Εθνικο Μετσοβιο Πολυτεχνειο

Διπλωματικη Εργασια

Εφαρμογές της Μηχανικής Μάθησης στην Κρυπτογραφία

Φοιτητής:
Δουμήτρου Δανιήλ Νιμάρα

Επιβλέπων:
Αριστείδης Παγουρτζής

Μέλος Επιτροπής:
Μιχαήλ Λουλάκης

Μέλος Επιτροπής:
Γιώργος Στάμου

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών

2 Ιουλίου 2019

Περίληψη

Η παρούσα εργασία διερευνά και παρουσιάζει βασικές εφαρμογές της Μηχανικής Μάθησης στην Κρυπτογραφία. Το πρώτο κεφάλαιο εφοδιάζει τον αναγνώστη με θεμελιώδεις εισαγωγικές γνώσεις που κρίθηκαν απαραίτητες για την αποτελεσματικότερη ανάλυση και παρακολούθηση των μετέπειτα περιεχομένων. Τα επόμενα τρία κεφάλαια παρουσιάζουν και αξιολογούν την χρήση και αποτελεσματικότητα μοντέλων Μηχανικής Μάθησης για την κρυπτανάλυση, την ενίσχυση κρυπτοσυστημάτων και την δυνατότητα υποκατάστασης τους από σύνθετα μοντέλα όπως αυτά των GAN. Στο τελευταίο κεφάλαιο προτείνουμε σύγχρονα μοντέλα τα οποία σκοπεύουμε να αξιοποιήσουμε σε καινοτόμες εφαρμογές, με σκοπό την περαιτέρω ανάπτυξη του κλάδου. Η συνεισφορά της εργασίας είναι διπλή. Πρώτον, αποτελεί μια συγκεντρωτική και κατάλληλα οργανωμένη αναφορά σε διάφορες σπουδαίες εφαρμογές αυτών των δυο κλάδων. Αξίζει να σημειωθεί ότι κατά την μελέτη αυτού του αντικειμένου δεν εντοπίσαμε ανάλογες εργασίες που να εστιάζουν στην παρουσίαση τέτοιων εφαρμογών. Επιχειρήσαμε να απομονώσουμε την ουσία από κάθε εφαρμογή, ενώ παράλληλα εμπλουτίσαμε την θεωρία στην οποία βασίζεται, εφοδιάζοντας τον αναγνώστη με γνώσεις που κρίνουμε ότι παραλείπονται από τις αρχικές εργασίες. Επιπλέον, μελλοντικοί ερευνητές δύναται να ανατρέξουν στην εργασία μας για την υιοθέτηση παρεμφερών μοντέλων σε ανάλογα προβλήματα. Τέλος, το code obfuscation και το crypto-mining που αναλύονται στο τελευταίο κεφάλαιο αποτελούν ακμαία και ραγδαία αναπτυσσόμενα πεδία στην Επιστήμη των Υπολογιστών.

Abstract

This paper examines and presents basic applications of Machine Learning in Cryptography. The First Chapter provides the reader with fundamental introductory knowledge which was deemed necessary for a more effective analysis and comprehension of this work's content. The following three chapters present and evaluate the usage and efficacy of Machine Learning Models for cryptanalysis, for the reinforcement of cryptosystems and their substitutability from complex models such as GANs. In the last chapter we propose modern models that we intend to use in innovative applications, with the aim of further developing the field. The contribution of our work is twofold. Firstly, it constitutes a centralized and appropriately organized reference to several important applications of these two fields. It is worth noting that during the study of this object we did not find similar work focusing on the presentation of such applications. We attempted to encapsulate the essence from every application, while enriching the theory on which it is based, providing the reader with knowledge that we consider to be omitted from the original work. In addition, prospective researchers may refer to our work on adopting the presented models to similar problems. Finally, code obfuscation and crypto-mining discussed in the last chapter are booming and rapidly growing fields in Computer Science.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πρωτίστως, τους γονείς και την οικογένεια μου, που στάθηκαν πλάι μου συμβουλεύοντας και βοηθώντας με να γίνω πιο ολοκληρωμένος άνθρωπος. Χρωστάω επίσης μεγάλη ευγνωμοσύνη σε όλους τους φίλους και συμφοιτητές μου, για τον χρόνο και τις συζητήσεις που μοιραστήκαμε. Να ευχαριστήσω επιπλέον, την Δώρα, τον Κώστα και την Αγγελική, που στάθηκαν δίπλα μου από το γυμνάσιο και συνεχίζουν να με υποστηρίζουν. Επίσης, τα ξαδέρφια μου Νικήτα και Αλέξη, με τα οποία συναναστρέφομαι συχνά, αναπτύσσοντας ιδέες στα κοινά μας ενδιαφέροντα.

Οφείλω να τονίσω την συνεισφορά των καθηγητών στην ακαδημαϊκή μου πορεία. Ιδιαιτέρως, να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Αριστείδη Παγουρτζή, που με εισήγαγε στην Κρυπτογραφία και με βοήθησε, παρέχοντάς μου υλικό και καθοδήγηση στην διπλωματική μου εργασία. Έπειτα, τον κύριο Ζάχο, που μου δίδαξε Θεωρητική πληροφορική και με ψυχαγώγησε με τις αμέτρητες ιστορίες του. Θέλω να ευχαριστήσω επίσης, τον κύριο Συμβώνη που με μύησε στον τομέα της πληροφορικής και στον προγραμματισμό και μου έμαθε να τα εκτιμάω. Τέλος, ευχαριστώ τα μέλη της επιτροπής, κύριο Μιχαήλ Λουλάκη και τον κύριο Γιώργο Στάμου που μελέτησαν και αξιολόγησαν την εργασία και την παρουσίαση μου.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η Κρυπτογραφία και η Μηχανική Μάθηση είναι δυο επιστημονικά πεδία που γνωρίζουν ολοένα και εντονότερη ανάπτυξη η οποία εντυπώνεται τόσο ακαδημαϊκά όσο και σε πρακτικές εφαρμογές. Αναφορές σε έννοιες όπως το Blockchain [2] και τα Νευρωνικά Δίκτυα (Neural Networks/Deep Learning) πληθαίνουν εκθετικά. Αυτό γίνεται αντιληπτό, τόσο στο ευρύ κοινό, μέσω εφαρμογών αυτών των εννοιών σε κρυπτονομίσματα όπως το Bitcoin ή σε παραγωγή εικόνων με χρήση ειδικών νευρωνικών δικτύων ονόματι GAN (Generative Adverserial Network) [1], όσο και στην ακαδημαϊκή κοινότητα, όπου δεκάδες συνέδρια διοργανώνονται κάθε χρόνο, όπως τα COLT (Conference on Learning Theory), NeurIPS, ICML, AtheCrypt, Crypto, Eurocrypt.

Η παρούσα εργασία έχει σκοπό την σύνθεση αυτών των δυο ενεργών πεδίων έρευνας και την παρουσίαση αξιοσημείωτων εφαρμογών της Μηχανικής Μάθησης στην Κρυπτογραφία. Παράλληλα, για την βαθύτερη κατανόηση των αποτελεσμάτων και της σημασίας τους, θα παρουσιαστούν εν συντομία βασικές έννοιες των παραπάνω κλάδων.

Βασικό κίνητρο της συγγραφής αυτής της εργασίας δεν αποτέλεσε μόνο το ενδιαφέρον που παρουσιάζουν τα επιμέρους επιστημονικά πεδία, άλλα και το γεγονός ότι συχνά δε δίνεται ιδιαίτερη έμφαση στη μεταξύ τους αλληλεπίδραση. Η επιφύλαξη αυτή δεν είναι αβάσιμη, καθώς η εξόρυξη οποιουδήποτε ουσιωδούς αποτελέσματος απαιτεί την προσεκτική εφαρμογή της Μηχανικής Μάθησης στην Κρυπτογραφία. Το βασικότερο εμπόδιο αυτής της εφαρμογής είναι δυο αντικρουόμενες θεμελιώδεις αρχές που διέπουν αυτού τους δύο κλάδους: η Μηχανική Μάθηση είναι πολύ αποτελεσματική στον εντοπισμό μοτίβων μεταξύ δεδομένων, ενώ η Κρυπτογραφία στην απόκρυψή τους. Για τον λόγο αυτό, όπως θα σας παρουσιάσουμε στην συνέχεια, τα περισσότερα αξιόλογα αποτελέσματα βασίζονται κυρίως σε τεχνικές υλοποιήσεις κρυπτοσυστημάτων (χρήση νευρωνικών για side channel attack). Παρόλα αυτά, υπάρχουν ενδιαφέρουσες απόπειρες απομίμησης κρυπτοσυστημάτων, όπως ενός απλού Simon cipher[3], ή τη δημιουργία ενός καινούργιου με χρήση adverserial training (GAN)[4].

Η εργασία δομείται με τον ακόλουθο τρόπο: Στο παρόν κεφάλαιο, θα ακολουθήσουν δυο υποενότητες που θα εξηγήσουν συνοπτικά, βασικά χαρακτηριστικά της Κρυπτογραφίας και της Μηχανικής Μάθησης τα οποία θα μας χρειαστούν στην συνέχεια. Στο δεύτερο κεφάλαιο, θα διερευνήσουμε βασικές εφαρμογές για την επίθεση κρυπτοσυστημάτων, στο τρίτο για την προστασία και ενίσχυση τους, ενώ στο τέταρτο θα αναρωτηθούμε και θα εξετάσουμε σε ποιο βαθμό μπορούμε πραγματικά να προσομοιάσουμε τη δομή και την λειτουργία ενός σύνθετου κρυπτοσυστήματος με ένα νευρωνικό δίκτυο. Στο τελευταίο κεφάλαιο παρουσιάζονται σύγχρονα μοντέλα τα οποία σκοπεύουμε να αξιοποιήσουμε σε καινοτόμες εφαρμογές, με σκοπό την περαιτέρω ανάπτυξη του κλάδου.

1.1 Κρυπτογραφία

Η κρυπτογραφία ασχολείται με την μεταφορά ενός μηνύματος m από έναν πομπό σε ένα δέκτη, κατά τέτοιο τρόπο, ώστε η μυστικότητα του περιεχομένου του να μένει αλώβητη σε οποιονδήποτε εξωτερικό παραλήπτη. Τα πρώτα δείγματά της μπορεί κανείς να τα διακρίνει 2500 χρόνια πριν, στην αρχαία Ελλάδα όπου οι Σπαρτιάτες είχαν την δυνατότητα να ανταλλάσσουν πληροφορία κρυφά μέσω μιας σκυτάλης (κύλινδρος). Σε αυτήν, τύλιγαν μια κορδέλα και έγραφαν πάνω της το μήνυμα που ήθελαν να στείλουν. Η κορδέλα αποτελούσε το κρυπτογραφημένο κείμενο, του οποίου η ορθή αποκρυπτογράφηση θα μπορούσε να πραγματοποιηθεί μόνο εφόσον, ο παραλήπτης το τύλιγε σε μια παραπλήσια σκυτάλη.

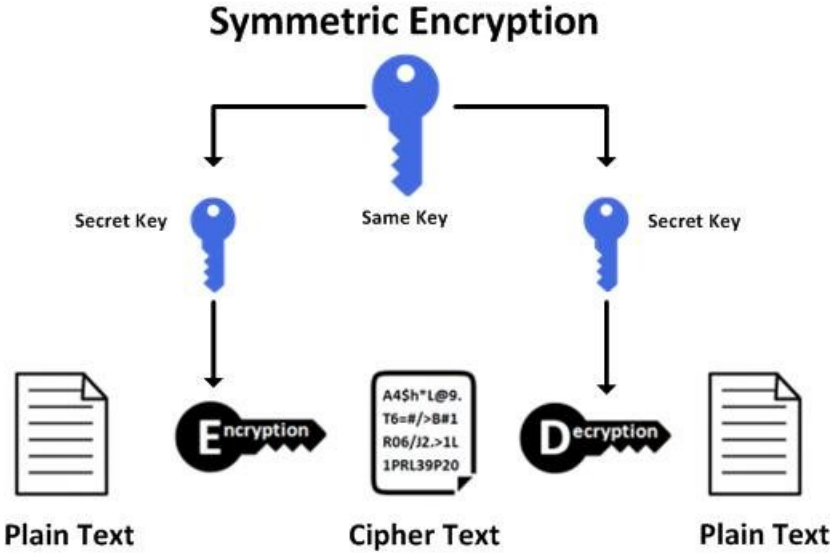
Σχήμα 1.1: Αρχικό μήνυμα killkingtomorrowmidnight κρυπτογραφείται σε ktm ioi lmd lon kri irg noh gwt

Παρατηρήστε τις έννοιες του μηνύματος, κρυπτομηνύματος και του κλειδιού (σκυτάλη). Ένα άλλο κρυπτοσύστημα που εμφανίστηκε αργότερα είναι αυτό του Καίσαρα, όπου το κάθε γράμμα του αρχικού μηνύματος αντικαθίσταται από ένα άλλο που προκύπτει ολισθαίνοντας την αλφάβητο κατά k γράμματα. Για παράδειγμα, το μήνυμα "ΝΑΙ" με ολίσθηση $k = 1$, μετατρέπεται "ΞΒΚ". Το κλειδί σε αυτή την περίπτωση είναι η τιμή του k.

Με το πέρασμα των χρόνων και την εξέλιξη των Μαθηματικών και της Επιστήμης των Υ-πολογιστών, τα κρυπτοσυστήματα έγιναν ολοένα και πιο σύνθετα. Χαρακτηριστικό παράδειγμα αποτελεί η μηχανή-Αίνιγμα (Enigma) που χρησιμοποιήθηκε από τους Γερμανούς για την ανταλλαγή μηνυμάτων κατά τον Δεύτερο Παγκόσμιο Πόλεμο (την οποία έσπασε ο Alan Turing με την βοήθεια Πολωνών μαθηματικών). Τέλος, τα πιο σύγχρονα κρυπτοσυστήματα (AES, RSA, El Gamal) βασίζονται σχεδόν εξ' ολοκλήρου σε αριθμοθεωρία, καθώς και σε υποθέσεις πέρι δυσεπίλυτων προβλημάτων[5].

Ας υποθέσουμε, λοιπόν, ότι η Alice και ο Bob θέλουν να επικοινωνήσουν μεταξύ τους (να ανταλλάξουν ένα μήνυμα m), χωρίς να καταλάβει το πραγματικό του περιεχόμενο ένας τρίτος ακροατής, η Eve. Όπως είδαμε και στα ιστορικά παραδείγματα, η Alice πρώτα κρυπτογραφεί το μήνυμα της m (plaintext) αξιοποιώντας ένα κλειδί k, και στέλνει το κρυπτογραφημένο μήνυμα c (ciphertext) στον Bob. Η διαδικασία αυτή πραγματοποιείται με μια κατάλληλη συνάρτηση $E(k, m)$, ώστε $c = E(k, m)$. Στη συνέχεια, ο Bob αναιρεί την κρυπτογράφηση, αποκρυπτο-

γραφώντας το μήνυμα με το ίδιο κλειδί, παίρνοντας πίσω το $m = D(k, c)$. Αυτό αποτελεί την γενική ιδέα της ***συμμετρικής*** κρυπτογραφίας.



Σχήμα 1.2: Συμμετρική κρυπτογραφία: Κρυπτογράφηση και Αποκρυπτογράφηση γίνεται με το ίδιο κλειδί

Όλα τα διαφορετικά σχήματα κρυπτογράφησης υφίστανται λόγω διαφορετικής επιλογής κατάλληλων συναρτήσεων $E(k, m), D(k, c)$. Όπως θα δούμε και στα επόμενα κεφάλαια, δεν έχει σημασία απλά η σωστή επιλογή των E, D, άλλα και η προσεκτική τους υλοποίηση, καθώς μια αμελής υλοποίηση πιθανώς να προσδώσει επιπρόσθετες πληροφορίες για το κλειδί και το μήνυμα μας στην Eve. Ενδεικτικά παραδείγματα συμμετρικής κρυπτογραφίας είναι το κρυπτοσύστημα του Καίσαρα, Vigenere, DES, AES, one-time pad[5].

Στην εργασία μας εστιάζουμε σε συμμετρική κρυπτογραφία, άλλα για λόγους πληρότητας θα σας αναφέρουμε και την ασύμμετρη, όπου οι Alice, Bob έχουν ένα ζεύγος δημόσιου (pk) και ιδιωτικού κλειδιού (sk). Έτσι, η Alice κρυπτογραφεί με το δημόσιο κλειδί του Bob (pk_{Bob}) ο οποίος με την σειρά του το αποκρυπτογραφεί με το ιδιωτικό του, δηλαδή $D(sk_{Bob}, E(pk_{Bob}, m)) = m$. Κλασικά παραδείγματα ασύμμετρης κρυπτογραφίας είναι το RSA, El Gamal[5].

Γεννάται φυσικά το ερώτημα για το πως κρίνουμε ότι ένα κρυπτοσύστημα επιτελεί επιτυχώς την λειτουργία του, δηλαδή η συνάρτηση κρυπτογράφησης $E(k, m)$ είναι τέτοια, ώστε η Eve να μη μπορεί να μάθει τίποτα απολύτως για το m εξετάζοντας μόνο το κρυπτοκείμενο c. Καταλήγουμε, λοιπόν στον ακόλουθο ορισμό του Shannon ο οποίος διατυπώνει την προηγούμενη απαίτηση με πιο αυστηρό τρόπο:

Ορισμός 1. Θεωρούμε το αρχικό μήνυμα m, το κρυπτοκείμενο c, και το κλειδί k τυχαίες μεταβλητές που παίρνουν τιμές στους χώρους M, C, K αντίστοιχα. Τότε το κρυπτοσύστημα (E, D) έχει τέλεια μυστικότητα κατά Shannon όταν:

$$\forall x \in M, y \in C \; P_{m\in M,k\in K}[m = x|c = y] = P_{m\in M}[m = x]$$

Με άλλα λόγια η πιθανότητα να έχει επιλεγεί το μήνυμα m προς κρυπτογράφηση, δεν μεταβάλλεται από την επιπρόσθετη γνώση του κρυπτοκειμένου c. Επομένως, λαμβάνουμε μηδενική πληροφορία από το c.

Στην πράξη, η απαίτηση ένα κρυπτοσύστημα (E, D) να έχει τέλεια μυστικότητα κατά *Shannon*, είναι αρκετά επιβλαβής, υπό την έννοια ότι ισχύει η παρακάτω πρόταση:

Πρόταση 1. Έστω ένα κρυπτοσύστημα (E, D) πάνω στους χώρους M, C, K. Τότε ισχύει ότι

$$|M| \leq |C| \leq |K|$$

Η παραπάνω πρόταση μας δεσμεύει στο να έχουμε μεγάλα μήκη κλειδιών (της τάξης των μηνυμάτων/κρυπτομηνυμάτων). Έτσι, καταφεύγουμε σε μια ασθενέστερη απαίτηση μυστικότητας, την υπολογιστική μυστικότητα/ασφάλεια (αφού ορίσουμε πρώτα την έννοια της αμελητέας συνάρτησης):

Ορισμός 2. Μια συνάρτηση $f(x) : \mathbb{N} \longrightarrow \mathbb{R}$ είναι αμελητέα (negligable) αν για κάθε πολυώνυμο $p(\cdot)$, υπάρχει ένας ακέραιος $N_p > 0$, ώστε για κάθε $x > N_p$:

$$|f(x)| < \frac{1}{p(x)}$$

Ορισμός 3. Θεωρούμε αντίπαλο (adversary) A ο οποίος προσπαθεί να υπολογίσει ένα κατηγόρημα $q : M \to \{0, 1\}$ πάνω στο αρχικό μήνυμα m. Ορίζουμε επίσης το πλεονέκτημα του A ως

$$Adv_q(A) = |P[A(c) = q(D(key, c))] - \frac{1}{2}|$$

Λέμε ότι ένα κρυπτοσύστημα με παράμετρο ασφαλείας λ είναι σημασιολογικά ασφαλές, αν για κάθε PPT A, για κάθε q:

$$Adv_q(A) = negl(\lambda)$$

Ζητάμε, δηλαδή, να μη μπορεί μια περιορισμένη (PPT) Eve να βρει την αληθοτιμή για οποιοδήποτε κατηγόρημα (π.χ. αν $m \in \{0, 1\}^*$ τότε να μαντέψει αν το τελευταίο *bit* είναι 0 ή 1) πάνω στο αρχικό μήνυμα, στηριζόμενη αποκλειστικά στο κρυπτοκείμενο c.

Στα επόμενα κεφάλαια, θα δούμε πώς με την χρήση της Μηχανικής μάθησης θα μπορέσουμε να εκμεταλλευτούμε τις τεχνικές υλοποιήσεις ([6],[7],[8]) ή την ίδια την κατανομή που δημιουργεί

το κρυπτοσύστημα πάνω στα κρυπτοκείμενα για την εξόρυξη μοτίβων και να κατασκευάσουμε στην πράξη έναν τέτοιο αντίπαλο A (π.χ. ένα Νευρωνικό Δίκτυο). Μάλιστα, ένας ισοδύναμος ορισμός με την σημασιολογική ασφάλεια, είναι η μη διακρισιμότητα (indistinguishability)[5], όπου ανάγουμε την ασφάλεια σε ένα παίγνιο μεταξύ δυο παικτών C, A. Σε αυτό, ο A επιλέγει δυο μηνύματα, τα οποία στέλνει στον C. Ο C κρυπτογραφεί τυχαία ένα από αυτα και στέλνει το κρυπτογραφημένο c στον A ο οποίος θέλει να βρει (με πιθανότητα πάνω του $\frac{1}{2}$) ποιο από τα δυο κρυπτογραφήθηκε. Αυτό το παίγνιο δρα ως έναυσμα για την υιοθέτηση ενός adversarial μοντέλου νευρωνικών δικτύων (GAN), όπου ο κάθε παίκτης είναι ο ίδιος ένα νευρωνικό δίκτυο. Η ιδέα αυτή μελετάται στο [4], και θα αναλυθεί περαιτέρω στο τέταρτο κεφάλαιο.

