κάποιο είδος προεργασίας για την εξαγωγή ενός συνόλου χαρακτηριστικών (features) που παρέχουν μια αφηρημένη απεικόνιση του λογισμικού/κακόβουλου λογισμικού. Στη συνέχεια, τα χαρακτηριστικά χρησιμοποιούνται για την εκπαίδευση και την επικύρωση ενός μοντέλου για την εκάστοτε εργασία. Υπάρχουν δύο κύριες κατηγορίες στις οποίες μια λύση μηχανικής μάθησης, ενάντια σε επιθέσεις κακόβουλου λογισμικού, θα μπορούσε να εμπίπτει. Από τη μία πλευρά, τα συστήματα ανίχνευσης κακόβουλου λογισμικού εξάγουν απλώς μια τιμή $y = f(\mathbf{x})$, στο εύρος 0 έως 1, υποδεικνύοντας το επίπεδο εμπιστοσύνης ότι ένα εκτελέσιμο είναι κακόβουλο ή όχι. Από την άλλη πλευρά, τα συστήματα ταξινόμησης κακόβουλου λογισμικού παράγουν την πιθανότητα ενός δεδομένου εκτελέσιμου να ανήκει σε μια συγκεκριμένη κλάση, $y = f(\mathbf{x}), y \in R^N$, όπου N είναι ο αριθμός διαφορετικών κατηγοριών κακόβουλου λογισμικού. Κατά τη διάρκεια του εισαγωγικού κεφαλαίου αναφέραμε δύο βασικούς τύπους τεχνικών που χρησιμοποιούν οι AV, τη στατική προσέγγιση και τη δυναμική προσέγγιση. Είναι λογικό και επόμενο ότι τα χαρακτηριστικά που χρησιμοποιούνται από τις λύσεις μηχανικής μάθησης θα ακολουθούν την ίδια κατηγοριοποίηση, (1) στατικά χαρακτηριστικά και (2) δυναμικά χαρακτηριστικά.

3.2 Στατικά Χαρακτηριστικά

Τα στατικά χαρακτηριστικά θεωρούνται γενικά "ασφαλέστερα" στην εξαγωγή τους επειδή δεν υπάρχει η ανάγκη εκτέλεσης του εκτελέσιμου. Όσον αφορά τα αρχεία φορητών εκτελέσιμων αρχείων (PE), τα στατικά χαρακτηριστικά μπορούν να εξαχθούν είτε από το περιεχόμενο του αρχείου είτε από τη γλώσσα assembly.

3.2.1 Συμβολοσειρές

Οι συμβολοσειρές είναι, συνήθως, ακολουθίες εκτυπώσιμων χαρακτήρων εντός του εκτελέσιμου. Οι συμβολοσειρές μερικές φορές είναι ο ευκολότερος τρόπος για να μάθουμε για τη λειτουργία ενός προγράμματος χωρίς πολλή φασαρία. Για παράδειγμα, οι συμβολοσειρές μπορούν να δώσουν πληροφορίες σχετικά με διευθύνσεις URL, διαδρομές αρχείων, κωδικούς πρόσβασης, μενού επιλογών κ.λπ. Οι συμβολοσειρές θα αποκαλύψουν πληροφορίες για όλα όσα είναι τοποθετημένα από τους προγραμματιστές και είναι εκτυπώσιμα.

Ye et al. (2008α) [22] ανέπτυξαν ένα σύστημα ανίχνευσης κακόβουλου λογισμικού που βασίζεται σε ερμηνεύσιμες συμβολοσειρές που εξάγονται από τα δυαδικά αρχεία. Η προτεινόμενη λύση συνίστατο σε έναν αναλυτή για την εξαγωγή των συμβολοσειρών και την προετοιμασία τους ως χαρακτηριστικά και ένα ensemble SVM με bagging για τη δημιουργία του ανιχνευτή. Το σύστημα εκπαιδεύτηκε και αξιολογήθηκε χρησιμοποιώντας ένα σύνολο δεδομένων που συντέθηκε από το εργαστήριο προστασίας από ιούς Kingsoft, με 39838 δείγματα, από τα οποία τα 8320 ήταν καλοήθη εκτελέσιμα. Χρησιμοποιώντας τη μετρική F1 το σύστημα πέτυχε 92.3% και 92.22% για micro-F1 και macro-F1 αντίστοιχα.

Markel and Bilzor (2014) [23] επικεντρώθηκαν στην εργασία τους στις κεφαλίδες των PE αρχείων. Εκτός από τις ερμηνεύσιμες συμβολοσειρές ένα εκτελέσιμο σε μορφή PE έχει τις κεφαλίδες που μπορούν να δώσουν αρκετές πληροφορίες για την συμπεριφορά του αρχείου. Στην εργασία τους θέλησαν να μελετήσουν την συμπεριφορά των ταξινομητών όταν δεν υπάρχει ισορροπία στα σύνολα δεδομένων. Δηλαδή, όταν σε κάποιο ρεαλιστικό σενάριο, τα καλοήθη δείγματα αποτελούν το 99.9% των δειγμάτων και το κακόβουλο μόλις

το 0.1%. Δοκιμάζοντας τους ταξινομητές, NB, LR και DT. Σε ένα σύνολο 25000 δειγμάτων, με μεταβλητό αριθμό καλοηθών μεταξύ των πειραμάτων, ο αλγόριθμος DT ήταν πάντα ο καλύτερος απο θέμα ακρίβειας.

Balram et al. (2019) [24] παρουσίασαν έναν σύστημα εντοπισμού κακόβουλου λογισμικού βασισμένο στον εντοπισμό συμβολοσειρών. Χρησιμοποιώντας για ελάχιστο μήκος ακολουθίας τους 5 χαρακτήρες, εξήγαγαν από τα δεδομένα τους όλες τις ερμηνεύσιμες συμβολοσειρές και χρησιμοποιώντας το hashing trick για να μειώσουν τις διαστάσεις ετοίμασαν τα διανύσματα που θα τροφοδοτήσουν τους ταξινομητές τους. Δοκίμασαν τους ταξινομητές, SVM, LR, RF και XGB. Σε ένα σύνολο 1416 δειγμάτων εκ των οποίων τα 989 ήταν καλοήθη ο συνδυασμός του LR με τον XGB έδωσαν το καλύτερο αποτέλεσμα με μια ακρίβεια 98%.

Δυστυχώς, η χρήση συμβολοσειρών ως κύριο χαρακτηριστικό δεν προτιμάται συχνά, επειδή οι επιτιθέμενοι χρησιμοποιούν πολλές τεχνικές για να αποκρύψουν ή να κρυπτογραφήσουν σε μεγάλο βαθμό το περιεχόμενο του κακόβουλου λογισμικού και καθιστώντας έτσι αδύνατη την αποκάλυψη της πρόθεσης ή του στόχου τους μόνο με την εξέταση των εκτυπώσιμων συμβολοσειρών.

3.2.2 Bytes και εντολές

Η προσέγγιση από μια πιο αφηρημένη διαδρομή και η αντιμετώπιση ενός δυαδικού αρχείου ως ακολουθία byte οδηγεί στο πιο συχνά χρησιμοποιούμενο χαρακτηριστικό στην ανίχνευση και ταξινόμηση κακόβουλου λογισμικού, τα n-grams. Ένα n-gram είναι μια συνεχόμενη ακολουθία n στοιχείων από μια δεδομένη ακολουθία κειμένου. Ένα καλό μέρος για την εξαγωγή χρήσιμων n-grams θα μπορούσε να είναι τα byte που αντιπροσωπεύουν το δυαδικό περιεχόμενο του κακόβουλου λογισμικού ή/και ο πηγαίος κώδικας assembly (opcodes). Τα εξαγόμενα n-grams αναφέρονται στον μοναδικό συνδυασμό κάθε n διαδοχικού byte/opcode ως μεμονωμένο χαρακτηριστικό.

Santos et al. (2013) [25] ανέπτυξαν έναν ανιχνευτή κακόβουλου λογισμικού με βάση τη συχνότητα εμφάνισης εντολών (opcodes) και τη συνάφειά τους χρησιμοποιώντας τη μετρική Mutual Information [26]. Χρησιμοποίησαν n-grams, όπου το n κυμαίνεται από 1 έως 4, αλλά ο υπολογιστής τους μπορούσε να χειριστεί τον όγκο για μείωση χαρακτηριστικών, επιλογή των 1000 καλύτερων, χρησιμοποιώντας την μετρική Information Gain σε δείγματα 1-grams και 2-grams. Η προσέγγισή τους επικυρώθηκε σε 17000 κακόβουλα και 1000 καλοήθη προγράμματα και τα αποτελέσματα δείχνουν ότι η υψηλότερη ακρίβεια επιτεύχθηκε με την χρήση ενός ταξινομητή Support Vector Machine με κανονικοποιημένο πολυώνυμο ως πυρήνα, φτάνοντας το ποσοστό 95.9%.

Boujnouni et al. (2014) [27] εργάστηκαν για την βελτίωση του αλγορίθμου Support Vector Domain Description (SVDD)[28]. Συγκεκριμένα χρησιμοποιώντας 4-grams σαν δεδομένα εισόδου, προσπάθησαν να τροποποιήσουν τον αλγόριθμο με τέτοιον τρόπο ώστε όχι μόνο να βρίσκει την υπερσφαίρα με τον μικρότερο όγκο για τον καλύτερο διαχωρισμό των δεδομένων αλλά επιτρέπει λάθη, σε ποσοστό που ορίζεται από μια υπερπαράμετρο, με σκοπό να βρεθεί η υπερσφαίρα με τον ελάχιστο όγκο. Έπειτα, μειώνοντας το πλήθος των n-grams με την χρήση του αλγορίθμου "information gain" σε ένα σύνολο 1258 δειγμάτων, εκ των οπίων τα 600 είναι καλοήθη, έδειξε ότι η προσέγγιση που πρότειναν πέτυχε μια ακρίβεια 94% σε σχέση με την απλή εκδοχή που πέτυχε μόλις 44%.

Fuyong et al. (2017)[29] χρησιμοποιώντας 3-grams πρότειναν μια μέθοδο που επιλέγει τα χαρακτηριστικά με βάση την απόκτηση πληροφοριών από τα 3-grams και τα συγκρίνουν με βάση την ομοιότητα μεταξύ των χαρακτηριστικών. Στη συνέχεια, υπολόγισαν τους μέσους όρους κάθε χαρακτηριστικού των διανυσμάτων χαρακτηριστικών για το κακόβουλο

λογισμικό και για τα καλοήθη δείγματα ξεχωριστά. Τέλος, ένα νέο κομμάτι λογισμικού ανατέθηκε σε μία από τις δύο κατηγορίες ανάλογα με την ομοιότητα μεταξύ του διανύσματος χαρακτηριστικών του άγνωστου δείγματος και του μέσου όρου των δύο κατηγοριών, εφαρμόζοντας ουσιαστικά τον 1-NN ως αλγόριθμο ταξινόμησης. Οι δοκιμές τους αποκάλυψαν ότι ο τελικός αριθμός των 3-grams δεν ήταν τόσο σημαντικός, αλλά 499 χαρακτηριστικά ήταν η καλύτερη επιλογή. Το σύνολο δεδομένων τους αποτελείται από 2540 δείγματα, εκ των οποίων τα 1252 ήταν καλοήθη.

Raff et al. (2019) [30] έχοντας μελετήσει εργασίες άλλων ερευνητών διαπίστωναν ότι οι αναφορές για n-grams με n μεγαλύτερο του 6 είναι εξαιρετικά σπανίες λόγω του ασύμφορου υπολογιστικού κόστους. Έτσι στην εργασία τους παρουσιάζουν μια μέθοδο ικανή να χειριστεί μεγάλα n≥1024. Για να μπορέσουν να χειριστούν τόσο μεγάλα n-grams καταφεύγουν στο να χρησημοποιήσουν τα top-k πιο συχνά n-grams. Για να επιταχύνουν αυτή την διαδικασία χρησιμοποιούν τα hash-grams που βασίζονται στην κατανομή Zipf[31]. Παρ' όλη την προσπάθεια τους, τα αποτελέσματα έδειξαν ότι σε ένα σύνολο 200000 δεδομένων, η καλύτερη ακρίβεια επιτεύχθηκε για n ίσο με 8, και έπεφτε σταδιακά όσο αυξανόταν ο αριθμός n.

Yuxin et al. (2019)[32] υιοθέτησαν μια πιο σύγχρονη προσέγγιση για να μειώσουν τη διαστατικότητα των χαρακτηριστικών εισόδου, 3-gras. Χρησιμοποιώντας ένα Deep Belief Network (DBN), ένα restricted Boltzmann machine ως αυτόματο κωδικοποιητή, μπόρεσαν να μειώσουν τις διαστάσεις του διανύσματος χαρακτηριστικών κάτω από 400, με τις 200 να είναι η καλύτερη δοκιμή τους. Τέλος, χρησιμοποιώντας τα εξαγόμενα και μειωμένα σε διάσταση χαρακτηριστικά, ο αλγόριθμος Decision Tree πέτυχε ακρίβεια 96.5% σε ένα σύνολο δεδομένων 11200 από τα οποία 4600 ήταν καλοήθη και 2000 χωρίς ετικέτα.

Είναι φανερό ότι τα n-grams είναι μια καλή προσέγγιση για την δημιουργία χαρακτηριστικών από εκτελέσιμα τα οποία θα τροφοδοτήσουν ταξινομητές μηχανικής μάθησης. Δεν σημαίνει όμως ότι είναι και τέλεια. Στην πραγματικότητα έχουν κάποια πολύ σοβαρά ελαττώματα που πρέπει να αναφερθούν. Πρώτον, όσο αυξάνεται ο αριθμός n αυξάνεται δραματικά και το πλήθος των χαρακτηριστικών που θα δημιουργηθούν, καθιστώντας τα μεγάλα n-grams υπολογιστικά ασύμφορα. Αυτή η δραματική αύξηση πολλές φορές ωθεί τους ερευνητές στην χρήση τεχνικών μείωσης διαστάσεων, οι οποίες εισάγουν έναν νέο πρόβλημα. Οι τεχνικές μείωσης διαστάσεων επιλέγουν τα πιο σημαντικά χαρακτηριστικά συνήθως με την συχνότητα εμφάνισής τους, κάτι το οποίο δεν είναι θεμιτό σε ένα εκτελέσιμο αρχείο καθώς τα πιο κοινά εμφανιζόμενα n-grams θα είναι πιθανών συμβολοσειρές ή ειδικοί χαρακτήρες, χαρακτηριστικά που δεν εμπεριέχουν πολλές πληροφορίες για την αναγνώριση ενός εκτελέσιμου ως κακόβουλου ή όχι. Τέλος, υπάρχει το πρόβλημα της γενίκευσης. Τα n-grasm μοιάζουν αρκετά με τις υπογραφές που μελετήσαμε στο κεφάλαιο του συναφούς υποβάθρου, δηλαδή για να γίνει μια σωστή ταυτοποίηση πρέπει να ταιριάξει πλήρως το n-gram στο νέο δείγμα αλλιώς μια μικρή αλλαγή στο εκτελέσιμο θα οδήγηση στην λανθασμένη ταυτοποίησή του.

3.2.3 API Κλήσεις

Οι διεπαφές προγραμματισμού εφαρμογών (API) και οι κλήσεις λειτουργιών που πραγματοποιούν θεωρούνται ως ιδιαίτερα διακριτικά χαρακτηριστικά, παρά τον εξαιρετικά μεγάλο αριθμό διαθέσιμων API. Ουσιαστικά, οι λειτουργίες API και οι κλήσεις συστήματος παρέχουν μια γέφυρα μεταξύ των εκτελέσιμων και των υπηρεσιών που παρέχονται από το λειτουργικό σύστημα, όπως η δικτύωση, η ασφάλεια, η διαχείριση αρχείων κ.ο.κ. Τα API είναι ο μόνος τρόπος για να έχουν πρόσβαση τα εκτελέσιμα μέσα σε ορισμένους πόρους του συστήμα-

τος, οπότε με την παρατήρηση ορισμένων κλήσεων API ανακτώνται πολύτιμες πληροφορίες σχετικά με τη συμπεριφορά του κακόβουλου λογισμικού.

Sami et al. (2010)[33] θέλοντας να ξεπεράσουν το προηγούμενο σύστημα από τους Ye et al.[34] ανέπτυξαν το δικό τους σύστημα τριών τμημάτων για τον εντοπισμό κακόβουλου λογισμικού. Πρώτα, έρχεται πάντα ο αναλυτής για να εξαχθούν οι κλήσεις API από το εκτελέσιμο. Σε αυτήν την εφαρμογή επέλεξαν να χρησιμοποιήσουν τον αλγόριθμο Clospan[35] για να μειώσουν τον αριθμό των περιπτώσεων κατά έναν παράγοντα της τάξης του 10. Τέλος, τροφοδοτώντας τα χαρακτηριστικά σε έναν ταξινομητή Random Forest κατάφεραν να επιτύχουν ακρίβεια 98.31% σε 34820 σύνολα δειγμάτων, από τα οποία 2951 ήταν καλοήθη.

Baldangombo et al. (2013) [36] προσέγγισαν την ανίχνευση του κακόβουλου λογισμικού με βάση τρία χαρακτηριστικά. Τις κεφαλίδες PE, τις κλήσεις API και τα DLLs. Στις πειράματα τους δοκίμασαν τους ταξινομητές NB, SVM και J48. Συνέκριναν τα αποτελέσματα των περιπτώσεων να τροφοδοτήσουν τα δεδομένα απευθείας στους ταξινομητές ή να εφαρμόσουν πρώτα τον αλγόριθμο PCA για να μειώσουν τις διαστάσεις και να εξάγουν τα σημαντικά χαρακτηριστικά. Σε ένα σύνολο 247348 δειγμάτων εκ των οποίων τα 10592 ήταν καλοήθη, ο συνδυασμός του PCA με τα χαρακτηριστικά API και τον ταξινομητή J48 έδωσαν τα καλύτερα αποτελέσματα με μια ακρίβεια 99.1%.

Singla et al. (2015) [37] δεν περιορίστηκαν απλά στην μελέτη των κλήσεων API. Αλλά ανέλυσαν όλες τις κλήσεις συναρτήσεων και opcode για να παράξουν σαν χαρακτηριστικά εισόδου τις συχνότητες με τις οποίες εμφανίζεται η κάθε κλήση ή εντολή. Αρχικά, έχοντας τα κακόβουλα λογισμικά στα χέρια τους τα σάρωσαν με ένα antivirus για να τους δώσουν τα σωστά labels. Έπειτα, εξήγαγαν από το κάθε λογισμικό τα χαρακτηριστικά συχνότητας που αναφέραμε προτού τα τροφοδοτήσουν στους ταξινομητές, MLP, Kstar, J48 και DT. Σε ένα σύνολο 1230 αρχείων, εκ των οποίων τα 430 ήταν καλοήθη, ο ταξινομητής DT είχε την καλύτερη ακρίβεια με 97.37%.

3.2.4 Εντροπία

Το πρώτο μέλημα ενός δημιουργού κακόβουλου λογισμικού είναι να παρακάμψει όλους τους στατικούς ελέγχους προκειμένου να εξαπατήσει τον χρήστη για να εκτελέσει το πρόγραμμα. Κατά τη βιβλιογραφική μας ανασκόπηση είδαμε ανιχνευτές εξαιρετικά υψηλής ακρίβειας που δεν χρησιμοποιούν παρά μόνο απλά στατικά χαρακτηριστικά. Δυστυχώς, η εξαγωγή αυτών των χαρακτηριστικών δεν είναι πάντα μια βιώσιμη επιλογή, επειδή οι επιτιθέμενοι χρησιμοποιούν μια ποικιλία μεθόδων για να αποκρύψουν τον πραγματικό σκοπό του εκτελέσιμου, "ασφάλεια" μέσω της αφάνειας (security through obscurity). Οι δύο πιο συχνά χρησιμοποιούμενες μέθοδοι είναι η συμπίεση και η κρυπτογράφηση. Έτσι, οι ερευνητές ασφαλείας μπορεί να μην είναι σε θέση να εντοπίσει ξεκάθαρο κώδικα κακόβουλου λογισμικού, αλλά εξακολουθεί να επωφελείται από τον εντοπισμό αποκρυμμένων ή συμπιεσμένων τμημάτων κώδικα. Για την απόκτηση αυτής της γνώσης χρησιμοποιήθηκε η ανάλυση της εντροπίας. Η εντροπία είναι ένα μέτρο τυχαιότητας, που σημαίνει ότι μια τιμή 0 δείχνει τον απόλυτο ντετερμινισμό, δηλαδή μόνο ένας χαρακτήρας ή μια ακολουθία επαναλαμβάνεται ξανά και ξανά στο αναλυμένο κομμάτι. Αντίθετα μια τιμή 8 δείχνει καθαρή τυχαιότητα, δηλαδή το αναλυμένο κομμάτι περιέχει εντελώς ξεχωριστούς χαρακτήρες ή ακολουθίες. Στην εργασία τους οι Lyda et al. (2007)[38] ανέλυσαν ένα σύνολο αρχείων που περιείχαν απλό κείμενο, απλά, συμπιεσμένα και κρυπτογραφημένα εκτελέσιμα και κατέληξαν στο συμπέρασμα ότι η μέση εντροπία ήταν 4.35, 5.1, 6.8 και 7.17 αντίστοιχα.

Φυσικά οι επιτιθέμενοι μελέτησαν επίσης ερευνητικά έγγραφα και προσπάθησαν να εκμεταλλευτούν περαιτέρω την απλή λογική ότι μια υψηλή εντροπία σημαίνει κρυπτογραφημένα και

συμπιεσμένα τμήματα. Για να γίνει αυτό έπρεπε να χρησιμοποιήσουν πιο έξυπνους τρόπους για να αποκρύψουν τον κακόβουλο κώδικα. Μια κοινή τακτική είναι να τοποθετηθεί στο εκτελέσιμο μια δέσμη οδηγιών "nop" για να μειωθεί σημαντικά την εντροπία του σε κομβικά σημεία του κακόβουλου λογισμικού. Παρ 'όλα αυτά, ακόμη και με τις ανανεωμένες εντολές, οι επιτιθέμενοι δεν μπορούν να κρύψουν τα μεμονωμένα τμήματα εντροπίας ενός εκτελέσιμου. Έτσι, οι ερευνητές ασφάλειας [39] άρχισαν να αναλύουν αυτό που είναι γνωστό ως δομική εντροπία ενός αρχείου. Η δομική εντροπία είναι στην πραγματικότητα η ακολουθία byte του αρχείου ως ένα ρεύμα τιμών εντροπίας, όπου κάθε τιμή υποδηλώνει την εντροπία σε ένα μικρό κομμάτι κώδικα. Με απλά λόγια, οι ερευνητές μελετούν τώρα την εντροπία των εκτελέσιμων σε μικρότερα κομμάτια/τμήματα.

Baysa et al. (2013)[40] πήγαν την έρευνα εντροπίας ένα βήμα παραπέρα και τη χρησιμοποίησαν για τον εντοπισμό μεταμορφωμένου κακόβουλου λογισμικού. Το μεταμορφικό κακόβουλο λογισμικό μπορεί να αλλάξει την εσωτερική του δομή χωρίς να αλλάξει τη λειτουργικότητά του. Συνεπώς, δεν υπάρχει μια κοινή υπογραφή σε κάποιο εξαιρετικά μεταμορφώσιμο κακόβουλο λογισμικό. Κατά συνέπεια, ένα τέτοιο κακόβουλο λογισμικό μπορεί να παραμείνει κρυμμένο ακόμη και κάτω από σαρώσεις με συνδυασμό προσομοίωσης και ανίχνευση υπογραφών. Εφαρμόζοντας ανάλυση κυματισμών (wavelet), για να εντοπιστούν περιοχές με σημαντικές αλλαγές εντροπίας και να μετρηθεί η ομοιότητα των αρχείων χρησιμοποιώντας την απόσταση Levenshtein, μπόρεσαν με δεδομένο ένα άγνωστο εκτελέσιμο αρχείο να το κατατάξουν μαζί με το πιο όμοιο δείγμα στο σύνολο εκπαίδευσης και να επιτύχουν, στις καλύτερες συνθήκες, ακρίβεια 92%.

Pham et al. (2018) [41] εμπνευσμένοι από τους δημιουργούς του dataset EMBER[42] θέλησαν να δημιουργήσουν ένα σύστημα εντοπισμού κακόβουλου λογισμικού χωρίς να γίνεται κάποια παραβίαση ιδιωτικότητας στα καλοήθη λογισμικά. Ο τρόπος για να το πετύχουν αυτό είναι να μην χρησιμοποιήσουν τα εκτελέσιμα αυτούσια αλλά να αρκεστούν σε περιγραφικά και στατιστικά δεδομένα όπως αυτά που παρέχει το dataset EMBER. Ένα από τα βασικά χαρακτηριστικά που υπολόγισαν είναι το ιστόγραμμα Byte-Εντροπία. Υπολογίζουν την εντροπία σε περιοχές του αρχείου και την συνδέουν με κάθε Byte στην περιοχή που την υπολόγισαν. Έχουν διαλέξει με τέτοιο τρόπο τις παραμέτρους τους έτσι ώστε η τελική πληροφορία να είναι η μισή της αρχικής, καταλήγοντας σε ένα διάνυσμα 256 διαστάσεων. Σε ένα σύνολο 800000 δεδομένων, εκ των οποίων τα 400000 είναι καλοήθη, ο αλγόριθμος Gradient Boosting Decision Trees έδωσε τα καλύτερα αποτελέσματα με 99.394% detection rate.

3.2.5 Αναπαράσταση κακόβουλου λογισμικού σαν εικόνες

Ο τομέας των εφαρμογών ταξινόμησης εικόνων έχει λάβει μεγάλη προσοχή σαν πεδίο έρευνας της μηχανικής μάθησης. Τα εκτελέσιμα κακόβουλα προγράμματα είναι στην πραγματικότητα μια ακολουθία byte, οπότε είναι αρκετά εύκολο να μετατραπεί ένα δυαδικό σε εικόνα ακολουθώντας έναν πολύ απλό κανόνα: κάθε byte είναι ένα μεμονωμένο pixel με τιμή που κυμαίνεται από 0 έως 255 (0: μαύρο και 255: λευκό). Μετά τον μετασχηματισμό, μπορούν να εφαρμοστούν τεχνικές επεξεργασίας εικόνας για να βοηθήσουν τη διαδικασία ανίχνευσης/ταξινόμησης κακόβουλου λογισμικού.

Nataraj et al. (2011)[43] εξήγαγαν GIST[44] χαρακτηριστικά από την αναπαράσταση του περιεχομένου κακόβουλου λογισμικού σε γκρι κλίμακα. Στη συνέχεια, χρησιμοποιώντας τον αλγόριθμο k-NN, τα αποτελέσματά τους έδειξαν ότι με $k = 3$ αυτή η μέθοδος θα μπορούσε να επιτύχει μια ακρίβεια ταξινόμησης 98.08% σε 9581 δείγματα, από τα οποία τα 123 ήταν καλοήθη.

Η τεχνική αναπαράστασης εκτελέσιμων ως εικόνες γκρι κλίμακας φυσικά δεν είναι αλεξίσφαιρη. Αντίθετα, αν κάποιος δεν είναι αρκετά προσεκτικός, θα μπορούσε να αποφέρει αποτελέσματα αντίθετα από αυτά που αναμένονται. Εξ' ορισμού ένα δυαδικό δεν είναι ένα δισδιάστατο αντικείμενο και όταν μετατραπεί σε εικόνα 2-D αναπόφευκτα θα διπλωθεί πολλές φορές για να σχηματιστεί ένας πίνακα 2-D. Αυτά τα πτυσσόμενα σημεία θα εισάγουν μια νέα, ρυθμιζόμενη, υπερ-παράμετρο, το πλάτος και το ύψος της εικόνας. Αξίζει να σημειωθεί επίσης ότι, λόγω του διπλώματος, ανύπαρκτοι χωρικοί συσχετισμοί μεταξύ διαφορετικών "pixel" μπορεί να προκύψουν και αυτό θα μπορούσε να οδηγήσει σε ψευδείς παραδοχές εάν το μέγεθος της εικόνας δεν επιλεγεί προσεκτικά. Τέλος, όπως κάθε άλλη τεχνική που ασχολείται με τα στατικά χαρακτηριστικά πάσχει από τεχνικές απόκρυψης/αλλοίωσης κώδικα. Συγκεκριμένα, τεχνικές όπως η κρυπτογράφηση και η συμπίεση μπορεί να αλλάξουν εντελώς τη δομή των byte ενός δυαδικού προγράμματος και, επομένως, οι μέθοδοι που βασίζονται σε αυτό το είδος αναπαράστασης δεν θα κατατάξουν στην σωστή κλάση το εκτελέσιμο.

3.2.6 Γραφήματα κλήσεων συναρτήσεων

Ένα γράφημα κλήσεων συναρτήσεων (Function Call Graph, FCG) είναι ένα κατευθυνόμενο γράφημα του οποίου οι κορυφές αντιπροσωπεύουν τις συναρτήσεις από τις οποίες αποτελείται ένα πρόγραμμα λογισμικού και οι άκρες συμβολίζουν τις κλήσεις συναρτήσεων. Μια κορυφή αντιπροσωπεύεται από έναν από τους ακόλουθους δύο τύπους συναρτήσεων:

1. Τοπικές συναρτήσεις, που υλοποιούνται από τον εκάστοτε προγραμματιστή για την εκτέλεση συγκεκριμένων εργασιών.

2. Εξωτερικές συναρτήσεις, που παρέχονται από το υποκείμενο λειτουργικό σύστημα ή από εξωτερικές βιβλιοθήκες.