1.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση είναι το επιστημονικό πεδίο που ασχολείται με την μελέτη αλγορίθμων οι οποίοι επιτρέπουν στον υπολογιστή να εξορύξει κανόνες, βασιζόμενος σε μοτίβα που εντοπίζει σε δεδομένα. Διαφοροποιείται, λοιπόν, από τους συνηθισμένους αλγορίθμους, όπου η σχέση μεταξύ εισόδου και εξόδου είναι σαφώς διατυπωμένη εκ των προτέρων από τον προγραμματιστή. Στην Μηχανική Μάθηση το ίδιο το πρόγραμμα συνθέτει αυτή την σχέση, μελετώντας ένα σύνολο εκπαίδευσης (training set). Για τον λόγο αυτό κατατάσσεται στον ευρύτερο επιστημονικό κλάδο της Τεχνητής Νοημοσύνης (Artificial Intelligence).

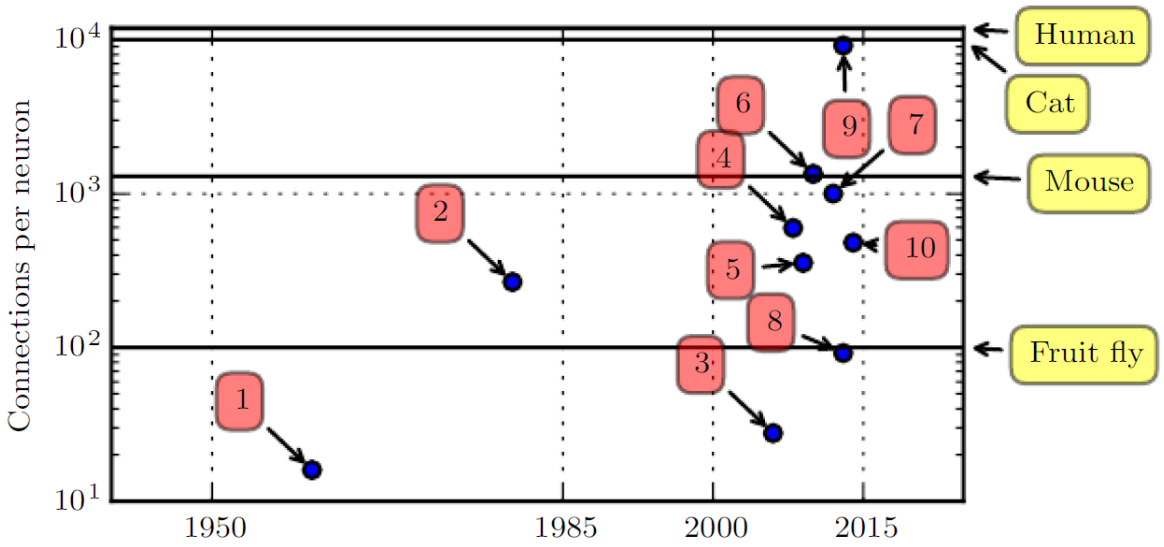
Τέτοιου τύπου αλγόριθμοι προτιμούνται σε περιπτώσεις όπου η σχέση εισόδου-εξόδου δεν διατυπώνεται εύκολα και επομένως επωφελούμαστε να επιτρέψουμε στο ίδιο το πρόγραμμα να την συνθέσει. Κάθε τέτοιο πρόγραμμα λαμβάνει ένα διάνυσμα από χαρακτηριστικά (features) $\overrightarrow{x}$, τα οποία συνήθως περιγράφουν ορισμένες ποιοτικές μεταβλητές του προβλήματος μας, και εξωτερικεύει ένα δίανυσμα τιμών $\overrightarrow{y}$. Για παράδειγμα, αν έχουμε σκοπό την πρόγνωση του καιρού, τότε το $\overrightarrow{x}$ μπορεί να περιέχει πληροφορίες για την θερμοκρασία, την ατμοσφαιρική πίεση και την υγρασία, ενώ το $\overrightarrow{y}$ την τιμή 0 ή 1 (για το αν θα βρέξει ή όχι).

Αν το $\overrightarrow{y}$ είναι διακριτό και πεπερασμένο, τότε λέμε ότι έχουμε πρόβλημα κατηγοριοποίηση-ταξινόμησης (classification), ενώ σε αντίθετη περίπτωση έχουμε παλινδρόμηση (regression). Σε κάθε περίπτωση, θέλουμε να προσεγγίσουμε την κατανομή $p(\overrightarrow{y}|\overrightarrow{x})$. Για τον σκοπό αυτό, το πρόγραμμα τροφοδοτείται με δεδομένα (σύνολο εκπαίδευσης-training data) τα οποία απαρτίζονται από στιγμιότυπα του προβλήματος μας. Αν τα χαρακτηριστικά $\overrightarrow{x}$ κάθε στιγμιοτύπου συνοδεύονται και με την έξοδο $\overrightarrow{y}$ (label) που τους αναλογεί, τότε έχουμε ένα πρόβλημα supervised learning. Αν κανένα δεν περιέχει label, έχουμε unsupervised learning, ενώ αν είμαστε σε ενδιάμεση κατάσταση, πρόκειται για semi-supervised learning. Σε αυτήν την εργασία θα εστιάσουμε σε supervised learning.

Αν και η δημοτικότητα της Μηχανικής Μάθησης τα τελευταία χρόνια έχει επανέλθει με

την ανάπτυξη του deep learning, ο κλάδος υφίσταται από τα μέσα του περασμένου αιώνα. Ήδη, στην δεκαετία του 1950, εμφανίζεται ένας από τους πρώτους αλγορίθμους σε αυτόν τον κλάδο, ο perceptron [9] και λίγα χρόνια αργότερα, δημοσιοποιείται και ένας regressor, ο adaptive linear element (ADALINE) [10], ο οποίος είναι στενά συνδεδεμένος (αποτελεί ειδική περίπτωση) με την στοχαστική μέθοδο των κλίσεων (stohastic gradient descent). Μολονότι υπάρχουν και άλλοι σπουδαίοι αλγόριθμοι (SVM[11], random trees, random forests[12]), τα δημοφιλέστερα, ίσως, αποτελέσματα οφείλονται σε αυτούς που αφορούν τα νευρωνικά δίκτυα (deep learning), τα οποία λόγω της αυξημένης πολυπλοκότητας τους έχουν την δυνατότητα να προσεγγίσουν πιο σύνθετες κατανομές (capacity). Ενδεικτικά παραδείγματα αποτελούν τα Convolutional Neural Networks (CNN), τα οποία κατάφεραν να αναγνωρίσουν χειρόγραφα ψηφία, αξιοποιώντας την βάση δεδομένων MNIST, ή η συγγραφή και επεξεργασία της φυσικής μας γλώσσας, αξιοποιώντας LSTM [13] (νευρωνικό δίκτυο με μνήμη).

Η άνοδος της δημοτικότητας της Μηχανικής Μάθησης είναι άρρηκτα συνδεδεμένη με την επίδοση των deep learning αλγορίθμων, η οποία βελτιώθηκε λόγω της αύξησης της υπολογιστικής ισχύος και του διαθέσιμου όγκου δεδομένων προς εκπαίδευση. Ενδεικτικά, ένα από τα παλαιότερα (πρώτο μισό του $20^{ου}$ αιώνα) δείγματα στα οποία εφαρμόστηκαν μέθοδοι μηχανικής μάθησης, το iris, έχει τάξη μεγέθους $10^2 - 10^3$, ενώ το WMT 2014 English to French dataset κυμαίνεται στα 10^9. Η πολυπλοκότητα των νευρωνικών δικτύων παρουσιάζει παρόμοια αύξηση, όπως φαίνεται στο παρακάτω σχήμα (θα ορίσουμε, στη συνέχεια, αναλυτικότερα την έννοια του νευρώνα) [14]:



Σχήμα 1.3: Πλήθος συνδέσεων κάθε νευρώνα, για διάφορες δομές/υλοποιήσεις νευρωνικών δικτύων και αντιπαραβολή τους με βιολογικούς νευρώνες διάφορων οργανισμών [14]

Όπου:

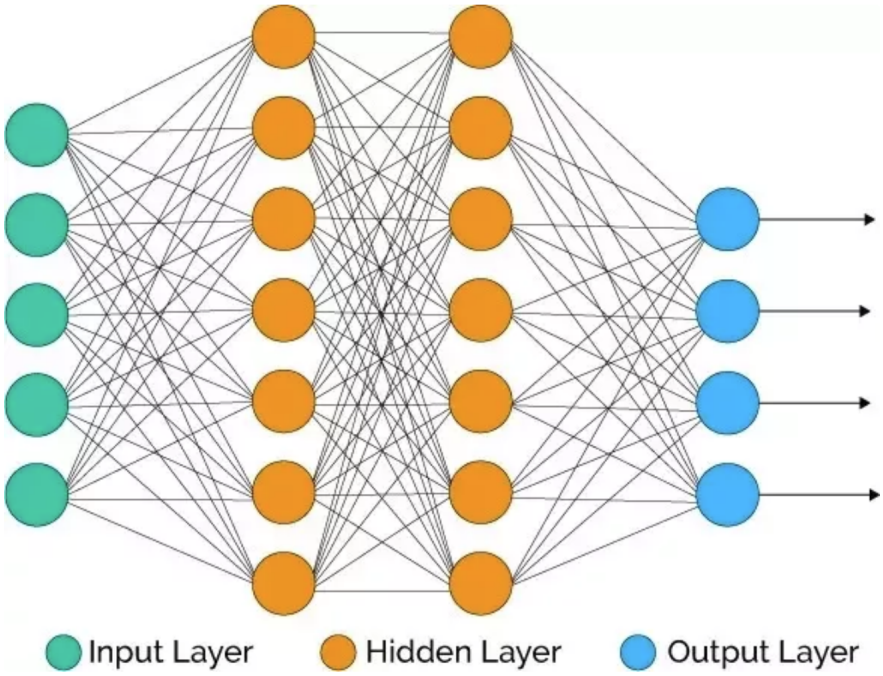
1. Adaptive linear element [10]
2. Neocognitron [15]
3. GPU-accelerated convolutional network [16]
4. Deep Boltzmann machine [17]
5. Unsupervised convolutional network [18]
6. GPU-accelerated multilayer perceptron [19]
7. Distributed autoencoder [20]
8. Multi-GPU convolutional network [21]
9. COTS HPC unsupervised convolutional network [22]
10. GoogLeNet [23]

1.2.1 Deep Learning

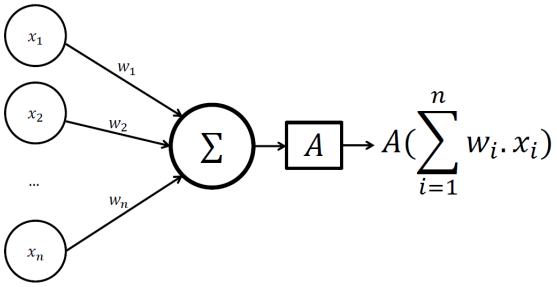
Το deep learning είναι υποκλάδος της Μηχανικής Μάθησης, το οποίο εστιάζει στην λειτουργία των Νευρωνικών Δικτύων. Ο όρος προέρχεται από τα πολλά στρώματα νευρώνων που χαρακτηρίζουν τα Νευρωνικά Δίκτυα (ή αλλιώς MLP-MultiLayerPerceptron). Πιο συγκεκριμένα, αποτελούνται από ένα input layer, όπου εισάγονται τα χαρακτηριστικά $\overrightarrow{x}$ (features), ένα ή περισσότερα hidden layers και ένα τελικό output layer, στο οποίο εξάγει το ζητούμενο $\overrightarrow{y}$. Παρατηρήστε ότι τα input, output layers είναι άμεσα συσχετισμένα με τη δομή των δεδομένων ($\overrightarrow{x}$, $\overrightarrow{y}$), σε αντίθεση με τα hidden layers.

Στο σχήμα 1.4, η κάθε ακμή σχετίζεται με την τιμή μιας παραμέτρου βάρους, η οποία εκφράζει τον βαθμό αλληλεπίδρασης μεταξύ των δυο νευρώνων που ενώνει. Η λειτουργία κάθε νευρώνα φαίνεται πιο ξεκάθαρα στο σχήμα 1.5.

Υπολογίζει δηλαδή το εσωτερικό γινόμενο $< \overrightarrow{w}, \overrightarrow{x} >$ το οποίο τίθεται ως όρισμα σε μια κατάλληλη συνάρτηση ενεργοποίησης A (activation function). Συχνά στην βιβλιογραφία, αντί να αναφερόμαστε σε επιμέρους διανύσματα βαρών, μπορούμε να ομαδοποιήσουμε όλα τα βάρη ενός στρώματος σε έναν πίνακα, όπου η κάθε γραμμή αναλογεί σε έναν νευρώνα και επομένως το συνολικό output του στρώματος είναι της μορφής $A(Wx + b)$, όπου το b είναι το bias. Η χρήση μιας **μη γραμμικής** A είναι απαραίτητη αν θέλουμε το δίκτυο μας να προσεγγίζει μια πιο σύνθετη συνάρτηση/κατανομή. Διαφορετικά, η τελική συνάρτηση $f(\overrightarrow{w}, \overrightarrow{x})$ του δικτύου μας θα ήταν γραμμική.



Σχήμα 1.4: Σχηματική αναπαράσταση MLP



Σχήμα 1.5: Σχηματική αναπαράσταση λειτουργίας νευρώνα [6]

Συνήθεις συναρτήσεις που αξιοποιούνται στα Νευρωνικά Δίκτυα (και θα συναντήσουμε στα επόμενα κεφάλαια) ως συναρτήσεις ενεργοποίησης:

- Σιγμοειδής (sigmoid) $s(x) = \frac{1}{1+e^{-x}}$. Επειδή $\forall x \; s(x) \in [0,1]$, η συνάρτηση εκφράζει ένα είδος πιθανότητας και συνήθως αξιοποιείται στο output layer σε προβλήματα δυαδικής ταξινόμησης (binary classification) ($label(x) = 1 \Leftrightarrow s(x) \geq 0.5$).
- Συνάρτηση softmax $softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, όπου z ένα διάνυσμα. Αποτελεί γενίκευση της σιγμοειδούς συνάρτησης (παρατηρήστε ότι για $z \in R^2$, τότε $P[y = 1|x] = s(x) = softmax(z)_1$) και αξιοποιείται στο output layer για την ταξινόμηση σε k κλάσεις. Πιο συγκεκριμένα $label(x) = argmax_i(softmax(z)_i)$.

- Υπερβολική εφαπτομένη $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Παρόμοια με την σιγμοειδή ($tanh(x) = 2s(2x) - 1$). Αξιοποιείται κυρίως σε hidden units (ιδιαίτερα σε LSTM), ή σε output layer σε προβλήματα παλινδρόμησης.

- Συνάρτηση ReLU (Rectified Linear Unit) $relu(x) = max\{0, x\}$. Παρόμοια χρήση με αυτή της υπερβολικής εφαπτομένης. Πιο αποδοτική και αποτελεσματική από την $tanh$, λόγω της απλότητας της και της αποφυγής κορεσμού παραγώγου: οι $s(x), softmax(z)_i, tanh(x)$ έχουν το πρόβλημα ότι για μεγάλες ή μικρές τιμές των x, z οι παράγωγοί τους είναι σχεδόν μηδενικές. Το γεγονός αυτό τις καθιστά προβληματικές, αν δεν χειριστούν σωστά, για την βελτιστοποίηση της προσέγγισης με τον στοχαστικό αλγόριθμο των κλίσεων.

Η αποτελεσματικότητα ενός Νευρωνικού Δικτύου επηρεάζεται σημαντικά και από την αρχιτεκτονική του. Μια πρώτη έννοια αρχιτεκτονικής είναι ο τρόπος διασύνδεσης μεταξύ στρωμάτων. Αν κάθε νευρώνας ενός στρώματος συνδέεται με όλους του προηγούμενου, τότε χαρακτηρίζεται ως fully connected (FC) layer. Αν αυτοί συσχετίζονται μέσω κατάλληλης συνάρτησης παραθύρου (η τιμή του νευρώνα εξαρτάται μόνο από μια περιοχή γειτωνικών νευρώνων του προηγούμενου στρώματος) (kernel), τότε ονομάζεται convolutional layer. Η χρήση του ενδείκνυται σε εφαρμογές που αφορούν εικόνες, όπου τα δεδομένα έχουν μια επιπλέον χωρική πληροφορία η οποία μπορεί να επεξεργαστεί με χρήση αυτών των kernel. Τέλος, όταν ένα στρώμα μπορεί να μεταβιβάσει πληροφορία και προς τα πίσω (σε προγενέστερο στρώμα), το δίκτυο λέγεται Recurrent Neural Network (RNN) (αξιοποιείται σε εφαρμογές που αφορούν την επεξεργασία της γλώσσας).

Στην αρχιτεκτονική εντάσσεται και το μέγεθος του δικτύου. Το συνολικό πλήθος στρωμάτων λέγεται βάθος (depth), ενώ το πλήθος νευρώνων σε κάθε στρώμα χαρακτηρίζει το πλάτος (width). Αυξάνοντας το πλάτος και το βάθος, βελτιώνουμε το capacity του μοντέλου μας, δηλαδή την κλάση συναρτήσεων που μπορούμε να προσεγγίσουμε. Το καθολικό θεώρημα προσέγγισης (universal approximation theorem) ([24], [25], [26]) μας αναφέρει ότι, υπό χαλαρές υποθέσεις για την συνάρτηση ενεργοποίησης (μη σταθερή, συνεχής και φραγμένη), μας αρκεί ένα hidden layer και ένας πεπερασμένος αριθμός νευρώνων, ώστε να μπορούμε να προσεγγίσουμε οποιαδήποτε συνεχή συνάρτηση σε συμπαγή χωρίο του R^n.

Θεώρημα 1. *Έστω $\varphi: \mathbb{R} \to \mathbb{R}$ μη σταθερή, φραγμένη και συνεχής συνάρτηση, $I_m = [0,1]^m$, $C(I_m) = \{f : I_m \to \mathbb{R}, f$ συνεχής $\}$. Τότε, για κάθε $\varepsilon > 0$ και $f \in C(I_m)$, υπάρχει ακέραιος N (πλήθος νευρώνων μοναδικού hidden layer), πραγματικοί $v_i, b_i \in \mathbb{R}$, και διανύσματα $w_i \in \mathbb{R}^m$, $i \in \{1, ..., N\}$, ώστε αν*

$$F(x) = \sum_{i=1}^{N} v_i \varphi({w_i}^T x + b_i)$$

τότε $|F(x) - f(x)| < \varepsilon$ *για κάθε* $x \in I_m$. *Επομένως, το σύνολο συναρτήσεων* $\{F(x),\ F(x) = \sum_{i=1}^{N} v_i \varphi(w_i{}^T x + b_i)\}$ *είναι πυκνό στον χώρο* $C(I_m)$.

Η έννοια της σύγκλισης σε μια κατανομή δε θα μπορούσε να έχει νόημα αν δεν ορίσουμε κάποια έννοια απόστασης από αυτήν. Στην Μηχανική Μάθηση, αυτό εκδηλώνεται μέσω της συνάρτησης κόστους $J(\theta)$. Στη πράξη, προσπαθούμε να εκτιμήσουμε την κατανομή μέσω της Εκτιμήτριας Μεγίστης Πιθανοφάνειας (ΕΜΠ) και επομένως το κόστος προς ελαχιστοποίηση (στην βιβλιογραφία μπορεί κανείς να την συναντήσει ως cross-entropy) είναι

$$J(\theta) = -E_{x,y\sim\widehat{p}_{data}}(\log p_{model}(y|x))$$

Ένας εναλλακτικός τρόπος να καταλήξουμε στο ίδιο αποτέλεσμα είναι μέσω του kullback-leibler (KL) divergence το οποίο εκφράζει την απόσταση μεταξύ δυο κατανομών p, q (δεν αποτελεί μετρική καθώς είναι ασύμμετρη και δεν ισχύει η τριγωνική ανισότητα). Ορίζεται ως:

$$D_{KL}(p||q) = E_{x\sim p}[\log p(x) - \log q(x)]$$

Στην μηχανική μάθηση, θέλουμε να ελαχιστοποιήσουμε την διαφορά μεταξύ της κατανομής του μοντέλου p_{model} και την εμπειρική κατανομή (όπως προκύπτει από την δειγματοληψία μας) του δείγματος $\widehat{p}_{data}$, άρα να ελαχιστοποιήσουμε την ποσότητα

$$D_{KL}(\widehat{p}_{data}||p_{model}) = E_{x\sim\widehat{p}_{data}}[\log \widehat{p}_{data}(x) - \log p_{model}(x)]$$

που ισοδυναμεί (καθώς ο πρώτος όρος δεν είναι συνάρτηση του μοντέλου) στην ελαχιστοποίηση της ποσότητας $-E_{x\sim\widehat{p}_{data}}[\log p_{model}(x)]$.

Ανάλογα με το μοντέλο που θεωρούμε ότι έχει η αρχική κατανομή, η παραπάνω έκφραση απλοποείται:

- Αν θεωρήσουμε ότι $p_{model}(y|x) = N(y; f(x;\theta), I)$, όπου $f(x;\theta)$ η έξοδος του Νευρωνικού Δικτύου με παραμέτρους θ (βάρη και bias), τότε η έκφραση εκφυλίζεται στο Μέσο Τετραγωνικό Σφάλμα (MSE):

  $$J(\theta) = \frac{1}{2}E_{x,y\sim\widehat{p}_{data}}||y - f(x;\theta)||^2 + const = \frac{1}{2n}\sum_{i=1}^{n}||y_i - f(x_i;\theta)||^2 + const$$

  Αξιοποιείται κυρίως για προβλήματα παλινδρόμησης.

- Σε προβλήματα δυαδικής ταξινόμησης προτιμάται η binary cross-entropy η οποία προκύπτει αυτούσια από τον ορισμό, θεωρώντας p_{model} να είναι Bernoulli, όπου πλέον:

  $$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log f(x_i;\theta) + (1 - y_i)\log(1 - f(x_i;\theta)))$$

Ο ορισμός αυτός γενικεύεται σε ταξινομήση κ-κλάσεων:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{k}(\mathbb{1}(y_i = j)\log f(x_i;\theta)))$$

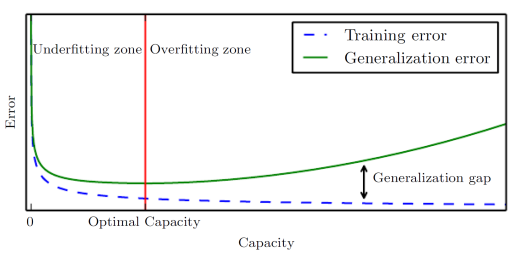
Οι δυο αυτές συναρτήσεις κόστους αξιοποιούνται σε προβλήματα ταξινόμησης, όπου εφαρμόζονται στο τελευταίο output layer. Ο λογάριθμος σε συνδυασμό με τα εκθετικά που εμφανίζονται στα συνήθη εξωτερικά στρώματα (sigmoid, softmax) εξουδετερώνουν τον κορεσμό της παραγώγου με αποτέλεσμα να αποφέρουν καλύτερα αποτελέσματα από ότι το MSE [14].

Ο λογισμός πίσω από την παράσταση $E_{x,y\sim\widehat{p}_{data}}[L(f(x;\theta),y)] = \frac{1}{n}\sum_{i=1}^{n} L(f(x;\theta),y)$, όπου $L(f(x;\theta,y)$ εκφράζει την απόκλιση από την πραγματική τιμή (loss function), ονομάζεται ERM (Empirical Risk Minimization). Ιδανικά, θα θέλαμε να ελαχιστοποιήσουμε την ποσότητα **ρίσκου** $J^*(\theta) = E_{x,y\sim p_{data}}[L(f(x;\theta),y)]$, όπου p_{data} είναι η πραγματική κατανομή των δεδομένων από το οποίο παράχθηκε το δείγμα εκπαίδευσης (με αντίστοιχη εμπειρική κατανομή $\widehat{p}_{data}$). Για την επίλυση αυτού του προβλήματος, ελαχιστοποιούμε την αντίστοιχη ποσότητα στην κατανομή του δείγματος, ευελπιστώντας αντίστοιχο αποτέλεσμα στην πραγματική κατανομή, δηλαδή $J(\theta) = E_{x,y\sim\widehat{p}_{data}}[L(f(x;\theta),y)]$. Το ERM προσεγγίζει αυτήν την ποσότητα με την ίσως πιο λογική εκτιμήτρια της μέσης τιμής, την δειγματική μέση τιμή: $E_{x,y\sim\widehat{p}_{data}}[L(f(x;\theta),y)] = \frac{1}{n}\sum_{i=1}^{n} L(f(x;\theta),y)$.

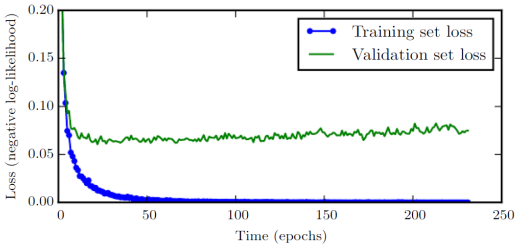
Οι τρεις συναρτήσεις κόστους που αναφέρθηκαν παραπάνω είναι οι συνηθέστερες που αξιοποιούνται στην Μηχανική Μάθηση και είναι αυτές που θα συναντήσουμε στην συνέχεια. Σε ορισμένες περιπτώσεις, ανάλογα με τη φύση του προβλήματος και των δεδομένων του, μπορεί να είναι επικερδές να δημιουργήσουμε δικές μας συναρτήσεις κόστους, όπως θα δούμε και στο τέταρτο κεφάλαιο, όπου θα αναλύσουμε το adversarial training [4].

Εφοδιασμένοι με τα παραπάνω εργαλεία, είμαστε πλέον σε θέση να περιγράψουμε τη γενική διαδικασία εκπαίδευσης και αξιολόγησης ενός Νευρωνικού Δικτύου. Αρχικά, χωρίζουμε το δείγμα μας σε σύνολο εκπαίδευσης και αξιολόγησης (training/test). Στη συνέχεια, διοχετεύουμε στο δίκτυο μας το σύνολο εκπαίδευσης, ελαχιστοποιώντας την $J(\theta)$ (συνήθως με την στοχαστική μέθοδο των κλίσεων). Επειδή εν γένει το σύνολο εκπαίδευσης είναι μεγάλο, είναι προτιμότερο να προπονούμε το Νευρωνικό μας Δίκτυο σε μικρότερα υποσύνολα του (batch) τα οποία το διαμερίζουν. Κάθε πλήρης διαπέραση του συνόλου εκπαίδευσης ονομάζεται epoch. Όσα περισσότερα epochs περάσουν, τόσο περιορίζουμε την συνάρτηση κόστους. Απώτερος σκοπός μας, όμως, είναι η γενίκευση, δηλαδή ευελπιστούμε η $J(\theta)$ να παρουσιάζει παρόμοια καθοδική πορεία και στο σύνολο αξιολόγησης. Η χρήση πολλών epochs οδηγεί στην απομνημόνευση των δεδομένων και όχι στην εξόρυξη ουσιωδών μοτίβων και πληροφοριών με αποτέλεσμα να έχουμε μεγάλο σφάλμα γενίκευσης (overfit). Για την αποφυγή αυτού του φαινομένου, δημιουργούμε

επιπλέον σύνολο, το validation set, στο οποίο ελέγχουμε την επίδοση του Νευρωνικού μας Δικτύου μετά από κάθε στάδιο εκπαίδευσης. Επειδή προσομοιάζει το test set, η συνάρτηση κόστους σε αυτό θα παρουσιάσει ελάχιστο (μετάβαση από underfit σε overfit), το οποίο επιδιώκουμε να βρούμε. Μόλις εντοπιστεί, τερματίζουμε την εκπαίδευση και ελέγχουμε την επίδοση του στο σύνολο αξιολόγησης.



(α′) Ποιοτική αναπαράσταση overfit-underfit [14]



(β′) Αναπαράσταση συνάρτησης κόστους συναρτήσει των epochs σε valdiation-test set [14]

1.3 Ανακεφαλαίωση

Έχουμε πλέον εφοδιαστεί με βασικές γνώσεις πάνω στην Κρυπτογραφία και την Μηχανική Μάθηση, εστιάζοντας στο Deep Learning, στο οποίο βασίζονται η πλειοψηφία των εφαρμογών που θα παρουσιαστούν στην παρούσα εργασία. Η επίγνωση της θεωρίας που διέπει αυτά τα πεδία, μας επιτρέπει μια ομαλότερη μετάβαση στις εφαρμογές, τις οποίες, πλέον μπορούμε να αξιολογήσουμε ορθότερα.

Μολονότι κάθε εφαρμογή που θα εξερευνήσουμε στην συνέχεια εξετάζει κάτι διαφορετικό, η γενικότερη φιλοσοφία είναι η απόπειρα ενσωμάτωσης ενός αλγορίθμου Μηχανικής Μάθησης (π.χ. Νευρωνικού Δικτύου) σε κάποιο παραδοσιακό στάδιο της Κρυπτογραφίας. Στο δεύτερο κεφάλαιο, θα παρεμβάλλουμε μια τέτοια μέθοδο για την κρυπτανάλυση ενός κρυπτοσυστήματος (DES [8], Masked AES [6], [7]), στο τρίτο για την προστασία και ενίσχυση του, ενώ στο τέταρτο θα αντικαταστήσουμε το ίδιο το κρυπτοσύστημα.

Η συγγραφή αυτής της εργασίας έγινε κατά τέτοιο τρόπο, ώστε να επωφελήσει τον αναγνώστη, ανεξαρτήτου προγενέστερου γνωστικού αντικειμένου. Η εισαγωγική θεωρία θα δώσει την δυνατότητα σε οποιονδήποτε να παρακολουθήσει το υπόλοιπο της εργασίας και ευελπιστούμε να του προσδώσει νέα ερεθίσματα, ώστε να επιδιώξει την περαιτέρω εμβάθυνση σε αυτούς τους δύο κλάδους. Ένας Κρυπτογράφος μπορεί να εστιάσει περισσότερο στις εφαρμογές, από τις οποίες μπορεί να αναθεωρήσει την ασφάλεια ορισμένων κρυπτοσυστημάτων, να δώσει μεγαλύτερη έμφαση στις τεχνικές τους υλοποιήσεις και να καταφύγει στην ενίσχυση τους αξιοποιώντας Νευρωνικά Δίκτυα. Ένας αφοσιωμένος ερευνητής στο πεδίο της Μηχανικής Μάθησης μπορεί να εμπνευστεί από τα Κρυπτογραφικά παίγνια (π.χ. προς δημιουργία νέων δομών GAN) και από

τις δυσκολίες που προκύπτουν από την προσέγγιση των σύνθετων συναρτήσεών της. Τέλος, μελλοντικοί ερευνητές δύναται να ανατρέξουν στην εργασία μας για την υιοθέτηση παρεμφερών μοντέλων σε ανάλογα προβλήματα.

Κεφάλαιο 2

Επίθεση Κρυπτοσυστημάτων με Μηχανική Μάθηση

Η επίθεση Κρυπτοσηστημάτων (Κρυπτανάλυση) ανάγεται στην εξαγωγή συμπερασμάτων για το αρχικό μήνυμα m ή το κλειδί k που χρησιμοποιήθηκε για την κρυπτογράφηση (θυμηθείτε τον ορισμό της σημασιολογικής ασφάλειας). Σε αυτήν την ενότητα θα εστίασουμε στην αποκάλυψη του κλειδιού k. Ο αντίπαλος (adversary A) αναζητά εμμέσως την κατανομή που δημιουργείται στα κλειδιά από το κρυπτοσύστημα, είτε βασιζόμενος στην επίγνωση μιας ποσότητας ζευγών μηνυμάτων-κρυπτομηνυμάτων (m_i, c_i) που προέκυψαν από το ίδιο κλειδί($\forall i\ c_i = E(k, m_i)$), είτε εκμεταλλευόμενος την τεχνική του υλοποίηση (side-channel attacks [27]), όπου πλέον θα έχει στην διάθεσή του διάφορες φυσικές ποσότητες (ακτινοβολία, θερμοκρασία επεξεργαστή, χρόνο εκτέλεσης) που προκύπτουν κατά την εκτέλεση αυτών των κρυπτογραφήσεων.

2.1 Κρυπτανάλυση DES, Triple-DES

Το DES[28] αποτελεί, ίσως, το χαρακτηριστικότερο παράδειγμα block cipher συμμετρικής κρυπτογραφίας. Τα block ciphers αποτελούν μια ειδική κατηγορία κρυπτοσυστημάτων, όπου το αρχικό μήνυμα m τεμαχίζεται σε υπομηνύματα m_i συγκεκριμένου μήκους, ώστε $m_1||m_2||...||m_n = m$. Το τελικό κρυπτοκείμενο προκύπτει από την κατάλληλη σύνθεση των επιμέρους $c_i = E(k, m_i)$ (π.χ. $c = E(k, m_1)||E(k, m_2)||...||E(k, m_n)$, όπως συμβαίνει στο ECB mode του DES).

Πιο συγκεκριμένα, το DES αποτελεί ένα δίκτυο Feistel 16 γύρων με $m_i, c_i, k \in \{0,1\}^{64}$. Ενδεικτικά, η κρυπτογράφηση:

$$m_i = L_0||R_0, |L_0| = |R_0|$$

$$L_{j+1} = R_j$$

$$R_{j+1} = L_j \oplus F(R_j, K_j),\ j = 0, ..., 15$$

$$c_i = L_{16}||R_{16}$$

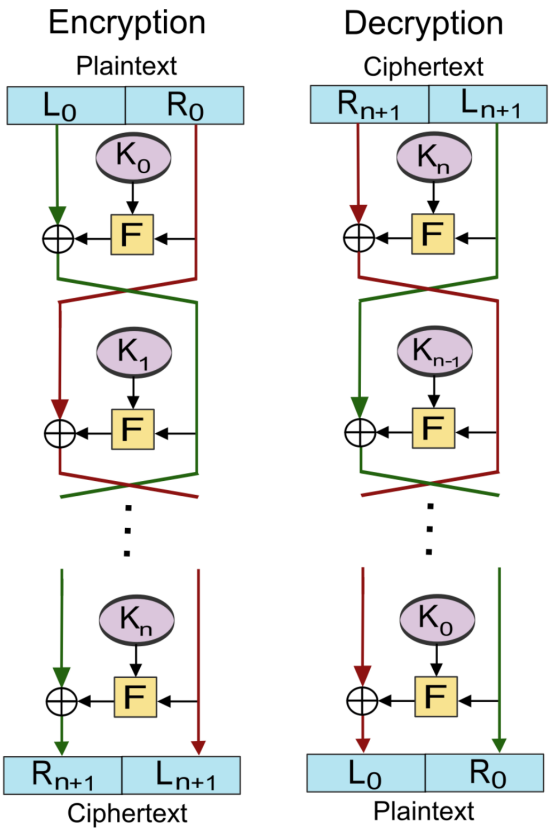
ενώ η αποκρυπτογράφηση:

$$R_j = L_{j+1}$$

$$L_j = R_{j+1} \oplus F(L_{j+1}, K_j),\ j = 15, ..., 0$$

$$m_i = L_0||R_0$$

όπου F μια κατάλληλη ψευδοτυχαία μετάθεση.

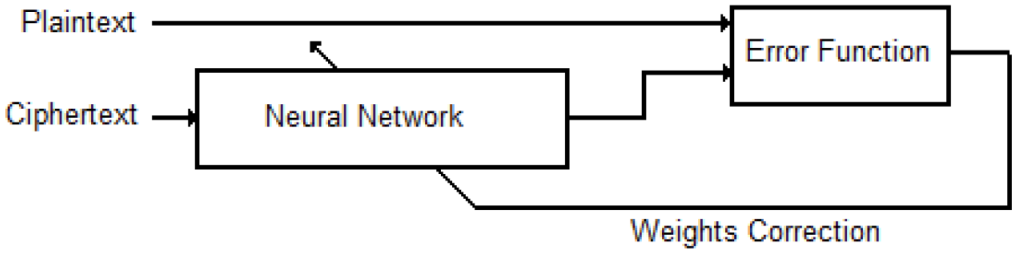


Σχήμα 2.1: Σχηματική Αναπαράσταση δικτύου Feistel

Ας υποθέσουμε ότι αναζητούμε το κλειδί k που χρησιμοποιήθηκε για την κρυπτογράφηση ενός ζεύγους (m, c). Αυτό ισοδυναμεί στην εξερεύνηση του k στον χώρο των κλειδιών K. Επειδή όμως, τα τελευταία 8 bits του κλειδιού είναι γνωστά (bits ισοτιμίας), ο χώρος αναζήτησης περιορίζεται σε $|K| = 2^{56}$. Επομένως, μια απλή brute force επίθεση χρειάζεται 2^{56} δοκιμές/χρονικές μονάδες. Αφού το συγκεκριμένο μέγεθος, με την σημερινή υπολογιστική ισχύ, φαντάζει προσιτό το DES αντικαταστήθηκε από πιο σύνθετα κρυπτοσυστήματα, όπως το AES, Triple-DES ($E_{3-DES}(k_1, k_2, k_3, m) = E_{DES}(k_1, D_{DES}(k_2, E_{DES}(k_3, m)))$, όπου ο χώρος αναζήτησης είναι 2^{128} και 2^{118} αντίστοιχα.

Γεννάται φυσικά το ερώτημα για το πόσο βελτιώνεται η απόδοση της κρυπτανάλυσης σε περίπτωση που έχουμε στην διάθεση μας περισσότερα ζεύγη (m_i, c_i) και επιχειρήσουμε μια πιο σύνθετη αναζήτηση από brute force. Μια πρώτη προσέγγιση είναι η κρυπτανάλυση διαφορών (differential cryptanalysis), όπως πρωτοαναπτύχθηκε από τους Shamir, Biham [30], η οποία κατάφερε να σπάσει το DES αξιοποιώντας 2^{47} ζεύγη με 2^{37} χρονική πολυπλοκότητα [32]. Εναλλακτική προσέγγιση αποτελεί η γραμμική κρυπτανάλυση (linear cryptanalysis [31]) που επιτυγχάνει τον ίδιο στόχο με μόλις 2^{43} ζεύγη σε 2^{39} χρόνικες μονάδες [29]. Η κρυπτανάλυση με νευρωνικό δίκτυο που θα παρουσιάσουμε [8] απαιτεί μέσο όρο ζευγών 2^{11} και έχει μέση διάρκεια 51 λεπτά. Παρόμοια βελτίωση παρουσιάζεται και στο 3-DES, όπου μια παραδοσιακή προσσέγιση, όπως ο αλγόριθμος του Lucks [33] χρειάζεται 2^{32} γνωστά ζεύγη και 2^{113} χρονικές μονάδες, ενώ η χρήση νευρωνικού θέλει κατά μέσο όρο 2^{12} ζεύγη και 72 λεπτά.

Η επίθεση που θα σας παρουσιάσουμε κατηγοριοποιείται ως Known Plaintext Attack (KPA), εφόσον έχουμε στην διάθεση μας ζεύγη (m_i, c_i) και αναζητούμε το κοινό k, ώστε $c_i = E(k, m_i)$. Σκοπός τους είναι η εκπαίδευση ενός νευρωνικού δικτύου το οποίο θα δέχεται ως *input* κρυπτοκείμενα c και θα εξωτερικεύει τα m που τους αναλογεί. Σχηματικά:



Σχήμα 2.2: Συνοπτική Αναπαράσταση δομής Νευρωνικού Δικτύου [8]

Υπό αυτήν την έννοια, μετά την εκπαίδευση, τα βάρη θα είναι τέτοια ώστε όλο το MLP να προσομοιάζει την κατανομή του DES για αυτό το κρυφό κλειδί k. Έτσι, μολονότι το κλειδί k παραμένει κρυφό και άγνωστο σε μας, πλέον θα είμαστε σε θέση να αποκρυπτογραφήσουμε οποιοδήποτε μήνυμα m το οποίο κρυπτογραφήθηκε με αυτό το κλειδί.

Το DES είναι σε λειτουργία ECB, ενώ ως συνάρτηση σφάλματος προτιμήθηκε η MSE. Το κάθε m, c κωδικοποιείται ως ένα διάνυσμα μήκους 64, όπου σε κάθε κελί αποθηκεύεται η τιμή του αντίστοιχου *bit*. Για να βρίσκονται όλες οι τιμές στο $[0, 1]$ αξιοποιήθηκε η σιγμοειδής συνάρτηση, ενώ η αρχιτεκτονική του Νευρωνικού Δικτύου ήταν cascaded-forward neural network, το οποίο μοιάζει με το παραδοσιακό MLP που περιγράψαμε στην εισαγωγή, με την διαφορά ότι το $i-$οστό στρώμα δε συνδέεται μόνο με το αμέσως επόμενο, αλλά και με όσα έπονται αυτού. Ο αλγόριθμος ελαχιστοποίησης του κόστους ήταν η συζυγής μέθοδος των κλίσεων, ενώ η εκπαίδευση τερματίζει μετά από 10^4 epochs ή όταν το MSE πέσει κάτω του 10^{-4}.

Για να αποδειχθεί ότι η επιτυχία της μεθόδου δεν εξαρτάται από τυχόν ιδιότητες ενός μεμονωμένου κλειδιού, αξιοποιήθηκαν 100 διαφορετικά σύνολα με 2^{20}, κατά μέσο όρο, ζεύγη (m, c), ώστε κάθε ένα από αυτά τα 100 να αντιστοιχούν σε διαφορετικά (ανεξάρτητα) κλειδιά. Τα σύνολα αυτά (και τα αντίστοιχα κλειδιά) δημιουργήθηκαν αξιοποιώντας κατάλληλη ψευδοτυχαία γεννήτρια αριθμών.