Είναι λογικό ότι μόνο οι τοπικές συναρτήσεις έχουν τη δυνατότητα να επικαλούνται εξωτερικές συναρτήσεις και όχι το αντίστροφο. Τα γραφήματα κλήσεων συναρτήσεων μπορεί να είναι είτε δυναμικά με την έννοια ότι δημιουργήθηκαν κατά την διάρκεια της εκτέλεσης ενός λογισμικού είτε στατικά. Τα δυναμικά γραφήματα αντιπροσωπεύουν μόνο μια εκτέλεση του λογισμικού ενώ τα στατικά αποσκοπούν στην πλήρη εποπτεία, για κάθε εκτέλεση του λογισμικού. Προφανώς, τα στατικά γραφήματα είναι αυτά που παρέχουν το μεγαλύτερο ποσοστό πληροφοριών αλλά δυστυχώς δεν είναι πάντα εύκολο να δημιουργηθούν.

Hassen and Chan (2017)[45] προσπάθησαν να ξεπεράσουν ένα κρίσιμο ελάττωμα, στις τεχνικές που αξιοποιούν για χαρακτηριστικά τα γραφήματα κλήσεων συνάρτησης (FCG). Οι περισσότερες δημοσιευμένες έρευνες βασίζονται στον υπολογισμό της ομοιότητας γραφήματος μεταξύ των δειγμάτων για τη δημιουργία συστάδων, μια σπάταλη σε χρόνο και πόρους συστήματος, τεχνική υπολογισμού. Αντ' αυτού, πρότειναν μια γραμμική διαδικασία για την εξαγωγή FCG ως διανύσματα. Για τη μετατροπή ενός FCG σε ένα διάνυσμα ακολουθείται η εξής διαδικασία: (1) Εξαγωγή του FCG και επισήμανση των κορυφών με εξωτερικές συναρτήσεις με τα ονόματα των συναρτήσεων. Οι εσωτερικές συναρτήσεις αναπαρίστανται ως η ακολουθία εντολών που εκτελούνται. (2) Πέρασμα του προκύπτον FCG σε έναν αλγόριθμο ομαδοποίησης για την συγκέντρωση των τοπικών συναρτήσεων μαζί, χρησιμοποιώντας τις υπογραφές Minhash και μετονομασία με το αναγνωριστικό συμπλέγματος. (3) Μετατροπή του FCG σε ένα διάνυσμα χαρακτηριστικών με βάση τα αποτελέσματα ομαδοποίησης. Τέλος, χρησιμοποίησαν αυτά τα χαρακτηριστικά για την ταξινόμηση κακόβουλων προγραμμάτων χρησιμοποιώντας έναν ταξινομητή Random Forest και έναν Meta-Classifier σε ένα σύνολο δεδομένων 10260 δειγμάτων, με ακρίβεια 97.9%.

Rajeswaran et al. (2018) [46] παρουσίασαν μια μεθοδολογία αναγνώρισης κακόβουλου λογισμικού χρησιμοποιώντας μετρικές ομοιότητας μεταξύ των γραφημάτων κλήσεων συναρτήσεων. Προτείνουν μια μετρική που απαρτίζεται από δύο μέρη. Το πρώτο μέρος αφορά την ομοιότητα των γραφημάτων που αφορά τις εξωτερικές συναρτήσεις, και αυτό είναι κάτι εύκολο να μετρηθεί καθώς οι εξωτερικές συναρτήσεις έχουν πάντα το ίδιο όνομα οπότε η αντιστοιχία, αν υπάρχει, βρίσκεται άμεσα. Το δεύτερο μέρος αφορά τις εσωτερικές συναρτήσεις, δεδομένου ότι το όνομα δεν μπορεί να βοηθήσει σε αυτή την περίπτωση καταφεύγουν στην σύγκριση ομοιότητας μεταξύ opcodes εντός των εσωτερικών συναρτήσεων. Συνδυάζοντας τα δύο μέρη εξάγουν την τελική μετρική. Τα πειράματά τους όμως έδειξαν ότι ένας συνδυασμός του αλγορίθμου SVM με πολύ πιο απλά χαρακτηριστικά ομοιότητας μπορεί να προσφέρει πολύ καλύτερα γενικευμένα αποτελέσματα από την μετρική που πρότειναν.

3.2.7 Πίνακας σύγκρισης στατικών χαρακτηριστικών

Έχοντας καλύψει όλες τις κύριες τεχνικές για τον εντοπισμό κακόβουλου λογισμικού χρησιμοποιώντας στατικές λειτουργίες και κλασική μηχανική μάθηση, ήρθε η ώρα να παρουσιάσουμε έναν κοινό πίνακα σύγκρισης για ευκολία και γρήγορη αναφορά, αν χρειαστεί.

Δημοσίευση	Χαρακτηριστικά	Επιλογή χαρακτηριστικών Μείωση Διαστάσεων	Αλγόριθμος Μηχανικής Μάθησης
Ye et al. (2008a)	Strings	-	SVM ensemble with bagging
Markel and Bilzor (2014)	PE headers	-	NB, LR, DT
Barlam et al. (2019)	Strings	Hashing trick	SVM, LR, XGB
Santos et al. (2013)	1,2-grams	Information Gain	SVM
Boujnouni et al. (2014)	4-grams	Information Gain	SSPV-SVDD
Fuyong et al. (2017)	3-grams	Information Gain	1-NN
Raff et al. (2019)	n-grams	hash-grams and Zipf	-
Yuxin et al. (2019)	n-grams	-	DBN, SVM, K-NN, DT
Sami et al. (2010)	API Calls	Fisher Score + Clospan Algorithm	RF
Baldangombo et al. (2013)	API Calls	PCA	NB, SVM, J48
Singla et al. (2015)	API + Opcodes	-	MLP, Kstar, J48, DT
Baysa et al. (2013)	Structural Entropy	Discrete Wavelet Transform	Sequence Similarity
Pham et al. (2018)	Byte-Entropy	-	GBDT
Nataraj et al. (2011)	Gray Scale IMG	GIST Features	K-NN
Hassen and Chan (2017)	Function Call Graph	Vector representation algorithm	RF, meta-classifier
Rajeswaran et al. (2018)	Function Call Graph	-	-

Πίνακας 1: Αλγόριθμοι: Support Vector Machine (SVM), Random Forests (RF), Naive Bayes (NB), Artificial Neural Networks (ANN), Decision Trees (DT), Instance-based Learner (IL), K-Nearest Neighbor (K-NN), Logistic Regression (LR), Sequential Minimal Optimization (SMO), Voted Perceptron (VT)

Δημοσίευση	Πηγή Δεδομένων	Συνολικά Δείγματα	Αναλογία Δειγμάτων (B/M)	Στόχος	Καλύτερη επίδοση
Ye et al. (2008a)	Kingsoft Lab	39838	20.88%	Detection	SVM, 92.3% F1
Markel and Bilzor (2014)	OpenMalware	25000	Variable	Detection	DT, variable
Barlam et al. (2019)	APT1	1416	69.8%	Detection	LR+XGB, 98% acc.
Santos et al. (2013)	VXHeavens, Windows OS	18000	5.55%	Detection	SVM, 95.9% acc.
Boujnouni et al. (2014)	VxHeaven	1258	47.6%	Detection	SSPV-SVDD, 94% acc.
Fuyong et al. (2017)	Open Malware Benchmark, Windows Os	2540	49.29%	Detection	1-NN
Raff et al. (2019	VirusShare	200000	-	Classification	-
Yuxin et al. (2019)	-	9200 + 2000 unlabeled	50%	Detection	DT, 96.5% acc.
Sami et al. (2010)	-	34820	8.47%	Detection	RF, 98.31% acc.
Baldangombo et al. (2013)	Various Sites	247348	4%	Detection	J48, 99.1% acc.
Singla et al. (2015))	Various Sites	1230	35%	Detection	DT, 97.37% acc.
Baysa et al. (2013)	-	-	-	-	Similarity, 92% acc.
Pham et al. (2018)	EMBER	800000	50%	Classification	GBDT, 99.394% TPR
Nataraj et al. (2011)	-	9581	1.28%	Classification	3-NN, 98.08% acc.
Hassen and Chan (2017)	Microsoft Malware Classification Challenge	10260	0%	Classification	RF, 97.9% acc.
Rajeswaran et al. (2018)	Malicia	-	-	Detection	SVM

Πίνακας 2: Μια συνολική σύγκριση των αποτελεσμάτων και των χαρακτηριστικών που επιλέχθηκαν από τα σύνολα δεδομένων

Αλλά ο πίνακας 1 λέει μόνο τη μισή αλήθεια για τις μεθόδους που ελέγχθηκαν. Για μια πλήρη, συμπαγή, επισκόπηση χρειαζόμαστε τον πίνακα 2 που συγκρίνει τα σύνολα δεδομένων που χρησιμοποιούνται από κάθε ερευνητικό έγγραφο και τα αποτελέσματα που έδωσε κάθε έρευνα. Από την ανάλυση στατικών χαρακτηριστικών μπορούμε να δηλώσουμε με ασφάλεια ότι τα στατικά χαρακτηριστικά παρέχουν αρκετές πληροφορίες για να αποφασίσουμε αν ένα εκτελέσιμο είναι κακόβουλο ή όχι, όπως φαίνεται ξεκάθαρα από τις τιμές υψηλής ακρίβειας. Αυτό που πραγματικά ξεχωρίζει όμως, είναι η μεγάλη ανισορροπία που υπάρχει ανάμεσα στις κλάσεις σε μερικά από τα αναφερόμενα δημοσιευμένα άρθρα.

3.3 Δυναμικά Χαρακτηριστικά

Για να συμπληρώσει τα στατικά χαρακτηριστικά, η ανάλυση κακόβουλου λογισμικού βασίζεται μερικές φορές στα λεγόμενα "δυναμικά χαρακτηριστικά". Τα δυναμικά χαρακτηριστικά εξάγονται κατά την εκτέλεση ενός προγράμματος. Η δυναμική ανάλυση περιλαμβάνει την παρακολούθηση του εκτελέσιμου κατά τη διάρκεια του χρόνου εκτέλεσης ή την εξέταση τυχόν αλλαγών στο σύστημα μετά την εκτέλεση του προγράμματος. Έτσι μπορεί να αποκαλυφθεί, η δημιουργία υπο-διαδικασιών (μια κοινή τεχνική απόκρυψης), ο χειρισμός αρχείων και άλλων ενεργειών που πιθανόν να καταδεικνύουν την παρουσία κακόβουλου λογισμικού.

3.3.1 Εξέταση μνήμης και καταχωρητών

Όλες οι τεχνικές απόκρυψης/αλλοίωσης αργά ή γρήγορα πρέπει να αποκαλύψουν το κακόβουλο ωφέλιμο φορτίο του εκτελέσιμου, καθώς η CPU μπορεί να εκτελέσει μόνο μη κρυπτογραφημένες/αλλοιωμένες εντολές, προκειμένου να μολύνει το σύστημα. Έτσι, η συμπεριφορά ενός εκτελέσιμου μπορεί να καθοριστεί από τη μνήμη και το περιεχόμενο των καταχωρητών κατά τη διάρκεια του χρόνου εκτέλεσης και τελικά να διακρίνει μεταξύ καλοήθων και κακόβουλων προγραμμάτων.

Ghiasi et al. (2015) [47] επέκτειναν την προηγούμενη εργασία τους [48] εισάγοντας ένα νέο βήμα στη διαδικασία τους. Το νέο βήμα είναι στην πραγματικότητα μια τεχνική μείωσης χαρακτηριστικών όπου παρουσιάστηκαν τα λεγόμενα "Πρωτότυπα". Τα πρωτότυπα είναι ένα μικρό σύνολο αρχείων που αντιπροσωπεύουν ολόκληρο το σύνολο δεδομένων, τα οποία παρέχουν μια αποδεκτή προσέγγιση σε ανάλυση απόσταση ομοιότητας κατά ζεύγη. Με άλλα λόγια, εάν πολλές αναφορές είναι παρόμοιες, δημιουργείται ένα πρωτότυπο για να τα αντικαταστήσει. Σε ένα σύνολο δεδομένων 1240 δειγμάτων, εκ των οποίων 390 ήταν καλοήθεις, κατάφεραν να επιτύχουν ακρίβεια 93% με το 20% των αρχείων να λειτουργούν ως πρωτότυπα. Μια μικρή πτώση της ακρίβειας με αντάλλαγμα την απόδοση και την ταχύτητα.

Fraley et al. (2016) [49] επικεντρώθηκαν στην εργασία τους στον εντοπισμό κακόβουλου λογισμικού που αλλάζει μορφή κατά την διάρκεια της εκτέλεσης. Προφανώς, επειδή το εκτέλεσιμο μεταβάλει την μορφή του κατά την διάρκεια της εκτέλεσης είναι απαραίτητη η συλλογή και στατικών χαρακτηριστικών πριν την εκτέλεση αλλά κυρίως δυναμικών χαρακτηριστικών κατά την διάρκεια της εκτέλεσης. Με την βοήθεια του αλγορίθμο K-means σε ένα σύνολο 3637 δεδομένων, εκ των οποίων τα 2400 είναι καλοήθη, πέτυχαν μια ακρίβεια της τάξης του 99%.

3.3.2 Ίχνη Εντολών

Ένα δυναμικό ίχνος εντολών (dynamic instruction trace) είναι μια ακολουθία οδηγιών επεξεργαστή που καλείται κατά την εκτέλεση ενός προγράμματος. Σε αντίθεση με το στατικό

ίχνος εντολών, τα δυναμικά ίχνη διατάσσονται με την σειρά εκτελούνται ενώ τα στατικά διατάσσονται όπως εμφανίζονται στο δυαδικό αρχείο. Τα δυναμικά ίχνη εντολών παρέχουν το μεγάλο πλεονέκτημα ότι όταν δημιουργούνται, εκτός του ότι παρέχουν τις εντολές με την σειρά που εκτελέστηκαν, τις παρέχουν και αποκρυπτογραφημένες. Δηλαδή, οι συνηθισμένες τεχνικές απόκρυψης των στατικών εντολών παύουν να ισχύουν.

Storlie et al. (2014) [50] εισήγαγαν μια διαδικασία που αντιπροσωπεύει ένα ίχνος εντολών με δομή αλυσίδας Markov στην οποία ο πίνακας μετάβασης, **P**, έχει σειρές που διαμορφώνονται ως διανύσματα Dirichlet. Η κατηγορία λογισμικού (κακόβουλη ή καλοήθης) διαμορφώνεται χρησιμοποιώντας ένα εύκαμπτο μοντέλο flexible spline logistic regression με μεταβλητή επιλογή στα στοιχεία του P. Χρησιμοποιώντας τη μέθοδο κανονικοποίησης Elastic Net σε ένα μοντέλο λογιστικής παλινδρόμησης κατάφεραν να επιτύχουν ακρίβεια 97% σε δείγματα 21988 , εκ των οποίων οι 3046 ήταν καλοήθεις.

O'kane et al. (2016) [51] διερεύνησε το βέλτιστο σύνολο λειτουργικών εντολών (opcodes) που δημιουργούν έναν ισχυρό δείκτη κακόβουλου λογισμικού και τη βέλτιστη διάρκεια του χρόνου εκτέλεσης του προγράμματος για την ακριβή ταξινόμηση καλοήθων και κακόβουλων αρχείων. Τροφοδοτώντας τα συνδυασμένα ιστογράμματα (χρόνος, opcodes και διάρκεια προγράμματος), μετά τη χρήση του PCA για τη μείωση του υπολογιστικού κόστους, σε ένα SVM πέτυχαν ακρίβεια 86.3% σε ένα σύνολο δεδομένων που αποτελούνταν από 650 δείγματα από τα οποία 300 καλοήθη αρχεία. Αξίζει να αναφερθεί ότι τα αποτελέσματα ποικίλλουν σημαντικά ως προς τη διάρκεια του προγράμματος. Παρόλο που η υψηλότερη ακρίβεια επιτυγχάνεται σε μεγάλα προγράμματα 32k-opcodes, μικρότερα, περίπου 1k-opcodes, έδωσαν πολύ πιο συνεπή και αξιόπιστα αποτελέσματα.

Carlin et al. (2017) [52] παρουσίασαν μια προσέγγιση που εξάγει τα ίχνη εντολών από εικονικές μηχανές μετά την εκτέλεση καλοήθους ή κακόβουλου λογισμικού. Αναλύοντας την ακολουθία των opcodes, δοκίμασαν δύο αλγόριθμους προκειμένου να εντοπίσουν επιτυχώς τις περιπτώσεις κακόβουλου λογισμικού. Πρώτον, δοκίμασαν έναν ταξινομητή Random Forest για να ταξινομήσουν όλα τα δεδομένα που βασίζονται στην καταμέτρηση. Δεύτερον, δοκίμασαν ένα μοντέλο Hidden Markov για να ταξινομήσουν δεδομένα με βάση τις χρονικές σχέσεις στις ακολουθίες των opcodes. Σε ένα σύνολο δεδομένων 1000000 δειγμάτων, ο ταξινομητής Random Forest ξεπέρασε το μοντέλο HMM σε κάθε σενάριο, επιτυγχάνοντας μια ακρίβεια 98.4%. Αν και οι συγγραφείς δηλώνουν ότι αυτή η ακρίβεια μπορεί να είναι παραπλανητική λόγω μεγέθους δείγματος ή προβλημάτων ανισορροπίας κλάσεων.

3.3.3 Network Traffic

Με τις αυξανόμενες αναφορές περιστατικών ransomware είναι σαφές ότι η μόλυνση ενός συστήματος δεν είναι ο τελικός στόχος ενός κακόβουλου λογισμικού. Αντ' αυτού, μετά από μια επιτυχή μόλυνση συνδέεται με έναν εξωτερικό διακομιστή για να λάβει τις εντολές που θα εκτελέσει στο προσβεβλημένο μηχάνημα ή για να διαρρεύσει ευαίσθητες πληροφορίες σχετικά με τον μολυσμένο χρήστη/μηχάνημα. Ως αποτέλεσμα, η παρακολούθηση της κίνησης του δικτύου που εισέρχεται ή εξέρχεται από το προσβεβλημένο μηχάνημα θα μπορούσε να αποκαλύψει σημαντικές πληροφορίες για τον εντοπισμό κακόβουλης συμπεριφοράς.

Bekerman et al. [53] παρουσίασαν ένα σύστημα που μπορεί να εντοπίσει κακόβουλο λογισμικό με βάση την ανάλυση δικτύου. Στην εργασία τους, εξήγαγαν 972 χαρακτηριστικά από την ανάλυση της κίνησης του δικτύου στο Διαδίκτυο, τα επίπεδα μεταφοράς και εφαρμογών. Στη συνέχεια, ένα υποσύνολο των χαρακτηριστικών επιλέχθηκε χρησιμοποιώντας τον αλγόριθμο επιλογής χαρακτηριστικών συσχέτισης [54]. Στη συνέχεια, τα χαρακτηρισ-

τικά που προέκυψαν χρησιμοποιήθηκαν για τον έλεγχο τριών διαφορετικών αλγορίθμων ταξινόμησης, συμπεριλαμβανομένων των Naïve Bayes, του Decision Tree (J48) και του Random Forest. Σε ένα σύνολο δεδομένων 50720 εγγραφών, ο ταξινομητής Random Forest απέδωσε καλύτερα με μια ακρίβεια 98% στις περισσότερες οικογένειες κακόβουλων προγραμμάτων.

Zhao et al. (2015) [55] πρότειναν ένα σύστημα εντοπισμού κακόβουλου λογισμικού APT (Advance Persistent Threat) χρησιμοποιώντας DNS και ανάλυση κίνησης δικτύου. Από τη μία πλευρά, ο κακόβουλος ανιχνευτής DNS εξάγει 14 χαρακτηριστικά που βασίζονται σε μεγάλα δεδομένα για να χαρακτηρίσουν διαφορετικές ιδιότητες DNS ερωτημάτων (DNS queries) που σχετίζονται με κακόβουλο λογισμικό. Από την άλλη πλευρά, ο αναλυτής κίνησης δικτύου συνδυάζει ένα σύστημα βασισμένο σε αναζήτηση υπογραφών και ένα σύστημα αναζήτησης ανωμαλιών, το οποίο ανιχνεύει προσβολές κακόβουλου λογισμικού με βάση την ακρίβεια των κανόνων από τα σύνολα VRT (τώρα ονομάζεται "Talos") του Snort και ανωμαλίες που συμβαίνουν σε επίπεδο πρωτοκόλλου και λογισμικού, αντίστοιχα. Στη συνέχεια, ένα δέντρο απόφασης J48 ταξινομεί την απειλή. Η ομάδα του Zhao εξέτασε 400,000,000 εγγραφές και πέτυχε true positive rate 95.8%.

Banin et al. (2016) [56] ανέπτυξαν ένα σύστημα εντοπισμού κακόβουλου λογισμικού που βασίζεται σε ανίχνευση ομοιοτήτων προσπέλασης μνήμης. Συγκεκριμένα, η μεθοδολογία που ακολούθησαν ήταν να εκτελούν το κακόβουλο λογισμικό εντός εικονικών μηχανών και να καταγράφουν με την βοήθεια n-grams με n 16,20,24,36,48,72,96, ακολουθίες μεγέθους 100,000 ή 1,000,000 ή 10,000,000 από προσπελάσεις που εκτέλεσε το λογισμικό στην μνήμη κατά την διάρκεια της εκτέλεσής του. Το τελικό διάνυσμα που τροφοδοτεί τον αλγόριθμο έχει διάσταση 800, δηλαδή εμπεριέχει 800 ακολουθίες. Τα καλύτερα αποτελέσματα σε ένα σύνολο 1204 δεδομένων εκ των οποίων τα 445 ήταν καλοήθη τα έδωσε ο συνδυασμός: 96-grams, 1,000,000 μέγεθος ακολουθιών και ο αλγόριθμος k-NN σαν ταξινομητής. Ο συγκεκριμένος συνδυασμός έδωσε ακρίβεια 98.92%.

Boukhtouta et al. (2016) [57] εισήγαγαν ένα σύστημα ανίχνευσης και ταξινόμησης κακόβουλου λογισμικού που βασίζεται στην ταξινόμηση Deep Packet Inspection (DPI) και IP κεφαλίδων από τα πακέτα. Εκτέλεσαν κάθε δείγμα κακόβουλου λογισμικού σε ένα sandbox για 3 λεπτά για να δημιουργήσουν αντιπροσωπευτική κακόβουλη κίνηση δικτύου. Στη συνέχεια, εξήχθησαν πολλά χαρακτηριστικά διπλής κατεύθυνσης, όπως ο αριθμός των πακέτων προς τα εμπρός και προς τα πίσω, ο μέγιστος και ο ελάχιστος χρόνος μεταξύ των αφίξεων για τα πακέτα προώθησης, το μέγεθος του πακέτου κλπ. Τα συνολικά χαρακτηριστικά τροφοδοτήθηκαν στους ακόλουθους ταξινομητές: J48, Naïve Bayes, Boosted Naïve Bayes και SVM, τα οποία εντόπισαν αν η κίνηση ήταν κακόβουλη ή όχι. Μετά από μια επιτυχημένη ανίχνευση κακόβουλου λογισμικού, τα Hidden Markov Models (HMMs) δημιούργησαν μη-ντετερμινιστικά μοντέλα που χαρακτήρισαν οικογένειες κακόβουλου λογισμικού, χρησιμοποιώντας ροές μονής κατεύθυνσης, σαν ένα σύνολο 45 χαρακτηριστικών. Η εργασία τους έδειξε ότι ο καλύτερος αλγόριθμος ήταν ο Boosted J48 με ρυθμό ανίχνευσης στο 99%.

Hoang et al. (2018) [58] προσπάθησαν να δημιουργήσουν ένα σύστημα εντοπισμού botnet. Τα Botnet είναι και αυτά είδος κακόβουλου λογισμικού και επικοινωνούν διαρκώς με έναν απομακρυσμένο διακομιστή για να λάβουν εντολές που θα εκτελέσουν στο προσβεβλημένο σύστημα. Επειδή επικοινωνούν συνέχεια μέσω του διαδικτύου, δημιουργούν μεγάλη κίνηση σε αιτήματα DNS. Διαβάζοντας τα αιτήματα DNS και εξάγοντας 18 διαφορετικά χαρακτηριστικά, που αφορούν κυρίως την γλωσσική μορφολογία του αιτήματος, ο ταξινομητής RandomForest είχε την καλύτερη επίδοση με ακρίβεια άνω του 88% σε όλα τα πειράματα που εκτέλεσαν σε ένα σύνολο 60000 δεδομένων από τα οποία τα 30000 ήταν καλοήθη.

3.3.4 Ίχνη κλήσεων API

Για άλλη μια φορά οι κλήσεις API μπορούν να δώσουν πολύτιμες πληροφορίες σχετικά με τη συμπεριφορά κακόβουλων εφαρμογών. Αυτή την φορά δεν συγκεντρώνονται όλες οι κλήσεις API που είναι προγραμματισμένες εντός του κακόβουλου λογισμικού αλλά μόνο εκείνες οι οποίες καταλήγουν όντως να εκτελεστούν.

Ravi et al. (2012) [59] εργάστηκαν πάνω στον εντοπισμό κακόβουλου λογισμικού σε λειτουργικά συστήματα Windows με την χρήση κλήσεων API κατά την διάρκεια της εκτέλεσης του λογισμικού. Από την παρακολούθηση των κλήσεων API δημιούργησαν 4-grams, τα οποία τα τροφοδότησαν σε έναν αλγόριθμο δημιουργίας κανόνων. Στην συνέχεια ένας ταξινομητής λαμβάνει μια ακολουθία API κλήσεων που χαρακτηρίζουν το εκτελέσιμο και σε συνδυασμό με τους κανόνες που έχουν εξαχθεί κατηγοριοποιεί το εκτελέσιμο σε κακόβουλο ή μη. Σε ένα σύνολο 273 δειγμάτων, εκ των οποίων τα 94 είναι καλοήθη, η μέθοδος τους πέτυχε μια ακρίβεια της τάξης του 90%.

Uppal et al. (2014) [60] παρουσίασαν μια προσέγγιση αναγνώρισης κακόβουλου λογισμικού βασισμένη σε χαρακτηριστικά από τις ακολουθίες API. Η μέθοδος παρακολουθεί την εκτέλεση ενός προγράμματος για να καταγράφει τις κλήσεις API που επικαλούνται. Στη συνέχεια, δημιουργούνται grams κλήσης API και υπολογίζεται ο λόγος πιθανότητας κάθε gram. Αυτός ο λόγος πιθανότητας χρησιμοποιείται για την κατάταξη των χαρακτηριστικών και την επιλογή των κορυφαίων χαρακτηριστικών για να σχηματίσει το διάνυσμα χαρακτηριστικών. Για την ταξινόμηση, προτάθηκαν διάφοροι αλγόριθμοι συμπεριλαμβανομένων των Naïve Bayes, Random Forest, Decision Tree και Support Vector Machine. Η αξιολόγηση της προσέγγισής τους πραγματοποιήθηκε σε ένα σύνολο δεδομένων σε 270 δυαδικά αρχεία που ελήφθησαν από το VXHeavens με τον ταξινομητή SVM να επιτυγχάνει ακρίβεια 98.5%.

Fan et al. (2015) [61] επικεντρώθηκαν στον εντοπισμό κακόβουλου λογισμικού με την βοήθεια χαρακτηριστικών που εξήχθησαν από την εκτέλεση λογισμικού εντός εικονικών μηχανών. Συγκεκριμένα, κατά την εκτέλεση των λογισμικών παρακολουθούν και καταγράφουν τις κλήσεις αλλά και τον αριθμό κλήσεων σε API. Επίσης, έχουν ρυθμίσει τα εργαλεία τους με τέτοιον τρόπο ώστε να καταγράφουν και κλήσεις API από διεργασίες που δημιουργήθηκαν μέσω της κλήσης του εκτελέσιμου. Είναι πολύ συνηθισμένο φαινόμενο κακόβουλο λογισμικό να δημιουργεί υπο-διεργασίες για να κρύψει την πραγματική του πρόθεση. Σε ένα σύνολο 7251 δειγμάτων εκ των οποίων τα 251 ήταν καλοήθη, ο ταξινομητής J48 είχε την καλύτερη επίδοση με ακρίβεια 95.9%.

Galal et al. (2016) [62] συγκέντρωσαν και επεξεργάστηκαν ακατέργαστες πληροφορίες που δημιουργήθηκαν από API hooking call, προκειμένου να δημιουργήσουν ένα σύνολο ενεργειών που περιγράφουν κακόβουλες συμπεριφορές λογισμικού. Μια ενέργεια είναι ένα αντιπροσωπευτικό σημασιολογικό χαρακτηριστικό που συνάγεται από τις ακολουθίες κλήσεων API χρησιμοποιώντας ένα σύνολο ευρετικών συναρτήσεων. Τέλος, η βιωσιμότητα των ενεργειών αξιολογήθηκε από διάφορους αλγορίθμους ταξινόμησης όπως Decision Trees, Random Forests και Support Vector Machines. Σε ένα σύνολο δεδομένων 4000 δειγμάτων, εκ των οποίων τα 2000 ήταν καλοήθεις, ο αλγόριθμος του Decision Tree ξεπέρασε τους άλλους επιτυγχάνοντας ακρίβεια 97.19%.

Damodaran et al. (2017) [63] επικεντρώθηκαν στην ταξινόμηση κακόβουλου λογισμικού σε οικογένειες κακόβουλου λογισμικού. Συγκεκριμένα είχαν 6 κατηγορίες στις οποίες ένα δείγμα από τις δοκιμές τους μπορούσε να ενταχθεί. Τα χαρακτηριστικά που χρησιμοποίησαν ήταν κλήσεις API κατά την διάρκεια εκτέλεσης των λογισμικών από ένα Sandbox. Σε ένα σύνολο 745 δειγμάτων, τα εκπαιδευμένα Hidden Markov Models - HMMs ήταν σε θέση να

πετύχουν ακρίβεια άνω του 97% για όλες τις επιμέρους κατηγορίες.