Διαδικασία εκπαίδευσης (για κάθε ένα από τα 100 παραπάνω σύνολα):

1. Το σύνολο διαμερίζεται σε training/test
2. Το δίκτυο διοχετεύεται με δεδομένα της μορφής (όπως αναπτύξαμε στην εισαγωγή) $\overrightarrow{x} = c \in \{0,1\}^{64}, \overrightarrow{y} = m \in \{0,1\}^{64}$
3. Αν μετά την εκπαίδευση, το MSE είναι μεγαλύτερο του 10^{-2}, τότε η εκπαίδευση χρήζεται ανεπιτυχής και το βήμα 2 επαναλαμβάνεται με νέα (τυχαία) αρχικοποίηση παραμέτρων θ και αρχιτεκτονική. Σε αντίθετη περίπτωση μεταβαίνουν στο στάδιο αξιολόγησης

Διαδικασία αξιολόγησης:

1. Υπολογίζονται τα $m' = round(f(c;\theta)), c \in$ training set.
2. Συγκρίνουν τα m' με τα πραγματικά m του συνόλου εκπαίδευσης ανά bit:

$$inside - error = \frac{\sum_{i=1}^{n}\sum_{j=1}^{l} m'(i,j) \oplus m(i,j)}{n * l}$$

   όπου $n = 64$ (πλήθος των bits ανά block), $l =$πλήθος των m', $m'(i,j) = j-$οστό bit $i-$οστού μηνύματος. Το συγκεκριμένο μέγεθος αποτελεί ένδειξη για το αν το μοντέλο έχει αφομιώσει πληροφορίες και μοτίβα από το σύνολο εκπαίδευσης (underfit).

3. Ύστερα παράγονται $m'_{new} = round(f(c;\theta))$, $c \in$ test set
4. Επαναλαμβάνεται η ίδια ακριβώς διαδικασία με το (2):

$$outside - error = \frac{\sum_{i=1}^{n}\sum_{j=1}^{l_{new}} m'_{new}(i,j) \oplus m_{new}(i,j)}{n * l_{new}}$$

   Οι παράμετροι είναι όπως πριν, με την διαφορά ότι $l_{new} \simeq 9 * l$ (αξιολογήθηκαν περισσότερα στοιχεία στο σύνολο αξιολόγησης). Χαρακτηρίζει την ικανότητα γενίκευσης (overfit).

Η διαδικασία αυτή επαναλαμβάνεται σε κάθε ένα από τα 100 σύνολα τόσο για το DES όσο για το 3-DES. Συνοπτικά, τα αποτελέσματα:

Table 1. Results of Implementing Neuro-cryptanalysis on DES and Triple-DES

Calculation	DES	Triple-DES
Total Number of trials	833	1093
Number of successful trials	100	100
Number of failure trials	733	993
Average number of plaintext-ciphertext pairs needed for training	2^{11}	2^{12}
Average time of successful training (min.)	51	72
Average time of failure trial (min.)	3	5
Average time until success (min.)	21	52
Average number of trials required to reach success	7.33	9.93
Average inside-error	0.022	0.028
Average outside-error	0.083	0.114

Table 2. Results of most successful 10 trials on DES

Net. Layout	text pairs needed	Inside error	Outside error	MSE reached	Training time (min.)	Epochs
128-256-256-128	2048	0.0279	0.0859	0.0133	39	308
128-256-256-128	2048	0.0273	0.0997	0.0104	39	298
128-256-256-128	2048	0.0298	0.1116	0.0274	41	334
128-256-512-256	2048	0.0341	0.1310	0.0151	42	356
128-256-256-128	2048	0.0283	0.0999	0.0049	42	341
128-512-256-256	2048	0.0378	0.1239	0.0474	42	351
128-256-256-128	2048	0.0344	0.1352	0.0359	42	355
128-256-256-128	2048	0.0295	0.0858	0.0006	43	388
128-256-512-128	2048	0.0415	0.1348	0.0768	43	379
128-256-256-128	2048	0.0267	0.1017	0.0759	43	411

Table 3. Results of most successful 10 trials on Triple-DES

Net. Layout	text pairs needed	Inside error (%)	Outside error (%)	MSE reached	Training time (min.)	Epochs
128-512-512-128	4096	0.0303	0.1170	0.0437	62	242
128-512-512-128	4096	0.0495	0.2103	0.0590	63	248
128-512-512-128	4096	0.0473	0.1259	0.0309	63	239
64-128-256-512-1024	4096	0.0407	0.1639	0.0373	63	218
128-512-512-128	4096	0.0408	0.1748	0.0455	63	265
128-512-512-128	4096	0.0538	0.2171	0.0575	64	233
128-256-512-256	4096	0.0319	0.1507	0.0236	64	253
128-512-512-128	4096	0.0459	0.1947	0.0155	64	235
128-512-512-128	4096	0.0334	0.1313	0.0242	64	229
128-256-512-512	4096	0.0362	0.1724	0.0350	64	230

Παρατηρούμε ότι, μολονότι το μέσο σύνολο εκπαίδευσης απαρτιζόταν από περίπου 2^{20} δεδομένα, το πείραμα επιτυγχάνει μια από τις συνθήκες τερματισμού με μόλις 2^{11}. Ακόμα πιο εντυπωσιακό είναι το αποτέλεσμα στο 3-DES, όπου με μόλις τα διπλάσια δεδομένα, πετυχαίνουν το ίδιο αποτέλεσμα σε ένα αρκετά πιο σύνθετο κρυπτοσύστημα. Επιπλέον, επειδή οι χρόνοι εκτέλεσης είναι μικροί, η εφαρμογή παρουσιάζει, πέρα από ακαδημαϊκή, και ουσιαστική πρακτική σημασία.

Το αποτέλεσμα ίσως γίνεται ακόμα πιο αξιόλογο, αν αναλογιστεί κανείς ότι η παρούσα εφαρμογή [8] δημοσιεύτηκε το 2012 και επομένως μπορεί να βελτιωθεί περαιτέρω, τόσο στην

ακρίβεια της (αξιοποιώντας πιο σύγχρονες έννοιες μηχανικής μάθησης που αναπτύχθηκαν και αναλύθηκαν αργότερα, όπως το dropout [34] και τα relu [35]) όσο και στον χρόνο εκτέλεσης της (εκμεταλλευόμενοι την ραγδαία ανάπτυξη της υπολογιστικής ισχύος).

Κλείνουμε αυτήν την υποενότητα, επισημαίνοντας τα σημεία που θεωρούμε ότι οφείλει ο αναγνώστης να δώσει περισσότερη σημασία. Αρχικά, όπως γίνεται φανερό και από την προηγούμενη παράγραφο, δεν έχει ιδιαίτερο νόημα να συγκρατήσει ακριβή νούμερα για τον χρόνο εκτέλεσης, πέρα του ότι γίνεται σε πρακτικά εφικτό χρόνο. Μάλιστα, μια σύγχρονη υλοποίηση δύναται να επιφέρει σημαντική μείωση χρόνου εκτέλεσης, μιας και η εφαρμογή πραγματοποιήθηκε αξιοποιώντας CPU (AMD Athlon X2, 1.9 Gigahertz), έναντι πολύ αποδοτικότερων GPU που χρησιμοποιούνται στις μέρες μας, και με μόλις 4 GB RAM. Έπειτα, ουσιώδους σημασίας είναι η μείωση στην τάξη μεγέθους των ζευγών (m, c). Τις περισσότερες φορές, η μεγαλύτερη δυσκολία τέτοιου είδους επιθέσεων είναι η απόκτηση επαρκούς πλήθους (m, c). Έτσι, ένα data leak μεγέθους $2^{11} \simeq 2000$ ζευγών είναι σημαντικά πιο πιθανό απ΄ ότι ένα με $2^{43} \simeq 10^9$ που χρειάζεται η γραμμική κρυπτανάλυση. Τέλος, αν και η χρήση των DES, 3-DES έχει γενικά εξαλειφθεί, το παραπάνω θετικό παράδειγμα δρα ως έναυσμα για την εφαρμογή της Μηχανικής Μάθησης σε πιο σύνθετα κρυπτοσυστήματα τα οποία χρησιμοποιούνται ευρέως, όπως το AES, με το οποίο θα ασχοληθούμε στις παρακάτω δυο ενότητες.

2.2 Μηχανική Μάθηση για την κρυπτανάλυση Masked AES μέσω profiled attack

Στην προηγούμενη εφαρμογή διαπιστώσαμε πως με την χρήση νευρωνικών δικτύων μπορούμε με αποδοτικό τρόπο, αξιοποιώντας 2^{11} ζεύγη (m, c), να σπάσουμε το DES. Σε αυτή την ενότητα θα εξετάσουμε το AES, ένα κρυπτοσύστημα που χρησιμοποείται ευρέως λόγω της αυξημένης πολυπλοκότητας του. Για τον λόγο αυτό, η εφαρμογή που θα σας παρουσιάσουμε βασίζεται σε απώλεια φυσικής πληροφορίας κατά την εκτέλεση του κρυπτοσύστηματος σε πραγματικό χρόνο (side channel attack).

Σε ένα side channel attack στοχεύουμε την εξόρυξη χρήσιμης πληροφορίας εκμεταλλευόμενοι διαρροές (trace) που προκύπτουν κατά την κρυπτογράφηση. Επειδή η $E(k, m)$ συχνά απαρτίζεται από μια σειρά συναρτήσεων, εμείς συνήθως εστιάζουμε σε μια από αυτές $f : M \times K \longrightarrow F$ (την πιο σύνθετη, όπως τα $SBox$ στο DES, AES). Στη γενική περίπτωση, καλούμαστε να υποθέσουμε ένα μοντέλο L που θα συσχετίζει την τιμή αυτής της συνάρτησης με την διαρροή που επιφέρει, δηλαδή μια συνάρτηση

$$L : F \longrightarrow Q \subseteq \mathbb{R}$$

$$L = L(f(k, m))$$

Ενδεικτικά, ένα κλασικό μοντέλο L που υιοθετείται είναι το Hamming Weight (HW), Hamming Distance [36]. Το Hamming Distance δυο αλφαριθμητικών ίσου μήκους, είναι το πλήθος των θέσεων στα οποία εμφανίζουν διαφορετικούς χαρακτήρες, ενώ το Hamming Weight είναι το Hamming Distance από το αλφαριθμητικό που αποτελείται μόνο από τον μηδενικό χαρακτήρα του αφλαβήτου. Για $\varepsilon \sim N(0,1)$(τυχαίος θόρυβος), θεωρούμε τη $j-$οστή διαρροή πληροφορίας τη χρονική στιγμή t:

$$^{j}_{t}T_i = L(f(k_i, m)) + \varepsilon$$

Με βάση αυτό, ορίζουμε το σύνολο που περιέχει τα $j-$οστά ίχνη που αφορούν το $i-$οστό κλειδί για όλες τις χρονικές στιγμές:

$$^{j}T_i = \{^{j}_{t}T_i \in R | t \in [1, ..., n]\}$$

Αξίζει να σημειωθεί ότι, επειδή η f αποτελεί τμήμα της συνολικής κρυπτογράφησης, δεν είναι απαραίτητο τα m, k να αναφέρονται σε ολόκληρο το μήνυμα, κλειδί αντίστοιχα (π.χ. μπορεί να είναι το τελευταίο bit). Στην ανάλυση που θα ακολουθήσει θεωρούμε ότι στην $f(m, k)$ το m είναι γνωστό και το ζητούμενο μας είναι η εύρεση του k. Πιθανώς να φαντάζει ισχυρή απαίτηση αυτή, αλλά στη πραγματικότητα δεν είναι παράλογο να έχουμε κάποια πληροφορία για την δομή τμήματος m του συνολικού M την οποία μπορούμε να εκμεταλλευτούμε. Έτσι, η ενίσχυση κρυπτοσυστημάτων απέναντι σε τέτοιες επιθέσεις, συχνά, επιτυγχάνεται αξιοποιώντας masks, δηλαδή d ομοιόμορφα επιλεγμένες τιμές $\{v_1, ..., v_d\}$ στο $V = \{0,1\}^l$ οι οποίες "κρύβουν" το μήνυμα m (π.χ. εφαρμόζοντας xor $f(k, m \oplus v_i)$ σε κάθε εκτέλεση).

Η παρούσα εφαρμογή δρα σε δυο στάδια (εστιάζοντας κυρίως στο πρώτο):

- profiling attack με χρήση Μηχανικής Μάθησης για την εξουδετέρωση της επίδρασης των $\{v_1, ..., v_d\}$.
- Παραδοσιακό non-Profiling Attack (π.χ. Correlation Power Analysis (CPA)) για την εύρεση του k.

Στο profiling attack, δημιουργούμε μια εκτίμηση (ένα profile) της κατανομής $p[^{j}T_i | L(f(k_i, m); \theta_i]$ (θ_i πιθανοί παράμετροι της κατανομής) για κάθε τιμή $k_i \in K$. Πρόκειται για την ισχυρότερη μορφή side channel attack αφού απαιτεί την χρήση μιας συσκευής που αποτελεί ακριβές αντίγραφο (profiling device) της συσκευής στόχου. Αξιοποιώντας το profiling device σχηματίζουμε μια a priori γνώση για την κατανομή $p[^{j}T_i | L(f(k_i, m); \theta_i]$ για κάθε κλειδί. Για παράδειγμα, η εκτίμηση της κατανομής στην περίπτωση του Template Attack (ειδική κατηγορία profiling attack) πραγματοποιείται θεωρώντας ότι τόσο το μοντέλο διαρροής $L(f(k_i, m))$, όσο και ο θόρυβος, ακολουθούν κατάλληλη προσδιοριστέα κανονική κατανομή $\forall k_i \in K$.

Μετά το profiling στάδιο θα είμαστε σε θέση, δεχόμενοι ένα νέο ίχνος T από την συσκευή στόχο, να βρούμε το κατάλληλο k:

$$k = argmax_{k\in K}(p[L(f(k,m))|T]) = argmax_{k\in K}(\frac{p[T|L(f(k,m)) * p(L(f(k,m))]}{p(T)})$$

$$\Leftrightarrow k = argmax_{k\in K}(p[T|L(f(k,m)] * p[L(f(k,m)])$$

όπου το $p[L(f(k,m))]$ εκτιμάται από τον χρήστη.

Μια παραδοσιακή προσέγγιση της κατανομής $p[T|L(f(k,m))]$ γίνεται με την στοχαστική επίθεση (stochastic attack) [38]. Σε αυτήν μετασχηματίζουμε την αναζήτηση μας σε ένα πρόβλημα παλινδρόμησης (τη χρονική στιγμή t):

$${}^{j}_{t}T_i = L(f(k_i,m)) + \epsilon =_t h(f(k_i,m)) +_t R =_t c + \sum_{u=1}^{U}({}_t a_u g_u(f(k_i,m))) +_t R$$

όπου ${}_tR \sim N(0,\Sigma)$ θόρυβος, $\{{}_tc, {}_t a_1, ..., {}_t a_U\}$ οι παράμετροι του μοντέλου παλινδρόμησης ${}_th$ και τα $\{g_1, ..., g_U\}$ η βάση της (ο αναγνώστης ίσως έχει συνηθίσει τη βάση να είναι πολυωνυμική ή γραμμική). Συνηθίζεται:

$$g_j(f(k_i,m)) = Bit_j(f(k_i,m))$$

Έτσι, η κατανομή $p[T|L(f(k,m))$ ακολουθεί $N(h(f(k_i,m)),\Sigma), h(f(k_i,m)) = \{{}_1h(x), {}_2h(x), ..., {}_nh(x)\}$. Η επίθεση αυτή γενικεύτηκε και σε masked υλοποιήσεις [37].

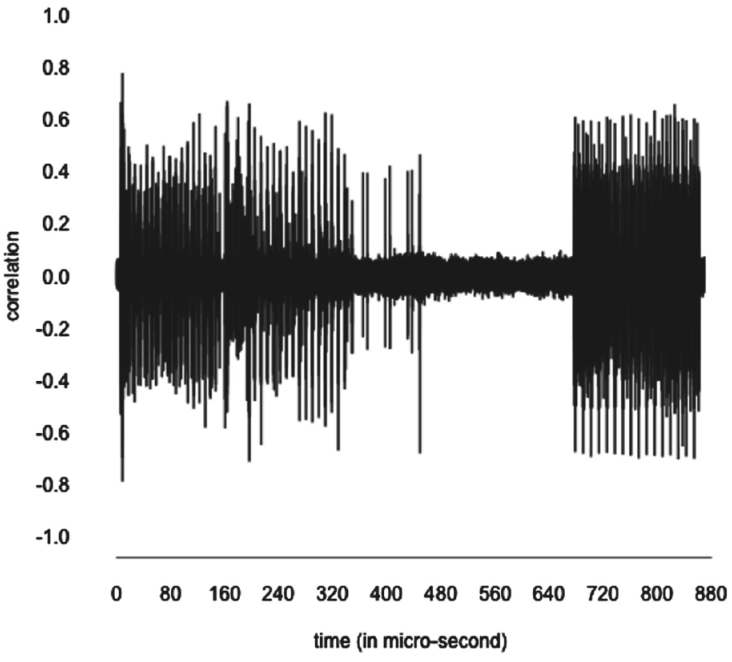
Η Template Attack είναι πιο ισχυρή επίθεση από την stochastic attack υπό την λογική ότι προτείνει μια πολύ πιο σύνθετη ανάλυση, απαιτώντας παράλληλα και πολλά περισσότερα δεδομένα. Πιο συγκεκριμένα και οι δυο επιθέσεις θέλουν να χαρακτηρίσουν το σήμα $L(f(k,m))$ και τον θόρυβο ϵ. Στο Template Attack, το σήμα ${}_th(f(k_i,m))$ προσεγγίζεται με κανονικές κατανομές (για κάθε κλειδί), ενώ στο stochastic attack επιδιώκουμε τον προσδιορισμό του γραμμικού, ως προς την βάση g_u, τμήματος του. Τέλος, παρατηρήστε ότι στην στοχαστική επίθεση ο πίνακας διασποράς του θορύβου δεν εξαρτάται από το κλειδί που εξετάζουμε. Αντιθέτως, στο Template Attack έχουμε διαφορετικό θόρυβο (και επομένως πίνακα συνδιασποράς) για κάθε κλειδί. Είναι σύνηθες $|K| = 256$ και $U = 8$ (ανά byte). Τότε, το Stochastic Attack απαιτεί τον προσδιορισμό ενός κοινού πίνακα συνδιασποράς και 8 συντελεστών, ενώ το Template Attack την εκτίμηση 256 πινάκων συνδιασποράς και $L(f(k,m))$ (η εκτίμηση γίνεται του L γίνεται παίρνοντας την μέση τιμή των ιχνών). Οι [41] ανέλυσαν και σύγκριναν περαιτέρω τις δυο αυτές επιθέσεις.

Παράλληλα, αναπτύχθηκαν και άλλες επιθέσεις [39], όπως οι Templates Before Processing, Templates During Processing, Templates After Processing. Η εφαρμογή στοχεύει σε μια αποτελεσματικότερη εναλλακτική, με αποδοτικό χρόνο εκτέλεσης, χαμηλή εκμετάλλευση μνήμης και υψηλό ποσοστό επιτυχίας. Σε αυτά τα κριτήρια θα εξεταστούν δυο μέθοδοι της Μηχανικής Μάθησης: τα Random Forest και Support Vector Machines.

Η επίθεση βασίστηκε σε ίχνη (traces) ηλεκτρομαγνητικής ακτινοβολίας ([40]), ενώ η μηχανή στόχος (Atmel ATMega-163 smart card) υλοποιεί masked (Rotating Sbox Masking)AES-258 [87]. Η συνάρτηση f (όπως αναπτύξαμε προηγουμένως) είναι ο πρώτος γύρος του AES-256, ενώ σκοπός αποτελεί η εξόρυξη των 128 πρώτων bits του κλειδιού. Τα ίχνη που έχουμε στην διάθεση μας απαρτίζονται από 435002 χαρακτηριστικά/πληροφορίες για το κλειδί (ηλεκτρομαγνητική ακτινοβολία σε διάφορες χρονικές στιγμές). Έχουμε επίγνωση των masks, τα οποία παράγονται από μια τυχαία (επίσης γνωστή) τιμή (offset value).

Αρχικά, δεχόμενοι ένα ίχνος, θα πρέπει να είμαστε σε θέση να εντοπίσουμε ποια μάσκα αξιοποιήθηκε για να αναιρέσουμε τη δράση της. Για τον εντοπισμό της μελετήθηκε η αποδοτικότητα ενός Random Forest (RF) (500 δέντρων), Support Vector Machine (SVM) (με Gaussian kernel), και παραδοσιακού Template Attack (TA), Stochastic Attack (SA) αξιοποιώντας σύνολα εκπαίδευσης και αξιολόγησης που απαρτίζονταν από 1500 ίχνη.

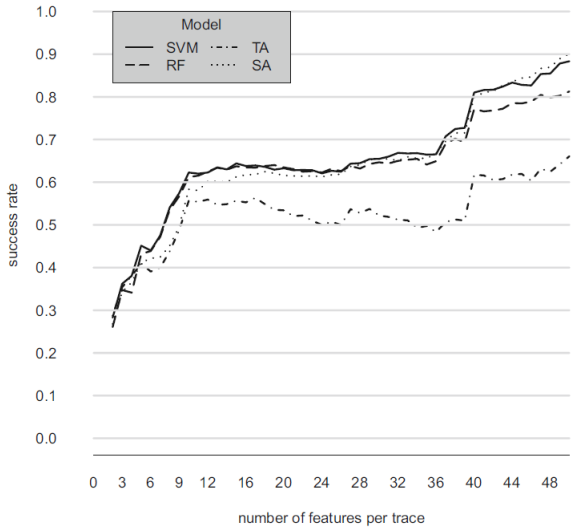
Πρώτο βήμα είναι η μελέτη των χαρακτηριστικών και η απομόνωση των σημαντικότερων. Αποφεύχθηκαν παραδοσιακές μέθοδοι (Principal Component Analysis) για λόγους μνήμης ($1500 * 453002$ δεδομένα). Η ανάλυση έγινε εξετάζοντας την ευστοχία στο σύνολο αξιολόγησης σαν συνάρτηση των χαρακτηριστικών που παρουσίαζαν το μεγαλύτερο βαθμό γραμμικής εξάρτησης (Pearson) με το offset value. Ενδεικτικά:



Σχήμα 2.3: Συσχέτιση μεταξύ των τιμών ακτινοβολίας και της offset value [7]

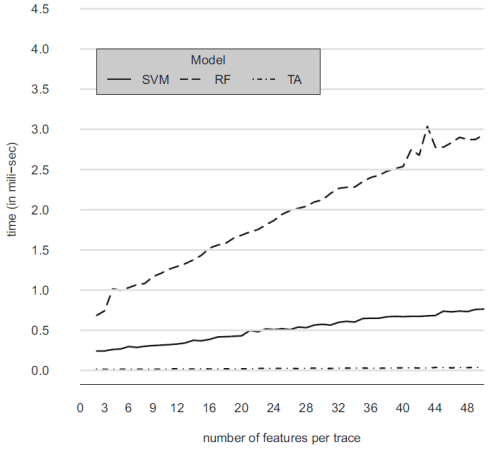
Σύμφωνα με το παραπάνω σχήμα, αναμένουμε τα περισσότερα χαρακτηριστικά να εμφανίζονται από το δεξί τμήμα του γραφήματος.

Τα 50 χαρακτηριστικά που απομονώθηκαν ταξινομήθηκαν σε φθίνουσα σειρά (ως προς Pearson correlation), ενώ η ευστοχία συναρτήσει του πλήθους τους παρουσιάζεται στο παρακάτω σχήμα:

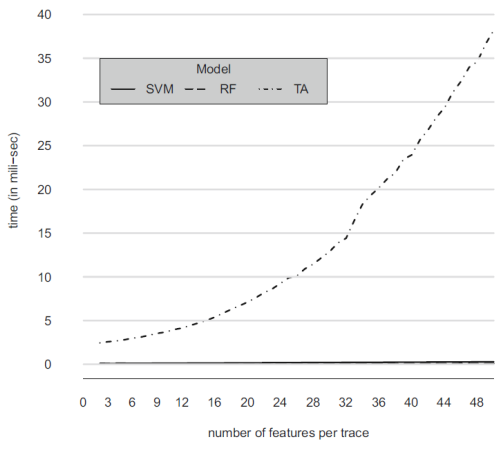


Σχήμα 2.4: Ευστοχία ως συνάρτηση του πλήθους χαρακτηριστικών [7]

Όπως παρουσιάζεται και στο σχήμα, τα SVM, RF, SA έχουν παρεμφερή ευστοχία, η οποία υπερβαίνει αυτή του TA. Όσον αφορά τη χρονική πολυπλοκότητα, παρατηρήθηκε ότι τα RF παρουσιάζουν με διαφορά την μεγαλύτερη στο στάδιο εκπαίδεσης, ενώ το TA σε αυτό της επίθεσης. Λόγω της χρονικής τους υστέρησης, το RF δε θα αξιολογηθεί στο στάδιο επίθεσης.



(α′) Χρόνος επεξεργασίας ενός ίχνους στο στάδιο εκπαίδευσης, συναρτήσει του πλήθους των χαρακτηριστικών [7]



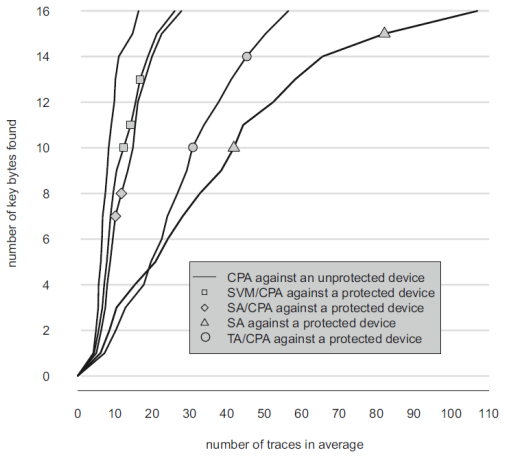
(β′) Χρόνος επεξεργασίας ενός ίχνους στο στάδιο επίθεσης, συναρτήσει του πλήθους των χαρακτηριστικών [7]

Η επίθεση χωρίζεται σε δυο στάδια: την εξαλειφή της μάσκας και την έυρεση του κλειδιού.

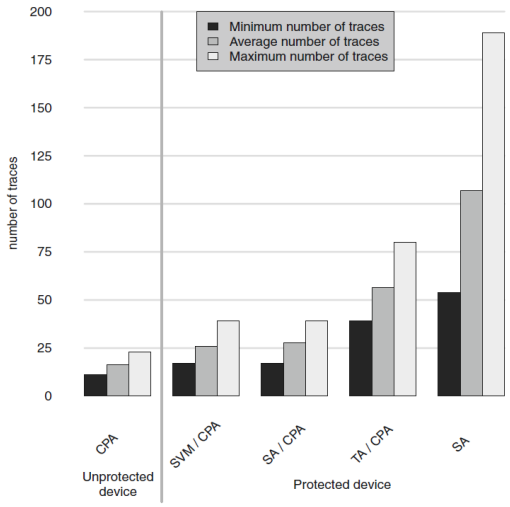
Μελετήσανε:

- CPA σε AES χωρίς μάσκα (16.3 ίχνη)
- SVM για την εξουδετέρωση των μασκών, ακολουθούμενο από CPA (26.3 ίχνη)
- SA για την εξουδετέρωση των μασκών, ακολουθούμενο από CPA
- TA για την εξουδετέρωση των μασκών, ακολουθούμενο από CPA (56.4 ίχνη)
- SA αυτούσιο (εύρεση κλειδιού χωρίς το non-profiling CPA βήμα)

Τα αποτελέσματα παρουσιάζονται παρακάτω (το πείραμα αξιολόγησης πραγματοποιήθηκε 4 φορές):



(α′) Σύγκριση ευστοχίας (Bytes κλειδιού) [7]



(β′) Ελάχιστος, μέσος και μέγιστος αριθμός ιχνών που χρειάζεται η κάθε επίθεση για την εξόρυξη του κλειδίου k [7]

Παρατηρούμε ότι το μοντέλο επίθεσης που βασίζεται στη Μηχανική Μάθηση (SVM/CPA) χρειάζεται κατά μέσο όρο τα μισά ίχνη (26) από ότι το TA/CPA (56.4) και το ένα τέταρτο σε σχέση με την ενιαία στοχαστική επίθεση SA. Εξίσου αξιόλογη είναι η επίδοση της SVM/CPA επίθεσης από άποψη χρόνου, η οποία χρειάζεται περίπου 20 δευτερόλεπτα, έναντι των 80 που απαιτεί το SA. Ενδεικτικά, η CPA non-profiled επίθεση στο AES χωρίς μάσκα, θέλει 17 ίχνη και 5 δευτερόλεπτα. Έτσι, βλέπουμε ότι η Μηχανική Μάθηση μας παρέχει την δυνατότητα να επιλύσουμε ένα αρκετά πιο σύνθετο πρόβλημα, με μόλις τα διπλάσια ίχνη και τετραπλάσιο χρόνο. Μια πιθανή εξήγηση της επιτυχίας των αποτελεσμάτων είναι το γεγονός ότι οι SA, TA, σε αντίθεση με το SVM, υποθέτουν ότι η κατανομή $p[T|L(f(k,m))]$ είναι κανονική, στην οποία καλούμαστε να προσδιορίσουμε τις παραμέτρους. Η χαλάρωση αυτής της υπόθεσης (πέρι κανονικότητας) μοιάζει να αποφέρει ανάλογα αποτελέσματα και σε άλλες δημοσιεύσεις[44],

όπου σε παρεμφερές πρόβλημα (επίθεση απλής και Masked υλοποίησης AES) διερευνήθηκε η αποτελεσματικότητα του Deep Learning και των Νευρωνικών Δικτύων.

Ένα κεντρικό κομμάτι της επίθεσης είναι το non-profiling attack αξιοποιώντας την παραδοσιακή μέθοδο CPA. Είναι λογικό να αναρωτηθεί κανείς για το αν θα μπορούσε να εφαρμοστεί η Μηχανική Μάθηση σε ένα non-profiling attack σενάριο (αντί του CPA). Πράγματι, στην επόμενη υποενότητα θα αναλύσουμε μια side channel attack που θα αξιοποιήσει Νευρωνικά Δίκτυα [6].

2.3 Deep Learning για την κρυπτανάλυση Masked AES μέσω non-profiled attack

Όπως είδαμε και προηγουμένως, η χρήση μεθόδων Μηχανικής Μάθησης για Profiling Attack φαίνεται φυσιολογική, αφού, λόγω της συσκευής αντιγράφου (profiling device) που έχουμε στην διάθεση μας, μπορούμε να γεννήσουμε επαρκώς μεγάλα σύνολα δεδομένων για εκπαίδευση και αξιολόγηση. Σε περίπτωση που δεν έχουμε profiling device, τότε καταφεύγουμε σε Non-Profiled side channel attacks όπου αρκούμαστε σε έναν περιορισμένο αριθμό ιχνών τα οποία προέρχονται εξ΄ ολοκλήρου από την συσκευή στόχο. Η μόνη απαίτηση τέτοιων επιθέσεων είναι ότι όλα αυτά τα ίχνη αντιστοιχούν στο ίδιο εκτιμητέο κλειδί $k^* \in K$ και ότι γνωρίζουμε την είσοδο m (ή μερικές φορές και έξοδο c) που τους αναλογεί.

Σε περιπτώσεις Profiling attack η χρήση Deep Learning μοιάζει απλή. Στο profiling phase παράγουμε ένα σύνολο N ιχνών για κάθε $k \in K$, $T_k = \{T_{i,k} | i = 1, 2, ..., N\}$ αξιοποιώντας το profiling device. Συνθέτουμε έτσι ένα σύνολο εκπαίδευσης:

$$X = \cup_{i=1}^{K} T_k$$

με labels $y(T_{i,k}) = k$. Μετά το στάδιο εκπαίδευσης, το νευρωνικό, δεχόμενο ένα σύνολο από M νέα ίχνη (από την συσκευή στόχο) $(T_i)_{1 \leq i \leq M}$, παράγει M διανύσματα $y_i = y(T_i) \in R^{|K|}$ των οποίων σε κάθε συντεταγμένη αντιστοιχεί μια τιμή (σκορ) για την πιθανότητα $P[k^* = k]$. Το k^* είναι το κλειδί που αναλογεί στο μεγαλύτερο αθροιστικό σκορ:

$$k^* = argmax_{j \in K} (\sum_{i=1}^{M} y_i)[j]$$

Προφανώς σε non-profiling attack μια τέτοια προσέγγιση αποτυγχάνει, αφού δεν έχουμε στην διαθεσή μας κανένα label (έχουμε μόνο την πληροφορία ότι όλα τα ίχνη προέρχονται από το ίδιο κλειδί k^* το οποίο θέλουμε να βρούμε). Για τον σκοπό αυτό, θα εμπνευστούμε από το CPA (Correlation Power Analysis) που αναφέραμε και στην προηγούμενη ενότητα. Φανταστείτε ότι η υπό μελέτη συνάρτηση (που αποτελεί τμήμα του συνολικού AES) είναι η $f(m, k) = Sbox(m \oplus k)$, όπου τα m, k μπορεί να αναφέρονται σε τμήματα του συνολικού

μηνύματος/κλειδιού αντίστοιχα. Μάλιστα, πολλές φορές (επειδή ο χώρος κλειδιών K είναι μεγάλος : 2^{256}), προτιμάμε να αποκρυπτογραφούμε τμηματικά (ανά *byte*). Έτσι, πλέον $k, m \in \{0, 1, ..., 255\}$. Για την επίτευξη αυτής της επίθεσης έχουμε στην διάθεση μας N ίχνη $(T_i)_{1 \leq i \leq N}$ τα οποία αντιστοιχούν στη διαρροή εκτέλεσης του $f(m_i, k)$ όπου m_i γνωστά. Για αυτά τα m_i υπολογίζουμε όλα τα πιθανά k, τα $(V_{i,k})_{1 \leq i \leq N}$ ώστε $V_{i,k} = f(m_i, k)$. Στη συνέχεια, εφαρμόζουμε σε αυτά ένα μοντέλο διαρροής L, όπως για παράδειγμα το Hamming Weight $H_{i,k} = HW(V_{i,k})$. Τέλος, υπολογίζουμε τη γραμμική (Pearson) συσχέτιση μεταξύ του διανύσματος $(H_{i,k})_{1 \leq i \leq N}$ και του $(T_i)_{1 \leq i \leq N}$. Υποθέτοντας ότι το μοντέλο που χρησιμοποιήσαμε είναι εύστοχα επιλεγμένο, τότε για το σωστό k^*, το $(H_{i,k^*})_{1 \leq i \leq N}$ θα είναι ισχυρά συσχετισμένο με τα ίχνη, ενώ τα υπόλοιπα αρκετά ασθενέστερα. Έτσι:

$$k = argmax_{k \in K} Pearson[(H_{i,k^*})_{1 \leq i \leq N}, (T_i)_{1 \leq i \leq N}]$$

όπου για δυο διανύσματα (δείγματα) X, Y μεγέθους n με μέσες τιμες $\overline{X}, \overline{Y}$:

$$Pearson(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

Ο παραπάνω λογισμός περιγράφει την περίπτωση όπου τα ίχνη δεν έχουν χρονική διάρκεια, δηλαδή τα T_i, $H_{i,k} \in \mathbb{R}$. Η μέθοδος επεκτείνεται με τον πιο φυσικό τρόπο, όταν η διαρροή έχει διάρκεια T, όπου πλέον T_i, $H_{i,k} \in \mathbb{R}^T$. Πλέον, για κάθε κλειδί και για κάθε $0 \leq t \leq T$, υπολογίζουμε την γραμμική συσχέτιση $r_{k,t}$. Πλέον, $k = argmax_{k \in K} \max_{0 \leq t \leq T} r_{k,t}$

Αυτή είναι η κεντρική φιλοσοφία του Deep Learning Power Analysis (DLPA) που αναπτύσσει η εφαρμογή που θα σας παρουσιάσουμε. Όμοια με πριν, για κάθε $k \in K$ υπολογίζουμε τα $(V_{i,k})_{1 \leq i \leq N}$ και $(H_{i,k})_{1 \leq i \leq N}$. Έπειτα, εκπαιδεύουμε το νευρωνικό μας δίκτυο έχοντας ως χαρακτηριστικά εκπαίδευσης τα $(T_i)_{1 \leq i \leq N}$ με αντίστοιχα label τα $y_i = (H_{i,k})_{1 \leq i \leq N}$. Για το σωστό $k = k^*$ τα $(T_i)_{1 \leq i \leq N}$ και y_i θα είναι ισχυρά συσχετισμένα με αποτέλεσμα το νευρωνικό να μπορεί να εκπαιδευτεί επιτυχώς (ως προς μετρικές όπως ευστοχία στο σύνολο εκπαίδευσης και αξιολόγησης και ελαχιστοποίηση συνάρτησης σφάλματος). Σε όλα τα υπόλοιπα κλειδιά αναμένουμε η εκπαίδευση να μην έχει το ίδιο αξιόλογο αποτέλεσμα. Έτσι, ο επιτιθέμενος επιλέγει το κλειδί k που εξασφαλίζει την βέλτιστη εκπαίδευση. Ως συνάρτηση σφάλματος προτιμήθηκε η MSE.

Algorithm 1 DLPA

Inputs: N traces $(T_i)_{1 \leqslant i \leqslant N}$ and corresponding plaintexts $(d_i)_{1 \leqslant i \leqslant N}$. Number of epochs n_e.

1: Set training data as $X = (T_i)_{1 \leqslant i \leqslant N}$
2: **for** $k \in \mathcal{K}$ **do**
3: Compute the series of hypothetical values $(H_{i,k})_{1 \leqslant i \leqslant N}$
4: Set training labels as $Y = (H_{i,k})_{1 \leqslant i \leqslant N}$
5: Perform DL training: $acc, loss = \texttt{DL}(X, Y, n_e)$
6: **end for**
7: **return** key k which leads to the best DL training metrics

Σχήμα 2.7: Αλγόριθμος DLPA [6]

Έστω ότι έχουμε ένα δείγμα 5000 ιχνών ($N = 5000$) ώστε:

- Το κάθε ίχνος περιέχει 50 χαρακτηριστικά (μπορείτε να φανταστείτε ότι το κάθε χαρακτηριστικό αντιστοιχεί στην διαρροή πληροφορίας σε χρόνο $1 \leq t \leq 50$)
- Θεωρούμε $T_{i,25} = f(m_i, k^*) + N(0, 1)$ $(1 \leq i \leq N)$
- $T_{i,t} = l$, l τυχαίος αριθμός στο $\{0, 1, ...255\}$, $t \in \{0, 1, ..., 50\} \setminus \{25\}$

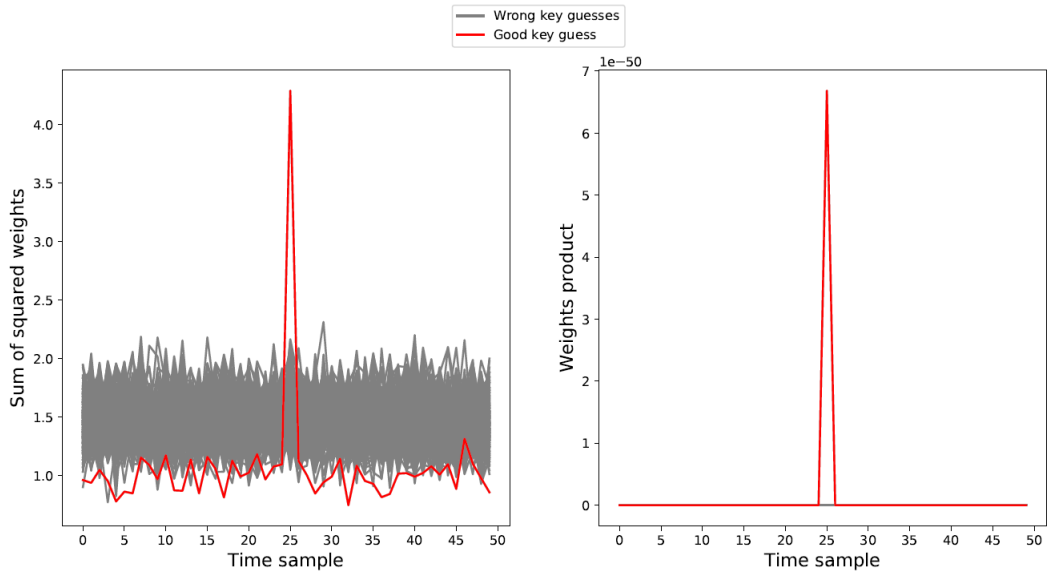
Η προσομοίωση αποσκοπεί στην μελέτη και αξιολόγηση ορισμένων πιθανών μετρικών που μπορεί να αξιοποιηθούν για την εύρεση του k^*.