Salehi et al. (2017) [64] ανέπτυξαν ένα "εσωτερικό" εργαλείο που αποτελείται από μια εικονική μηχανή, ένα εργαλείο πρόσδεσης (hooking) και ένα σύστημα καταγραφής, το οποίο χρησιμοποιήθηκε για την ανάλυση των δυαδικών αρχείων και την παρακολούθηση της συμπεριφοράς τους. Η προσέγγισή τους βασίζεται στην υπόθεση ότι τα ονόματα των API από μόνα τους μπορεί να μην αντιπροσωπεύουν την πρόθεση των λειτουργιών που εκτελεί η συνάρτηση. Για το λόγο αυτό, η ρύθμιση των χαρακτηριστικών κακόβουλων και καλοήθων συμπεριφορών δημιουργήθηκε χρησιμοποιώντας τις κλήσεις API, τα ορίσματα εισόδου τους και τις τιμές επιστροφής. Στη συνέχεια, το σύνολο χαρακτηριστικών μειώθηκε μέσω μιας διαδικασίας δύο σταδίων. Στο πρώτο στάδιο, η βαθμολογία Fisher εφαρμόστηκε για την επιλογή των χαρακτηριστικών με την μεγαλύτερη διακριτική ικανότητα. Στο δεύτερο στάδιο, ένα Support Vector Machine βασισμένο στην μέθοδο "Recursive Feature Elimination" μείωσε ακόμη περισσότερο το σύνολο χαρακτηριστικών. Στη συνέχεια, το σύνολο χαρακτηριστικών που δημιουργήθηκε χρησιμοποιήθηκε ως είσοδος στους αλγόριθμους ταξινόμησης Random Forest, J48, Sequential minimal optimization και Bayesian logistics regression. Σε ένα σύνολο δεδομένων 4386 δειγμάτων, ο ταξινομητής J48 πέτυχε 96.3% σε F1-Score.

3.3.5 Πίνακας σύγκρισης δυναμικών χαρακτηριστικών

Δημοσίευση	Χαρακτηριστικό	Επιλογή Χαρακτηριστικών Μείωση Διαστάσεων	Αλγόριθμος Μηχανικής Μάθησης
Ghiasi et al. (2015)	Register's Usage	Prototype Extraction	Matching based on Jaccard's Similarity
Fraley et al. (2016)	Memmory Usage	-	K-means
Storlie et al. (2014)	Instruction Traces	-	Flexible spline logistic regression
O' kane et al. (2016)	Instruction Traces	PCA	SVM
Carlin et al. (2017)	Instruction Traces	Opcode counts	RF, Hidden Markov Model
Bekerman et al. (2015)	Network Traffic	Correlation Feature Selection	Naive Bayes, J48 DT, RF
Zhao et al. (2015)	DNS and Network Traffic	-	Reputation Engine
Banin et al. (2016)	DNS Traffic	-	k-NN
Boukhtouta et al. (2016)	Network Traffic	-	Boosted J48, j48, NB, Boosted NB, SVM, HMMs
Hoang et al. (2018)	DNS Traffic	Linguistic Statistics	RF
Ravi et al. (2012)	API Call Traces	-	Rule based
Uppal et al. (2014)	API Call Traces	Odds ratio	NB, RF, DT, SVM
Fan et al. (2014)	API Call Traces	-	J48
Galal et al. (2016)	API Call Traces	Hand-Crafted Heuristics	DT, RF, SVM
Damodaran et al. (2017)	API Call Traces	-	HMMs
Salehi et al. (2017)	API Call Traces	Fisher Score, SVM base on Recursive Feature Elimination	RF, J48 DT, Bayesian Logistic Regression, Sequential Minimal Optimization

Πίνακας 3: Αλγόριθμοι: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Sequential Minimal Optimization

Δημοσίευση	Πηγή Δεδομένων	Συνολικά Δείγματα	Αναλογία Δειγμάτων (B/M)	Στόχος	Καλύτερη Επίδοση
Ghiasi et al. (2015)	-	1240	31.5%	Detection	Matching, 93% acc.
Fraley et al. (2016)	ClamAV, VirusTotal	3637	66%	Detection	K-means, 99% acc.
Storlie et al. (2014)	Offensive Computing Repository	21988	13.9%	Detection	Logistic Regression, 97% acc.
O' kane et al. (2016)	-	650	46.2%	Detection	SVM, 86.3% acc.
Carlin et al. (2017)	VirusShare	1000000	-	Detection	Random Forest, 98.4% acc.
Bekerman et al. (2015)	Vering, Emerging Threats	50720 (records)	-	Classification	Random Forest, 98% acc.
Zhao et al. (2015)	Alexa's TOP 1000 sites	4000000 (records)	-	Detection	J48 Decision Tree, 95.8% TPR
Banin et al. (2016)	VirusShare	1204	37%	Detection	k-NN, 98.92% acc.
Boukhtouta et al. (2016)	-	-	-	Detection, Classification	Boosted J48, 99% detection rate
Hoang et al. (2018)	Self Created	60000	50%	Detection	RF, 88% acc.
Ravi et al. (2012)	VXHeavens	273	34%	Detection	Rule based classifier, 90% acc.
Uppal et al. (2014)	VXHeavens	270	-	Detection	SVM, 98.5% acc.
Fan et al. (2014)	Various internet sites	7251	3%	Detection	J48, 95.9% acc
Galal et al. (2016)	VirusSign, Windows 7	4000	50%	Detection	Decision Tree, 97.19% acc.
Damodaran et al. (2017)	Self Created	745	-	Classification	HMMs, 97% acc.
Salehi et al. (2017)	Windows XP	4368	-	Detection	J48, 96.3% F1-Score

Πίνακας 4: Μια συνολική σύγκριση των αποτελεσμάτων και των χαρακτηριστικών που επιλέχθηκαν από τα σύνολα δεδομένων για τις δυναμικές μεθόδους

Οι πίνακες 3 και 4 παρέχουν μια πλήρη και συμπαγή επισκόπηση των εργασιών που αναφέρονται σε αυτήν την υποενότητα. Θα πρέπει να είναι προφανές ότι η δυναμική ανάλυση ενός κακόβουλου λογισμικού μπορεί επίσης να προσφέρει εξαιρετικά πολύτιμα χαρακτηριστικά για να βοηθήσει τη διαδικασία ανίχνευσης/ταξινόμησης. Επιπλέον, κάθε ερευνητικό έγγραφο που αναφέρθηκε και έδωσε κάποια αποτελέσματα δοκιμών δείχνει ότι η χρήση δυναμικών χαρακτηριστικών μπορεί να αποφέρει εξαιρετικά καλά αποτελέσματα όσον αφορά την απόδοση ανίχνευσης καθώς βασίζεται, μερικώς, στο γεγονός ότι η δυναμική ανάλυση παρακολουθεί το εκτελέσιμο κατά τη διάρκεια του χρόνου εκτέλεσης και αυτό σημαίνει ότι οι περισσότερες τεχνικές συσκότισης/απόκρυψης δεν εφαρμόζονται πλέον. Αλλά, τίποτα δεν τέλειο από μόνο του, η δυναμική ανάλυση είναι πολύ πιο επικίνδυνη από τη στατική ανάλυση και ορισμένες μέθοδοι απαιτούσαν εκατομμύρια δείγματα να υποβληθούν σε επεξεργασία προκειμένου να επιτευχθεί το επιθυμητό αποτέλεσμα. Τέλος, ένα ακόμη περίεργο γεγονός που πρέπει να σημειωθεί είναι ότι πολλές εργασίες δεν ανέφεραν την αναλογία δειγμάτων τους (Καλοήθη/Κακόβουλο λογισμικό). Αντίθετα, ανέφεραν απλώς το συνολικό μέγεθος του συνόλου δεδομένων τους.

3.4 Προσεγγίσεις βαθιάς μάθησης

Μέχρι στιγμής μελετήσαμε τις κλασικές προσεγγίσεις μηχανικής μάθησης που βασίζονται σε μεγάλο βαθμό σε προσεκτικά επιλεγμένα χαρακτηριστικά βασισμένα σε ειδικές γνώσεις επί του τομέα. Τα χαρακτηριστικά πάντα σκοπεύουν να παρουσιάσουν μια πιο αφηρημένη οπτική των εκτελέσιμων αρχείων έτσι ώστε οι διάφοροι ταξινομητές να μπορούν να τα επεξεργαστούν και να εξάγουν γενικευμένα αποτελέσματα. Δυστυχώς, επειδή οι προηγούμενες μέθοδοι που αναφέραμε βασίζονται εξαιρετικά πολύ σε προ υπάρχουσες γνώσεις είναι τρομερά χρονοβόρο να επιλέγονται και να διαμορφώνονται τα κατάλληλα χαρακτηριστικά για την εκάστοτε εφαρμογή. Η ανάπτυξη που έχει γνωρίσει ο τομέας της βαθιάς μηχανικής μάθησης τείνει να αντικαταστήσει την διαδικασία επιλογής και διαμόρφωσης χαρακτηριστικών με δομές που έχουν σαν είσοδο την ακατέργαστη πληροφορία, αποφασίζουν μόνα τους ποια είναι τα ωφέλιμα χαρακτηριστικά και παράγουν την επιθυμητή έξοδο.

3.4.1 Αναπαράσταση Διανυσματικών Χαρακτηριστικών

Η αναπαράσταση διανυσματικών χαρακτηριστικών είναι μια τεχνική που έχουμε ήδη μελετήσει σε προηγούμενες υποενότητες. Αναφερόμαστε στην διαδικασία εξαγωγής ενός συνόλου χαρακτηριστικών που δίνουν μια αφηρημένη εικόνα ενός εκτελέσιμου.

Azeez et al. (2021) [65] πρότειναν μια μέθοδο βασισμένη σε συλλογική μάθηση. Αρχικά, μείωσαν τις διαστάσεις του αρχικού συνόλου με την χρήση του αλγορίθμου PCA σε 55 από 77. Στην συνέχεια εκτελούν διεξοδικές δοκιμασίες για την ανεύρεση των καλύτερων ταξινομητών αλλά και του καλύτερου τελικού ταξινομητή ο οποίος θα πάρει τα επιμέρους αποτελέσματα των ταξινομητών και θα δώσει την τελική κατηγοριοποίηση. Σε ένα σύνολο 19611 δειγμάτων εκ των οποίων τα 5012 είναι καλοήθη, η καλύτερη μέθοδος που είχαν ήταν ο συνδυασμός τριών διαφορετικών, από άποψη αρχιτεκτονικής, νευρωνικών δικτύων για 7 πρώτους ταξινομητές και τον ταξινομητή ExtraTrees για τελικό, επιτυγχάνοντας 100% ακρίβεια στο σύνολο των δεδομένων.

Damaševičius et al. (2021) [66] πειραματίστηκαν επίσης με την ιδέα της συλλογικής μάθησης. Τα δεδομένα τους ήταν πληροφορίες από την κεφαλίδα διάφορων PE αρχείων και κατέληξαν σε ένα διάνυσμα 68 χαρακτηριστικών. Έπειτα με την εφαρμογή του αλγορίθμου PCA μειώθηκαν οι διαστάσεις του διανύσματος σε 40. Τα καλύτερα αποτελέσματα για τους επιμέρους ταξινομητές τα έδωσε ένας συνδυασμός από 5 πλήρως συνδεδεμένα δίκτυα και CNN, ενώ για τελικός ταξινομητής δοκιμάστηκαν 13 ταξινομητές μηχανικής μάθησης με τον αλγόριθμο ExtraTrees να φέρνει άλλη μια φορά τα καλύτερα αποτελέσματα. Σε ένα σύνολο 5210 δειγμάτων, εκ των οποίων τα 2488 είναι καλοήθη, η συλλογική μέθοδος είχε μια ακρίβεια 99.9%. Τα αποτελέσματα αλλά και η προσέγγιση μοιάζουν εξαιρετικά πολύ με την εργασία των Azeez et al. (2021) [65], γεγονός που δείχνει ότι ο συνδυασμός νευρωνικών δικτύων και του αλγορίθμου ExtraTrees μπορεί να δώσει εκπληκτικά αποτελέσματα.

Rizvi et al. (2021) [67] εισήγαγαν μια μη επιβλεπόμενη μέθοδο για τον εντοπισμό κακόβουλου λογισμικού. Αρχικά, εξήγαγαν τα χαρακτηριστικά που χρησιμοποίησαν από τις κεφαλίδες των PE αρχείων, καταλήγοντας σε ένα διάνυσμα 38 χαρακτηριστικών για το κάθε εκτελέσιμο. Δεδομένου ότι το 80% των δειγμάτων που είχαν στην διάθεσή τους ήταν χωρίς ετικέτες, εφάρμοσαν τον αλγόριθμο k-means για την δημιουργία, ψευδο-ετικετών. Στην συνέχεια, τα διανύσματα χαρακτηριστικών μαζί με τις ψευδο-ετικέτες τροφοδοτήθηκαν σε ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο με την προσθήκη ενός μηχανισμού προσοχής. Ο μηχανισμός προσοχής, εισέρχεται αμέσως μετά το δεύτερο πλήρως συνδεδεμένο επίπεδο και προσπαθεί να καθοδηγήσει το υπόλοιπο δίκτυο να δώσει περισσότερη προσοχή ή βάρος στα ενδιάμεσα αποτελέσματα που θεωρεί σημαντικότερα. Σε ένα σύνολο 15457, εκ των οποίων

τα 6681 ήταν καλοήθη, η μέθοδος που προτάθηκε είχε μια ακρίβεια 98.09%.

3.4.2 Αναπαράσταση εικόνων

Οι προσεγγίσεις που βασίζονται σε εικόνες παρέχουν τα βαθιά νευρωνικά δίκτυα με εισόδους εικόνες. Οι εικόνες αντιπροσωπεύουν το δυαδικό περιεχόμενο του κακόβουλου λογισμικού όπως έχει ήδη περιγραφεί σε προηγούμενη υποενότητα. Η διαφοροποίηση των ακόλουθων προσεγγίσεων βρίσκεται στο γεγονός ότι πλέον οι ερευνητές δεν βασίζονται σε τεχνικές επεξεργασίας εικόνων για την εξαγωγή των χαρακτηριστικών αλλά αφήνουν και αυτό το κομμάτι στα νευρωνικά δίκτυα, μαζί με την τελική ταξινόμηση.

Rezende et al. (2017) [68] πρότεινε μια μέθοδο που βασίζεται στη μεταφορά μάθησης προκειμένου να αυξηθεί η ακρίβεια της ανίχνευσης και να μειωθεί ο απαιτούμενος χρόνος εκπαίδευσης. Πιο συγκεκριμένα, χρησιμοποιώντας ένα προ-εκπαιδευμένο ResNet-50 και αντικαθιστώντας μόνο τα τελευταία πλήρως συνδεδεμένα στρώματα, μπόρεσαν να επιτύχουν μια ακρίβεια 98% ξεπερνώντας τη χειροποίητη προσέγγιση που παρουσίασαν οι Nataraj et al. [43] στο ίδιο σύνολο δεδομένων.

Ni et al. (2018) [69] προτείνουν μια ενδιαφέρουσα προσέγγιση για την κατηγοριοποίηση του κακόβουλου λογισμικού σε οικογένειες. Αντί να δημιουργήσουν εικόνες κατευθείαν από το εκτελέσιμο χρησιμοποιούν τον αλγόριθμο SimHash για να δημιουργήσουν ομάδες εντολών με παραπλήσιες τιμές κατακερματισμού. Έπειτα, δημιουργούν διανύσματα SimHash και χρησιμοποιούν αυτά για να παράξουν εικόνες. Επειδή τα διανύσματα SimHash που χρησιμοποιούν έχουν τιμές μόνο 0 ή 1 οι τελικές εικόνες έχουν μόνο άσπρα ή μαύρα εικονοστοιχεία. Σε ένα σύνολο δεδομένων 10805 δειγμάτων ένα συνελικτικό δίκτυο CNN πέτυχε μια μέση ακρίβεια της τάξης του 98.8%.

Vinayakumar et al. (2019) [70] δοκίμασαν πολλού αλγόριθμους μηχανικής μάθησης, όπως DT, RF, SVM, DNN, CNN σε δύο σύνολα δεδομένων, το MalImg και ένα που δημιούργησαν οι ίδιοι. Τα σύνολα δεδομένων είχαν 9339 και 15512 δείγματα κακόβουλου λογισμικού αντίστοιχα. Στόχος τους ήταν η κατάταξη των δειγμάτων σε οικογένειες κακόβουλου λογισμικού. Τα δεδομένα προτού τροφοδοτηθούν στους αλγορίθμους μετασχηματίστηκαν σε "μονοδιάστατες εικόνες" δηλαδή ξεδίπλωσαν τις υπάρχουσες εικόνες σε ένα μονοδιάστατο διάνυσμα. Έπειτα στα παραγόμενα διανύσματα εφαρμόστηκε κανονικοποίηση L2.Τα αποτελέσματα έδειξαν ότι και στα δύο σύνολα δεδομένων τα δίκτυα CNN είχαν τις καλύτερες επιδόσεις με ακρίβεια 93.3% και 96.3% αντίστοιχα.

Huang et al. (2020) [71] πρότειναν μια μέθοδο εντοπισμού κακόβουλου λογισμικού που βασίζεται σε ένα δίκτυο τύπου VGG16. Η διαφοροποίηση της πρότασής τους εισέρχεται στο γεγονός ότι οι εικόνες που δημιουργήθηκαν για να τροφοδοτήσουν το δίκτυο παράχθηκαν από στατικά αλλά και από δυναμικά χαρακτηριστικά για να δώσουν σαν τελικό αποτέλεσμα εικόνες RGB που κωδικοποιούν μεγαλύτερο μέρος πληροφορίας από της εικόνες γκρί κλίμακας. Σε ένα σύνολο 1923 δειγμάτων, εκ των οποίων τα 1310 ήταν καλοήθη, η προσέγγιση τους έδωσε ακρίβεια επαλήθευσης ίση με 94.7%.

3.4.3 Ίχνη κλήσεων API

Σε προηγούμενες υποενότητες παρουσιάσαμε αρκετές προσεγγίσεις που χρησιμοποίησαν διανύσματα χαρακτηριστικών με βάση τις κλήσεις API. Οι προσεγγίσεις που παρουσιάστηκαν είχαν το μειονέκτημα ότι έχαναν εντελώς την χρονική συσχέτιση των κλήσεων API. Στο πλαίσιο της βαθιά μηχανικής μάθησης το πρόβλημα αυτό μπορεί να ξεπεραστεί καθώς υπάρχουν αρχιτεκτονικές που επεξεργάζονται χρονικές σειρές με εξαιρετική ακρίβεια.

Kolosnjaji et al. (2016) [72] ερεύνησαν τη χρήση νευρωνικών δικτύων για τη βελτίωση της ταξινόμησης πρόσφατα ανακτημένων δειγμάτων κακόβουλου λογισμικού σε ένα προκαθορισμένο σύνολο οικογενειών κακόβουλου λογισμικού. Ανέλυσαν δύο τύπους στρωμάτων νευρωνικού δικτύου για τη μοντελοποίηση ακολουθιών κλήσεων συστήματος: συνελικτικές και επαναλαμβανόμενες στρώσεις. Κατασκεύασαν ένα Νευρωνικό Δίκτυο βασισμένο σε συνελικτικές και επαναλαμβανόμενες στρώσεις που συνδυάζει τη συνέλιξη των n-grams με πλήρη διαδοχική μοντελοποίηση. Η είσοδος του δικτύου είναι οι ακολουθίες κλήσεων API κακόβουλου λογισμικού χωρίς τις κλήσεις API που επαναλαμβάνονται περισσότερες από δύο φορές στη σειρά. Κάθε κλήση API κωδικοποιήθηκε με χρήση One-hot encoding για να βρεθεί ένα μοναδικό διάνυσμα για κάθε κλήση API. Το συνελικτικό μέρος του δικτύου αποτελείται από ένα στρώμα συνέλιξης που ακολουθείται από pooling με τη συνέλιξη να λειτουργεί ως εξαγωγέας χαρακτηριστικών. Οι έξοδοι του συνελικτικού τμήματος συνδέονται με το επαναλαμβανόμενο τμήμα που μοντελοποιεί διαδοχικές εξαρτήσεις στα ίχνη API. Για την εξαγωγή των χαρακτηριστικών υψίστης σημασίας από την έξοδο των επαναλαμβανόμενων στρώσεων (LSTM), χρησιμοποιήθηκε ένα επίπεδο mean-pooling. Επιπλέον, εφάρμοσαν dropout τεχνικές για να αποτρέψουν την υπερ-εκπαίδευση και ένα στρώμα softmax για να εξάγουν τις πιθανότητες κλάσης. Το προτεινόμενο μοντέλο τους πέτυχε μια μέση ακρίβεια 89.4%.

Athiwaratkun et al. (2017) [73] εξέτασαν επαναλαμβανόμενες αρχιτεκτονικές νευρωνικών δικτύων για να καταγράψουν καλύτερα τις μακροπρόθεσμες εξαρτήσεις στα ίχνη κλήσεων API. Πειραματίστηκαν με την αρχιτεκτονική Long Short-Term Memory (LSTM) και την Gated Recurrent Unit (GRU) ως μοντέλα γλώσσας. Η προτεινόμενη μέθοδος αποτελείται από δύο στάδια. Σε πρώτο στάδιο, το LSTM ή το GRU χρησιμοποιούνται για τη δημιουργία των χαρακτηριστικών που σχετίζονται με ένα συγκεκριμένο ίχνος κλήσης API. Στο δεύτερο στάδιο, αυτά τα χαρακτηριστικά ταξινομούνται είτε με ένα μόνο πλήρως συνδεδεμένο επίπεδο είτε με Logistic Regression με softmax. Επιπλέον, πρότειναν ένα συνελικτικό νευρωνικό δίκτυο σε επίπεδο χαρακτήρων. Αυτό το δίκτυο λαμβάνει ως είσοδο μια ακολουθία, μέγιστου μήκους, 1014 χαρακτήρων όπου κάθε χαρακτήρας είναι ένα συμβάν. Το δίκτυο σε επίπεδο χαρακτήρων που παρουσιάζεται αποτελείται από 9 επίπεδα, 6 συνελικτικά και 3 πλήρως συνδεδεμένα. Σε ένα σύνολο δεδομένων 75000 δειγμάτων, εκ των οποίων τα 37500 ήταν καλοήθεις, το μοντέλο με την καλύτερη απόδοση ήταν ένα δίκτυο LSTM με logistic regression και χρονικό max-pooling, επιτυγχάνοντας πάνω από 80% true positive rate.

Xu et al. (2021) [74] πρότειναν το εργαλείο ανίχνευσης “Malbert”. Το Malbert έχει ως δεδομένα εισόδου κλήσεις API οι οποίες έχουν εξαχθεί από τα εκτελέσιμα, έχουν καθαριστεί από ιδιαίτερα σύμβολα και έχουν μετατραπεί στο ίδιο μήκος, είτε με συμπλήρωση των μικρότερων ακολουθιών είτε με περικοπή των μεγαλύτερων. Στην καρδιά του Malbert βρίσκονται 12 δίκτυα Encoder με attention layers με έναν τελικό ταξινομητή MLP 2 επιπέδων. Η καινοτομία έρχεται με την μορφή της προ εκπαίδευσης. Συγκεκριμένα εκπαιδεύουν όλο το δίκτυο με τυχαίες κλήσεις API μέχρι και τον πρώτο layer του MLP και τον δεύτερο, που στην πραγματικότητα δίνει και το τελικό αποτέλεσμα, τον εκπαιδεύουν μόνο όταν χρειαστεί το Malbert σε μια εφαρμογή. Έτσι ευελπιστούν να γλυτώσουν πολύτιμους υπολογιστικούς πόρους σε περιπτώσεις που χρειαστεί η μεταφορά του Malbert σε νέα σύνολα δεδομένων. Πράγματι, η προσέγγιση τους σε ένα σύνολο δεδομένων 25262, από τα οποία τα 21166 ήταν καλοήθη, πέτυχαν μια ακρίβεια της τάξης του 99.98%.

3.4.4 Ίχνη Εντολών

Αντί να εντοπίζουμε απλώς τις κλήσεις API, η συμπεριφορά ενός προγράμματος είναι στην πραγματικότητα η ακολουθία οδηγιών που εκτελούνται από τον επεξεργαστή. Αυτές οι αλληλουχίες μπορούν να εξαχθούν τόσο μέσω στατικής όσο και δυναμικής ανάλυσης. Οι εξαγόμενες αλληλουχίες (είτε στατικά είτε δυναμικά) μπορούν να χρησιμοποιηθούν για την εκπαίδευση συστημάτων ταξινόμησης. Στην πραγματικότητα η μέθοδος αυτή παρακάμπτει το πρόβλημα που δημιουργείται στην ανάλυση n-grams όταν το n αυξάνει, καθώς πλέον έχουν εντοπιστεί όλες οι εντολές, δεν χρειάζεται η προσέγγιση των n-grams για να βρεθούν οι σημαντικές ακολουθίες byte.

Gibert et al. (2017) [75] πρότειναν μια αρχιτεκτονική νευρωνικού δικτύου που περιλαμβάνει: ένα επίπεδο ενσωμάτωσης, ένα συνελικτικό επίπεδο ακολουθούμενο από ένα ανώτερο επίπεδο συγκέντρωσης και ένα επίπεδο εξόδου. Το στρώμα συνέλιξης έμαθε εγγενώς να ανιχνεύει υπογραφές που μοιάζουν με n-gram ανιχνεύοντας opcodes που είναι ενδεικτικά κακόβουλου λογισμικού. Με τη ρύθμιση του μεγέθους του πυρήνα που χρησιμοποιείται στο στρώμα της συνέλιξης μπορούν να εντοπιστούν πολύ μεγάλες υπογραφές που μοιάζουν με n-gram, μια εργασία που θα ήταν ανέφικτη αν απαιτούνταν ρητή απαρίθμηση όλων των n-gram. Αυτό επιτυγχάνεται με τον καθορισμό φίλτρων με διάφορα μεγέθη. Στη δουλειά τους, το στρώμα συνέλιξης περιείχε $\{2, 3, 4, 5, 6, 7\} \times \{k\} \times \{64\}$ φίλτρα όπου k είναι το μέγεθος του διανύσματος ενσωμάτωσης. Στη συνέχεια, η μέγιστη τιμή ελήφθη ως το χαρακτηριστικό που αντιστοιχεί στο φίλτρο εφαρμόζοντας τον τελεστή μέγιστης συγκέντρωσης (max-pooling) πάνω στο χάρτη χαρακτηριστικών (γνωστό και ως παγκόσμια μέγιστη συσσώρευση, global max-pooling) για να διατηρήσει μια τιμή από κάθε φίλτρο. Αυτό επιτρέπει την εξαγωγή υπογραφών τύπου n-gram με n να κυμαίνεται από 2 έως 7. Τέλος, το στρώμα softmax εξάγει την κατανομή πιθανότητας στις κλάσεις. Σε ένα σύνολο δεδομένων 21741 δειγμάτων κατάφεραν να επιτύχουν ακρίβεια εκπαίδευσης 99.64%. Παρόλο που η ακρίβεια που έχει επιτευχθεί είναι εξαιρετικά υψηλή, εξακολουθεί να υστερεί σε άλλες προτεινόμενες λύσεις [76] που χρησιμοποίησαν χαρακτηριστικά που εξήχθησαν με το χέρι από ειδικούς στον τομέα και συνδυάστηκαν σχολαστικά.

Gibert et al. (2019) [77] με βάση την προηγούμενη προσέγγισή τους πρότειναν ένα νευρωνικό δίκτυο ιεραρχικής συνέλιξης (HCNN) για να αντιμετωπίσει την ιεραρχική δομή των εκτελέσιμων PE. Διατήρησαν το συνελικτικό στρώμα από την προηγούμενη εργασία τους λίγο πολύ άθικτο και πρόσθεσαν ένα άλλο μπλοκ που ομαδοποιεί εντολές από τις ίδιες συναρτήσεις προκειμένου να διατηρηθεί η ιεραρχική δομή ενός προγράμματος υπολογιστή. Για να βοηθηθεί αυτή η προσέγγιση, οι οδηγίες της assembly χωρίστηκαν σε συναρτήσεις, όπου κάθε συνάρτηση αντιπροσωπεύτηκε από μια ακολουθία μνημονικών. Κατά συνέπεια, το ιεραρχικό συνελικτικό νευρωνικό δίκτυο αναγνώρισε χαρακτηριστικά σε μνημονικό επίπεδο και σε επίπεδο συνάρτησης. Στο ίδιο σύνολο δεδομένων, αυτή η προσέγγιση πέτυχε 99.13% ακρίβεια σε 10-fold διασταυρούμενη επικύρωση.

3.4.5 Αναπαράσταση βασισμένη σε bytes

Ο πιο "καθαρός" τρόπος για να αναπαραστήσετε ένα πρόγραμμα υπολογιστή είναι ως μια ακολουθία byte και κάθε byte είναι μια μονάδα σε μια πιθανή ακολουθία εισόδου. Το κύριο πλεονέκτημα αυτής της αναπαράστασης είναι το επιπλέον επίπεδο αφαίρεσης που παρέχει. Τα Bytes είναι ανεξάρτητα από οποιονδήποτε υλικολογισμικό παράγοντα. Δηλαδή, η αναπαράσταση με Bytes παρέχει ίσως τον πιο καθολικό τρόπο αναπαράστασης ενός εκτελέσιμου. Όμως, προφανώς, δεν είναι τέλεια. Το πρώτο μεγάλο πρόβλημα που εμφανίζεται είναι το ίδιο με τα n-grams για μεγάλα n. Αν εντοπιστεί το κάθε Byte ξεχωριστά τότε

υπάρχουν εκατομμύρια, αν όχι παραπάνω, πιθανοί συνδυασμοί Bytes που καθιστούν το συγκεκριμένο πρόβλημα άλυτο. Επιπλέον τα Bytes όταν απομονωθούν είναι απλοί αριθμοί χωρίς κανένα παραπάνω νόημα, μόνο όταν εξετασθούν συνολικά μπορούν να αποκτήσουν νόημα για την πληροφορία που κωδικοποιούν (κάποια συμβολοσειρά, κάποια εντολή, κ.λπ.). Έπειτα, όταν εξετάζονται σε μικρότερες ομάδες για να αποκτήσουν νόημα και αυτές οι ομάδες κωδικοποιούν εντολές είναι αρκετά συχνό φαινόμενο να υπάρχουν εντολές που κανονικά θα μετέφεραν την εκτέλεση του προγράμματος σε άλλη διαφορετική ομάδα Bytes, σπάει η χωρική συνοχή των ομάδων. Αυτά τα προβλήματα καλούνται να τα αντιμετωπίσουν οι ερευνητές ασφάλειάς που αποφασίζουν να χρησιμοποιήσουν την συγκεκριμένη αναπαράσταση.

Chowdhury et al. (2017) [78] παρουσιάζουν μια μέθοδο εντοπισμού κακόβουλου λογισμικού που στηρίζεται στον συνδυασμό χαρακτηριστικών n-grams και στοιχείων από την κεφαλίδα των PE αρχείων. Συγκεκριμένα, ο συνδυασμός 5-grams με τα στοιχεία κεφαλίδων, αφού εφαρμοστεί ο αλγόριθμος PCA, τροφοδοτήθηκαν σε ένα πλήρως συνδεδεμένο δίκτυο. Σε ένα σύνολο δεδομένων 52185 δειγμάτων, εκ των οποίων τα 10920 είναι καλοήθη, το νευρωνικό δίκτυο είχε μια ακρίβεια της τάξης του 97.7%. Αξίζει να σημειωθεί ότι για την επίτευξη αυτής της ακρίβειας ο αριθμός των συνολικών επαναλήψεων εκπαίδευσης ήταν 35000 (epochs).

Raff et al. (2018) [79] πρότειναν ένα Συνελικτικό Νευρικό Δίκτυο για την αντιμετώπιση του προβλήματος της αναλλοίωτης τοποθεσίας υψηλού επιπέδου. Ο συνδυασμός των συνελικτικών στρώσεων με global max-pooling πριν από το πλήρως συνδεδεμένο επίπεδο επέτρεψε στο μοντέλο να παράγει την ενεργοποίησή του ανεξάρτητα από τη θέση των εντοπισμένων χαρακτηριστικών. Σε σύνολο δεδομένων με πάνω από 2 εκατομμύρια δείγματα, εκ των οποίων 1,000,020 ήταν καλοήθη, η προσέγγισή τους πέτυχε 94% ακρίβεια στο σύνολο δοκιμών.

3.4.6 Κίνηση Δικτύου

Για άλλη μια φορά η παρακολούθηση της κίνησης δικτύου που δημιουργείται από δυνητικά κακόβουλο λογισμικό παρέχει έναν άλλο τρόπο ανίχνευσης της παρουσίας κακόβουλου λογισμικού σε ένα σύστημα.

Prasse et al. (2017) [80] πρότειναν μια συνολική λύση για τον εντοπισμό κακόβουλου λογισμικού σε υπολογιστές με βάση την ανάλυση της κίνησης δικτύου HTTP. Εξήγαγαν διάφορα χαρακτηριστικά από τις ακολουθίες ροών που αποστέλλονται ή λαμβάνονται από υπολογιστές. Στη συνέχεια, ένας ταξινομητής LSTM λαμβάνει τις ακολουθίες ως είσοδο και μαθαίνει να καθορίζει εάν οι ροές προέρχονται ή όχι από κακόβουλες εφαρμογές. Η προσέγγισή τους διαρκεί κατά μέσο όρο 90 λεπτά για τον εντοπισμό κακόβουλου λογισμικού αφού το κακόβουλο λογισμικό ξεκινήσει την επικοινωνία.

Al-Hawawreh et al. (2018) [81] πρότειναν μια τεχνική ανίχνευσης ανωμαλιών για τον εντοπισμό εισβολών σε Συστήματα Βιομηχανικού Ελέγχου Διαδικτύου (Internet Industrial Control Systems - IICS) βάσει μοντέλων βαθιάς μάθησης. Το σύστημα περιλαμβάνει μια φάση μάθησης χωρίς επίβλεψη, όπου ένας Deep Autoencoder μαθαίνει φυσιολογικές συμπεριφορές δικτύου και μια εποπτευόμενη φάση μάθησης, όπου ένα Deep Neural Network χρησιμοποιεί τις εκτιμώμενες παραμέτρους του Autoencoder για να ρυθμίσει τις παραμέτρους του και να ταξινομήσει τις εισερχόμενες παρατηρήσεις του δικτύου. Σε ένα σύνολο δεδομένων δειγμάτων 148517 από τα οποία τα 77054 ήταν καλοήθη, το μοντέλο τους πέτυχε ακρίβεια 98.6%.

3.4.7 Πίνακας σύγκρισης μεθόδων βαθιάς μάθησης

Δημοσίευση	Χαρακτηριστικά	Επιλογή Χαρακτηριστικών Μείωση Διαστάσεων	Τύπος Τεχνητού Νευρωνικού Δικτύου
Azeez et al. (2021)	PE Headers	PCA	ANN Ensemble + ExtraTrees
Damaševičius et al. (2021)	PE Headers	PCA	ANN Ensemble + ExtraTrees
Rivizi et al. (2021)	PE Headers	-	k-NN + ANN + Attention
Rezende et al. (2017)	Gray-scale image	-	ResNet-50
Ni et al. (2018)	Black-White image	SimHash	CNN
Vinayakumar et al. (2019)	1D images	L2 Normalization	CNN
Huang et al. (2020)	RGB images	-	VGG16
Kolosnjaji et al. (2016)	API call sequence	Convolutional layer	CRNN
Athiwaratkun et al. (2017)	API call sequence	-	LSTM, GRU
Xu et al. (2021)	API call sequence	-	Encoders + MLP
Gibert et al. (2017)	Mnemonics sequence	-	Shallow CNN
Gibert et al. (2019)	Mnemonics sequence	-	Hierarchical CNN
Chowdhury et al. (2017)	PE headers 5-grams	PCA	Feed-Forward Network
Raff et al. (2018)	Bytes sequence	-	CNN
Prasse et al. (2017)	HTTP traffic	-	LSTM
Al-Hawawreh et al. (2018)	Network behavior	AutoEncoder	Feed-Forward Network

Πίνακας 5: Αλγόριθμοι: Convolutional Neural Network (CNN), Residual Network (ResNet), Autoencoder (AE), Recurrent Neural Network (RNN), Long Short-Term Memory Network (LSTM), Gated Redurrent Unit Network (GRU).

Δημοσίευση	Πηγή Δεδομένων	Συνολικά Δείγματα	Αναλογία Δειγμάτων (B/M)	Στόχος	Καλύτερη Επίδοση
Azeez et al. (2021)	Kaggle Windows PE	19611	25.5%	Detection	Ensemble + ExtraTrees, 100% acc.
Damaševičius et al. (2021)	ClaMP	5210	47%	Detection	Ensemble + ExtraTrees, 99.9% acc.
Rivizi et al. (2021)	Self created	15457	43%	Detection	ANN+Attention, 98.09% acc.
Rezende et al. (2017)	MalIMG dataset	9339	-	Classification	CNN, 98% acc.
Ni et al. (2018)	Microsoft Malware Classification Challenge	10805	-	Classification	CNN, 98.8% acc.
Vinayakumar et al. (2019)	MalIMG + self created	9339 15512	-	Classification	CNN, 93.3% and 96.3%
Huang et al. (2020)	VirusSign	1923	68%	Detection	VGG16, 94.7% acc
Kolosnjaji et al. (2016)	VirusShare Maltrieve private	-	-	Detection	CRNN, 89.4% acc.
Athiwaratkun et al. (2017)	Self created	75000	50%	Detection	LST, 80% TPR
Xu et al. (2021)	Ki + Gatak	25262	83.7%	Detection	Malbert, 99.98% acc.
Gibert et al. (2017)	Microsoft Malware Classification Challenge	21741	-	Classification	Shallow CNN, 99.64% training accuracy
Gibert et al. (2019)	Microsoft Malware Classification Challenge	21741	-	Classification	HCNN, 99.13% training accuracy
Chowdhury et al. (2017)	VXHeaven	52185	21%	Detection	FFN, 97.7% acc.
Raff et al. (2018)	Self created	2011786	50%	Detection	CNN, 94% acc.
Prasse et al. (2017)	Self created datasets	1) 44348879 2) 129005149	-	Detection	LSTM
Al-Hawawreh et al. (2018)	1) KDD Cup 99 2) UNSW-NB15	1) 148517 2) 257673	52%	Detection	FFN, 98.6% acc.

Πίνακας 6: Μια συνολική σύγκριση των αποτελεσμάτων και των χαρακτηριστικών που επιλέχθηκαν από τα σύνολα δεδομένων για τις μεθόδους βαθιάς μηχανική μάθησης

Η υποενότητα ανασκόπησης βαθιάς μάθησης αποκάλυψε ορισμένα γεγονότα εξαιρετικής σημασίας. Είναι μάλλον σαφές ότι οι μέθοδοι βαθιάς εκμάθησης μπορούν να χρησιμοποιηθούν αποτελεσματικά για τον εντοπισμό και την ταξινόμηση κακόβουλου λογισμικού, χωρίς την ανάγκη εντατικού feature engineering λόγω της αυξημένης αφηρημένης αναπαράστασης. Όμως, τα νευρωνικά δίκτυα υποφέρουν από προβλήματα υπερβολικής προσαρμογής και χρόνου εκμάθησης. Ως εκ τούτου, είναι εξαιρετικά σημαντικό να δοθεί μεγάλη προσοχή κατά τη διάρκεια της εκπαίδευσης για να αποφευχθούν περιστατικά υπερβολικής προσαρμογής και να επιλεγεί ένας σωστός συνδυασμός μεταξύ hardware, πολυπλοκότητας μοντέλου και μεγέθους δεδομένων, προκειμένου να μειωθεί ο χρόνος εκπαίδευσης και ο χρόνος εξαγωγής αποτελεσμάτων.

4 Περιγραφή του Προβλήματος

Μετά την εκτενή βιβλιογραφική ανασκόπηση, είναι καιρός να παρουσιάσουμε την προσέγγισή μας στο πρόβλημα εντοπισμού του κακόβουλου λογισμικού. Η πλειοψηφία των επιστημονικών δημοσιεύσεων αποκάλυψε δύο βασικά προβλήματα. Πρώτα απ 'όλα το βήμα εξαγωγής και επιλογής χαρακτηριστικών είναι υψίστης σημασίας. Αυτό σημαίνει ότι τα σχολαστικά συνδυασμένα και επιλεγμένα χαρακτηριστικά μπορούν να αποδώσουν τέλεια αποτελέσματα. Αλλά η επεκτασιμότητα και η αντοχή αυτής της προσέγγισης είναι σχεδόν μηδενική. Κανείς δεν πρέπει να ξεχνά ότι οι δημιουργοί κακόβουλου λογισμικού και οι αναλυτές ασφαλείας βρίσκονται σε διαρκή αγώνα, επομένως δεν υπάρχει πραγματική αξία στη δημιουργία του τέλειου ανιχνευτή χωρίς την ικανότητα να κλιμακώνεται καλά σε μελλοντικές απειλές. Δεύτερον, είναι ο χρόνος που απαιτείται από κάθε προσέγγιση. Από τη μία πλευρά, κάθε ανιχνευτής πρέπει να ενεργεί όσο το δυνατόν γρηγορότερα, άμεσα εάν είναι δυνατόν, προκειμένου να ελαχιστοποιήσει ή να αποτρέψει τη ζημιά που θα προκαλέσει ένα κακόβουλο λογισμικό στο σύστημα. Από την άλλη πλευρά, εμφανίζονται συνεχώς νέες παραλλαγές κακόβουλου λογισμικού, επομένως η επανεκπαίδευση όλων των αναπτυγμένων μοντέλων σε τακτά διαστήματα είναι σχεδόν υποχρεωτική.

Λαμβάνοντας υπόψη τα προηγούμενα "αδύνατα σημεία" η πρότασή μας είναι να δημιουργήσουμε ένα σύστημα που:

- Ανιχνεύει γρήγορα το κακόβουλο λογισμικό, έχει πολύ σύντομο χρόνο εξαγωγής συμπερασμάτων. Αν ορίσουμε ένα χρόνο δράσης για κακόβουλο λογισμικό, ο χρόνος που απαιτείται για να εγκατασταθεί και να απελευθερωθεί το επικίνδυνο φορτίο του, σαν t_{mal} ο προτεινόμενος ανιχνευτής μας πρέπει να εντοπίσει το κακόβουλο λογισμικό σε $t_{det} \ll t_{mal}$ προκειμένου να αποφευχθούν τυχόν κακόβουλες ενέργειες και να δοθεί χρόνος για σωστή αφαίρεση χωρίς υπολείμματα που θα μπορούσαν να μολύνουν εκ νέου το σύστημα.

- Χρησιμοποιεί πολύ αφηρημένα χαρακτηριστικά για να κάνει τον ανιχνευτή ανθεκτικό σε μελλοντικές εξελίξεις. Με τον όρο "αφηρημένα" υπονοείται η εξαγωγή υποκείμενων δομών, μοτίβων και ιδιοτήτων μιας μαθηματικής οντότητας. Ακολουθώντας την πορεία της αφαίρεσης, οποιαδήποτε εξάρτηση από αντικείμενο του πραγματικού κόσμου θα μειωθεί σημαντικά, αυξάνοντας έτσι τη γενίκευση με τρόπο που ο προκύπτων ανιχνευτής θα έχει ευρύτερες εφαρμογές.

- Έχει καλή ακρίβεια. Σύμφωνα με τη βιβλιογραφική μελέτη αναφοράς στο προηγούμενο κεφάλαιο, καλή ακρίβεια σημαίνει βαθμολογία 95%+ στην μετρική ακρίβειας ή κοντά στο 1.0 στις μετρικές τύπου F1.

- Απαιτεί σύντομο χρόνο επανεκπαίδευσης. Για να δώσουμε έναν συγκεκριμένο ορισμό σχετικά με τον όρο "χρόνος εκπαίδευσης" πρέπει να κάνουμε μια βαθιά βουτιά στην καρδιά της εκπαιδευτικής διαδικασίας. Οι περισσότερες τεχνικές εκπαίδευσης που χρησιμοποιούνται ευρέως βασίζονται σε μια παραλλαγή μιας διαδικασίας που ονομάζεται "Στοχαστική Κάθοδος Βαθμίδας - Stochastic Gradient Descent (SGD)". Αλλά πριν αντιμετωπίσουμε το SGD πρέπει να παρουσιάσουμε το αρχικό πρόβλημα, που προσπαθούμε να λύσουμε. Σύμφωνα με την δημοσίευση του Latz [82] αν ορίσουμε $(X, \|\cdot\|) := \left(\mathbb{R}^K, \|\cdot\|_2\right)$, και $\bar{\Phi} : X \to \mathbb{R}$ είναι κάποια συνάρτηση που έχει κάποιο ολικό ελάχιστο στο X. Υποθέτοντας ότι $\bar{\Phi}$ είναι της μορφής $\bar{\Phi} = \frac{1}{N}\sum_{i=1}^{N} \Phi_i$ όπου $N \in \mathbb{N} := \{1, 2, \ldots\}, N \geq 2$ και $\Phi_i : X \to \mathbb{R}$ είναι μια συνεχώς διαφορίσιμη συνάρτηση. Στόχος μας είναι να λύσουμε το πρόβλημα, χωρίς περιορισμούς, βελτιστοποίησης

$\theta^* \in \text{argmin}_{\theta \in X} \bar{\Phi}(\theta)$. Στις εφαρμογές μηχανικής εκμάθησης η συνάρτηση $\bar{\Phi}$ ονομάζεται συνάρτηση κόστους και κάθε $\Phi_i(\theta)$ αντιπροσωπεύει τη συνάρτηση κόστους για ένα κλάσμα y_i του συνόλου δεδομένων y. Το SGD αναφέρεται γενικά ως αλγόριθμος κατάβασης κλίσης με ανακριβείς εκτιμήσεις κλίσης επειδή το $\bar{\Phi}$ αντικαθίσταται τυχαία από κάποιο Φ_i. Οι επαναλήψεις του αλγόριθμου SGD, για μια σταθερή μικρή τιμή του συντελεστή μάθησης, μπορούν να διαμορφωθούν με μια στοχαστική διαφορική εξίσωση:

$$\mathrm{d}\theta(t) = -\nabla\Phi(\theta(t))\mathrm{d}t + \sqrt{\eta}\Sigma(\theta(t))^{1/2}\,\mathrm{d}W(t) \quad (t > 0), \tag{13}$$

$$\tilde{\theta}(0) = \theta_0 \tag{14}$$

Όπου, $\Sigma(\theta) : X \to X$ είναι ένας συμμετρικός, θετικός ημι-πεπερασμένος για $\theta \in X$ πίνακας και $W : [0, \infty) \to X$ είναι μια K-διάστατη κίνηση Brown. Ο στόχος μας λοιπόν είναι να παράγουμε ένα μοντέλο που ικανοποιεί την εξίσωση στο συντομότερο δυνατό χρόνο. Το οποίο, δυστυχώς, δεν είναι κάτι σταθερό λόγω της τυχαιότητας που κληρονομείται από τον όρο της κίνησης Brown.

Οι απαιτήσεις που παρουσιάσαμε σε συνδυασμό με την ανασκόπηση της διεθνούς βιβλιογραφίας μας οδηγούν στην δομή που θα επιλέξουμε για ν' αντιμετωπίσουμε το πρόβλημα που θέσαμε. Αρχίζοντας από την μορφή που θα έχουν τα δεδομένα που θα επεξεργαστούμε, επιλέγουμε την μορφή εικόνων, καθώς είναι ένα στατικό χαρακτηριστικό που περιγράφει πλήρως την δυαδική δομή κάθε εκτελέσιμου. Επίσης, εκτός από την πλήρη αντιστοιχία byte - pixel value που προσφέρουν οι εικόνες είναι και πολύ γρήγορο να δημιουργηθούν καθώς δεν χρειάζεται κάποιο επιπλέον βήμα ή επεξεργασία του εκτελέσιμου. Έπειτα, από την στιγμή που έχουμε επιλέξει ότι τα δεδομένα θα είναι σε μορφή εικόνων η χρήση δικτύων CNN είναι στην πραγματικότητα μονόδρομος καθώς η αποτελεσματικότητα τους έχει αποδειχτεί σε πολυάριθμες εφαρμογές. Έτσι έχουμε δημιουργήσει, τουλάχιστον σε θεωρητικό επίπεδο, την ροή που θα ακολουθήσουμε για την ανίχνευση ενός κακόβουλου λογισμικού μετατρέποντάς το σε εικόνα. Προφανώς, η ροή από μόνη της δεν μπορεί να καλύψει τις απαιτήσεις που θέσαμε και για αυτόν ακριβώς τον λόγο στο πειραματικό μέρος θα δοκιμάσουμε διαφορετικές αρχιτεκτονικές ώστε να καταλήξουμε ποια είναι αυτή που καλύπτει τις προϋποθέσεις που αναφέραμε.

Τέλος έχοντας αναφέρει και με μαθηματικούς ορισμούς τις απαιτήσεις που ζητάμε από τον ανιχνευτή είναι ακόμα πιο ξεκάθαρο ότι οι απαιτήσεις χωρίζονται σε δύο διακριτές κατηγορίες. Αυτές που αφορούν τον χρόνο και αυτές που αφορούν την ακρίβεια. Ο συγκεκριμένος διαχωρισμός δεν θα έπρεπε να εκπλήσσει κανέναν δεδομένου ότι εμφανίζεται πολύ συχνά στις εφαρμογές μηχανικής μάθησης. Οι έννοιες ακρίβεια και χρόνος στον τομέα της μηχανικής μάθησης είναι συνήθως αντικρουόμενες και ο εκάστοτε ερευνητής καλείται να βρει την χρυσή τομή μεταξύ των δύο, δηλαδή να έχει την καλύτερη δυνατή ακρίβεια εντός εύλογου χρόνου. Φυσικά, τα αποδεκτά όρια και για τις δύο ποσότητες εξαρτώνται από την εφαρμογή και το διαθέσιμο υλικό. Έτσι και εμείς στην συνέχεια της εργασίας καλούμαστε να αναπτύξουμε μια μέθοδο εντοπισμού κακόβουλου λογισμικού η οποία θα προσφέρει μια πολύ καλή ισορροπία μεταξύ των ποσοτήτων ακρίβεια και χρόνος. Με άλλα λόγια δεν μας ενδιαφέρουν περιπτώσεις στις οποίες τα μοντέλα εκπαιδεύονται για 5 ώρες σε έναν προσωπικό υπολογιστή για να δώσουν ακρίβεια 99% αλλά ούτε και οι περιπτώσεις στις οποίες η εκπαίδευση τελειώνει σε δέκα λεπτά και η ακρίβεια είναι κάτω του 90%. Αναζητούμε

τις ενδιάμεσες περιπτώσεις στις οποίες υπάρχει η ζητούμενη ισορροπία μεταξύ χρόνου και ακρίβειας.

5 Συλλογή δεδομένων, επεξεργασία και μετρικές αξιολόγησης

5.1 Πηγές δεδομένων

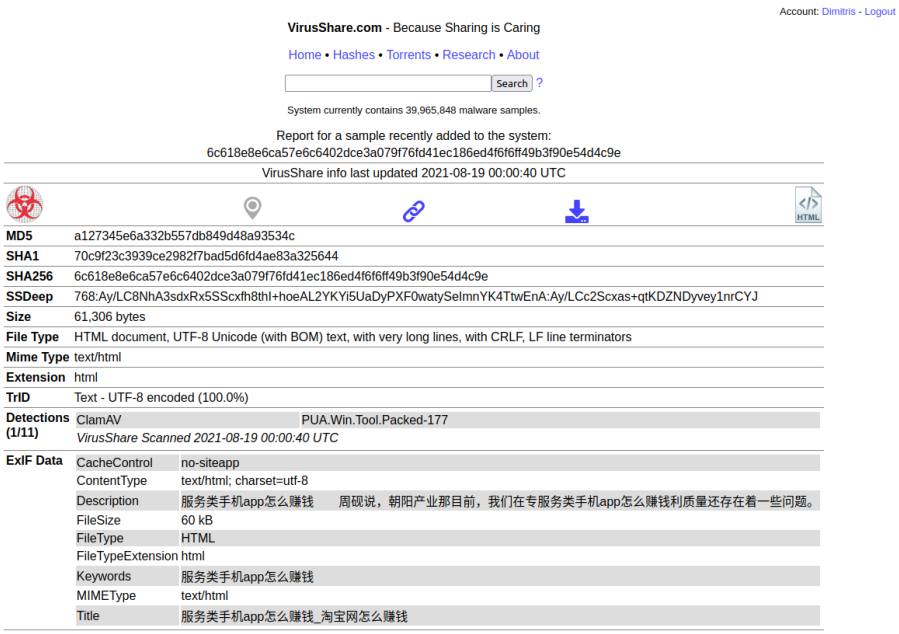
Η επιτυχία ενός μοντέλου ανίχνευσης και ταξινόμησης σε οποιαδήποτε εφαρμογή, όχι μόνο στα κακόβουλα λογισμικά, εξαρτάται άμεσα από την ποιότητα, πολλές φορές και από την ποσότητα, των δεδομένων. Στην παρούσα εργασία θα ασχοληθούμε με δεδομένα τα οποία προέρχονται από δύο διαφορετικές πηγές. Συγκεκριμένα, τα δεδομένα αφορούν κακόβουλα και καλοήθη λογισμικά. Επειδή, ένας από τους βασικούς στόχους που θέσαμε για τα μοντέλα είναι η ταχύτητα ανίχνευσης, θα ασχοληθούμε μόνο με στατικά χαρακτηριστικά, καθώς μπορούν να εξαχθούν γρήγορα και με την μέγιστη δυνατή ασφάλεια.

5.1.1 Windows

Προφανώς για την δημιουργία ενός ανιχνευτή κακόβουλου λογισμικού πρέπει να έχουμε και καλοήθη εκτελέσιμα στα δείγματα που θα αναλύσουμε. Από την στιγμή που θα δουλέψουμε με κακόβουλο λογισμικό που προσβάλλει Windows συστήματα, η καλύτερη πηγή για να αντλήσουμε καλοήθη δείγματα είναι από ένα σύστημα με λειτουργικό Windows. Δηλαδή, τα καλοήθη δείγματα είναι εκτελέσιμα που έρχονται μαζί με την εγκατάσταση των Windows.

5.1.2 VirusShare

Το dataset VirusShare είναι ένα διαδικτυακό καταθετήριο για δείγματα κακόβουλου λογισμικού. Το αποθετήριο είναι προσβάσιμο μόνο κατόπιν πρόσκλησης από τους διαχειριστές δεδομένου ότι περιέχει ενεργά δείγματα κακόβουλου λογισμικού. Ακριβώς, επειδή οι δημιουρ-


Account: Dimitris - Logout

VirusShare.com - Because Sharing is Caring

Home • Hashes • Torrents • Research • About

Search ?

System currently contains 39,965,848 malware samples.

Report for a sample recently added to the system:
6c618e8e6ca57e6c6402dce3a079f76fd41ec186ed4f6f6ff49b3f90e54d4c9e

VirusShare info last updated 2021-08-19 00:00:40 UTC

MD5	a127345e6a332b557db849d48a93534c
SHA1	70c9f23c3939ce2982f7bad5d6fd4ae83a325644
SHA256	6c618e8e6ca57e6c6402dce3a079f76fd41ec186ed4f6f6ff49b3f90e54d4c9e
SSDeep	768:Ay/LC8NhA3sdxRx5SScxfh8thI+hoeAL2YKYi5UaDyPXF0watySeImnYK4TtwEnA:Ay/LCc2Scxas+qtKDZNDyvey1nrCYJ
Size	61,306 bytes
File Type	HTML document, UTF-8 Unicode (with BOM) text, with very long lines, with CRLF, LF line terminators
Mime Type	text/html
Extension	html
TrID	Text - UTF-8 encoded (100.0%)
Detections (1/11)	ClamAV: PUA.Win.Tool.Packed-177 *VirusShare Scanned 2021-08-19 00:00:40 UTC*
ExIF Data	CacheControl: no-siteapp ContentType: text/html; charset=utf-8 Description: 服务类手机app怎么赚钱　周砚说，朝阳产业那目前，我们在专服务类手机app怎么赚钱利质量还存在着一些问题。 FileSize: 60 kB FileType: HTML FileTypeExtension: html Keywords: 服务类手机app怎么赚钱 MIMEType: text/html Title: 服务类手机app怎么赚钱_淘宝网怎么赚钱

Σχήμα 13: Η κεντρική σελίδα του αποθετηρίου VirusShare

γοί του VirusShare θέλουν να βοηθήσουν στην έρευνα κατά του κακόβουλου λογισμικού, ανά τακτά χρονικά διαστήματα ετοιμάζουν πακέτα που εμπεριέχουν χιλιάδες δείγματα μαζεμένα

για την διευκόλυνση των ερευνητών. Δεν χρειάζεται να κατεβάζουν ένα ένα δείγμα, κατεβάζουν απευθείας ένα πακέτο με τεράστιο όγκο. Αξιοποιώντας αυτή την δυνατότητα κατεβάσαμε ένα πακέτο που εμπεριέχει 30601 δείγματα κακόβουλου λογισμικού. Από αυτά τα 30601 πρέπει να αφαιρέσουμε όλα όσα δεν αφορούν Windows λειτουργικά. Τελικά, καταλήξαμε με 5000 δείγματα κακόβουλου λογισμικού που αφορούν μηχανήματα με λειτουργικό Windows.

5.2 Αρχική Επεξεργασία Δεδομένων

Προτού αρχίσει η οποιαδήποτε αλληλεπίδραση με τα δεδομένα που συλλέξαμε πρέπει να διασφαλίσουμε την ακεραιότητα των δεδομένων και των μηχανημάτων μας, διότι δεν πρέπει να ξεχνάμε ότι τα δείγματα από το VirusShare είναι ενεργά, δηλαδή μπορούν να εκτελεστούν και να προκαλέσουν ζημιά. Για τον λόγο αυτό η αρχική επεξεργασία θα γίνει σε περιβάλλον Linux και όχι Windows για να είμαστε σίγουροι ότι κανέναν εκτελέσιμο δεν μπορεί να εκτελεστεί και να προκαλέσει την οποιαδήποτε ζημιά.

Λαμβάνοντας υπ' όψη τους στόχους που έχουμε θέσει για τον ανιχνευτή που θέλουμε να δημιουργήσουμε αλλά και σε συνδυασμό με την διεθνή βιβλιογραφία καταλήξαμε στο γεγονός ότι ο καλύτερος τρόπος για να προσεγγίσουμε το συγκεκριμένο πρόβλημα είναι η μετατροπή των εκτελέσιμων σε εικόνες. Με την μετατροπή είμαστε σε θέση να χρησιμοποιήσουμε μέρη από πανίσχυρα προ-εκπαιδευμένα μοντέλα που ειδικεύονται στην διαχείριση εικόνων. Έχει ήδη αναφερθεί ότι όσο και αν μοιάζει "φυσική" η μετατροπή ενός εκτελέσιμου σε εικόνα γκρι κλίμακας εγκυμονεί έναν μεγάλο κίνδυνο που πρέπει να αποφευχθεί. Τα εκτελέσιμα είναι μονοδιάστατα αντικείμενα, αντιθέτως οι εικόνες είναι διδιάστατα αντικείμενα. Συνεπώς, το σημείο που θα "διπλώνεται" το εκάστοτε εκτελέσιμο είναι άκρως σημαντικό καθώς δημιουργούνται χωρικές συσχετίσεις που δεν υπήρχαν πριν. Οι Nataraj et al. [43] έθεσαν το πλάτος της εκάστοτε εικόνας ανάλογα με το μέγεθος του εκτελέσιμου αρχείου από το οποίο προέρχεται.