Ας υποθέσουμε ότι έχουμε ένα MLP και εστιάζουμε την προσοχή μας στα βάρη του πρώτου στρώματος $W_{t,j}$ (αντιστοιχεί στην ακμή που συνδέει το t-οστό χαρακτηριστικό με το j-οστό νευρώνα του πρώτου hidden layer). Τότε, αν εφαρμόσουμε τον DLPA αλγόριθμο, το μόνο βάρος που θα υφίσταται ουσιαστική εκπαίδευση είναι το $W_{25,j}$ για $k = k^*$ ενώ τα υπόλοιπα, επειδή αντιστοιχούν στον ίδιο τυχαίο θόρυβο, περιμένουμε να μη παρουσιάζουν ουσιαστική τροποποίηση για διαφορετικά κλειδία k. Πράγματι, αν το πρώτο στρώμα έχει R νευρώνες και ορίσουμε:

$$S(t) = \sum_{j=1}^{R} (W_{t,j})^2$$

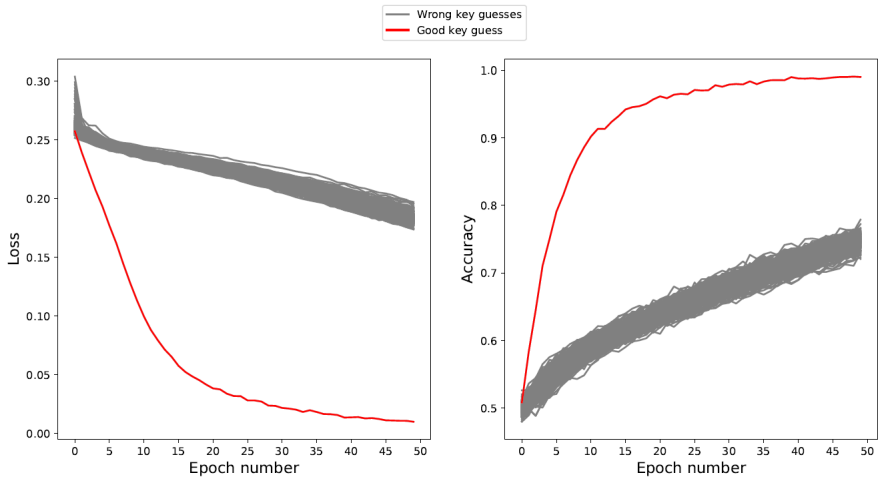
$$P(t) = \prod_{j=1}^{R} |W_{t,j}|$$

και εξετάσουμε την τιμή τους για κάθε μια από τις δυνατές τιμές του K μετά από $n_e = 50$ epochs, τότε το παρακάτω πείραμα που πραγματοποίησαν φανερώνει την ακόλουθη συμπεριφορά:



Σχήμα 2.8: Διαφοροποίηση $P(t)$, $S(t)$ για $k = k^*$ [6]

Ανάλογα μελετήθηκε η συμπεριφορά της ευστοχίας και της συνάρτησης κόστους πάνω στο σύνολο εκπαίδευσης:



Σχήμα 2.9: loss,accuracy συναρτήσει των epochs για κάθε $k \in K$ [6]

Παρατηρήστε ότι η δεύτερη προσέγγιση έχει νόημα για οποιαδήποτε δομή δικτύου, σε αντίθεση με την πρώτη που απαιτεί κάποια δομή/αρχιτεκτονική μεταξύ του στρώματος χαρακτηριστικών και του πρώτου hidden layer (είναι fully connected).

Το αμέσως επόμενο λογικό βήμα είναι η χρήση validation set (όπως αναπτύξαμε στην εισαγωγή). Πράγματι, σε πειράματα που διεξήχθησαν, κατάφεραν να δείξουν ότι το νευρωνικό κατάφερνε ανάλογη επίδοση ευστοχίας (accuracy) στο σύνολο εκπαίδευσης σε κλειδιά που

δεν αντιστοιχούσαν στο k^*. Έτσι, τροποποίησαν με τον πιο φυσιολογικό τρόπο τον αρχικό αλγόριθμο, ώστε να λαμβάνει υπ' όψιν του το καινούριο σύνολο.

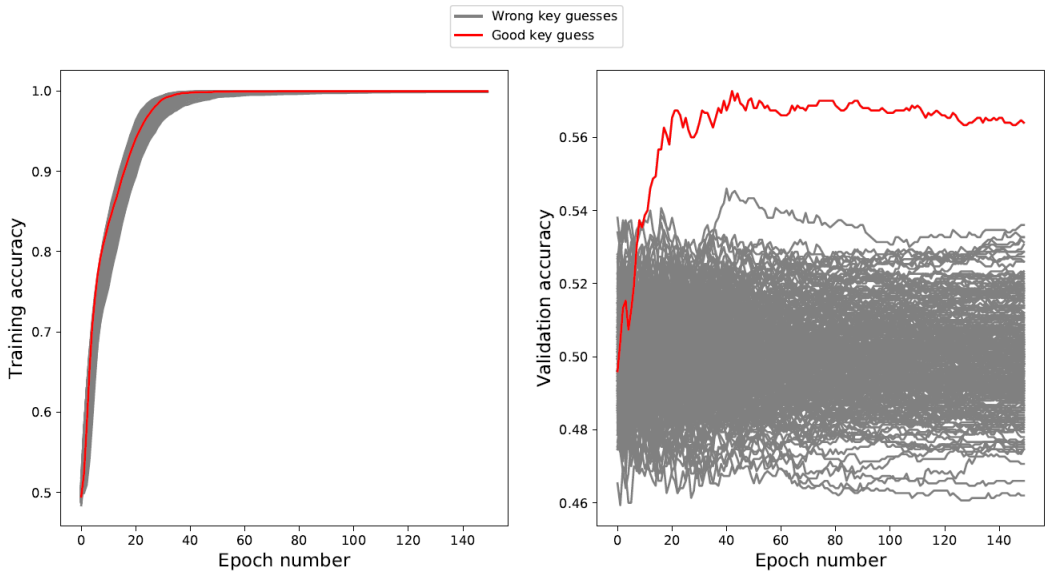
Algorithm 2 DLPA with validation data

Inputs: N traces $(T_i)_{1\leqslant i\leqslant N}$ and corresponding plaintexts $(d_i)_{1\leqslant i\leqslant N}$. Number of epochs n_e.

Choose N_T and N_V integers such that $N = N_T + N_V$ with $N_T > N_V$
Set training data as $X = (T_i)_{1\leqslant i\leqslant N_T}$
Set validation data as $X_{val} = (T_i)_{N_T+1\leqslant i\leqslant N}$
for $k \in \mathcal{K}$ **do**
    Compute the series of hypothetical values $(H_{i,k})_{1\leqslant i\leqslant N}$
    Set training labels as $Y = (H_{i,k})_{1\leqslant i\leqslant N_T}$
    Set validation labels as $Y_{val} = (H_{i,k})_{N_T+1\leqslant i\leqslant N}$
    Perform DL training: $acc, loss, acc_{val}, loss_{val} = \texttt{DL}(X, Y, X_{val}, Y_{val}, n_e)$
end for
return key k which leads to the best DL metrics

Σχήμα 2.10: Αλγόριθμος DLPA με validation set [6]

Το validation set διευκολύνει την ανάλυση σε ορισμένες περιπτώσεις όπου το σύνολο εκπαίδευσης δεν αρκεί, όπως φαίνεται στο ακόλουθο πείραμα. Σε αυτό, θεώρησαν $N = 8000$ ελαφρώς αποσυγχρονισμένα ίχνη τα οποία απαρτίζονται από 1000 χαρακτηριστικά. Λέμε ότι τα ίχνη μας είναι αποσυχρονισμένα όταν τα χαρακτηριστικά τους δεν παρουσιάζονται πάντα με την ίδια σειρά. Για παράδειγμα, στην προσομοίωση των 50 χαρακτηριστικών ανά ίχνος, μπορεί το 25^0 να εμφανιζόταν σε διαφορετικές θέσεις (π.χ. 24^0) σε ορισμένα ίχνη. Τέλος, $n_e = 150$, $N_V = 1500$, $N_T = 6500$.



Σχήμα 2.11: accuracy συναρτήσει των epochs για κάθε $k \in K$ σε training και validation set [6]

Η μορφή του label $\overrightarrow{y} = H_{i,k}$ έχει σημαντική επίπτωση στην ορθότητα του αλγορίθμου. Για παράδειγμα, αν θεωρήσουμε $H_{i,k} = f(m_i, k) = Sbox(m_i \oplus k)$ (ταυτοτική προσέγγιση), τότε παρατηρήστε ότι αν $(T_i)_{1 \leq i \leq N}$ ίχνος τέτοιο ώστε $f(m_i, k) = u \in \{0, ..., 255\}$ για κάποιο $k \in K$, τότε $f(m_i, k') = u' \in \{0, ..., 255\}$ $\forall k' \neq k$. Δηλαδή, η διαμέριση των ιχνών υπό την $H_{i,k}$ είναι αναλλοίωτη ως προς k, αφού η μόνη τροποποίηση που υπόκειται με την αλλαγή του κλειδιού είναι η τιμή του label και όχι τα στοιχεία που του αναλογούν. Πιο συγκεκριμένα, αν

$$E_u{}^{(k)} = \{T_i \in (T_i)_{1 \leq i \leq N} | Sbox(m_i \oplus k) = u\}$$

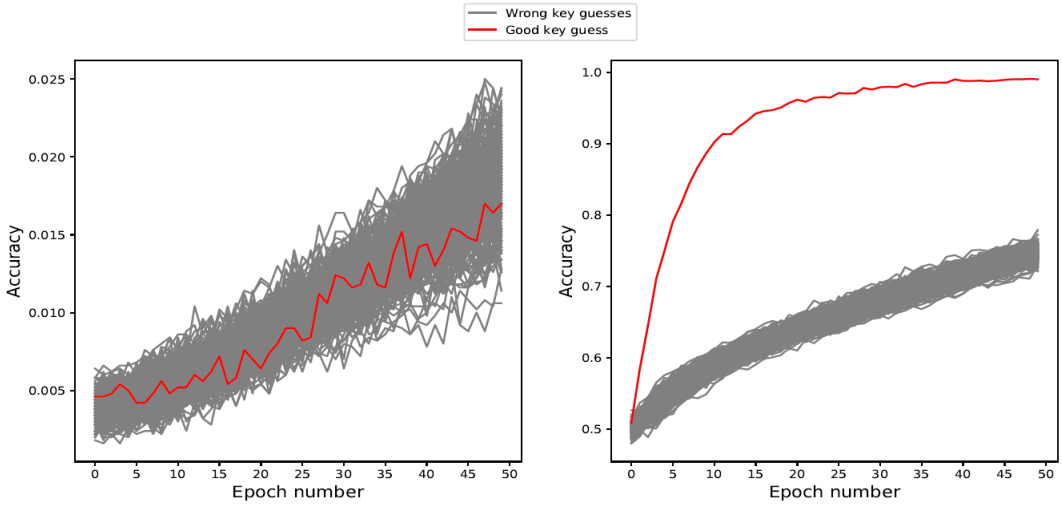
$$P_k = \{E_u{}^{(k)} | u \in \{0, ..., 255\}\}$$

Τότε $P_k \equiv P_{k'}$, και επομένως οι μετρικές μας θα έχουν παραπλήσια απόδοση. Οι $H_{i,k}$ που προτιμήθηκαν ήταν οι (Most Significant Bit/Least Significant Bit):

$$H_{i,k} = \begin{cases} 0 & \text{αν } V_{i,k} < 127 \\ 1 & \text{διαφορετικά} \end{cases}$$

$$H_{i,k} = V_{i,k}(mod 2)$$

που οδηγούν σε binary labelling και παρουσίαζαν καλύτερη επίδοση από την Hamming Weight labelling.



Σχήμα 2.12: MLP-DLPA χρησιμοποιώντας διαφορετικό labelling: ταυτοτική (αριστερά) και δυαδική (δεξιά) MSB [6]

Ένα σύνηθες ζήτημα σε τέτοιες επιθέσεις είναι τα αποσυγχρονισμένα ίχνη. Στο [46] δείχθηκε ότι η πετυχημένη επεξεργασία αποσυγχρονισμένων ιχνών μπορεί να πραγματοποιηθεί χρησιμοποιώντας Convolutional Neural Networks (CNN). Διαισθητικά, όπως και σε μια εφαρμογή εντοπισμού αντικειμένων σε μια εικόνα, ένα CNN, εκμεταλλευόμενο τις συναρτήσεις πυρήνα (Kernel), δύναται να εντοπίσει το ζητούμενο, ανεξαρτήτως της σχετικής του θέσης στην εικόνα (μια γάτα εξακολουθεί να είναι γάτα αν όλα τα pixel μετατοπιστούν μια θέση δεξιά), έτσι και στα ίχνη, πετυχαίνει την αναίρεση της επίδρασης της μετατόπισης των χαρακτηριστικών τους που αποφέρει ο αποσυγχρονισμός. Όπως θα δούμε παρακάτω, το CNN-DLPA παρουσιάζει αξιόλογα πειραματικά αποτελέσματα.

Ένα τελευταίο εμπόδιο που καλούμαστε να αντιμετωπίσουμε είναι το masking. Στην παρούσα εφαρμογή, μελετήθηκε το AES Boolean Masking, στο οποίο δεν απορρέει πληροφορία για το $Sbox(m \oplus k)$ αυτούσιο, αλλά καλύπτεται από ένα σύνολο τιμών $\{d_1, ..., d_s\}$ (masks). Πιο συγκεκριμένα, αντί του $Sbox(m \oplus k)$, έχουμε ίχνη για τα

$$(d_i)_{1 \leq i \leq s}$$

$$S = Sbox(m \oplus k) \oplus d_1 \oplus ... \oplus d_s$$

Ο παραδοσιακός αλγόριθμος CPA επεκτείνεται (High-Order CPA) για τέτοιες υλοποιήσεις, όπου αρχικά συνδιάζει κανείς την πληροφορία των παραπάνω ιχνών αξιοποιώντας κατάλληλες συναρτήσεις [47] και στη συνέχεια εφαρμόζει έναν κατάλληλο διαχωριστή (π.χ. μελετώντας πάλι την συσχέτιση Pearson) πάνω στην μετασχηματισμένη πληροφορία. Το πρόβλημα σε αυτή την προσέγγιση εμφανίζεται όταν δεν είναι γνωστό εκ των προτέρων ποιο ίχνος αναλογεί στα

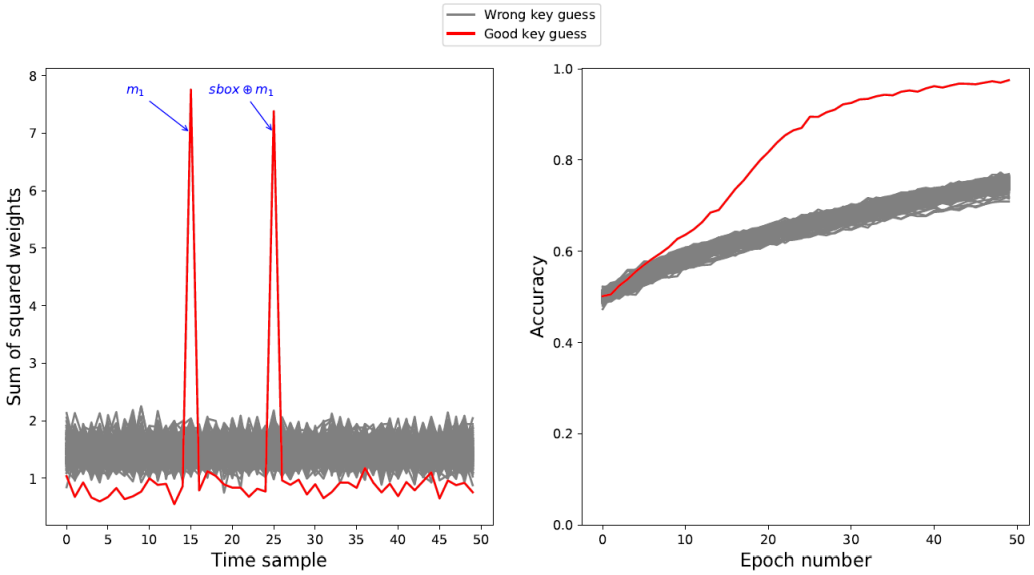
(d_i) και S. Μια ιδέα που χρησιμοποιείται σε τέτοιες περιπτώσεις είναι η εξέταση όλων των δυνατών συνδυασμών ιχνών. Το High Order DLPA προσφέρει μια εναλλακτική οπτική σε αυτή την επίθεση, αποφεύγοντας την παραπάνω μη αποδοτική λύση.

Τα βασικά πειράματα της εφαρμογής βασίστηκαν σε τρία είδη ιχνών:

- Τεχνητά που παρήχθησαν από τους ίδιους
- Ίχνη που συλλέχθηκαν από ένα ChipWhisperer-Lite board [49]
- Ίχνη από την δημόσια βάση δεδομένων ASCAD [48]

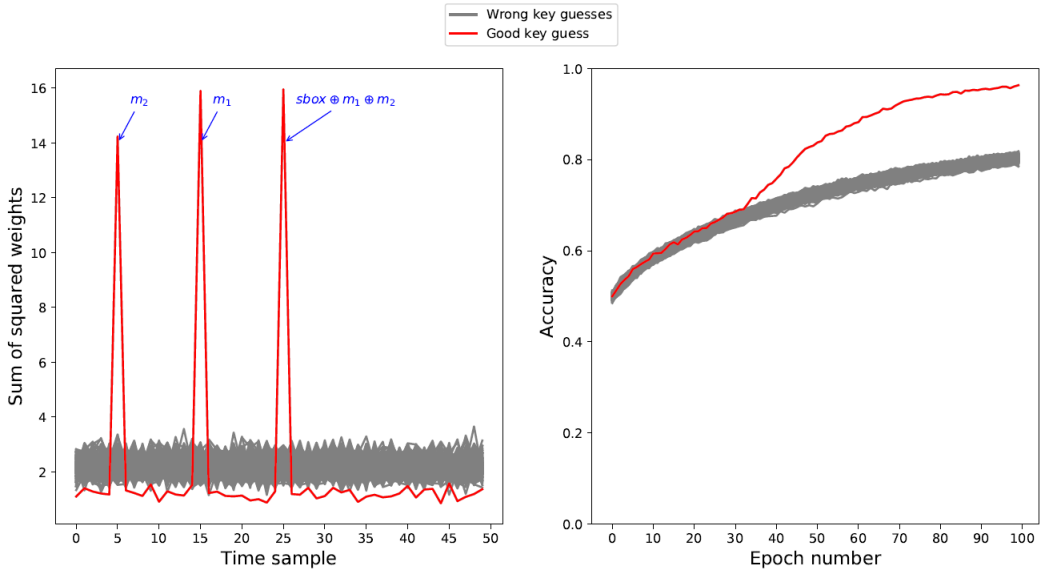
Αρχικά μελετήθηκε η συμπεριφορά του DLPA για 1 και 2 μάσκες. Πιο συγκεκριμένα παρήχθησαν $N = 5000$ ίχνη:

- 50 χαρακτηριστικά άνα ίχνος
- $T_{i,25} = Sbox(m_i \oplus k^*) \oplus d_1 + N(0, 1)$
- $T_{i,15} = d_i + N(0, 1)$
- $T_{i,t} = l$, l τυχαίος αριθμός στο $\{0, ..., 255\} \setminus \{15, 25\}$.



Σχήμα 2.13: Εφαρμογή MLP-DLPA για Sbox με μια μάσκα [6]

Εντελώς παρόμοια:



Σχήμα 2.14: Εφαρμογή MLP-DLPA για Sbox με δυο μάσκες [6]

Παρατηρούμε ότι και στις δύο προσομοιώσεις επιτεύχθηκε ο εντοπισμός του ορθού κλειδιού k^*.

Η τελευταία προσομοίωση που πραγματοποίησαν διερευνά την αποτελεσματικότητα του CNN-DLPA όταν εφαρμόζεται σε αποσυγχρονισμένα ίχνη (χωρίς μάσκες). Πιο συγκεκριμένα, παρήχθησαν ίχνη όπως στην αρχή αυτή της ενότητας τα όποια μετατοπίστηκαν τυχαία κάτα $p \in \{-6, -5, ..., 5, 6\}$ θέσεις, ενώ ο θόρυβος πλέον ακολουθούσε $N(0, 20)$.

Μελετήθηκαν και συγκρίθηκαν τρεις επιθέσεις:

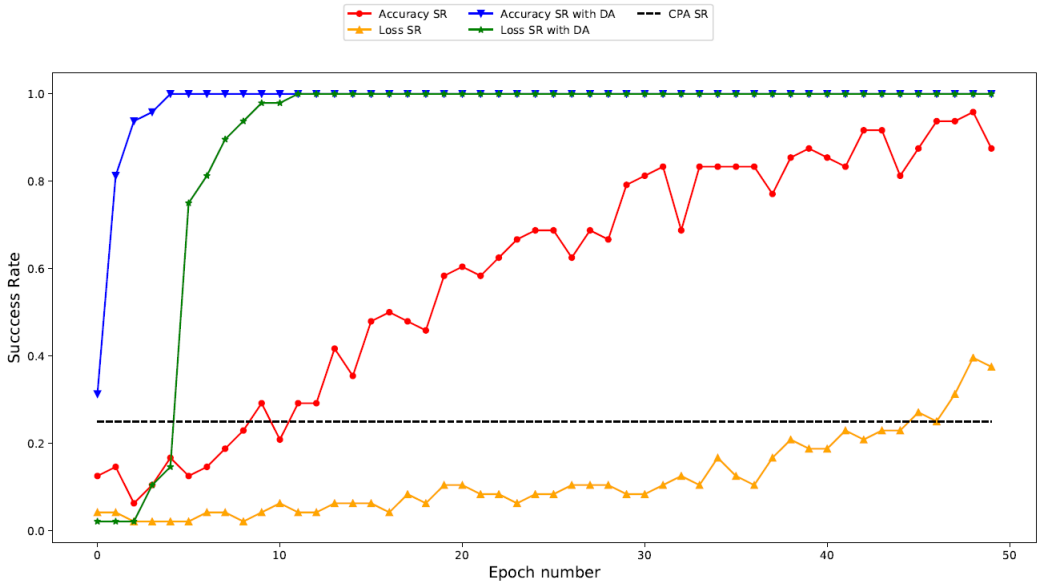
- Παραδοσιακό CPA
- CNN-DLPA χωρίς προσαύξηση δεδομένων (data augmentation)
- CNN-DLPA με προσαύξηση δεδομένων

Το data augmentation είναι ένα ισχυρό εργαλείο στην επιστήμη δεδομένων και ιδιαίτερα στο Deep Learning. Πρόκειται για μια τεχνική η οποία εκμεταλλεύεται ορισμένες πληροφορίες και γνώσεις που έχουμε για την δομή των δεδομένων μας για την δημιουργία νέων έγκυρων στιγμιοτύπων (συνήθως αξιοποιούμε συμμετρίες της τοπολογίας του χώρου χαρακτηριστικών). Γυρίζοντας πίσω στο παράδειγμα εντοπισμού αντικειμένων σε εικόνες, μπορούμε να παράξουμε νέα έγκυρα στιγμιότυπα (εικόνες με το ζητούμενο αντικείμενο) παίρνοντας ένα υπάρχον στο training set και μετατοπίζοντας ή περιστρέφοντας το αντικείμενο στον χώρο.