File Size Range	Image Width
<10 kB	32
10 kB – 30 kB	64
30 kB – 60 kB	128
60 kB – 100 kB	256
100 kB – 200 kB	384
200 kB – 500 kB	512
500 kB – 1000 kB	768
>1000 kB	1024

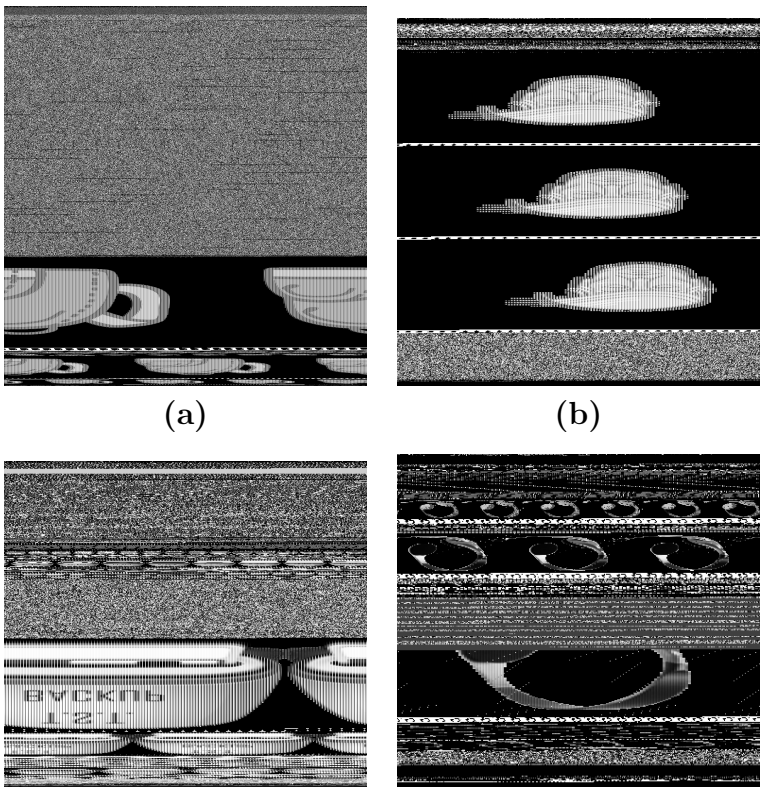
Σχήμα 14: Το πλάτος της εικόνας που θα δημιουργηθεί συναρτήσει του μεγέθους του εκτελέσιμου

Πηγή: Malware images: visualization and automatic classi-fication [43]

Με την συγκεκριμένη προσέγγιση το κάθε εκτελέσιμο αντιμετωπίζεται "διαφορετικά" ανάλογα με το μέγεθος του. Συνεπώς, έχει γίνει κάποιος εύλογος διαχωρισμός έτσι ώστε

οι όποιες χωρικές συσχετίσεις εμφανιστούν να εμφανίζονται με παρόμοιο τρόπο σε αρχεία παρόμοιου μεγέθους.

Τα δείγματα από το VirusShare είναι αρκετά εύκολο να μετατραπούν σε εικόνες δεδομένου ότι είναι πλήρη εκτελέσιμα. Παραθέτουμε κάποια "ενδιαφέροντα" δείγματα που προέκυψαν από τις μετατροπές:

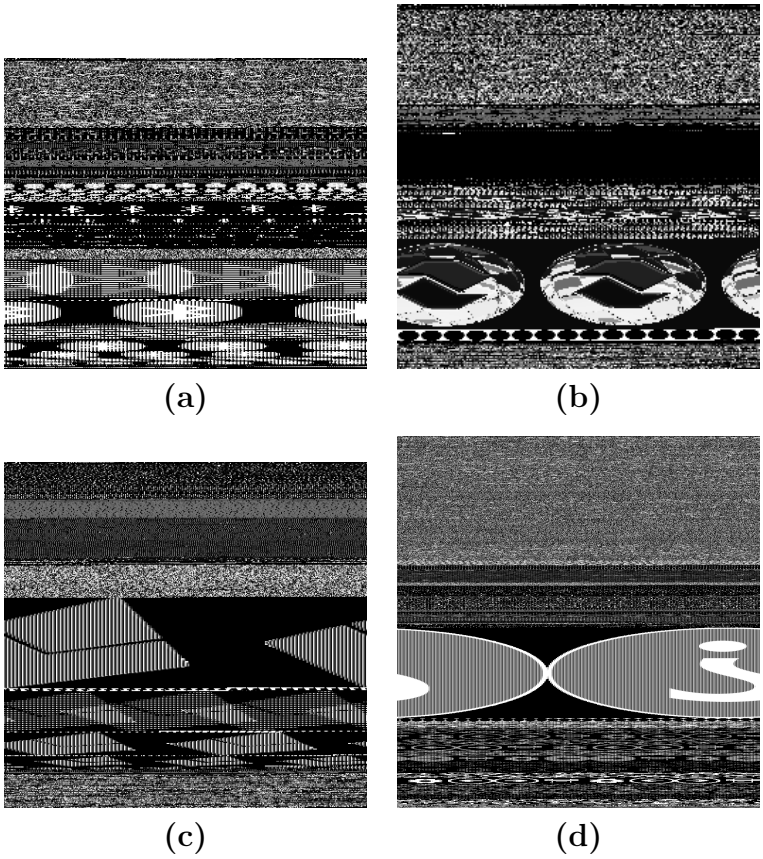


(a) **(b)**

(c) **(d)**

Σχήμα 15: Δείγματα εικόνων που δημιουργήθηκαν από κακόβουλα εκτελέσιμα **(a)** Trojan - "Payment Advice.exe" **(b)** Trojan - "TestDigitalControl.EXE" **(c)** Trojan - "Computer backup" **(d)** Backdoor - "TODDYSKEERNET.exe"

Το ίδιο εύκολο είναι να μετατραπούν και τα εκτελέσιμα αρχεία των Windows σε εικόνες δεδομένου ότι και αυτά είναι πλήρως λειτουργικά εκτελέσιμα.



Σχήμα 16: Δείγματα εικόνων που δημιουργήθηκαν από καλοήθη εκτελέσιμα **(a)** btpanui.exe **(b)** dxloader.exe **(c)** KiteService.exe **(d)** HelpPane.exe

5.3 Τεχνικές μείωσης διαστάσεων - Autoencoders

Στις απαιτήσεις που έχουμε θέσει, δύο αφορούν τον χρόνο. Θέλουμε οι λύσεις που θα προτείνουμε να είναι γρήγορες στην αναγνώριση και στην εκπαίδευση. Στο τελικό πείραμα θα βασιστούμε σε έτοιμα πανίσχυρα μοντέλα τα οποία θα τα τροποποιήσουμε ελαφρώς για να ταιριάξουν με τις ανάγκες μας. Σε αυτά τα δίκτυα η εκπαίδευση δεν είναι χρονοβόρα, καθώς το μεγαλύτερο μέρος είναι προ-εκπαιδευμένο, ανεξαρτήτως των δεδομένων εισόδου. Στην αρχή, όμως, πρέπει να θέσουμε κάποια κατώτατα όρια που οφείλουμε να ξεπεράσουμε χρησιμοποιώντας πιο απλά δίκτυα έτσι Από την στιγμή που η εκπαίδευση θα γίνει εξ' ολοκλήρου σε τοπικό-οικιακό μηχάνημα, ίσως κάποιες τεχνικές μείωσης διαστάσεων δεδομένων μπορέσουν να μειώσουν τον απαιτούμενο χρόνο εκπαίδευσης. Δεδομένου, πάντα, ότι δεν θυσιάζεται μεγάλο μέρος της ακρίβειας του δικτύου.

Επειδή μεγάλο μέρος της παρούσας εργασίας βασίζεται σε εφαρμογή τεχνικών Deep Learning δεν θα μπορούσαμε να παραλείψουμε και να μην δοκιμάσουμε να εφαρμόσουμε τα δίκτυα που ονομάζονται Autoencoders. Ένα δίκτυο Autoencoder απαρτίζεται από δύο μέρη, τον κωδικοποιητή (encoder) και τον αποκωδικοποιητή (decoder). Τα δύο μέρη μπορούν να οριστούν σαν μεταβάσεις φ και ψ, έτσι ώστε:
$\phi : \chi \to F$
$\psi : F \to \chi$
$\phi, \psi = argmin_{\phi,\psi} \|\chi - (\psi \circ \phi)\chi\|^2$
Με τον παραπάνω ορισμό είναι σαφές ότι ο σκοπός του δικτύου είναι η έξοδος του να μοιάζει όσο το δυνατόν με την είσοδο. Στο ενδιάμεσο υπάρχουν επίπεδα που μειώνουν τις διαστάσεις πριν αυξηθούν ξανά στην έξοδο. Έτσι, αν κρατήσουμε τις τιμές από αυτά τα ενδιάμεσα λανθάνοντα επίπεδα, έχουμε στα χέρια μας μια αναπαράσταση των δεδομένων με μειωμένη διάσταση.

5.4 Αξιολόγηση Αποτελεσμάτων

Το τελικό στάδιο εκπαίδευση ενός μοντέλου μηχανικής μάθησης είναι η αξιολόγηση των αποτελεσμάτων. Δηλαδή, η χρήση μετρικών για την ποσοτικοποίηση της αποτελεσματικότητας του τελικού μοντέλου στην εφαρμογή που εκπαιδεύτηκε.

Αρχικά, ως μετρική ορίζεται μια συνάρτηση που χρησιμοποιείται για την αξιολόγηση της απόδοσης του μοντέλου. Με δεδομένες τις προβλεπόμενες τιμές και τις πραγματικές τιμές η συνάρτηση αυτή δίνει ένα βαθμωτό μέτρο της καταλληλότητας του μοντέλου, με τα υπάρχοντα δεδομένα. Μια μετρική συνάρτηση είναι παρόμοια με τη συνάρτηση σφάλματος, εκτός από το ότι τα αποτελέσματα από την αξιολόγηση μιας μέτρησης δεν χρησιμοποιούνται κατά την εκπαίδευση του μοντέλου.

5.4.1 Πίνακας Σύγχυσης - Confusion Matrix

Ο πίνακας σύγχυσης (Confusion Matrix) παρέχει πληροφορίες σχετικά με τις πραγματικές και προβλεπόμενες κατηγοριοποιήσεις που πραγματοποιούνται από ένα σύστημα κατηγοριοποίησης. Η απόδοση αξιολογείται χρησιμοποιώντας τα δεδομένα στη μήτρα, τα οποία συμβολίζονται TP (True Positive), FN (False Negative), FP (False Positive), TN (True Negative) όπου το κάθε ένα συμβολίζει:

- TP: πλήθος θετικών δειγμάτων που έχουν προβλεφθεί σωστά από το μοντέλο.
- FN: πλήθος θετικών δειγμάτων που έχουν προβλεφθεί εσφαλμένα ως αρνητικά.

- FP: πλήθος αρνητικών δειγμάτων που έχουν προβλεφθεί εσφαλμένα ως θετικά.
- TN: πλήθος αρνητικών δειγμάτων που έχουν προβλεφθεί σωστά από το μοντέλο

Ο πίνακας σύγχυσης παρέχει μια πολύ εύκολη και ολοκληρωμένη συνολική εποπτεία απόδοσης του εκπαιδευμένου μοντέλου. Με βάση τον πίνακα σύγχυσης μπορεί να υπολογιστούν οι μετρικές που θα αναφέρουμε στην συνέχεια όπως η ορθότητα, η ακρίβεια, η ανάκληση, ο αρμονικός μέσος και η ειδικότητα.

5.4.2 Ορθότητα - Accuracy

Η ορθότητα είναι μια μετρική η οποία αξιολογεί το πόσο ακριβείς είναι οι προβλέψεις του μοντέλου σε σχέση με τα πραγματικά δεδομένα. Δηλαδή, η ορθότητα είναι ο λόγος των επιτυχών προβλέψεων (τόσο θετικών όσο και αρνητικών) προς το συνολικό αριθμό των περιπτώσεων που εξετάστηκαν. Όπως μπορεί κάποιος να παρατηρήσει πολύ εύκολα στο κεφάλαιο ανάλυσης της διεθνούς βιβλιογραφίας η ορθότητα είναι η βασική μετρική που αναφέρεται σχεδόν σε όλες τις δημοσιεύσεις.

$$\text{Accuracy } = \frac{\text{T}P + \text{TN}}{TP + FP + FN + TN} \tag{15}$$

5.4.3 Ακρίβεια - Precision

Ως ακρίβεια ορίζεται το ποσοστό των επιτυχών προβλέψεων προς το συνολικό αριθμό προβλέψεων μιας κλάσης από ένα μοντέλο. Η ακρίβεια μπορεί να θεωρηθεί ως μέτρο ποιότητας. Δίνεται από την παρακάτω σχέση:

$$\text{Precision } = \frac{TP}{TP + FP} \tag{16}$$

5.4.4 Ανάκληση - Recall

Η ανάκληση (γνωστή και ως ευαισθησία - sensitivity) είναι μια κλασική μετρική αξιολόγησης, η οποία στηρίζεται στην έννοια βασικής αλήθειας (ground truth). Ορίζεται ως το κλάσμα των επιτυχών προβλέψεων μιας κλάσης προς τα συνολικά παραδείγματα αυτής. Η ανάκληση μπορεί να θεωρηθεί ως μέτρο πληρότητας ή ποσότητας.

$$\text{Recall } = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

5.4.5 Αρμονικός Μέσος - F1-score

Η μετρική αυτή ορίζεται ως ο αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης. Το εύρος τιμών της μετρικής F1 είναι το διάστημα [0,1] με την τιμή 1 να αντιστοιχεί στην τέλεια ακρίβεια και ανάκληση. Η μετρική F1 είναι επίσης μια μετρική που εμφανίζεται αρκετά συχνά σε δημοσιεύσεις ειδικότερα σε προβλήματα ταξινόμησης με παραπάνω από δύο κλάσεις καθώς δίνει μια πιο "αντικειμενική" αξιολόγηση του μοντέλου σε όλες τις κλάσεις.

$$\text{F1-score } = \frac{2 \cdot \text{ Precision } \cdot \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{18}$$

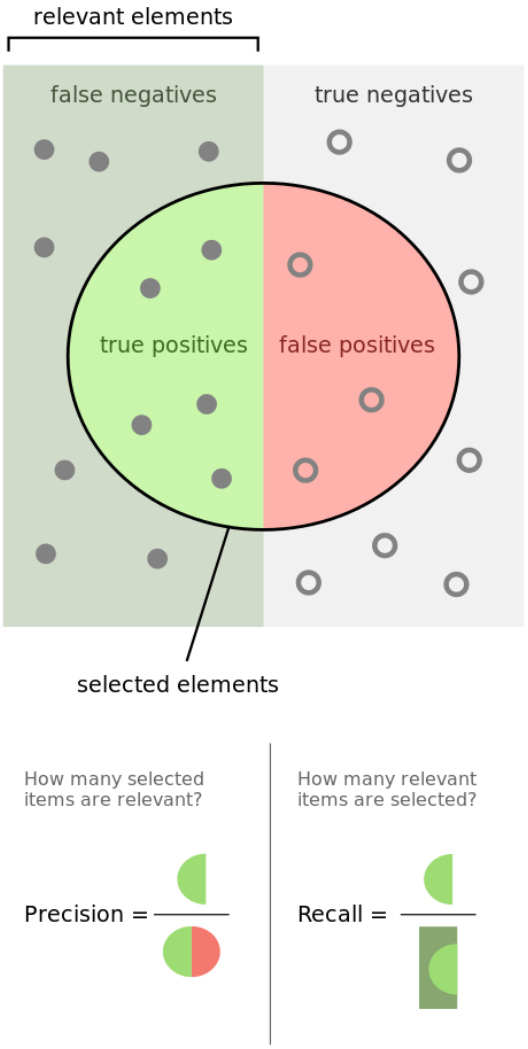
5.4.6 Ειδικότητα - Specificity

Η ειδικότητα αναφέρεται στο ποσοστό των πραγματικά αρνητικών δειγμάτων προς όλων των αρνητικών.

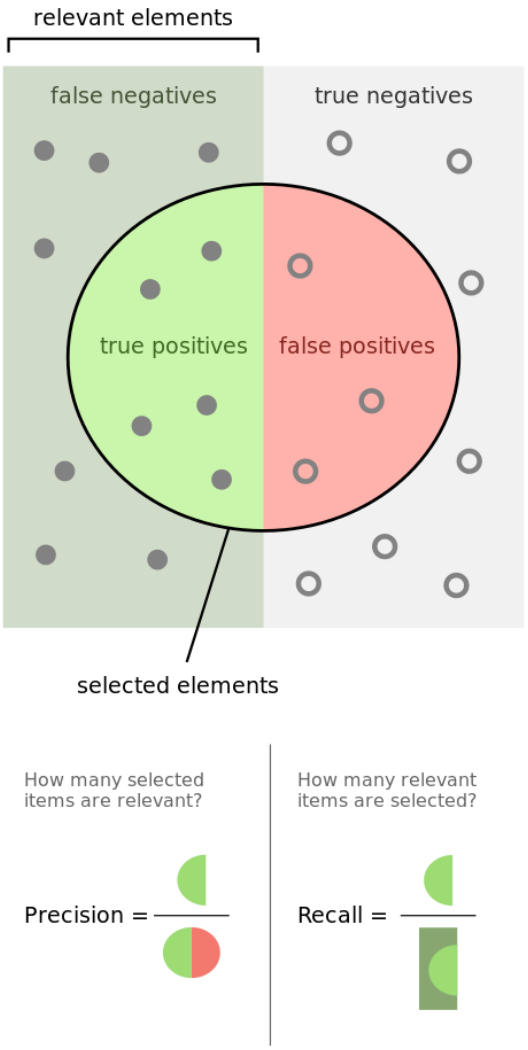
$$\text{Specificity} = \frac{TN}{TN + FN} \tag{19}$$

5.4.7 Γραφική Αναπαράσταση

Επειδή κάποιες μετρικές είναι αρκετά εύκολο να μπερδευτούν μεταξύ τους ο καλύτερος τρόπος για να τις θυμόμαστε είναι με την βοήθεια των ακόλουθων εικόνων:



Σχήμα 17: Σχηματική αναπαράσταση της ακρίβειας και της ανάκλησης
Πηγή: Precision and recall [83]



Σχήμα 18: Σχηματική αναπαράσταση της ανάκλησης και της ειδικότητας
Πηγή: Sensitivity and Specificity [84]

6 Πειραματική Αξιολόγηση

Έχοντας πλέον στην διάθεση μας τα απαραίτητα δεδομένα στην μορφή που θέλουμε, δεν μένει παρά να ξεκινήσουμε τα πειράματα που θα μας οδηγήσουν στην λύση που πληροί τις προϋποθέσεις που θέσαμε όσο το δυνατόν καλύτερα. Για κάθε πείραμα που θα περιγράψουμε στην συνέχεια χρησιμοποιήθηκε ένα τυπικό $80-20$ split των δεδομένων σε δεδομένα εκπαίδευση και ελέγχου. Επιπλέον ένα μικρό ποσοστό των δεδομένων εκπαίδευσης χρησιμοποιήθηκε για δεδομένα επαλήθευσης εντός της εκπαίδευσης. Δυστυχώς, ήρθαμε νωρίς αντιμέτωποι με ένα πρόβλημα που θα μας απασχολήσει σε όλη την έκταση της συγκεκριμένης ενότητας. Το πρόβλημα αφορά το μέγεθος των δεδομένων. Όπως είδαμε σε προηγούμενο κεφάλαιο φροντίσαμε οι διαστάσεις των εικόνων που προκύπτουν από την μετατροπή των εκτελέσιμων σε εικόνες να είναι μεταβλητές ανάλογα με το μέγεθος του εκτελέσιμου. Το "κβαντισμένο" μέγεθος ήταν το πλάτος της εικόνας αφού αφήναμε το μήκος ελεύθερο. Έτσι, ενώ είχαμε εικόνες με 8 πιθανά πλάτη, τα μήκη ήταν ελεύθερα. Η τεχνική αυτή οδήγησε σε δύο βασικά προβλήματα. Πρώτον, το framework του Pytorch όπου γράφτηκαν όλα τα πειράματα δεν υποστηρίζει εικόνες με διαφορετικά μεγέθη στο ίδιο batch. Μια λύση θα ήταν να ομαδοποιήσουμε τις εικόνες ανά μεγέθη και να φτιάξουμε μεταβλητά batches. Αυτό όμως εισάγει τρομερή μεροληψία στο σύστημα αφού οι εικόνες εκπαίδευσης δεν είναι τυχαίες στο κάθε batch αλλά μοιράζονται τα ίδια μεγέθη. Μια άλλη λύση θα ήταν να κάνουμε pad όλες τις εικόνες με μηδενικά για να φτάσουμε σε διαστάσεις την μεγαλύτερη εικόνα στο dataset. Αυτή η λύση μας οδηγεί στο δεύτερο πρόβλημα: τον χώρο που καταλαμβάνουν οι εν λόγω εικόνες στο σύστημα. Χωρίς καμία επεξεργασία ήταν αδύνατο να χωρέσουμε τις πολύ μεγάλες εικόνες στην μνήμη της κάρτας γραφικών. Σε αυτό το σημείο ίσως πρέπει ν' αναφερθούν τα χαρακτηριστικά του συστήματος που τρέχουμε τα πειράματα:

- Λειτουργικό σύστημα: Ubuntu και Windows 10
- CPU: Ryzen 3800x
- RAM: 32gb DRR4 3200MHZ
- GPU: RTX 2080 SUPER OC 8GB

Είναι λοιπόν σαφές ότι οποιοδήποτε μηχάνημα με πολύ καλύτερα χαρακτηριστικά δεν θα αποτελούσε έναν προσωπικό υπολογιστή αλλά ένα μηχάνημα συγκεκριμένα για εφαρμογές μηχανικής μάθησης. Οπότε, είμαστε αναγκασμένοι να βρούμε λύσεις στο πρόβλημα του χώρου και των διαστάσεων, ώστε να ολοκληρώσουμε τα πειράματα σωστά και σε εύλογο χρονικό διάστημα. Εξερευνώντας τις επιλογές μας στην διεθνή βιβλιογραφία είδαμε ότι η πιο συνηθισμένη τακτική για την αντιμετώπιση του συγκεκριμένου προβλήματος είναι ένας συνδυασμός περικοπής των εικόνων και η επεξεργασία/μετασχηματισμός των εικόνων για την εξαγωγή χαρακτηριστικών ή νέων εικόνων με σκοπό την τροφοδότηση των δικτύων. Το πρώτο βήμα, το βήμα της περικοπής, το υιοθετήσαμε και εμείς με μια διαφοροποίηση. Δεν περικόψαμε μόνο τις μεγάλες εικόνες, μεγαλώσαμε και τις μικρές με γέμισμα των στοιχείων με μηδενικά έτσι ώστε να έχουμε ένα σύνολο δεδομένων με εικόνες του ιδίου μεγέθους. Το μέγεθος που φαινόταν πιο λογικό είναι το τετράγωνο 1024×1024. Δηλαδή, κρατήσαμε όλο το πλάτος σε όλες τις εικόνες αλλά περιορίσαμε και το ύψος στα 1024 pixel.

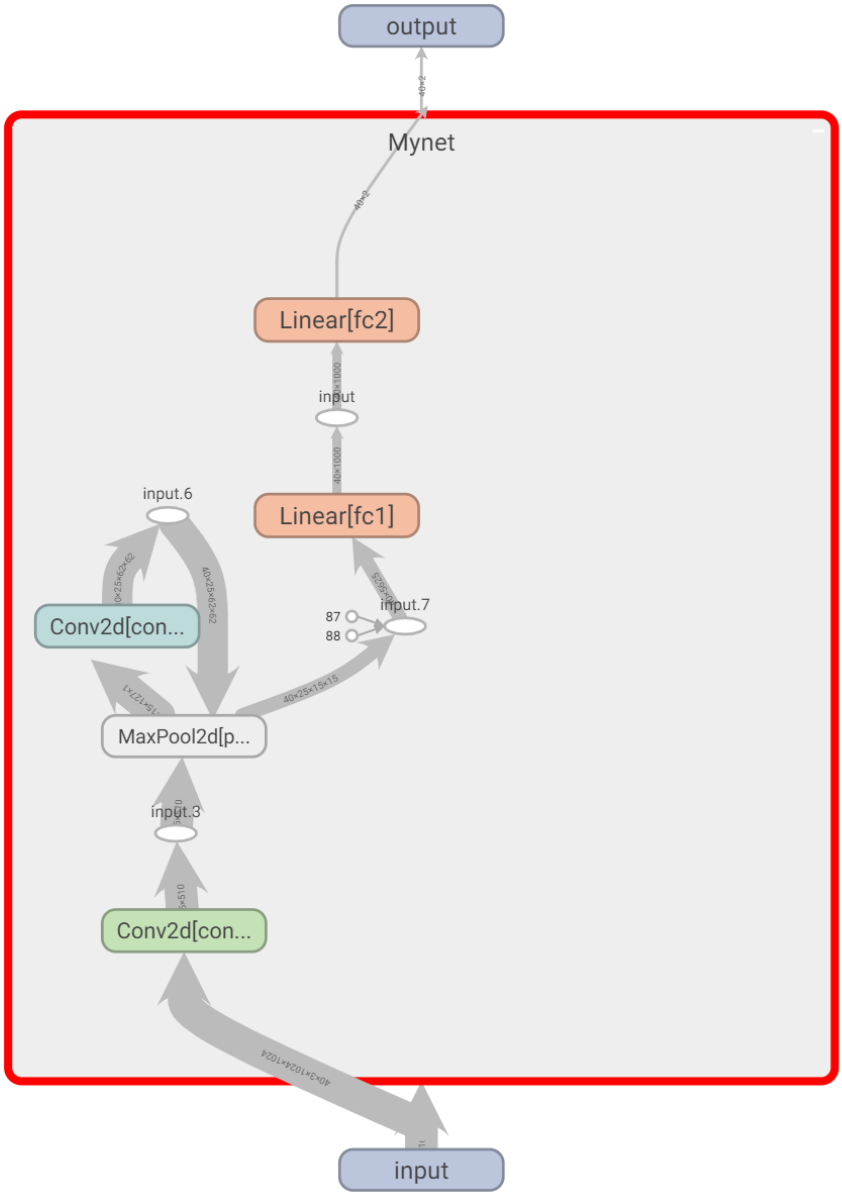
Η πορεία που ακολουθήσαμε στην εκτέλεση των πειραμάτων είχε την ακόλουθη λογική. Πρώτον, οφείλαμε να θέσουμε ένα κατώφλι, τον πιο απλό συνδυασμό δεδομένων και δικτύου για να έχουμε ένα σημείο εκκίνησης, οποιοδήποτε επόμενο πείραμα θα πρέπει να βελτιώνει το πρώτο και σε θέμα ακρίβειας και σε θέμα χρόνου. Στο επόμενο πείραμα, μελετάμε

την συμβολή ενός δικτύου autoencoder στην επεξεργασία των δεδομένων για την μείωση των διαστάσεών τους και για την εξαγωγή των σημαντικότερων χαρακτηριστικών. Έπειτα, αξίζει να επαναλάβουμε το πρώτο πείραμα με την διαφορά ότι τα δεδομένα έχουν περάσει πρώτα από το δίκτυο autoencoder, να δούμε δηλαδή πόσο μπορούμε να εκμεταλλευτούμε την πολύ απλή δομή του πρώτου πειράματος για την εξαγωγή καλύτερων και γρηγορότερων αποτελεσμάτων. Τέλος, επικεντρωνόμαστε στην επίτευξη καλύτερης ακρίβειας βασισμένοι στα δεδομένα που έχουν περάσει από το δίκτυο autoencoder. Για τον λόγο αυτό χρησιμοποιούμε ένα πολύ γνωστό και πανίσχυρο προ εκπαιδευμένο δίκτυο, το ResNet18.

6.1 Πείραμα 1^o - Μελέτη απλής δομής δικτύου CNN

Η πρώτη μας απόπειρα ήταν η δημιουργία ενός απλού δικτύου CNN για να έχουμε μια "βάση" που θα θέλαμε να ξεπεράσουμε σε πιο σύνθετες δομές. Το δίκτυο έχει την ακόλουθη δομή:

1. Το πρώτο συνελικτικό επίπεδο με πυρήνα 5×5 και έξοδο 15 feature maps.
2. MaxPooling επίπεδο με πυρήνα 4×4 και βήμα 4.
3. Δεύτερο συνελικτικό επίπεδο με πυρήνα 5×5 και έξοδο 25 feature maps.
4. MaxPooling επίπεδο με πυρήνα 4×4 και βήμα 4.
5. Το πρώτο Fully connected επίπεδο με είσοδο 5625 και έξοδο 1000.
6. Το τελικό Fully Connected επίπεδο με είσοδο 1000 και έξοδο 2, τις 2 πιθανές κατηγορίες.



Σχήμα 19: Το πρώτο δίκτυο CNN για τον εντοπισμό κακόβουλου λογισμικού

Οι υπερπαράμετροι του συγκεκριμένου πειράματος είναι οι εξής:

- **Ρυθμός μάθησης (lr):** 0.01 με εκθετική μείωση με συντελεστή 0.9 σε κάθε εποχή.
- **Αριθμός εποχών (epochs):** 50
- **Αριθμός δειγμάτων σε κάθε πέρασμα (mini-batch size):** 40
- **Αλγόριθμος εκπαίδευσης:** Adagrad [85]
- **Διαχωρισμός Δεδομένων:**
  - **Δεδομένα εκπαίδευσης:** 4600
  - **Δεδομένα επικύρωσης:** Από τα 4600 τα 460
  - **Δεδομένα αξιολόγησης:** 1150
- **Περιγραφή Δεδομένων Εισόδου:** Εικόνες 3-καναλιών διαστάσεων 1024×1024, συνολικά $1024 * 1024 * 3 = 3,145,728$ στοιχεία.

Με τις συγκεκριμένες υπερπαραμέτρους η συνολική εκπαίδευση του δικτύου ολοκληρώθηκε σε **1 ώρα και 20 λεπτά** με τα ακόλουθα αποτελέσματα:

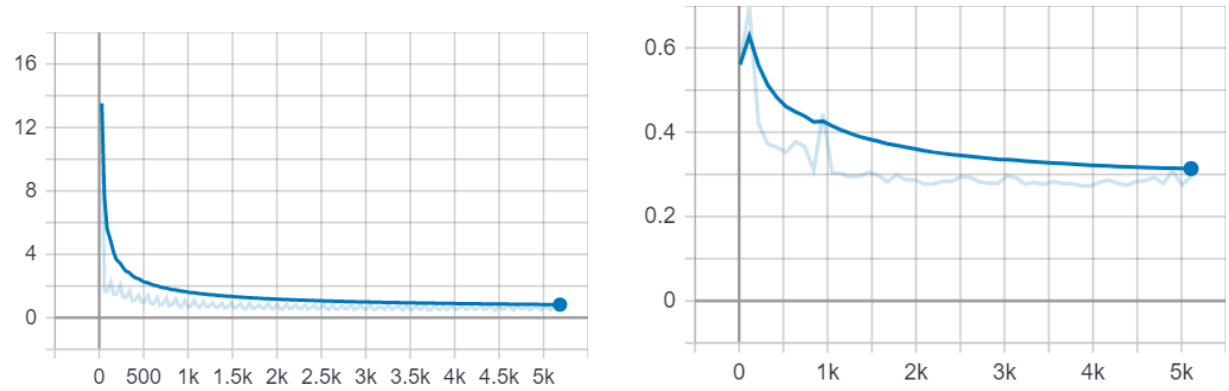
	precision	recall	f1-score	support
Benign	0.90	0.92	0.91	430
Malware	0.95	0.94	0.95	720
accuracy			0.93	1150
macro avg	0.93	0.93	0.93	1150
weighted avg	0.93	0.93	0.93	1150

Σχήμα 20: Αποτελέσματα του πρώτου πειράματος στο σύνολο αξιολόγησης

και πίνακα σύγχυσης:

	Predicted Benign	Predicted Malware
Benign	394	36
Malware	42	678

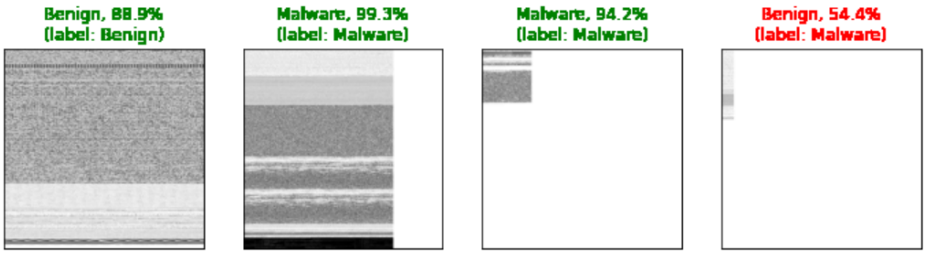
Ενώ η πορεία της εκπαίδευσης και της επικύρωσης, με δείγμα κάθε 30 mini-batch, φαίνεται στα κάτωθι γραφήματα:



(a) Loss στην εκπαίδευση (b) Loss στην επικύρωση

Σχήμα 21: Γραφικές αναπαραστάσεις των επιδόσεων του πρώτου πειράματος

Όπου με αχνή γραμμή είναι τα πραγματικά πειραματικά δεδομένα και με έντονη είναι μια ομαλοποιημένη εκδοχή της πειραματικής γραμμής για να φαίνεται καλύτερα η πτωτική τάση. Τέλος παραθέτουμε και μια εικόνα με 4 προβλέψεις του δικτύου στα δεδομένα επικύρωσης:



Σχήμα 22: 3 σωστές και 1 λανθασμένη πρόβλεψη του δικτύου στο πρώτο πείραμα

Στην λάθος πρόβλεψη το δίκτυο είχε το χαμηλότερο ποσοστό σιγουριάς, μόλις 54.4%, γεγονός το οποίο μας προδιαθέτει θετικά ότι με μερικές τροποποιήσεις θα μπορέσουμε να πετύχουμε καλύτερες επιδόσεις στις μετρικές.
Ολοκληρώνοντας το πρώτο πείραμα είναι ξεκάθαρο ότι έχουμε θέσει ένα πολύ καλό σημείο εκκίνησης πάνω στο οποίο θα επέμβουμε για να αντιμετωπίσουμε δύο βασικά προβλήματα. Πρωτίστως, τον χρόνο εκπαίδευσης, είναι απαγορευτικό για πέντε χιλιάδες δείγματα να χρειάζεται για ένα απλό δίκτυο σαν αυτό εκπαίδευση διάρκειας σχεδόν 90 λεπτών και έπειτα την ακρίβεια, τα αποτελέσματα είναι πάνω από το 93% αλλά απέχουν 2 ποσοστιαίες μονάδες για να επιτύχουμε το όριο του 95%+ που έχουμε θέσει στις απαιτήσεις του ανιχνευτή.

6.2 Ενδιάμεσο Πείραμα - Autoencoder δίκτυο για την μείωση διαστάσεων και την εξαγωγή χαρακτηριστικών

Προτού προχωρήσουμε σε πιο σύνθετες υλοποιήσεις, είναι σαφές ότι χρειαζόμαστε ένα σχέδιο για να μπορέσουμε να αντιμετωπίσουμε, έστω και θεωρητικά, τα δύο προβλήματα που παρουσιάστηκαν από το πρώτο πείραμα, ο χρόνος εκπαίδευσης και η ακρίβεια. Η μείωση της πολυπλοκότητας του δικτύου είναι μια μη βιώσιμη επιλογή καθώς θεωρούμε ότι η συγκεκριμένη μορφολογία που παρουσιάσαμε στο πρώτο πείραμα είναι το ελάχιστο δυνατό για την συγκεκριμένη εφαρμογή. Όμως, έχουμε ήδη αναφέρει ότι μια τεχνική μείωσης διαστάσεων θα φανεί ανεκτίμητη, καθώς μετά την εφαρμογή της το τελικό αποτέλεσμα θα έχει μικρότερες διαστάσεις και ότι απέμεινε από το αρχικό δείγμα, θεωρητικά, θα διατηρεί τα πιο σημαντικά χαρακτηριστικά με αποτέλεσμα το δίκτυο να μπορεί να επικεντρωθεί σε αυτά καλύτερα. Με γνώμονα τον προσανατολισμό της παρούσας εργασίας στην Βαθιά Μάθηση αποφασίσαμε να αναπτύξουμε ένα μοντέλο autoencoder για την αρχική επεξεργασία των δεδομένων. Το μοντέλο έχει την ακόλουθη δομή:

1. Το πρώτο συνελικτικό επίπεδο με πυρήνα 3×3, padding 1×1 και έξοδο 16 feature maps.

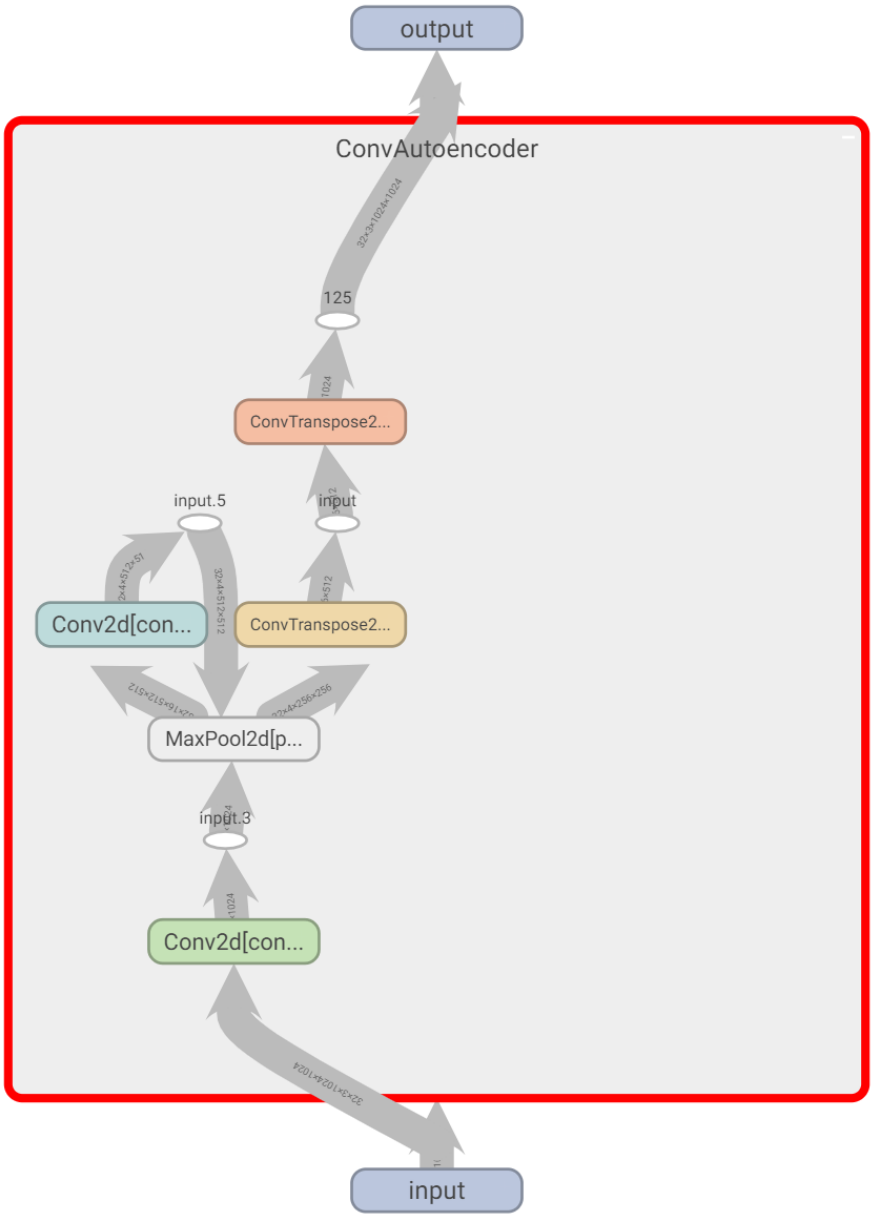
2. MaxPooling επίπεδο με πυρήνα 2×2 και βήμα 2.

3. Δεύτερο συνελικτικό επίπεδο με πυρήνα 3×3, padding 1×1 και έξοδο 4 feature maps.

4. MaxPooling επίπεδο με πυρήνα 2×2 και βήμα 2.

5. Τρίτο *ανάστροφο* [86] συνελικτικό επίπεδο με πυρήνα 2×2, stride 2×2 και έξοδο 16 feature maps.

6. Τέταρτο *ανάστροφο* [86] συνελικτικό επίπεδο με πυρήνα 2×2, stride 2×2 και έξοδο 3 feature maps.



Σχήμα 23: Το δίκτυο autoencoder για μείωση διαστάσεων

Έχουμε ήδη αναφέρει, και με μαθηματικό ορισμό, ότι ο ιδανικός στόχος ενός μοντέλου autoencoder είναι η ταύτιση της εξόδου με την είσοδο. Ακόμα και αν δεν το είχαμε επισημάνει, η δομή που περιγράψαμε μας προϊδεάζει για κάτι τέτοιο. Στα επίπεδα 1-4 υπάρχει μια συρρίκνωση της εισόδου ενώ τα επίπεδα 5 και 6 προσπαθούν να ξανά μεγαλώσουν τα δεδομένα στις αρχικές τους διαστάσεις. Αξίζει να δούμε με συγκεκριμένα νούμερα πόσο θα επηρεαστούν τα δεδομένα στην παρούσα εφαρμογή. Οι ακόλουθες εξισώσεις μας δίνουν το μέγεθος, μήκος και πλάτος, της εικόνας που θα προκύψει αν περάσει από ένα συνελικτικό επίπεδο.

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{ kernel\_size } [0] - 1) - 1}{\text{stride } [0]} + 1 \right\rfloor \tag{20}$$

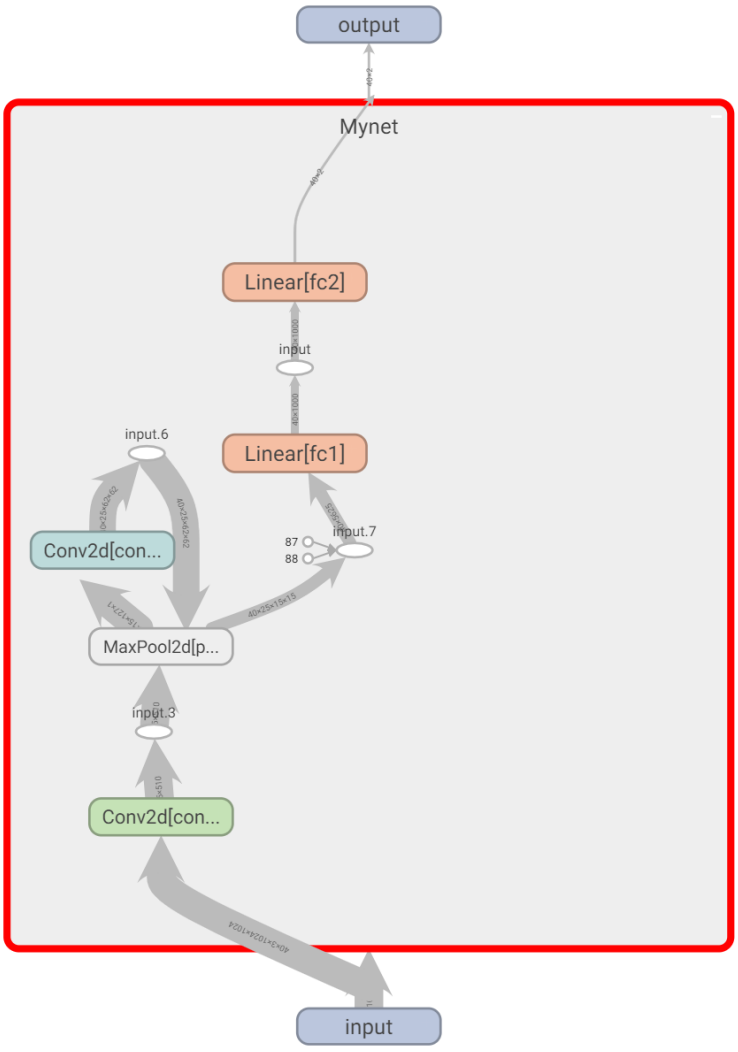
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{ kernel\_size } [1] - 1) - 1}{\text{stride } [1]} + 1 \right\rfloor \tag{21}$$

Οι αρχικές εικόνες έχουν διαστάσεις 1024×1024 οπότε αν γίνουν αναλυτικά οι πράξεις για το πρώτο και τρίτο επίπεδο θα διαπιστώσει κανείς ότι το δίκτυο είναι δομημένο με τέτοιον τρόπο ώστε τα συνελικτικά επίπεδα να μην αλλάζουν τις διαστάσεις των εικόνων. Άρα, η μείωση των διαστάσεων γίνεται εξ' ολοκλήρου από τους pooling layers. Το δίκτυο παρουσιάζει δύο πιθανά σημεία από τα οποία θα μπορούσαμε να εξάγουμε ενδιάμεσα αποτελέσματα και να τα χρησιμοποιήσουμε σαν είσοδο στο δίκτυο του ανιχνευτή. Συγκεκριμένα, μια ενδιάμεση έξοδος στο επίπεδο 3 θα δώσει ένα αντικείμενο - "εικόνα" με διαστάσεις $512 \times 512 \times 4$ ενώ μια ενδιάμεση έξοδος στο επίπεδο 4 θα δώσει ένα αντικείμενο - "εικόνα" με διαστάσεις $256 \times 256 \times 4$. Επιλέξαμε να συνεχίσουμε τις δοκιμές μας με την ενδιάμεση έξοδο από το επίπεδο 3 για τον λόγο ότι $1024 \times 1024 \times 1 = 512 \times 512 \times 4$ δηλαδή έχουμε ήδη υπο-τριπλασιάσει την είσοδο. Δεν χρειάζεται, εκτός αν κριθεί απολύτως απαραίτητο, να μειώσουμε και άλλο τις διαστάσεις από τα δεδομένα εισόδου. Ένα πρόβλημα παρουσιάζεται με την αποθήκευση των ενδιάμεσων αποτελεσμάτων, δεδομένου ότι δεν μπορούν όλα αν χωρέσουν ταυτόχρονα στην μνήμη του συστήματος ή της κάρτας γραφικών έπρεπε τα ενδιάμεσα αποτελέσματα να αποθηκεύονται στον δίσκο με την πλήρη τους μορφή ως tensors. Κάθε ενδιάμεσος tensor καταλαμβάνει χώρο 4MB, δηλαδή τα "νέα" δεδομένα καταλαμβάνουν συνολικό χώρο 22.4GB.

6.3 Πείραμα $2^ο$ - Συνδυασμός απλής δομής CNN και αποτελεσμάτων δικτύου Autoencoder

Φυσική συνέχεια της λογικής που ακολουθούμε είναι η επανάληψη του πρώτου δικτύου με τα καινούρια δεδομένα που αποθηκεύσαμε από τις ενδιάμεσες εξόδους του δικτύου autoencoder. Το δίκτυο έχει την ακόλουθη δομή:

1. Το πρώτο συνελικτικό επίπεδο με πυρήνα 5×5 και έξοδο 15 feature maps.
2. MaxPooling επίπεδο με πυρήνα 4×4 και βήμα 4.
3. Δεύτερο συνελικτικό επίπεδο με πυρήνα 5×5 και έξοδο 25 feature maps.
4. MaxPooling επίπεδο με πυρήνα 4×4 και βήμα 4.
5. Το πρώτο Fully connected επίπεδο με είσοδο 5625 και έξοδο 1000.
6. Το τελικό Fully Connected επίπεδο με είσοδο 1000 και έξοδο 2, τις 2 πιθανές κατηγορίες.



Σχήμα 24: Το δεύτερο δίκτυο CNN για τον εντοπισμό κακόβουλου λογισμικού

Οι υπερπαράμετροι του συγκεκριμένου πειράματος είναι οι εξής:

- **Ρυθμός μάθησης (lr):** 0.01 με εκθετική μείωση με συντελεστή 0.9 σε κάθε εποχή.
- **Αριθμός εποχών (epochs):** 50
- **Αριθμός δειγμάτων σε κάθε πέρασμα (mini-batch size):** 40
- **Αλγόριθμος εκπαίδευσης:** Adagrad [85]
- **Διαχωρισμός Δεδομένων:**
  - **Δεδομένα εκπαίδευσης:** 4600
  - **Δεδομένα επικύρωσης:** Από τα 4600 τα 460
  - **Δεδομένα αξιολόγησης:** 1150
- **Περιγραφή Δεδομένων Εισόδου:** Εικόνες 4-καναλιών διαστάσεων 512×512, συνολικά $512 * 512 * 4 = 1,048,576$ στοιχεία.

Με τις συγκεκριμένες υπερπαραμέτρους η συνολική εκπαίδευση του δικτύου ολοκληρώθηκε σε **15 λεπτά** με τα ακόλουθα αποτελέσματα:

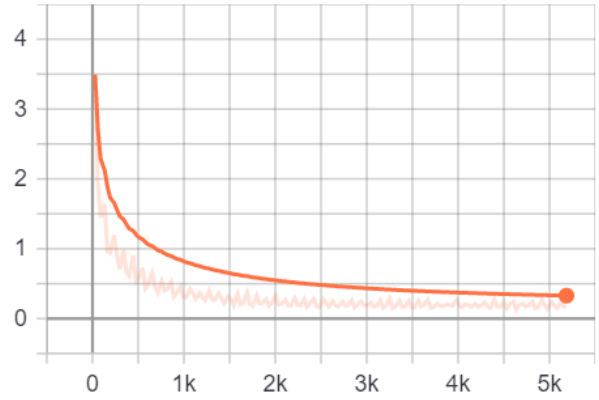
	precision	recall	f1-score	support
Benign	0.91	0.91	0.91	399
Malware	0.95	0.95	0.95	751
accuracy			0.94	1150
macro avg	0.93	0.93	0.93	1150
weighted avg	0.94	0.94	0.94	1150

Σχήμα 25: Αποτελέσματα του δεύτερου πειράματος στο σύνολο αξιολόγησης

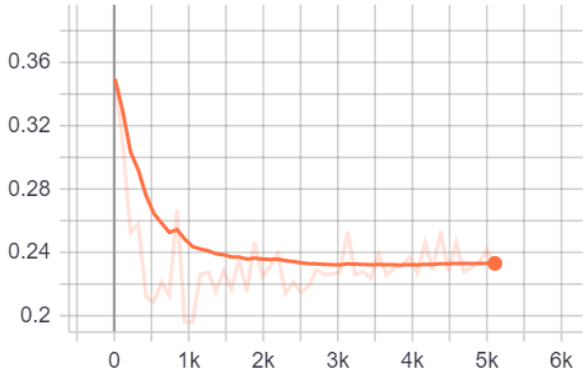
και πίνακα σύγχυσης:

	Predicted Benign	Predicted Malware
Benign	364	35
Malware	36	715

Ενώ η πορεία της εκπαίδευσης και της επικύρωσης, με δείγμα κάθε 30 mini-batch, φαίνεται στα κάτωθι γραφήματα:



(a) Loss στην εκπαίδευση



(b) Loss στην επικύρωση

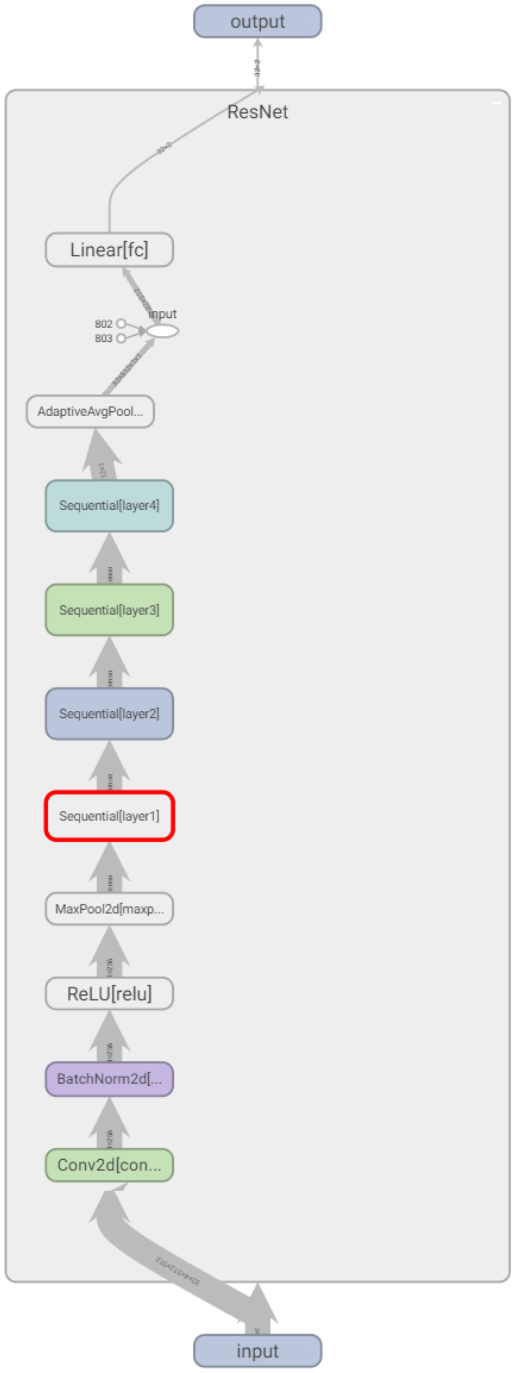
Σχήμα 26: Γραφικές αναπαραστάσεις των επιδόσεων του δεύτερου πειράματος

Όπου ακόμα μια φορά με αχνή γραμμή είναι τα πραγματικά πειραματικά δεδομένα και με την έντονη η ομαλοποιημένη τάση που ακολουθούν αυτά τα δεδομένα. Είναι πραγματικά απίστευτο το πόσο βελτιώθηκε και ο χρόνος εκπαίδευσης, μειώθηκε 6 φορές, και ότι ταυτόχρονα αυξήθηκε και η απόδοση του δικτύου στις μετρικές. Η μοναδική αλλαγή όπως είναι εμφανές από τα χαρακτηριστικά του πειράματος, σε σχέση με το πρώτο, είναι μόνο τα δεδομένα εισόδου. Πλέον, δεν έχουμε τις αρχικές εικόνες αλλά τις ενδιάμεσες εξόδους από το δίκτυο autoencoder. Το μόνο που μένει είναι να καλύψουμε την τελευταία απαίτηση· η ακρίβεια του ανιχνευτή να είναι πάνω από 95%.

6.4 Πείραμα $3^ο$ - Συνδυασμός σύνθετης δομής CNN ResNet18 και αποτελεσμάτων δικτύου Autoencoder

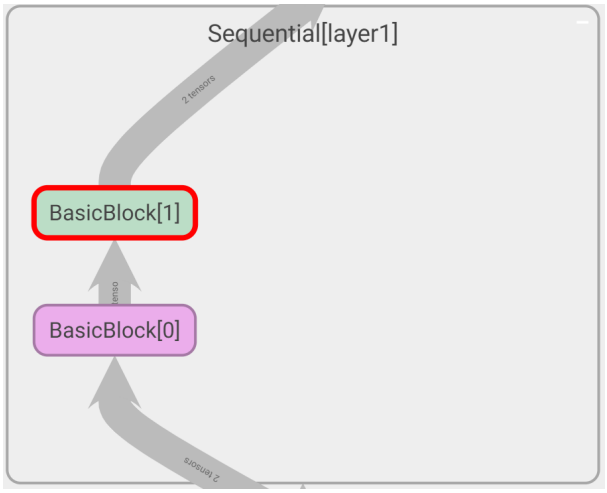
Φαίνεται ότι έχουμε καλύψει τους στόχους που είχαμε θέσει για τον ανιχνευτή, πλην ενός, της ακρίβειας. Είδαμε ότι το πέρασμα των δεδομένων από το δίκτυο autoencoder αύξησε την ακρίβεια και μείωσε το χρόνο εκπαίδευσης δραματικά. Συνεπώς δεν μοιάζει σωστή η επιλογή να ασχοληθούμε ακόμα περισσότερο με την μορφή των δεδομένων για την αύξηση της ακρίβειας. Αυτό που χρειάζεται είναι να περάσουμε σε πιο βαθιές και πολύπλοκες δομές δικτύου. Το να περάσουμε σε αυθαίρετες δομές δεν είναι κάτι θεμιτό καθώς όσο αυξάνεται το βάθος και η πολυπλοκότητα της δομής τόσο αυξάνεται και ο χρόνος εκπαίδευσης, και πολλές φορές η συσχέτιση δεν είναι καν γραμμική. Έτσι, στρέψαμε την προσοχή μας στο ResNet. Το ResNet είναι ένα δίκτυο που έχει αποδείξει την δύναμη του σε πολυάριθμες περιπτώσεις. Φυσικά, και πάλι η εκ του μηδενός εκπαίδευση του δεν συνίσταται καθώς ο χρόνος εκπαίδευσης για να έχουμε τα θεμιτά αποτελέσματα θα ήταν απαγορευτικός. Αντιθέτως, το framework του Pytorch προσφέρει την επιλογή να χρησιμοποιηθεί μια προ εκπαιδευμένη εκδοχή του εν λόγω δικτύου. Το δίκτυο είναι προ εκπαιδευμένο πάνω στο ImageNet[18] και είναι προφανές ότι οι "εικόνες" που έχουμε στο δικό μας δίκτυο δεν πλησιάζουν κοντά σε καμία κατηγορία του ImageNet. Συνεπώς δεν θα ακολουθήσουμε την συνηθισμένη τακτική στις εφαρμογές του transfer learning, η οποία είναι να εκπαιδεύονται μόνο οι τελικοί layers που κάνουν την ταξινόμηση και να αφήνουν τους άλλους απείραχτους με τα προ φορτωμένα βάρη και biases. Αντιθέτως, επιλέξαμε να εκπαιδεύσουμε ολόκληρο το δίκτυο, άρα από την προ εκπαίδευση αυτό που κερδίσαμε είναι τα βάρη να μην αρχικοποιούνται με τυχαίες τιμές, αλλά με τιμές που είναι κοντά στις βέλτιστες που αναζητούμε στην νέα εφαρμογή. Δεδομένου του μεγέθους του δικτύου είμαστε αναγκασμένοι να μειώσουμε το μέγεθος του batch

size στο παρόν πείραμα, αυξάνοντας τον χρόνο εκπαίδευσης. Για να περιγράψουμε την δομή του δικτύου ResNet18 που διατίθεται από το Pytorch είναι καλύτερο να βασιστούμε στην σχηματική αναπαράσταση παρά σε περιγραφική χάριν απλότητας. Ξεκινώντας από έξω προς τα μέσα έχουμε την εξής εικόνα:



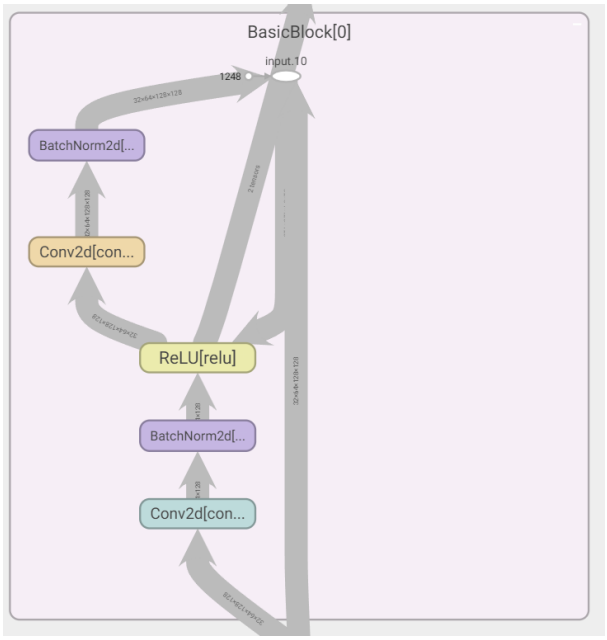
Σχήμα 27: Γενική εποπτεία του δικτύου ResNet18

Ήδη φαίνεται το βάθος και η πολυπλοκότητα του ResNet18 σε σχέση με το δικό μας δίκτυο που είχαμε υλοποιήσει στα προηγούμενα πειράματα. Παρατηρούμε ότι υπάρχουν κάποια block που επαναλαμβάνονται με το όνομα “Sequential”. Αυτά τα blocks κρύβουν δύο μικρότερα block:



Σχήμα 28: Ένα Sequential Block του δικτύου Resnet18

Επιτέλους φτάνουμε στην καρδιά του δικτύου, τα "Basic Blocks" είναι τα γνωστά residual blocks:



Σχήμα 29: Ένα Basic Block του δικτύου Resnet18

Πλέον φαίνεται ξεκάθαρα το μοναδικό χαρακτηριστικό που εισήγαγε η αρχιτεκτονική των ResNet που είναι το skip connection, υπάρχει σύνδεση της πληροφορίας από το ένα basic block στο άλλο χωρίς καμία αλλοίωση. Μια πολύ σημαντική προσθήκη που επιτρέπει στα εν λόγω δίκτυα να έχουν πολύ καλά αποτελέσματα χωρίς να χρειάζεται να καταφεύγουν σε πολύ βαθιές αρχιτεκτονικές που προφανώς ούτε αυτές είναι σε θέση να εγγυώνται καλύτερα αποτελέσματα. Στην πραγματικότητα ο λόγος που αναπτύχθηκαν τα ResNet είναι γιατί τα κλασσικά δίκτυα κάποια στιγμή σταμάτησαν να δίνουν καλύτερα αποτελέσματα όσο μεγάλωνε το βάθος τους και η πολυπλοκότητά τους. Ας περάσουμε στις υπερπαραμέτρους του συγκεκριμένου πειράματος:

- **Ρυθμός μάθησης (lr):** 0.001 με εκθετική μείωση με συντελεστή 0.9 σε κάθε εποχή.