Στο πείραμα που ακολουθεί πραγματοποίησαν Data Augmentation βασιζόμενοι στο Shifting Deformation [46]. Η κεντρική ιδέα είναι ιδιαίτερα κομψή και βασίζεται στην χρήση ενός Shifting

Window μεγέθους D'. Πιο συγκεκριμένα, μειώνουμε τον χώρο των χαρακτηριστικών (την χρονική διάρκεια του κάθε ίχνους) κάτα T^*, μετασχηματίζοντας το CNN μας ώστε να έχει Input Layer μεγέθους D'. Έστω ότι τα $(T_i)_{1\leq i\leq N}$ είναι μεγέθους $D > D'$. Τότε, επιλέγουμε ομοιόμορφα έναν αριθμό $t^* \in [0, T^*]$ και απομονώνουμε το παράθυρο που ξεκινά στην θέση t^*. Έτσι, για το ίχνος T_1 είναι εφικτό να δημιουργήσουμε T^* διαφορετικά ίχνη $\{T_1[0, T^*], T_1[1, T^* + 1], ..., T_1[T^*, D]\}$, όπου $T_1[i, j]$ συμβολίζουμε το ίχνος που περιέχεται στο T_1 και ξεκινά από το χαρακτηριστικό i και τελειώνει στο j.

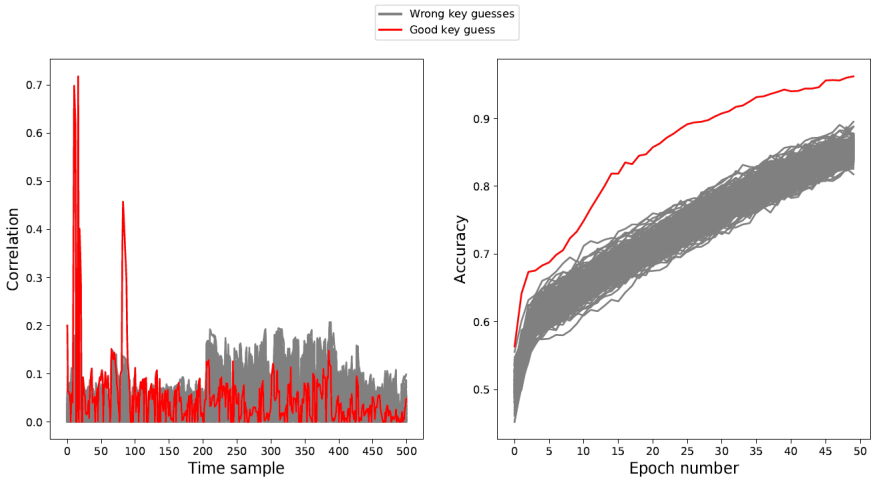
Το πείραμα που διεξήχθη απαρτιζόταν από 2000 αποσυγχρονισμένα ίχνη, τα οποία μετασχηματίστηκαν σε 20000 για το τρίτο πείραμα που αξιοποίησε το Shifting Deformation. Επαναλήφθηκε 50 φορές, παίρνοντας καινούρια ίχνη σε κάθε επανάληψη. Σχηματικά:



Σχήμα 2.15: Αποτελέσματα προσομοίωσης αποσυγχρωνισμένων ιχνών για τα CNN-DLPA με και χωρίς προσαύξηξη δεδομένων και σύγκριση με παραδοσιακό CPA[6]

Στο παραπάνω γράφημα παρουσιάζεται η καλύτερη επίδοση από τις 50 επαναλήψεις των τριών μεθόδων. Μάλιστα, μελέτησαν τα CNN-DLPA ως προς δυο μετρικές: την μεγιστοποίηση της ευστοχίας (μπλέ και κόκκινη) και την ελαχιστοποίηση της συνάρτησης κόστους (κόκκινη και πορτοκαλί). Παρατηρούμε ότι το CNN-DLPA βασιζόμενο στην ευστοχία εμφανίζει πολύ καλύτερη επίδοση από το παραδοσιακό CPA για αποσυγχρονισμένα ίχνη.

Στη συνέχεια μελετήθηκε η επίδοση του DLPA σε ίχνη που προσομοιάζονται από το ChipWhisperer-Lite. Εξέτασαν μια απλή υλοποίηση χωρίς μάσκες, έπειτα μια με αποσυγχρονισμένα ίχνη και τέλος έλεγξαν την αποδοτικότητα απέναντι σε 1 και 2 μάσκες. Σε κάθε περίπτωση αντιπαρέβαλλαν το αντίστοιχο CPA.

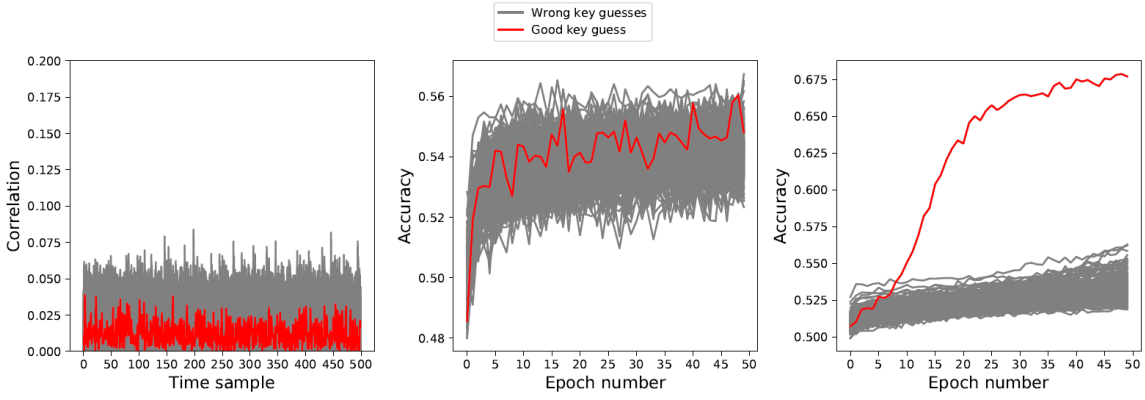


Σχήμα 2.16: Μη προστατευμένη υλοποίηση με συγχρονισμένα ίχνη. Αριστερα: CPA, Δεξιά: DLPA[6]

Όπως αναμέναμε, ούσα η απλούστερη περίπτωση, και οι δυο μέθοδοι βρίσκουν επιτυχώς το σωστό κλειδί k^*. Να σημειωθεί ότι αξιοποιήθηκαν $N = 3000$ ίχνη το οποίο αποτελεί ένα πρακτικά εφικτό νούμερο (σε πραγματική επίθεση).

Στην συνέχεια εξέτασαν μια απροστάτευτη υλοποίηση με $N = 3000$ αποσυχγρονισμένα ίχνη ως προς τρεις μεθόδους:

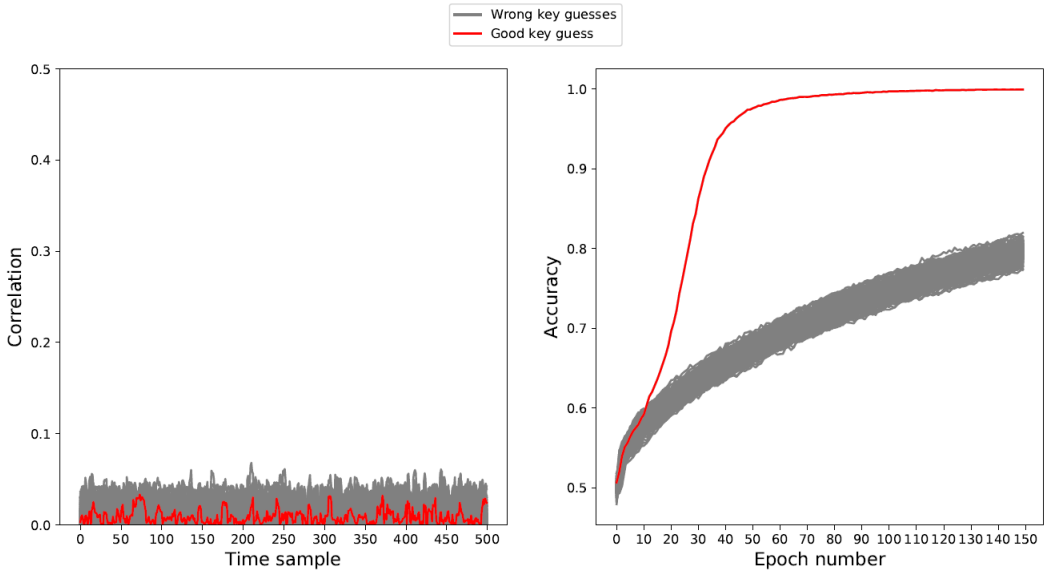
- CPA
- MLP-DLPA χωρίς προσαύξηση δεδομένων
- MLP-DLPA με προαύξηση δεδομένων ($N' = 3000 * 20 = 60000$)



Σχήμα 2.17: Μη προστατευμένη υλοποίηση με αποσυγχρονισμένα ίχνη. Αριστερα: CPA, Κέντρο: MLP-DLPA χωρίς προσαύξηση δεδομένων, Δεξιά: MLP-DLPA με προσαύξηση δεδομένων[6]

Παρατηρούμε ότι το CPA και το MLP-DLPA χωρίς προσαύξηση δεδομένων αδυνατούν να εντοπίσουν το σωστό κλειδί. Αντιθέτως, το MLP-DLPA με προσαύξηση δεδομένων πετυχαίνει τον ζητούμενο στόχο. Μάλιστα, το συγκεκριμένο αποτέλεσμα δείχνει την χρησιμότητα του Shifting Deformation ακόμα και σε διαφορετική αρχιτεκτονική (MLP έναντι CNN).

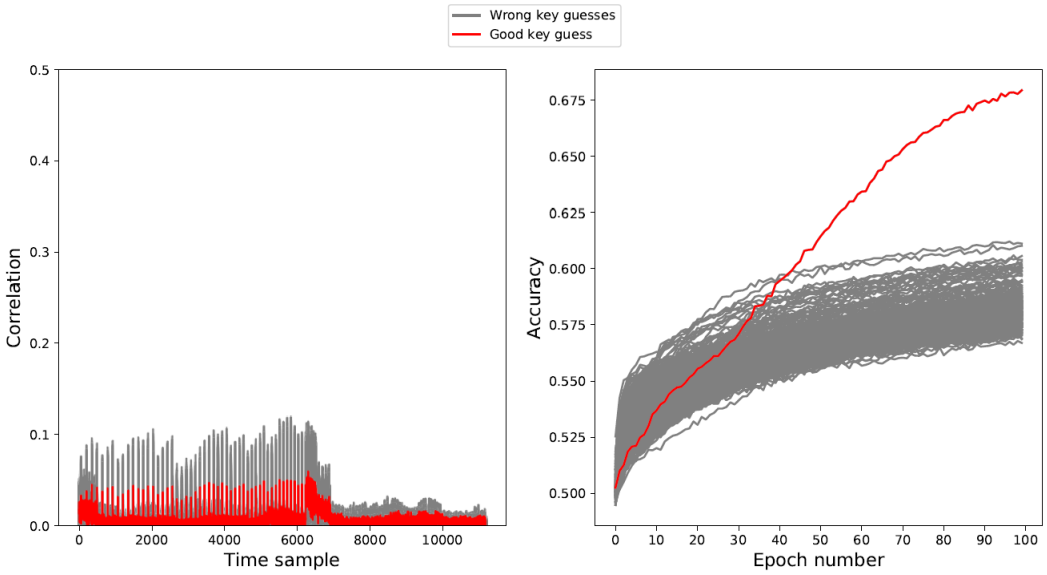
Έπειτα μελέτησαν την αποτελεσματικότητα του αλγορίθμου έναντι συγχρονισμένων ιχνών με προστατευμένες υλοποιήσεις 1 και 2 μασκών.



Σχήμα 2.18: Προστατευμένη υλοποίηση μιας μάσκας με συγχρονισμένα ίχνη. Αριστερα: CPA, Δεξιά: MLP-DLPA [6]

Όπως περιμέναμε, το παραδοσιακό CPA αδυνατεί να βρει το κλειδί, ενώ η ομοιομορφία της γραφικής φανερώνει ότι δεν υπάρχει οποιοδήποτε είδος διαρροής πληροφορίας.

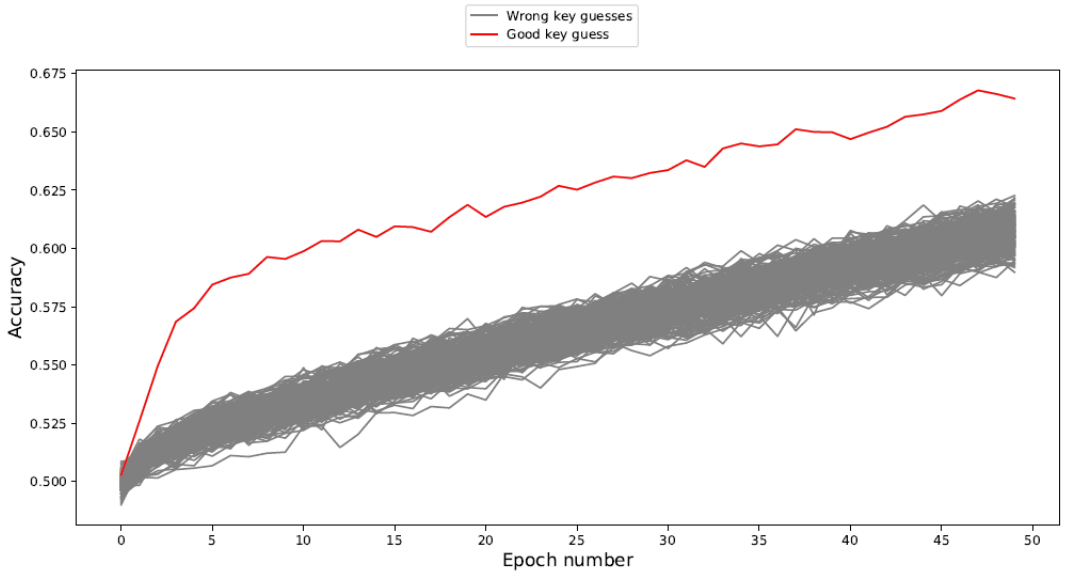
Στη συνέχεια μελετήθηκε η υλοποίηση με δυο μάσκες, όπου η CPA ανάλυση έγινε συνδυάζοντας κατάλληλα κάθε δυάδα χαρακτηριστικών από κάθε ίχνος, παίρνοντας ως συνάρτηση συνδυασμού την $f(x, y) = |x - y|$. Το σύνολο εκπαίδευσης απαρτιζόταν από 50000 ίχνη που πλέον αποτελούνταν από 150 χαρακτηριστικά, όποτε το CPA επεξεργάστηκε $\frac{150*149}{2}$ χαρακτηριστικά ανά ίχνος, σε αντίθεση με τα 150 του MLP-DLPA. Όπως φαίνεται και στο παρακάτω σχήμα, το CPA δεύτερης τάξης αποτυγχάνει, ενώ το MLP-DLPA βρίσκει το κλειδί μετά από 50 epochs.



Σχήμα 2.19: Προστατευμένη υλοποίηση με δύο μάσκες για συγχρονισμένα ίχνη. Αριστερα: Second Order CPA, Δεξιά: MLP-DLPA [6]

Η δεύτερη σειρά πειραμάτων (που αξιοποίησαν το ChipWhisperer-Lite) φανερώνει ότι η MLP-DLPA εμφανίζει αξιόλογα αποτελέσματα και σε ίχνη που παρήχθησαν με φυσικό τρόπο. Μένει η μελέτη της απόδοσης του αλγορίθμου σε μια πραγματική δημόσια βάση δεδομένων.

Η ASCAD είναι μια δημόσια βιβλιοθήκη ιχνών η οποία δημιουργήθηκε με σκοπό την έρευνα Side Channel επιθέσεων βασισμένες σε Deep learning. Στο πείραμα που διεξήχθη αξιοποιήθηκε η βάση ASCAD.h5 η οποία απαρτίζεται από 60000 ίχνη που αναλογούν στην διαρροή του τρίτου byte $Sbox(m[3] \oplus k[3]) \oplus d[3]$. Από αυτή την βάση απομόνωσαν όλα τα δείγματα που αναλογούν στο ίδιο κλειδί (20000 ίχνη) και μελέτησαν την επίδοση του MLP-DLPA για 50 epochs. Το προσδιοριστέο κλειδί εντοπίζεται επιτυχώς από το MLP-DLPA μετά από μόλις 5 epochs.



Σχήμα 2.20: Επίθεση MLP-DLPA πάνω στο ASCAD [6]

Συνοψίζοντας, η σειρά πειραμάτων που πραγματοποίησαν ενθαρρύνει την χρήση Νευρωνικών Δικτύων σε Non-Profiled Side Channel επιθέσεις. Μελετήθηκε η απόδοση του κεντρικού DLPA αλγορίθμου τους, εξετάζοντας και την χρήση CNN-DLPA για αποσυγχρονισμένα ίχνη. Μάλιστα, το DLPA φαίνεται να υπερτερεί της παραδοσιακής CPA προσέγγισης ως προς την δυνατότητα εύρεσης του κλειδιού. Επομένως, αποτελεί μια ιδανική εναλλακτική για None Profiled επιθέσεις και θα μπορούσε να συνδυαστεί με την μέθοδο που αναπτύξαμε στην προηγούμενη υποενότητα.

Η εφαρμογή αυτού του αλγορίθμου, όμως, θέλει προσοχή καθώς απαιτεί επανεκπαίδευση ενός νευρωνικού δικτύου για κάθε πιθανό κλειδί. Έτσι, αυτή η μεθοδολογία γίνεται απαγορευτική καθώς ο χώρος K αυξάνει. Ήδη, μελετώντας ένα byte κλειδιού ($|K| = 256$) το DLPA υστερεί (χρονικά) μιας CPA πρώτης τάξης [6]. Παρόλα αυτά, υπό κατάλληλες προϋποθέσεις (πολλά χαρακτηριστικά ανά κλειδί και λίγα epochs εκπαίδευσης) το DLPA μπορεί να επιτύχει ρεαλιστικούς και πρακτικούς χρόνους οι οποίοι, μάλιστα, είναι καλύτεροι από τους αντίστοιχους μιας δεύτερης τάξης CPA. Επιπλέον, ο αλγόριθμος είναι παραλληλοποιήσιμος (διαφορετικές εκπαιδεύσεις μπορούν να εκτελεστούν συγχρόνως).

Implementation	Attack	Nb traces	Nb samples	Nb epochs	Time (hours)
CW 1-mask	MLP-DLPA	5,000	500	150	0.92
CW 2-masks	MLP-DLPA	50,000	150	100	5.14
ASCAD	MLP-DLPA	20,000	700	50	1.19
ASCAD	MLP-DLPA	20,000	700	5	0.17
ASCAD	2nd order CPA	1,000	700	N/A	2.16

Σχήμα 2.21: Χρονική επίδοση DLPA. Υπολογιστική ισχύ 32 GB, GeForce GTX 1080 GPU, Intel Xeon E5-2687W CPU [6]

2.4 Ανακεφαλαίωση

Σε αυτή την ενότητα μελετήσαμε ορισμένες εφαρμογές της Μηχανικής Μάθησης για την επίθεση κρυπτοσυστημάτων. Είδαμε πως μπορούμε να σπάσουμε σε ρεαλιστικό χρόνο απλούστερα κρυπτοσυστήματα (DES, 3-DES) αξιοποιώντας δραματικά λιγότερα ζεύγη (m, c) από παραδοσιακές προσεγγίζεις. Στη συνέχεια παρουσιάσαμε Side Channel επιθέσεις στο πολύ πιο σύνθετο AES (απλή και προστατευμένη masked υλοποίηση). Τόσο στις Profiling όσο και στις ασθενέστερες Non Profiled επιθέσεις η Μηχανική Μάθηση έδωσε θετικά αποτελέσματα με επιδόσεις που συχνά υπερέβαιναν παλαιότερες προσεγγίσεις (CPA).

Ευελπιστούμε ότι το παρόν κεφάλαιο προσφέρει μια καλή συνοπτική εποπτεία στον αναγνώστη για τις δυνατότητες της Μηχανικής Μάθησης στην διεξαγωγή κρυπτοεπιθέσεων. Προσπαθήσαμε να παρουσιάσουμε ενδεικτικές εφαρμογές με τη σειρά που θεωρήσαμε πιο λογική. Έτσι, μεταβήκαμε φυσιολογικά από την επίθεση ενός πιο απλού κρυπτοσυστήματος στην αντιμετώπιση της πιο γενικής Side Channel επίθεσης (Non Profiled Attack) στο πολύ πιο σύνθετο masked AES.