- **Αριθμός εποχών (epochs):** 50
- **Αριθμός δειγμάτων σε κάθε πέρασμα (mini-batch size):** 32
- **Αλγόριθμος εκπαίδευσης:** Adagrad [85]
- **Διαχωρισμός Δεδομένων:**
  - **Δεδομένα εκπαίδευσης:** 4600
  - **Δεδομένα επικύρωσης:** Από τα 4600 τα 460
  - **Δεδομένα αξιολόγησης:** 1150
- **Περιγραφή Δεδομένων Εισόδου:** Εικόνες 4-καναλιών διαστάσεων 512×512, συνολικά $512 * 512 * 4 = 1,048,576$ στοιχεία.

Με μια απλή σύγκριση μεταξύ των πειραμάτων είναι εμφανείς οι αλλαγές. Πρώτον μειώθηκε ο συντελεστής μάθησης καθώς δεν θέλουμε να πειράξουμε πολύ το προ εκπαιδευμένο δίκτυο, θέλουμε απλά να το ωθήσουμε σιγά σιγά στην σωστή κατεύθυνση καθώς μεγαλύτερες αλλαγές σε έναν τόσο πολυδιάστατο χώρο θα μπορούσαν να καταστρέψουν την απόδοση του δικτύου εξ' ολοκλήρου. Η δεύτερη αλλαγή είναι το μέγεθος του batch size. Δοκιμές έδειξαν ότι όσο αυξάναμε το μέγεθος του batch size, μέχρι κάποια τιμή κοντά στο 256, υπήρχε βελτίωση στην τιμή της συνάρτησης κόστους. Δυστυχώς όμως, είμαστε αναγκασμένοι να μικρύνουμε το batch size για να μπορέσει να χωρέσει στην μνήμη της κάρτας γραφικών μαζί με το δίκτυο. Με τις συγκεκριμένες παραμέτρους η εκπαίδευση του δικτύου ολοκληρώθηκε σε **70 λεπτά** και έδωσε τα εξής αποτελέσματα:

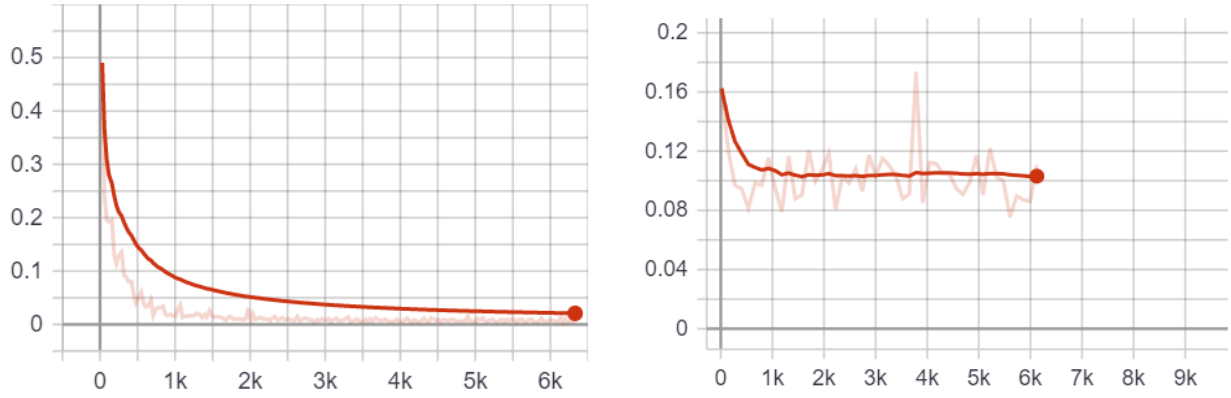
	precision	recall	f1-score	support
Benign	0.95	0.95	0.95	404
Malware	0.97	0.97	0.97	746
accuracy			0.96	1150
macro avg	0.96	0.96	0.96	1150
weighted avg	0.96	0.96	0.96	1150

Σχήμα 30: Αποτελέσματα του τρίτου πειράματος στα δεδομένα αξιολόγησης

με πίνακα σύγχυσης:

	Predicted Benign	Predicted Malware
Benign	383	21
Malware	20	726

Ενώ η πορεία της μάθησης κατά την διάρκεια της εκπαίδευση φαίνεται στα ακόλουθα γραφήματα:



(a) Loss στην εκπαίδευση

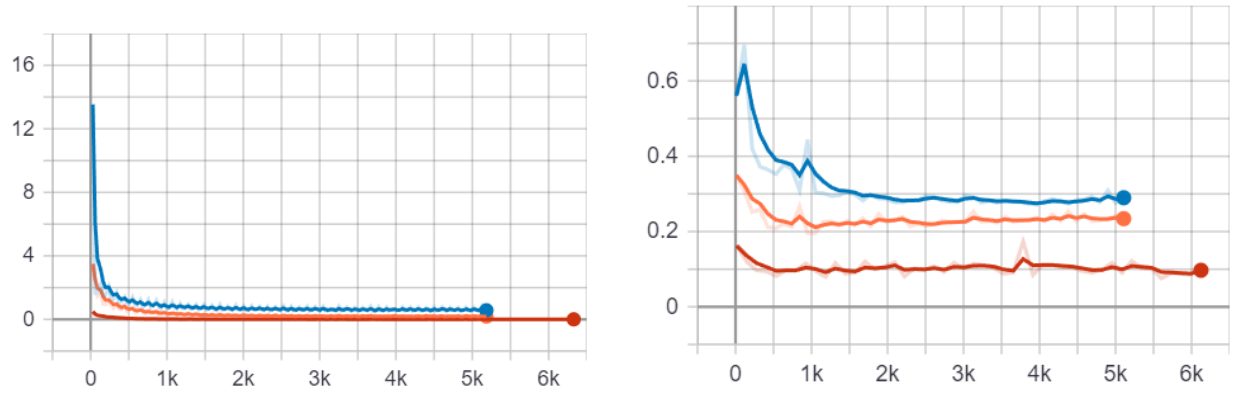
(b) Loss στην επικύρωση

Σχήμα 31: Γραφικές αναπαραστάσεις των επιδόσεων του τρίτου πειράματος

Είναι πραγματικά αξιοθαύμαστο το πόσο γρήγορα και καλά προσαρμόστηκε το προ εκπαιδευμένο μοντέλο στα δεδομένα μας, δεδομένου της πολυπλοκότητας του δικτύου. Συγκριτικά, από την τιμή loss που ξεκίνησε το δίκτυο ResNet εκεί κατέληξε, περίπου, το δίκτυο στο δεύτερο πείραμα. Επίσης, το γράφημα για τις τιμές loss στα δεδομένα επικύρωσης μπορεί να φαίνεται πολύ κακό, δεδομένου ότι ταλαντεύεται συνέχεια, αλλά αρκεί να κοιτάξουμε τον κάθετο άξονα και να παρατηρήσουμε την κλίμακα στην οποία βρισκόμαστε. Η ταλάντωση των τιμών υπάρχει σε ένα διάστημα εύρους 0.04, άρα δεν είναι κακό το γράφημα απλά για να δούμε την όποια μεταβολή χρειάστηκε να μεγεθύνουμε αρκετά την εικόνα. Με την ολοκλήρωση αυτού του πειράματος καλύψαμε όλες τις απαιτήσεις που θέσαμε για τον ανιχνευτή, οπότε το μόνο που μένει είναι μια σύγκριση μεταξύ των πειραμάτων για να παρουσιάσουμε πιθανές περιπτώσεις όπου ένα πείραμα είναι καλύτερο από κάποιο άλλο.

6.5 Σύγκριση Πειραμάτων και Αποτελεσμάτων

Έχουμε παρουσιάσει κάθε πείραμα χωριστά αλλά στην συγκεκριμένη ενότητα θα είναι πολύ χρήσιμο να δούμε τα συνδυαστικά γραφήματα των μοντέλων πως συμπεριφέρονται κατά την διάρκεια της εκπαίδευσης και της επικύρωσης.



(a) Loss στην εκπαίδευση (b) Loss στην επικύρωση

Μπλέ: πείραμα 1, **Πορτοκαλί**: Πείραμα 2, **Κόκκινο**: Πείραμα 3

Σχήμα 32: Κοινό γράφημα των επιδόσεων των τριών πειραμάτων

Αρχικά αξίζει να ξαναναφέρουμε τον λόγο για τον οποίο η κόκκινη καμπύλη εκτείνεται σε παραπάνω δείγματα σε σχέση με τις άλλες δύο. Ο λόγος είναι ότι το μοντέλο δεν μπορούσε να χωρέσει στην μνήμη της κάρτας γραφικών μαζί με τον αριθμό δειγμάτων που είχε το batch-size για τα προηγούμενα πειράματα. Έτσι, χρειάστηκε να μειωθεί το batch-size, με αποτέλεσμα να αυξηθούν οι φορές που γίνεται ανανέωση-εκπαίδευση των παραμέτρων του δικτύου. Από εκεί και πέρα οι διαφορές είναι εμφανέστατες και σημαντικές. Τα δίκτυα τα οποία χρησιμοποιούν τα δεδομένα από τον autoencoder είναι καλύτερα από θέμα ακρίβειας αλλά και από θέμα χρόνου εκπαίδευσης. Συνεπώς, μπορούμε με σιγουριά ν' απορρίψουμε σαν βιώσιμη πρόταση το πρώτο πείραμα. Αυτό μας αφήνει με δύο πειράματα:

1. Το δικό μας δίκτυο με δεδομένα από autoencoder
2. Το προ εκπαιδευμένο δίκτυο RensNet18 με δεδομένα από autoencoder

Στην πραγματικότητα μόνο το τρίτο πείραμα "κάλυψε" όλες τις απαιτήσεις που θέσαμε, αλλά υπάρχει μια πολύ μεγάλη διαφορά στον χρόνο εκπαίδευσης για μια σχετικά μικρή αύξηση της ακρίβειας, της τάξεως του 1-2%. Το γεγονός αυτό μας οδηγεί να κάνουμε μια σύνθετη πρόταση η οποία θα μπορεί να ανταποκριθεί σε δύο, ελαφρώς, διαφορετικά σενάρια. Συγκεκριμένα, εφαρμογές που απαιτούν λύσεις οι οποίες προσφέρουν μεγαλύτερη ασφάλεια και χρειάζονται επανεκπαίδευση σπανιότερα τότε προφανώς πρέπει να καταφύγουμε σε μια λύση ανάλογη του τρίτου πειράματος. Σε περίπτωση, όμως, που ο χρόνος είναι ζωτικής σημασίας και η ακρίβεια στις επιθέσεις μπορεί ν' αντισταθμιστεί με πιο συχνές επανεκπαίδευσης τότε δεν υπάρχει λόγος να καταφύγει κάποιος στο δίκτυο του τρίτου πειράματος αλλά μπορεί ν' αρκεστεί στην απλούστερη σύνθεση του δεύτερου πειράματος.

Συγκριτικά με την διεθνή βιβλιογραφία η πρόταση μας βελτιώνει κατά πολύ τον απαιτούμενο χρόνο εκπαίδευσης. Δυστυχώς, δεν μπορεί η πρόταση που υλοποιήσαμε να συγκριθεί επακριβώς με κάποια δημοσίευση καθώς δεν έχει μελετηθεί η χρήση εικόνων μόνο για

ανίχνευση σε συνδυασμό με κάποιο δίκτυο CNN. Όμως αυτό δεν μας εμποδίζει από το να κάνουμε κάποιες επιμέρους βασικές συγκρίσεις. Σε όλα τα πειράματα στην παρούσα εργασία ο αριθμός των εποχών ήταν 50, καθώς μια μεγαλύτερη αύξηση θα μπορούσε να εισάγει τον κίνδυνο της υπερεκπαίδευσης με τέτοιες επιδόσεις. Σε αντίστοιχη δημοσίευση [68], για κατηγοριοποίηση κακόβουλου λογισμικού χρειάστηκαν 2000(!) epochs για την επίτευξη των αποτελεσμάτων. Ο τεράστιος αριθμός epochs, σε συνδυασμό με το υποδεέστερο hardware και την αυξημένη πολυπλοκότητα του ResNet-50 είναι ισχυρές ενδείξεις, αν όχι αποδείξεις, ότι η εκπαίδευση του μοντέλου τους πήρε τουλάχιστον διπλάσιο χρόνο σε σχέση με την υλοποίηση που παρουσιάσαμε. Προφανώς ο τελικός στόχος διέφερε και είναι δικαιολογημένη η αύξηση στην πολυπλοκότητα και τις εποχές αλλά όχι σε τέτοιο βαθμό. Επίσης η συντριπτική πλειοψηφία των δημοσιεύσεων δεν αναφέρουν τον συνολικό χρόνο που χρειάστηκαν τα μοντέλα για την εκπαίδευση. Επιπλέον αναγνωρίζουμε ότι οι επιδόσεις που αναφέρονται σε κάποιες δημοσιεύσεις ξεπερνάνε κατά 2-3 ποσοστιαίες μονάδες την ακρίβεια από το καλύτερο πείραμα σε αυτή την εργασία, το πείραμα 3. Η ανταλλαγή μεταξύ απόδοσης και χρόνου δεν έγινε τυχαία. Από την αρχή είχαμε θέσει την απαίτηση η εκπαίδευση του δικτύου να είναι όσο το δυνατόν πιο σύντομη. Κάτι το οποίο δεν ισχύει για τις λύσεις που έχουν προταθεί από τις αντίστοιχες δημοσιεύσεις, καθώς φαίνεται ότι ο μόνος τους στόχος είναι η επίτευξη καλύτερων επιδόσεων (αύξηση πολυπλοκότητας μοντέλου, αύξηση πολυπλοκότητας χαρακτηριστικών, τεράστιος όγκος δεδομένων κοκ) και όχι ο συνδυασμός των επιδόσεων με χρήση σε πραγματικές συνθήκες όπου η επανεκπαίδευση θα επιβάλλεται από την ίδια την φύση του προβλήματος που καλούμαστε να αντιμετωπίσουμε. Συνεπώς, με τον συνδυασμό autoencoder και CNN έχουμε πετύχει μια πολύ καλή ισορροπία μεταξύ επιδόσεων και χρόνου εκπαίδευσης. Έτσι η λύση που δημιουργήσαμε είναι αρκετά ευέλικτη και κατάλληλη για εφαρμογές σε πραγματικές συνθήκες εκτός εργαστηρίου.

Όμως, ακόμα και στην περίπτωση που δεν μπορεί να υπάρξει απευθείας σύγκριση μπορούμε να μαζέψουμε και να παρουσιάσουμε σε έναν συγκεντρωτικό πίνακα τα βασικά αποτελέσματα των πιο κοντινών εργασιών, στην δική μας, από την διεθνή βιβλιογραφία. Προφανώς αναφερόμαστε σε εργασίες που χρησιμοποίησαν και αυτές ως χαρακτηριστικά εισόδου εικόνες που προήλθαν από εκτελέσιμα αρχεία. Έτσι, καταλήγουμε στον ακόλουθο πίνακα σύγκρισης:

Δημοσίευση	Χρόνος Εκπαίδευσης (λεπτά)	Ακρίβεια	Στόχος
Rezende et al. (2017)	>>	98%	Ταξινόμηση
Ni et al. (2018)	>	98.8%	Ταξινόμηση
Vinayakumar et al. (2019)	>>	93%	Ταξινόμηση
Huang et al. (2020)	≈	94.7%	Ανίχνευση
Εργασία - Πείραμα 2	15	94%	Ανίχνευση
Εργασία - Πείραμα 3	70	96%	Ανίχνευση

Πίνακας 7: Πίνακας σύγκρισης αποτελεσμάτων της παρούσας εργασίας με την διεθνή βιβλιογραφία

Κοιτώντας τον πίνακα σύγκρισης είναι πλέον ξεκάθαρο γιατί δεν μπορεί να γίνει μια καλή σύγκριση στο πεδίου του χρόνου. Συγκεκριμένα καμία από τις εργασίες που είναι κοντά στην προσέγγιση μας δεν αναφέρει ξεκάθαρα πόσο χρόνο χρειάστηκε η εκπαίδευση του μοντέλου που χρησιμοποίησαν. Για τον λόγο αυτό το μόνο που μπορούμε να κάνουμε είναι **προσεγγίσεις και υποθέσεις** για τον χρόνο που χρειάστηκαν οι λύσεις σε άλλες εργασίες σε σύγκριση με την δική μας, με τον μέσο όρο του πειράματος 2 και 3, περίπου στα 42 λεπτά. Οι υποθέσεις βασίστηκαν σε όσα στοιχεία αναφέρει η κάθε εργασία για τον

αριθμό των epochs, τον χρόνο που χρειάζονται επιμέρους τεχνικές κ.λπ. σε συνδυασμό με την αρχιτεκτονική του επιλεγμένου μοντέλου και τον αριθμό των δειγμάτων. Έτσι, είμαστε αρκετά σίγουροι ότι σε ό,τι αφορά τον χρόνο εκπαίδευσης η παρούσα εργασία είναι σε εξαιρετικό επίπεδο. Προφανώς, ο στόχος των εργασιών που χρειάστηκαν πολύ παραπάνω χρόνο είναι η ταξινόμηση και όχι η ανίχνευση, οπότε είναι αναμενόμενο να χρειάζονται περισσότερο χρόνο για εκπαίδευση. Όσο αφορά την ακρίβεια προφανώς ότι έχουμε "χάσει" από χρόνο στην εκπαίδευση έχει αντίκτυπο στο τελικό αποτέλεσμα της ακρίβειας. Όμως, ποτέ δεν ήταν ο σκοπός μας να φτάσουμε την τέλεια ακρίβεια, αλλά να βρούμε την καλύτερη ισορροπία μεταξύ χρόνου εκπαίδευσης και ακρίβειας. Έτσι, κάνοντας απευθείας σύγκριση με την εργασία των Huang et al. [71], καθώς είναι η πιο όμοια εργασία με την δικιά μας, μπορούμε με βεβαιότητα να αναφέρουμε ότι έχουμε πετύχει μια πολύ καλή ισορροπία μεταξύ χρόνου εκπαίδευσης και ακρίβειας, δεδομένου ότι κρίναμε ότι οι χρόνοι εκπαίδευσης στις δύο εργασίες είναι αρκετά κοντινοί αλλά η παρούσα προσέγγιση είχε μειωμένη πολυπλοκότητα χάρη στην χρήση του autoencoder για την εξαγωγή των χαρακτηριστικών αλλά και αυξημένη ακρίβεια.

7 Συμπεράσματα και μελλοντικές επεκτάσεις

Ολοκληρώνοντας την παρούσα εργασία είναι απαραίτητη μια συνολική ανασκόπηση των όσων μελετήθηκαν. Αρχικά, η εργασία προτάθηκε για να καλύψει, ή καλύτερα, να συμπληρώσει τις υπάρχουσες μεθόδους ανίχνευσης κακόβουλου λογισμικού. Μέχρι αρκετά πρόσφατα, όλες οι μέθοδοι βασίζονταν κατά κύριο λόγο σε τεχνικές αναζήτησης συγκεκριμένων "αποτυπωμάτων" στα διάφορα αρχεία έτσι ώστε σε περίπτωση ταυτοποίησης το εν λόγω αρχείο να χαρακτηριστεί ως κακόβουλο. Το μεγάλο πρόβλημα με αυτή την προσέγγιση είναι ότι η γνώση που αξιοποιείται για την κατηγοριοποίηση των εκτελέσιμων είναι πολύ "στεγανή" και πολύ συγκεκριμένη. Με άλλα λόγια αν οι δημιουργοί του κακόβουλου λογισμικού φροντίσουν να τροποποιήσουν κάποια σημεία για να κρύψουν τα συγκεκριμένα αποτυπώματα, το εκτελέσιμο θα περάσει απαρατήρητο μέχρι να ανανεωθεί η βάση με τα αποτυπώματα προς αναζήτηση. Ακριβώς πάνω σε αυτή την "έλλειψη" έρχεται η παρούσα εργασία να προτείνει μια πιο αυτόνομη λύση. Προφανέστατα η καλύτερη προσέγγιση είναι η συμβολή του ανιχνευτή μηχανικής μάθησης να είναι επικουρική καθώς η κλασική προσέγγιση δεν αφήνει περιθώρια λάθους σε περίπτωση εντοπισμού, σε αντίθεση με τα μοντέλα μηχανικής μάθησης που στηρίζονται σε πιθανότητες για ν' αποφανθούν.

Έχοντας διαβάσει κάποιος τις προηγούμενες ενότητες είναι σε θέση όχι μόνο να αναπτύξει τα μοντέλα και τις μεθόδους που μελετήσαμε στην ενότητα των πειραμάτων αλλά και να δοκιμάσει νέες προσεγγίσεις. Από το κεφάλαιο του "State of the Art" είδαμε ότι οι συνδυασμοί που μπορούν να υλοποιηθούν είναι πάρα πολλοί και ότι δεν υπάρχει το τέλειο, πολλές εφαρμογές απαιτούν διαφορετικές προσεγγίσεις. Εμείς, αποφασίσαμε να εργαστούμε με στατικά χαρακτηριστικά, καθώς έτσι μειώνεται δραματικά η έκθεση του συστήματος στα κακόβουλα λογισμικά. Επιπλέον, ακολουθήσαμε πλήρως το μονοπάτι της βαθιάς μάθησης με την έννοια ότι, την εξαγωγή των χρήσιμων χαρακτηριστικών, αλλά και την μείωση διαστάσεων, την ανάλαβε ένα δίκτυο autoencoder. Με αυτόν τον τρόπο κάνουμε τον ανιχνευτή να βασίζεται σε ακόμα πιο γενικευμένα και αφηρημένα χαρακτηριστικά αποσκοπώντας στην μακροβιότερη λειτουργία του ανιχνευτή χωρίς την ανάγκη επανεκπαίδευσης.

Τα πειράματα που εκτελέσαμε μας έδειξαν ότι η πρόταση για έναν ανιχνευτή κακόβουλου λογισμικού βασισμένο στην μηχανική μάθηση είναι πλήρως δικαιολογημένη καθώς τα αποτελέσματα ήταν εξαιρετικά. Επίσης, φάνηκε ότι στην συγκεκριμένη εφαρμογή έχει τεράστια σημασία η σωστή συλλογή, επεξεργασία και τροφοδότηση των δεδομένων. Γενικά, η μορφή των δεδομένων είναι ίσως πιο σημαντική και από το μοντέλο που θα χρησιμοποιηθεί. Βέβαια, με την χρήση του autoencoder έχουμε "χάσει" σχεδόν ολοκληρωτικά την ερμηνευσιμότητα των δεδομένων. Προτού τα δεδομένα εικόνων περάσουν από τον autoencoder, υπάρχει πλήρης 1-1 αντιστοιχία με τα εκτελέσιμα, δηλαδή μπορούμε από την εικόνα να ξαναγυρίσουμε στα εκτελέσιμα αρχεία, μετά το πέρασμα από τον autoencoder κάτι τέτοιο είναι αδύνατο.

Ολοκληρώνοντας, ήρθε η στιγμή να προτείνουμε κάποιες βελτιώσεις, οι οποίες σε συνδυασμό με τα ευρήματα της εργασίας, θα ισχυροποιήσουν περαιτέρω την παρουσία της μηχανικής μάθησης στο πεδίο της ασφάλειας από κακόβουλα λογισμικά.

- Μελέτη περισσότερων λειτουργικών συστημάτων. Εμείς, ασχοληθήκαμε μόνο με το λειτουργικό σύστημα Windows καθώς όλες οι έρευνες υποδεικνύουν ότι τα μεγαλύτερα ποσοστά επιθέσεων και κακόβουλου λογισμικού υπάρχουν στο συγκεκριμένο λειτουργικό σύστημα. Όμως, πιο πρόσφατες έρευνες δείχνουν ότι το λογισμικό Android ακολουθεί πολύ στενά τα Windows, άρα είναι πλήρως δικαιολογημένη μια επέκταση και σε άλλα λειτουργικά, ιδιαίτερα στο Android.

- Περαιτέρω μελέτη αρχιτεκτονικών δικτύου. Στην εργασία παρουσιάσαμε τα δύο άκρα ένα σχετικά απλό CNN δίκτυο και το ResNet18. Προφανώς, το ResNet πέτυχε καλύτερα αποτελέσματα, αλλά όχι τρομερά καλύτερα. Το γεγονός αυτό είναι ίσως μια ισχυρή ένδειξη ότι υπάρχει και μια ενδιάμεση λύση. Δηλαδή, ένα δίκτυο με μεγαλύτερη πολυπλοκότητα από αυτό που προτείναμε, χαμηλότερη από αυτή του ResNet, ικανό να πλησιάσει περισσότερο την απόδοση του ResNet με λιγότερο χρόνο εκπαίδευσης λόγω της μειωμένης πολυπλοκότητας.

- Το θέμα των δεδομένων. Χρησιμοποιήσαμε μια πολύ ενδιαφέρουσα και αναπτυσσόμενη τεχνική, τον autoencoder, για την επεξεργασία των δεδομένων μας και τα αποτελέσματα ξεπέρασαν οποιαδήποτε προσδοκία. Όμως, στην πορεία χάσαμε την ερμηνευσιμότητα των ενδιάμεσων αποτελεσμάτων. Στον χώρο της κυβερνοασφάλειας κάτι τέτοιο δεν αποτελεί ζωτικής σημασίας εμπόδιο, όπως στις επιστήμες υγείας, αλλά δεν αποκλείουμε να υπάρξουν περιπτώσεις όπου η ερμηνευσιμότητα να χρειάζεται για να βοηθήσει περαιτέρω την έρευνα εναντίων του κακόβουλου λογισμικού.

- Ίσως η δυσκολότερη, από θέμα χρόνου, βελτίωση που προτείνουμε είναι η μελέτη της συμπεριφοράς του ανιχνευτή σε βάθος χρόνου. Αρκετές φορές, δοκιμάζαμε τεχνικές με γνώμονα την θωράκιση του ανιχνευτή σε μελλοντικές εκδόσεις κακόβουλου λογισμικού για όσο το δυνατόν περισσότερο προτού χρειαστεί επανεκπαίδευση με νέα δείγματα. Δεδομένου ότι έχουμε καταφύγει σε χρήση πολύ αφηρημένων δεδομένων εισόδου θεωρούμε ότι τον στόχο τον έχουμε πετύχει αλλά δεν μπορούμε να ποσοτικοποιήσουμε επακριβώς το χρονικό διάστημα βέλτιστης λειτουργίας του ανιχνευτή χωρίς επανεκπαίδευση. Ο λόγος είναι ότι μια τέτοια μελέτη είναι αρκετά χρονοβόρα και πρέπει να παρακολουθεί πολύ στενά όλες τις εξελίξεις στο πεδίο ανάπτυξης του κακόβουλου λογισμικού για να βρεθούν τα κρίσιμα σημεία στα οποία η επανεκπαίδευση κρίνεται απαραίτητη.

Προφανώς, ο πόλεμος μεταξύ ερευνητών ασφαλείας και προγραμματιστών κακόβουλου λογισμικού είναι ατέρμονος και οι βελτιώσεις που προτείναμε δεν σκοπεύουν στην υλοποίηση του τέλειου ανιχνευτή, αλλά στην συντήρηση ενός ανιχνευτή μηχανικής μάθησης σε επίπεδο τέτοιο που να αποτελεί σημαντικό "αντίπαλο" στην μάχη εναντίων των κακόβουλων λογισμικών.

8 Επίλογος

Ολοκληρώνοντας την εργασία πάνω στο θέμα της κατασκευής ενός ανιχνευτή κακόβουλου λογισμικού με μεθόδους βασισμένες στην μηχανική μάθηση είναι το κατάλληλο σημείο να παραθέσουμε μια συνολική ανασκόπηση της εργασίας και της πορείας που ακολουθήθηκε.

Πρωταρχικό μέλημα είναι να κατανοήσει κανείς την απειλή που καλείται ν' αντιμετωπίσει. Το πρώτο κεφάλαιο εμπεριέχει το βασικό θεωρητικό υπόβαθρο πάνω στο οποίο θ' αναπτυχθεί η εργασία στο σύνολό της. Το πρώτο μέρος του κεφαλαίου είναι αφιερωμένο στην περιγραφή διαφόρων τύπων κακόβουλου λογισμικού. Είναι αξιοσημείωτο ότι με την πάροδο του χρόνου τα κακόβουλα λογισμικά, ανάλογα με την ζημιά που θέλουν να προξενήσουν, θολώνουν τα όρια μεταξύ των κατηγοριών, αποσκοπώντας στην καλύτερη απόκρυψη τους και στην μεγιστοποίηση της ζημιάς. Φυσικά, για να λειτουργήσει ένα κακόβουλο λογισμικό και να ενεργοποιήσει το φορτίο του είναι απαραίτητη η παρουσία ενός λειτουργικού συστήματος. Έτσι, το πρώτο κεφάλαιο συνεχίζει με την ανάλυση της δομής ενός αρχείου PE, την πλέον συνηθισμένη μορφή κακόβουλων λογισμικών για το πιο δημοφιλές λειτουργικό σύστημα, τα Windows. Στην συνέχεια, έχοντας αποκτήσει μια βασική γνώση για την γενική μορφολογία της απειλής, είναι η κατάλληλη στιγμή ν' αναφερθούν οι σύγχρονοι τρόποι πρόληψης και αντιμετώπισης. Οι τεχνικές εντοπισμού μπορούν αν χωριστούν χρονικά σε δύο ομάδες, την πρώτη και την δεύτερη γενιά. Η βασική διαφορά μεταξύ των δύο είναι ότι η δεύτερη γενιά χτίζει πάνω στην πρώτη με γνώμονα την καλύτερη και ταχύτερη αναγνώριση κακόβουλου λογισμικού. Επιπλέον, η δεύτερη γενιά εισάγει και την έννοια της δυναμικής ανάλυσης, δηλαδή την ταυτοποίηση λογισμικού την ώρα που εκτελείται, μια ριψοκίνδυνη, αν δεν εκτελεστεί σωστά, τεχνική που επιστρέφει εκπληκτικά αποτελέσματα. Κλείνοντας το πρώτο κεφάλαιο δεν θα μπορούσε να λείπει το στοιχειώδες θεωρητικό υπόβαθρο για τις βασικές τεχνικές μηχανικής μάθησης που θ' αξιοποιηθούν στην συνέχεια, όπως τα συνελικτικά τεχνητά νευρωνικά δίκτυα, για την κατασκευή του ανιχνευτή.

Προφανώς, η παγκόσμια ερευνητική κοινότητα έχει αναγνωρίσει την υπεροχή της μηχανικής σε πολλές εφαρμογές και αναζητά συνεχώς νέα πεδία για την δοκιμή και αξιολόγησή της, με το πεδίο της κυβερνοασφάλειας να είναι ένα από τα σημαντικότερα. Οι προσεγγίσεις στην διεθνή βιβλιογραφία διαφέρουν σε δύο βασικά σημεία, την μέθοδο, ή τις μεθόδους, μηχανικής μάθησης που ακολουθούν και την μορφή των δεδομένων που χρησιμοποιούν. Όσον αναφορά τις μεθόδους δεν υπάρχει κάποια έκπληξη στο γεγονός ότι έχουν δοκιμαστεί σχεδόν όλες με εξαιρετικά αποτελέσματα κατά περιπτώσεις. Το δεδομένα ωστόσο είναι το πραγματικά ενδιαφέρον κομμάτι, εκεί τα πράγματα είναι στην ευχέρεια του κάθε ερευνητή για το πως θα ταιριάξει, όσο το δυνατόν καλύτερα, τα δεδομένα με την μέθοδο μηχανικής μάθησης που επέλεξε. Ο πρώτος διαχωρισμός αφορά το πως θα εξαχθούν τα δεδομένα από το εκάστοτε εκτελέσιμο, αν θα εξαχθούν στατικά ή δυναμικά. Τα στατικά χαρακτηριστικά είναι τα πιο εύκολα και τα πιο ασφαλή να εξαχθούν αλλά είναι και αρκετά εύκολο να κρύβεται η πραγματική φύση κάποιο εκτελέσιμου. Αντιθέτως, τα δυναμικά χαρακτηριστικά, είναι σχεδόν σίγουρο ότι θα αποκαλύψουν τις πραγματικές προθέσεις κάθε εκτελέσιμου αλλά είναι αρκετά επικίνδυνα στην εξαγωγή τους και χρονοβόρα. Προφανώς, έχουν υπάρξει και μελέτες που χρησιμοποιούν έναν συνδυασμό μεθόδων αλλά και, στατικών και δυναμικών χαρακτηριστικών. Το τελευταίο μέρος του δεύτερου κεφαλαίου αφιερώνεται εξ' ολοκλήρου σε μεθόδους βαθιάς μάθησης, ανεξαρτήτως του τύπου χαρακτηριστικών, καθώς η κοινότητα τον τελευταίο καιρό, έχει στρέψει την προσοχή της πλήρως σε αυτή την κατεύθυνση κυρίως για τον λόγο ότι η βαθιά μάθηση τείνει να ξεπερνάει σε επιδόσεις κάθε άλλη προσέγγιση.

Έχοντας μελετήσει την διεθνή βιβλιογραφία μπορούν να οριστούν οι βασικές απαιτήσεις που θα πρέπει να καλύπτει ο ανιχνευτής για να μπορεί να σταθεί επάξια σε εφαρμογές έξω

από συνθήκες εργαστηρίου. Οι απαιτήσεις μπορούν να χωριστούν γενικά σε δύο κατηγορίες: απαιτήσεις απόδοσης και απαιτήσεις χρόνου, χωρίς αυτό να σημαίνει ότι η μια κατηγορία είναι ανεξάρτητη από την άλλη. Οι απαιτήσεις που αφορούν το κομμάτι της επίδοσης προφανώς τέθηκαν με βάση, έναν άτυπο μέσο όρο, επιδόσεων από αρχιτεκτονικές που παρουσιάστηκαν στην διεθνή βιβλιογραφία. Όμως, οι απαιτήσεις που τέθηκαν για τον χρόνο που θα ξοδεύει ο ανιχνευτής χωρίς να χρησιμοποιείται, είτε αυτός ο χρόνος αναφέρεται σε εκπαίδευση ή επανεκπαίδευση, είναι το κομμάτι που η παρούσα εργασία προσπαθεί να διαφοροποιηθεί. Συγκεκριμένα, παρατηρήθηκε ότι στην διεθνή βιβλιογραφία πολύ συχνά η αυξημένη πολυπλοκότητα σε προτεινόμενες λύσεις ναι μεν αύξανε ελαφρώς τις επιδόσεις αλλά η αύξηση στον απαιτούμενο χρόνο εκπαίδευσης ήταν τρομερά δυσανάλογη. Έτσι η εργασία επικεντρώνεται στην εύρεση του καλύτερου συνδυασμού για την μορφή των δεδομένων και το μοντέλο μηχανικής μάθησης που θα πετύχει την ιδανική ισορροπία μεταξύ επιδόσεων και χρόνου εκτός λειτουργίας. Τα δεδομένα που χρησιμοποιήθηκαν είναι στατικές εικόνες γκρι κλίμακας που αναπαριστούν τα bytes του εκάστοτε εκτελέσιμου, καθώς επιτυγχάνεται καλή ταχύτητα εξαγωγής και μέγιστη ασφάλεια, ενώ τα μοντέλα είναι παραλλαγές συνελικτικών τεχνητών νευρωνικών δικτύων, CNN. Για την εξαγωγή των εικόνων είναι απαραίτητη και η ύπαρξη των εκτελέσιμων, των πραγματικών δεδομένων για την συγκεκριμένη υλοποίηση. Ευτυχώς, για τις ανάγκες τις εργασίας, στο διαδίκτυο υπάρχουν πολλά αποθετήρια που δια μοιράζουν για εκπαιδευτικού σκοπούς κακόβουλο λογισμικό. Έτσι δημιουργώντας ένα ισορροπημένο σύνολο δεδομένων, με σχεδόν ίσα δείγματα κακόβουλου και ακίνδυνου λογισμικού, δεν χρειάζεται να καταφύγουμε σε ιδιαίτερες τεχνικές για περιπτώσεις όπου η μια κλάση δειγμάτων υπερτερεί κατά πολύ της άλλης. Έχοντας όλα τα δείγματα στην σωστή μορφή δεν μένει παρά να ξεκινήσουν τα πειράματα.

Το τελευταίο κεφάλαιο της εργασίας είναι και το πιο σημαντικό. Σε αυτό το κεφάλαιο υλοποιούνται οι ανιχνευτές κακόβουλου λογισμικού. Αρχικά, ακολουθείται μια εντελώς αφελής προσέγγιση με ένα απλό δίκτυο CNN για την οριοθέτηση ενός κατώτατου ορίου. Παραδόξως, ακόμα και αυτή η αφελέστατη προσέγγιση έδωσε καλά αποτελέσματα, σε καμία περίπτωση δεν είναι κοντά στις απαιτήσεις που τέθηκαν αλλά είναι ενθαρρυντικά για την συνέχεια. Έπειτα, αντί να γίνει κάποια τροποποίηση στο απλό δίκτυο έγινε μια τροποποίηση στα δεδομένα. Συγκεκριμένα, στόχος ήταν να πιεστεί το απλό δίκτυο στα όριά του προτού αντικατασταθεί από μια πιο σύνθετη αρχιτεκτονική. Για να επιτευχθεί αυτός ο στόχος ο μοναδικός τρόπος είναι μέσω των δεδομένων. Πλέον, τα δεδομένα περνάνε μέσα από ένα δίκτυο autoencoder με στόχο την μείωση των διαστάσεων των εικόνων αλλά και την εξαγωγή των πιο σημαντικών χαρακτηριστικών από τις εικόνες. Αυτός ο καινούριος τρόπος προσέγγισης είναι που επέτρεψε στο απλό δίκτυο να εκπαιδευτεί σε μόλις δεκαπέντε λεπτά και να πετύχει μια ακρίβεια της τάξεως του 94%, δύο αποτελέσματα που κατατάσσουν την μέθοδο αρκετά ψηλά στην διεθνή βιβλιογραφία. Όμως, υπάρχουν ακόμα περιθώρια βελτίωσης. Σε αυτό το σημείο είναι προφανές ότι το απλό δίκτυο έχει φτάσει, αν όχι τις μέγιστες δυνατότητες του, υπερβολικά κοντά στο μέγιστο. Ο μόνος τρόπος για να βελτιωθούν τα αποτελέσματα είναι η προσφυγή σε βαθύτερες αρχιτεκτονικές. Επιλέχθηκε, το προ εκπαιδευμένο δίκτυο ResNet18 για τον λόγο ότι παρουσιάζει εξαιρετικές επιδόσεις χωρίς να είναι υπερβολικά πολύπλοκο στην δομή του. Πράγματι με χρήση της τεχνικής transfer learning, το ResNet18 με τα δεδομένα από τον autoencoder ανέβασε την ακρίβεια στο 96%, όμως χρειάστηκε τον πενταπλάσιο χρόνο εκπαίδευσης, αναμενόμενο αν υπολογιστεί η τεράστια διαφορά στις πολυπλοκότητες των δικτύων. Το πείραμα με το ResNEt18 είναι το μόνο που καλύπτει απολύτως όλες τις απαιτήσεις που τέθηκαν. Το πραγματικό ερώτημα είναι αν αξίζει τον τόσο χρόνο μια μικρή, αναλογικά, βελτίωση της ακρίβειας. Η απάντηση είναι ότι σε ορισμένες περιπτώσεις η ακρίβεια είναι πάνω απ' όλα ενώ σε άλλες είναι προτιμότερος

ένας μικρός συμβιβασμός. Εν κατακλείδι, φαίνεται ότι το δίλημμα απόδοση ή χρόνος θα υπάρχει πάντα, αλλά με την παρούσα εργασία φάνηκε ότι η προσθήκη του ενδιάμεσου δικτύου autoencoder το χάσμα μεταξύ των δύο περιορίστηκε σε μεγέθη που πλέον δεν είναι απαγορευτική μια προσπάθεια για την επίτευξη του καλύτερου, είτε από θέμα χρόνου είτε από θέμα απόδοσης.

Συνοψίζοντας, η παρούσα εργασία χτίζει πάνω σε εξαιρετικές προσεγγίσεις που έχουν αναφερθεί στην διεθνή βιβλιογραφία με στόχο την επίτευξη μιας ισορροπίας μεταξύ του χρόνου εκπαίδευσης αλγορίθμων μηχανικής μάθησης, δηλαδή του μη ωφέλιμου χρόνου, και της ακρίβειας στην ανίχνευση κακόβουλου λογισμικού. Με την χρήση ενός δικτύου autonecoder μπορέσαμε να ξεφύγουμε από την ανάγκη για την χειροκίνητη εξαγωγή και διαλογή χαρακτηριστικών από τα εκτελέσιμα. Δηλαδή, προσθέσαμε ακόμα ένα επιπλέον επίπεδο αφαίρεσης το οποίο εκπαιδεύεται, συνεπώς αν κάποιο κακόβουλο λογισμικό προγραμματιστεί με τρόπο ώστε να διαφεύγει της χειροκίνητης διαλογής θα έπρεπε ν' αλλάξει όλη η προσέγγιση, ενώ με το δίκτυο autoencoder αρκεί μια απλή επανεκπαίδευση. Επιπλέον, η χρήση του autoencoder αποδείχτηκε αναντικατάστατη στην μείωση του χρόνου εκπαίδευσης των δικτύων. Με αυτή την προσέγγιση θα μπορούσαν και άλλες εργασίες που έχουν τρέξει τα πειράματά τους σε πολύ πιο σύνθετο και καλύτερο hardware να επωφεληθούν από την δραματική μείωση χρόνου, με το μειονέκτημα ότι θα αλλάξει αρκετά η αρχική τους προσέγγιση καθώς η εξαγωγή των χαρακτηριστικών θα βασίζεται εξ' ολοκλήρου στον autoencoder. Βέβαια, κάτι τέτοιο δεν αποκλείει την ύπαρξη υβριδικών λύσεων όπου θα αξιοποιούνται και τα χαρακτηριστικά του autoencoder αλλά και άλλα που έχουν εξαχθεί με το χέρι. Ολοκληρώνοντας, αξίζει να αναφέρουμε ότι τα αποτελέσματα της εργασίας έδειξαν ότι η μηχανική μάθηση, και πιο συγκεκριμένα ο τομέας της βαθιάς μηχανικής μάθησης, βρίσκεται σε τέτοιο επίπεδο όπου πλέον μπορούμε να τον εμπιστευτούμε για την προστασία των προσωπικών μας δεδομένων και μηχανημάτων, χωρίς να καταφεύγουμε σε χειροκίνητες λύσεις για την εξαγωγή χαρακτηριστικών. Βέβαια, η εμπιστοσύνη ποτέ δεν πρέπει να είναι τυφλή αλλά θα πρέπει να υπάρχει η σχετική συντήρηση και αξιολόγηση των αποτελεσμάτων σε τακτά χρονικά διαστήματα για την αποφυγή δυσάρεστων γεγονότων.

Βιβλιογραφία

[1] Rob Sobers Updated: 3/16/2021. *134 Cybersecurity Statistics and Trends for 2021: Varonis*. Mar. 2021. URL: https://www.varonis.com/blog/cybersecurity-statistics/.

[2] Posted by John Love and John Love. *A Brief History of Malware-Its Evolution and Impact*. Sept. 2019. URL: https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/.

[3] SUN TZU. *ART OF WAR*. WELLFLEET, 2021.

[4] André Ricardo Abed Grégio et al. The Computer Journal 58.10, 2015.

[5] Eric Filiol. *Computer viruses: from theory to applications*. Springer-Verlag France, 2005.

[6] Jeremy Z. Kolter and Marcus A. Maloof. "Learning to detect malicious executables in the wild". In: *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 04* (2004). DOI: 10.1145/1014052.1014105.

[7] *Business Home*. URL: https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/what-is-malware.html.

[8] Karl-Bridge-Microsoft. *PE Format - Win32 apps*. URL: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format.

[9] September 7 et al. *Malware researchers handbook (demystifying PE file)*. Aug. 2021. URL: https://resources.infosecinstitute.com/topic/2-malware-researchers-handbook-demystifying-pe-file/.

[10] Peter Szor. *The art of computer virus research and defense*. Addison-Wesley, 2005.

[11] *Eugene Kaspersky, CEO of Kaspersky*. URL: https://www.kaspersky.com/about/team/eugene-kaspersky.

[12] *Macro Viruses*. URL: https://www.kaspersky.com/resource-center/definitions/macro-virus.

[13] *What is a virtual machine?* URL: https://www.vmware.com/topics/glossary/content/virtual-machine.

[14] Tom M. Mitchell. *Machine learning*. McGraw Hill, 2017.

[15] Ethem Alpaydin. *Introduction to machine learning*. The MIT Press, 2020.

[16] Michael A. Nielsen. *Neural Networks and Deep Learning*. Jan. 1970. URL: http://neuralnetworksanddeeplearning.com/chap6.html.

[17] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[18] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[19] *ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks*. Jan. 2019. URL: https://neurohive.io/en/popular-networks/resnet/.

[20] Hyungeun Choi, Seunghyoung Ryu, and Hongseok Kim. "Short-Term Load Forecasting based on ResNet and LSTM". In: Oct. 2018, pp. 1–6. DOI: 10.1109/SmartGridComm.2018.8587554.

[21] Daniel Gibert, Carles Mateu, and Jordi Planes. "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges". In: *Journal of Network and Computer Applications* 153 (2020), p. 102526.

[22] Yanfang Ye et al. "SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging". In: *Journal in Computer Virology* 5.4 (2008), pp. 283–293. DOI: 10.1007/s11416-008-0108-y.

[23] Zane Markel and Michael Bilzor. "Building a machine learning classifier for malware detection". In: *2014 second workshop on anti-malware testing research (WATeR)*. IEEE. 2014, pp. 1–4.

[24] Neil Balram, George Hsieh, and Christian McFall. "Static Malware Analysis Using Machine Learning Algorithms on APT1 Dataset with String and PE Header Features". In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE. 2019, pp. 90–95.

[25] Igor Santos et al. "Opcode sequences as representation of executables for data-mining-based unknown malware detection". In: *Information Sciences* 231 (2013), pp. 64–82. DOI: 10.1016/j.ins.2011.08.020.

[26] Hanchuan Peng, Fuhui Long, and C. Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (2005), pp. 1226–1238. DOI: 10.1109/tpami.2005.159.

[27] Mohamed El Boujnouni, Mohamed Jedra, and Noureddine Zahid. "New malware detection framework based on N-grams and support vector domain description". In: *2015 11th international conference on information assurance and security (IAS)*. IEEE. 2015, pp. 123–128.

[28] David MJ Tax and Robert PW Duin. "Support vector data description". In: *Machine learning* 54.1 (2004), pp. 45–66.

[29] Zhang Fuyong and Zhao Tiezhu. "Malware detection and classification based on n-grams attribute similarity". In: *2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC)*. Vol. 1. IEEE. 2017, pp. 793–796.

[30] Edward Raff et al. "Kilograms: Very large n-grams for malware classification". In: *arXiv preprint arXiv:1908.00200* (2019).

[31] *Zipf Distribution*. URL: https://mathworld.wolfram.com/ZipfDistribution.html.

[32] Ding Yuxin and Zhu Siyi. "Malware detection based on deep learning algorithm". In: *Neural Computing and Applications* 31.2 (2019), pp. 461–472.

[33] Ashkan Sami et al. "Malware detection based on mining API calls". In: *Proceedings of the 2010 ACM symposium on applied computing*. 2010, pp. 1020–1025.

[34] Yanfang Ye et al. "An intelligent PE-malware detection system based on association mining". In: *Journal in computer virology* 4.4 (2008), pp. 323–334.

[35] Xifeng Yan, Jiawei Han, and Ramin Afshar. "Clospan: Mining: Closed sequential patterns in large datasets". In: *Proceedings of the 2003 SIAM international conference on data mining*. SIAM. 2003, pp. 166–177.

[36] Usukhbayar Baldangombo, Nyamjav Jambaljav, and Shi-Jinn Horng. "A static malware detection system using data mining methods". In: *arXiv preprint arXiv:1308.2831* (2013).

[37] Sanjam Singla et al. "A novel approach to malware detection using static classification". In: *International Journal of Computer Science and Information Security* 13.3 (2015), p. 1.

[38] Robert Lyda and James Hamrock. "Using entropy analysis to find encrypted and packed malware". In: *IEEE Security & Privacy* 5.2 (2007), pp. 40–45.

[39] Ivan Sorokin. "Comparing files using structural entropy". In: *Journal in computer virology* 7.4 (2011), pp. 259–265.

[40] Donabelle Baysa, Richard M Low, and Mark Stamp. "Structural entropy and metamorphic malware". In: *Journal of computer virology and hacking techniques* 9.4 (2013), pp. 179–192.

[41] Huu-Danh Pham, Tuan Dinh Le, and Thanh Nguyen Vu. "Static PE malware detection using gradient boosting decision trees algorithm". In: *International Conference on Future Data and Security Engineering*. Springer. 2018, pp. 228–236.

[42] Hyrum S Anderson and Phil Roth. "Ember: an open dataset for training static pe malware machine learning models". In: *arXiv preprint arXiv:1804.04637* (2018).

[43] Lakshmanan Nataraj et al. "Malware images: visualization and automatic classification". In: *Proceedings of the 8th international symposium on visualization for cyber security*. 2011, pp. 1–7.

[44] Aude Oliva and Antonio Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope". In: *International journal of computer vision* 42.3 (2001), pp. 145–175.

[45] Mehadi Hassen and Philip K Chan. "Scalable function call graph-based malware classification". In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 2017, pp. 239–248.

[46] Deebiga Rajeswaran et al. "Function call graphs versus machine learning for malware detection". In: *Guide to Vulnerability Analysis for Computer Networks and Systems*. Springer, 2018, pp. 259–279.

[47] Mahboobe Ghiasi, Ashkan Sami, and Zahra Salehi. "Dynamic VSA: a framework for malware detection based on register contents". In: *Engineering Applications of Artificial Intelligence* 44 (2015), pp. 111–122.

[48] Mahboobe Ghiasi, Ashkan Sami, and Zahra Salehi. "Dynamic malware detection using registers values set analysis". In: *2012 9th International ISC Conference on Information Security and Cryptology*. IEEE. 2012, pp. 54–59.

[49] James B. Fraley and Marco Figueroa. "Polymorphic malware detection using topological feature extraction with data mining". In: *SoutheastCon 2016*. 2016, pp. 1–7. DOI: 10.1109/SECON.2016.7506685.

[50] Curtis Storlie et al. "Stochastic identification of malware with dynamic traces". In: *The Annals of Applied Statistics* (2014), pp. 1–18.

[51] Philip O'kane, Sakir Sezer, and Kieran McLaughlin. "Detecting obfuscated malware using reduced opcode set and optimised runtime trace". In: *Security Informatics* 5.1 (2016), pp. 1–12.

[52] Domhnall Carlin, Philip O'Kane, and Sakir Sezer. "Dynamic analysis of malware using run-time opcodes". In: *Data analytics and decision support for cybersecurity*. Springer, 2017, pp. 99–125.

[53] Dmitri Bekerman et al. "Unknown malware detection using network traffic classification". In: *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2015, pp. 134–142.

[54] Mark Andrew Hall. "Correlation-based feature selection for machine learning". In: (1999).

[55] Guodong Zhao et al. "Detecting APT malware infections based on malicious DNS and traffic analysis". In: *IEEE access* 3 (2015), pp. 1132–1142.

[56] Sergii Banin, Andrii Shalaginov, and Katrin Franke. "Memory access patterns for malware detection". In: (2016).

[57] Amine Boukhtouta et al. "Network malware classification comparison using DPI and flow packet headers". In: *Journal of Computer Virology and Hacking Techniques* 12.2 (2016), pp. 69–100.

[58] Xuan Dau Hoang and Quynh Chi Nguyen. "Botnet detection based on machine learning techniques using DNS query data". In: *Future Internet* 10.5 (2018), p. 43.

[59] Chandrasekar Ravi and R Manoharan. "Malware detection using windows api sequence and machine learning". In: *International Journal of Computer Applications* 43.17 (2012), pp. 12–16.

[60] Dolly Uppal et al. "Malware detection and classification based on extraction of API sequences". In: *2014 International conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2014, pp. 2337–2342.

[61] Chun-I Fan et al. "Malware Detection Systems Based on API Log Data Mining". In: *2015 IEEE 39th Annual Computer Software and Applications Conference*. Vol. 3. 2015, pp. 255–260. DOI: 10.1109/COMPSAC.2015.241.

[62] Hisham Shehata Galal, Yousef Bassyouni Mahdy, and Mohammed Ali Atiea. "Behavior-based features model for malware detection". In: *Journal of Computer Virology and Hacking Techniques* 12.2 (2016), pp. 59–67.

[63] Anusha Damodaran et al. "A comparison of static, dynamic, and hybrid analysis for malware detection". In: *Journal of Computer Virology and Hacking Techniques* 13.1 (2017), pp. 1–12.

[64] Zahra Salehi, Ashkan Sami, and Mahboobe Ghiasi. "MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values". In: *Engineering Applications of Artificial Intelligence* 59 (2017), pp. 93–102.

[65] Nureni Ayofe Azeez et al. "Windows PE Malware Detection Using Ensemble Learning". In: *Informatics*. Vol. 8. 1. Multidisciplinary Digital Publishing Institute. 2021, p. 10.

[66] Robertas Damaševičius et al. "Ensemble-Based Classification Using Neural Networks and Machine Learning Models for Windows PE Malware Detection". In: *Electronics* 10.4 (2021), p. 485.

[67] Syed Khurram Jah Rizvi et al. “PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable”. In: *Complex & Intelligent Systems* (2021), pp. 1–13.

[68] Edmar Rezende et al. “Malicious software classification using transfer learning of resnet-50 deep neural network”. In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2017, pp. 1011–1014.

[69] Sang Ni, Quan Qian, and Rui Zhang. “Malware identification using visualization images and deep learning”. In: *Computers & Security* 77 (2018), pp. 871–885.

[70] R Vinayakumar et al. “Robust intelligent malware detection using deep learning”. In: *IEEE Access* 7 (2019), pp. 46717–46738.

[71] Xiang Huang et al. “A method for windows malware detection based on deep learning”. In: *Journal of Signal Processing Systems* 93.2 (2021), pp. 265–273.

[72] Bojan Kolosnjaji et al. “Deep learning for classification of malware system call sequences”. In: *Australasian joint conference on artificial intelligence*. Springer. 2016, pp. 137–149.

[73] Ben Athiwaratkun and Jack W Stokes. “Malware classification with LSTM and GRU language models and a character-level CNN”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 2482–2486.

[74] Zhifeng Xu, Xianjin Fang, and Gaoming Yang. “Malbert: A novel pre-training method for malware detection”. In: *Computers & Security* 111 (2021), p. 102458.

[75] Daniel Gibert et al. “Convolutional Neural Networks for Classification of Malware Assembly Code”. In: Oct. 2017. DOI: 10.3233/978-1-61499-806-8-221.

[76] Mansour Ahmadi et al. “Novel feature extraction, selection and fusion for effective malware family classification”. In: *Proceedings of the sixth ACM conference on data and application security and privacy*. 2016, pp. 183–194.

[77] Daniel Gibert, Carles Mateu, and Jordi Planes. “A hierarchical convolutional neural network for malware classification”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[78] Mozammel Chowdhury, Azizur Rahman, and Rafiqul Islam. “Protecting data from malware threats using machine learning technique”. In: *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE. 2017, pp. 1691–1694.

[79] Edward Raff et al. “Malware detection by eating a whole exe”. In: *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[80] Paul Prasse et al. “Malware detection by analysing encrypted network traffic with neural networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pp. 73–88.

[81] AL-Hawawreh Muna, Nour Moustafa, and Elena Sitnikova. “Identification of malicious activities in industrial internet of things based on deep learning models”. In: *Journal of Information security and applications* 41 (2018), pp. 1–11.

[82] Jonas Latz. “Analysis of stochastic gradient descent in continuous time”. In: *Statistics and Computing* 31.4 (2021), pp. 1–25.

[83] *Precision and recall*. Oct. 2021. URL: https://en.wikipedia.org/wiki/Precision_and_recall.

[84] *Sensitivity and specificity*. Oct. 2021. URL: https://en.wikipedia.org/wiki/Sensitivity_and_specificity.

[85] Agnes Lydia and Sagayaraj Francis. "Adagrad an optimizer for stochastic gradient descent". In: *Int. J. Inf. Comput. Sci* 6.5 (2019).

[86] *ConvTranspose2d*. URL: https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html.

Ακρωνύμια

ANN Artificial Neural Network.

API Application Programming Interface.

BDT Boosted Decision Trees.

BNB Bernoulli Naïve Bayes.

CFG Call Function Graphs.

CNN Convolutional Neural Network.

DBN Deep Belief Network.

DBSCN Density-Based Spatial Clustering of Applications with Noise.

DLL Dynamic-link Library.

DOS Disk Operating System.

DT Decision Tree.

FFN Feed Forward Network.

GBDT Gradient Boosting Decision Trees.

GRU Gated Recurrent Unit.

HMM Hidden Markov Model.

IL Instance-based Learner.

K-NN K-Nearest Neighbors.

LR Logistic Regression.

LSTM Long Short-Term Memory.

NB Naive Bayes.

Opcode Operation Code.

PCA Principal Component Analysis.

PE Portable Executable.

RBM Restricted Boltzmann Machine.

RF Random Forest.

SGD Stochastic Gradient Descent.

SMO Sequential Minimal Optimization.

SVDD Support Vector Domain Description.

SVM Support Vector Machine.

TF Term Frequency.

TF-IDF Term Frequency - Inverse Document Frequency.

TPR True Positive Rate.

VP Voted Perceptron.

XGB Extreme Gradient Boosting.

Σ.Η.Μ.Μ.Υ Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών.

κ.λπ. και λοιπά.

κ.ο.κ. και ούτω καθεξής.