Στο επόμενο κεφάλαιο θα εστιάσουμε στις εφαρμογές της Μηχανικής Μάθησης για την προστασία των κρυπτοσυστημάτων. Πιο συγκεκριμένα, θα σας παρουσιάσουμε έναν τρόπο ανίχνευσης Side Channel επιθέσεων [51] οι οποίες ενδεχομένως να προστατεύσουν το κρυπτοσύστημα από τις επιθέσεις που αναφέραμε στις δυο τελευταίες υποενότητες. Έπειτα, θα αναρωτηθούμε την δυνατότητα χρήσης ενός νευρωνικού σε ένα κρυπτοσύστημα (ως τμήμα ή ολόκληρο), εστιάζοντας στην ανθεκτικότητα του έναντι επιθέσεων που εκμεταλλεύονται την διαφορισιμότητα των συναρτήσεων που το απαρτίζουν [50].

Κεφάλαιο 3

Ενίσχυση Κρυπτοσυστημάτων με Μηχανική Μάθηση

Η προστασία ενός κρυπτοσυστήματος υπάγεται στην απόκρυψη οποιασδήποτε πληροφορίας που αναφέρεται στο κλειδί κρυπτογράφησης k ή το αρχικό μήνυμα m. Όπως είδαμε και στα προηγούμενα κεφάλαια, ο αντίπαλος μπορεί να στοχεύσει στην εξόρυξη τέτοιας πληροφορίας είτε μελετώντας την κατανομή του ίδιου του κρυπτοσυστήματος είτε εκμεταλλεύοντας την αποκάλυψη και εκροή φυσικών ποσοτήτων (ακτινοβολία, θερμοκρασία, χρόνος εκτέλεσης) οι οποίες σχετίζονται με το κλειδί ή μήνυμα και μας προσδίδουν κάποια επιπλέον γνώση για αυτά. Επομένως, τελείως φυσικά, η προστασία ενός κρυπτοσύστηματος μπορεί να επιτευχθεί ενισχύοντας την κατανομή του (κάνοντας την πιο σύνθετη), περιορίζοντας τις εν λόγω διαρροές ή εντοπίζοντας την συσκευή ή πρόγραμμα που αξιοποιεί ο αντίπαλος για να τις ανιχνεύσει.

Η χρήση ενός μοντέλου Μηχανικής Μάθησης (π.χ. Νευρωνικό Δίκτυο) σε τομείς ασφάλειας χρήζει ιδιαίτερης προσοχής, καθώς διαθέτει ιδιότητες οι οποίες μπορεί να εκμεταλλευτούν από έναν αντίπαλο. Χαρακτηριστικό παράδειγμα αποτελεί το γεγονός ότι στα Νευρωνικά Δίκτυα, τα οποία εξετάζονται τελευταία για να αποτελέσουν τμήμα ή αυτούσια ένα κρυπτοσύστημα ([57], [3], [4]), ο επιτιθέμενος μπορεί να εκμεταλλευτεί την διαφορισιμότητά τους για να τα οδηγήσει σε δυσλειτουργία ([59]).

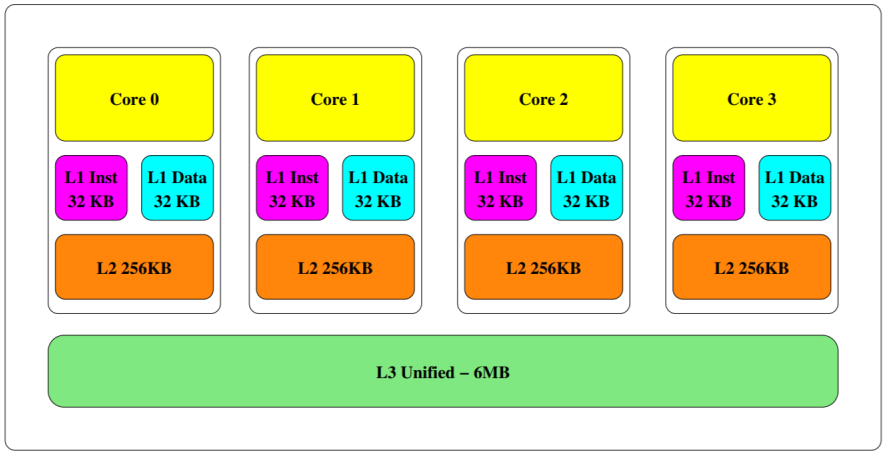
Επομένως, στην πρώτη ενότητα θα σας παρουσιάσουμε μια εφαρμογή που ενισχύει την χρήση μοντέλων μηχανικής μάθησης για τον εντοπισμό Side Channel επιθέσεων, ενώ στην δεύτερη θα μελετήσουμε και θα αναλύσουμε έναν τρόπο ενίσχυσης νευρωνικών ενάντια σε επιθέσεις που εκμεταλλεύονται την διαφορισιμότητά τους.

3.1 Ανίχνευση Side Channel επιθέσεων με Μηχανική Μάθηση

Στην παρούσα ενότητα θα μελετήσουμε Side Channel επιθέσεις που βασίζονται στην επεξεργασία και μελέτη της μνήμης Cache (CSCA). Πιο συγκεκριμένα, οι [51] εξέτασαν την αποδοτικότητα διαφόρων Machine Learning αλγορίθμων για την ανίχνευση CSCA πάνω στην αρχιτεκτονική

Intel x86 ενάντια σε Flush+Reload (F+R) [53] και Flush + Flush (F+F) [54] τεχνικές. Τα κρυπτοσυστήματα που μελετήθηκαν ήταν οι συναρτήσεις κρυπτογράφησης RSA[52] και AES, ενώ όλες οι μέθοδοι ανίχνευσης αξιολογήθηκαν και συγκρίθηκαν με βάση την ευστοχία τους, την επιβάρυνση που επιφέρουν στο σύστημα κατά την λειτουργία τους και την ταχύτητα εντοπισμού της επίθεσης. Ιδανικά, αναζητάμε ένα μοντέλο που θα πετυχαίνει γρήγορο (πριν ολοκληρωθεί η επίθεση) και εύστοχο εντοπισμό, με όσο το δυνατόν μικρότερη επιβάρυνση στην μνήμη.

Η βασική ιδέα πίσω από τις (F+R), (F+F) είναι αρκετά απλή. Ο υπολογιστής, κατά την εκτέλεση οποιασδήποτε διεργασίας (π.χ. προγράμματος) μεταφέρει δεδομένα από την κεντρική μνήμη του (σκληρός δίσκος) στην μνήμη Cache όπου μπορεί να πραγματοποιήσει οποιαδήποτε πράξη (read/write) πιο γρήγορα. Η Cache έχει ιδιαίτερα μικρή χωρητικότητα (από μερικά KB έως λίγα MB[53]) και επομένως για πιο σύνθετες μεθόδους χρειάζεται η διαγραφή και επαναφόρτιση (Reload) διαφόρων τμημάτων κώδικα ή δεδομένων. Τα επεξεργασμένα δεδομένα, πριν διαγραφούν, γράφονται στην κύρια μνήμη μέσω της εντολής flush. Ο χρόνος που χρειάζεται η flush ή reload εντολή μιας υποψήφιας θέσης μνήμης ποικίλει ανάλογα με την διαθεσιμότητα της στην μνήμη Cache. Σε αυτήν ακριβώς την λειτουργία βασίζονται οι επιθέσεις.



Σχήμα 3.1: Ενδεικτικό σχεδιάγραμμα αρχιτεκτονικής Intel i5-3470. Παρατηρήστε την ιεραρχία μνήμης Cache (L1,L2,L3) και την χωρητικότητα τους. Στο F+R έχουμε πρόσβαση μόνο στο L3 επίπεδο (το οποίο περιέχει μεταξύ άλλων αντίγραφο της πληροφορίας των L1,L2 cache) [53]

Ας υποθέσουμε ότι ο αντίπαλος υποψιάζεται ότι η λειτουργία του προγράμματος θα αξιοποιήσει μια συγκεκριμένη θέση μνήμης. Η (F+R) επίθεση έχει 3 στάδια:

1. Ο αντίπαλος κάνει flush την υποψήφια θέση μνήμης από όλη την ιεραρχία cache
2. Ο αντίπαλος περιμένει, αναμένοντας την ενέργεια του θύματος-στόχου
3. Ο αντίπαλος ξαναφορτώνει (reload) την υποψήφια θέση μνήμης και ελέγχει τον χρόνο που χρειάζεται αυτή η ενέργεια. Αν πραγματοποιείται γρήγορα, τότε η συσκευή στόχος

αξιοποίησε την συγκεκριμένη θέση μνήμης.

Συγκρίνοντας έτσι τους χρόνους επαναφόρτισης, μπορεί κανείς να εξάγει συμπεράσματα για το ποια δεδομένα καλέστηκαν από την μνήμη. Για παράδειγμα, για την εξόρυξη του κλειδιού ενός κρυπτοσυστήματος μπορούμε να εστιάσουμε σε διεργασίες που σχετίζονται με τα bit του. Έστω, ότι επιχειρούμε να σπάσουμε το RSA. Θυμηθείτε ότι σε αυτό:

$$p_k = (n = p * q, e), s_k = (d = e^{-1} mod \varphi(n), p, q)$$

$$Enc(m, e) = c = m^e mod n$$

$$Dec(c, d) = m = c^d mod n$$

Υποθέστε ότι η αποκρυπτογράφηση πραγματοποιείται με τον γνωστό αλγόριθμο επαναλαμβανόμενου τετραγωνισμού, ο οποίος συνδέεται άμεσα με την δυαδική αναπαράσταση του d. Επομένως, εξετάζοντας σε κάθε στάδιο αυτού του αλγορίθμου τη συμπεριφορά της cache ο αντίπαλος θα μπορέσει να υπολογίσει τα bits του εκθέτη (ιδιωτικού κλειδιού) και να σπάσει το κρυπτοσύστημα. Κάτι παρεμφερές μελετήθηκε και στο [53] στην οποία πρωτοπαρουσιάστηκε η επίθεση.

Η ιδέα του F+F είναι παρόμοια:

- Ο αντίπαλος κάνει flush την υποψήφια θέση μνήμης
- Ο αντίπαλος περιμένει για την ενέργεια του θύματος-στόχου
- Ο αντίπαλος ξανακάνει flush την υποψήφια θέση μνήμης. Αν η διαγραφή πάρει πολύ χρόνο, τότε αξιοποιήθηκε από τον χρήστη (flush θέσης μνήμης που βρίσκεται στην cache παίρνει περισσότερο χρόνο από όταν δεν βρίσκεται)

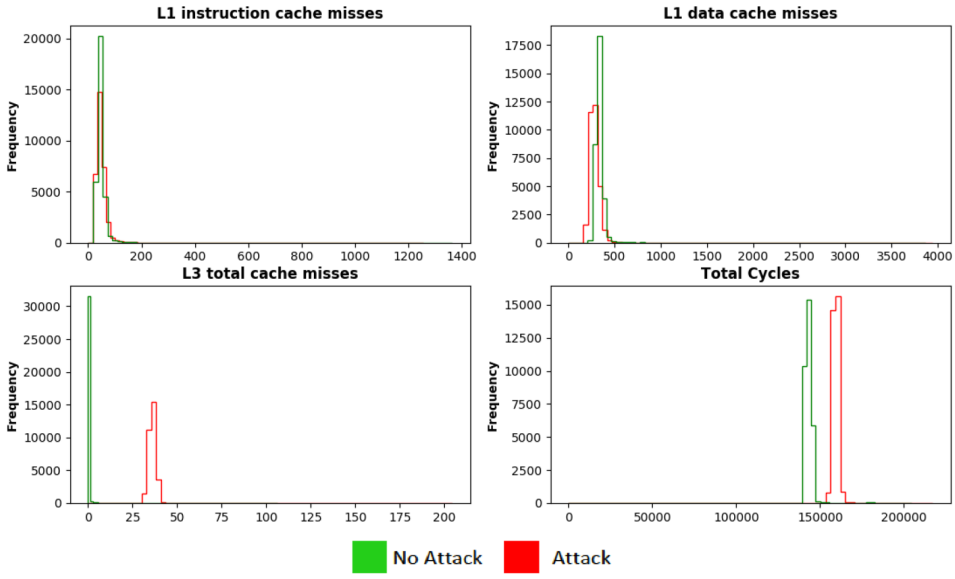
Δε θα εστιάσουμε περαιτέρω στου λόγους προτίμησης της μιας επίθεσης έναντι της άλλης, καθώς είναι κατά βάση τεχνικοί (σχετίζονται με την αποφυγή χρήσης του reload).

Σκοπός αυτής της εφαρμογής, όπως προαναφέραμε, είναι ο αποδοτικός εντοπισμός F+F, F+R επιθέσεων. Για τον σκοπό αυτό, μελετήθηκαν, ως προς τις μετρικές που αναπτύξαμε προηγουμένως, διάφοροι διαχωριστές Μηχανικής Μάθησης. Ως βάση δεδομένων αξιοποιήθηκε η βιβλιοθήκη PAPI η οποία προσφέρει διάφορες πληροφορίες για γεγονότα (events) που αφορούν επεξεργαστές Intel i7. Μάλιστα, εστίασαν σε χαρακτηριστικά που επηρεάζονται περισσότερο από τέτοιες επιθέσεις, όπως τα cache misses που αυξάνονται λόγω των flush-reload εντολών. Πιο συγκεκριμένα, λέμε ότι έχουμε cache miss όταν το πρόγραμμα προσπαθεί να αξιοποιήσει μια θέση μνήμης η οποία δεν είναι στην cache. Σε μια τέτοια περίπτωση αναγκάζεται να περιμένει μέχρι να επαναφορτιστεί σε αυτήν (από μια άλλη cache ή από τον σκληρό δίσκο). Αναλυτικότερα, τα χαρακτηριστικά που μελετήθηκαν:

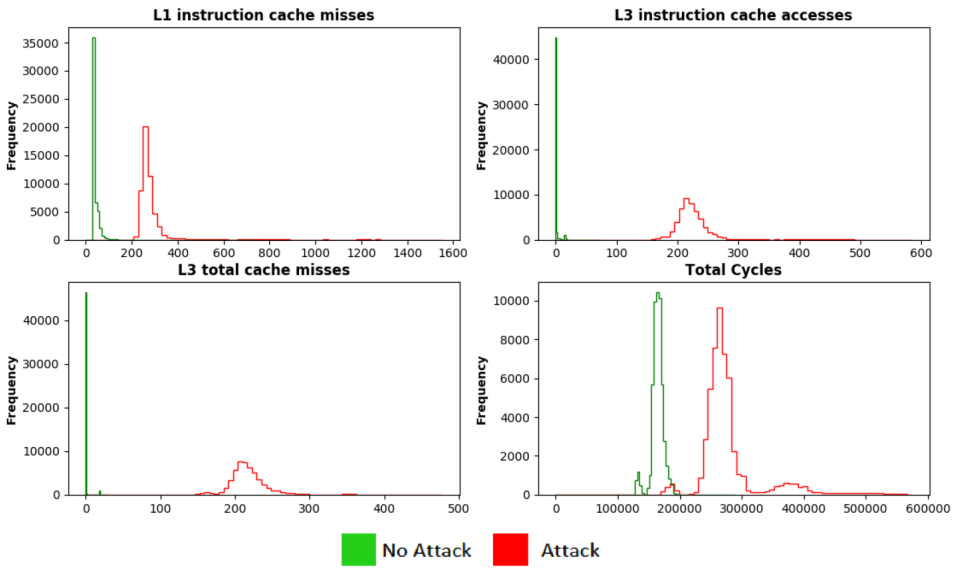
Scope of Event	Hardware Event as Feature	Feature ID
L1 Caches	Data Cache Misses	L1-DCM
	Instruction Cache Misses	L1-ICM
	Total Cache Misses	L1-TCM
L2 Caches	Instruction Cache Accesses	L2-ICA
	Instruction Cache Misses	L2-ICM
	Total Cache Accesses	L2-TCA
	Total Cache Misses	L2-TCM
L3-Caches	Instruction Cache Accesses	L3-ICA
	Total Cache Accesses	L3-TCA
	Total Cache Misses	L3-TCM
System-wide	Total CPU Cycles	TOT_CYC
	Branch Miss-Predictions	BR_MSP

Σχήμα 3.2: Πίνακας χαρακτηριστικών που μελετήθηκαν για την ανίχνευση των επιθέσεων. [51]

Η κατανομή ορισμένων τέτοιων χαρακτηριστικών σε εξιδανικευμένες συνθήκες (zero load condition, δηλαδή στον επεξεργαστή πραγματοποιείται μόνο η κρυπτογράφηση) παρουσιάζονται παρακάτω:



Σχήμα 3.3: Κατανομή χαρακτηριστικών για AES υπό την παρουσία και απουσία F+F[51]



Σχήμα 3.4: Κατανομή χαρακτηριστικών για RSA υπό την παρουσία και απουσία F+R [51]

Υπό την σκοπιά της μηχανικής μάθησης, έχουμε ένα πρόβλημα δυαδικής ταξινόμησης, με χαρακτηριστικά τα events που σας παρουσιάσαμε παραπάνω και labels 0 ή 1 (υπάρχει ή όχι αντίπαλος). Τα πειράματα διεξήχθησαν υπό τρεις διαφορετικές συνθήκες φορτίου του επεξεργαστή (Zero Load-ZL, Medium Load-ML, Heavy Load-HL) αξιοποιώντας κατάλληλο πλήθος προγραμμάτων (που επιβαρύνουν την μνήμη), όπως παρέχονται από το SPEC [55]. Τα μοντέλα μηχανικής μάθησης που αξιολογήθηκαν παρουσιάζονται στο σχήμα 3.5.

No.	Machine Learning Model	Category
1	Linear Regression (LR)	Linear
2	Linear Discriminant Analysis (LDA)	Linear
3	Support Vector Machine (SVM)	Linear
4	Quadratic Discriminant Analysis (QDA)	Non-linear
5	Random Forest (RF)	Non-linear
6	K-Nearest Neighbors (KNN)	Non-linear
7	Nearest Centroid	Linear
8	Naive Bayes	Linear
9	Perceptron	Linear
10	Decision Tree	Non-linear
11	Dummy	Non-linear
12	Neural Networks	Non-linear

Σχήμα 3.5: Μοντέλα Μηχανικής Μάθησης που μελετήθηκαν [51]

Συνοπτικά (για δύο κλάσεις):

- LR: Γραμμικό μοντέλο, στο οποίο θεωρούμε ως σύνορο απόφασης (διαχωρισμού) την ευθεία
$$Y = bX + a \ , b = r\frac{s_X}{s_Y}, \ a = \overline{Y} - b\overline{X}$$
όπου s_X, s_Y οι δειγματικές διασπορές των X, Y αντίστοιχα, ενώ r είναι η μεταξύ τους Pearson συσχέτιση. Η τεχνική γενικεύεται και σε μεγαλύτερες διαστάσεις, όπου πλέον $y =< \overrightarrow{b}, \overrightarrow{x} >$ και $\overrightarrow{b} = (X^T X)^{-1} X^T Y$, όπου X ο πίνακας με γραμμές τις παρατηρήσεις x_i και Y το διάνυσμα με τα αντίστοιχα $y_i =< \overrightarrow{b}, \overrightarrow{x_i} >$.

- LDA: Θεωρούμε σύνορο απόφασης την
$$\frac{P[y=1|X]}{P[y=0|X]} = 1, \ P[y=i|X] = \frac{P[X|y=i]P[y=i]}{P[X|y=1]P[y=1] + P[X|y=0]P[y=0]}$$
Στην LDA υποθέτουμε ότι οι $P[X|y=i]$ ακολουθούν πολυδιάστατες κανονικές κατανομές με ίδιο πίνακα συνδιασποράς (γεγονός που οδηγεί σε γραμμικό σύνορο απόφασης).

- QDA: Σαν την LDA, μόνο που δεν θεωρεί ίδιο πίνακα συνδιασποράς για κάθε κλάση.

- Naive Bayes: Ίδια με την QDA, μόνο που απαιτεί ότι οι πίνακες συνδιασποράς είναι διαγώνιοι.

- Perceptron: Γραμμικός διαχωριστής που αντιστοιχεί σε νευρωνικό δίκτυο χωρίς hidden layer. Αποσκοπεί να βρει ευθεία (υπερεπίπεδο)
$$y = Wx + b$$
ώστε όλα τα στοιχεία μιας κλάσης να βρίσκονται στο ίδιο ημιεπίπεδο που ορίζει.

- SVM: Παρόμοιο με το Perceptron, μόνο που επιλέγει την ευθεία που διαχωρίζει τα σημεία των κλάσεων κατά τέτοιο τρόπο, ώστε να απέχει την μεγαλύτερη δυνατή απόσταση από τα ακριανά (support vectors). Αν και υπολογιστικά πιο σύνθετη από τον Perceptron, ευελπιστούμε ότι η επιπλέον απαίτηση θα οδηγήσει σε μικρότερο σφάλμα γενίκευσης (δες σχήμα [3.6]).

- Decision tree: Δέντρο που απαρτίζεται από εσωτερικούς κόμβους, οι οποίοι αποτελούν ένα $if - else$ κριτήριο πάνω στα χαρακτηριστικά, και φύλλα, στα οποία αναγράφεται η κλάση στην οποία κατατάσσει το συγκεκριμένο διάνυσμα χαρακτηριστικών. Η δομή του (π.χ. πλήθος κορυφών, ύψος) καθορίζεται από το σύνολο εκπαίδευσης αξιοποιώντας κατάλληλες συναρτήσεις.

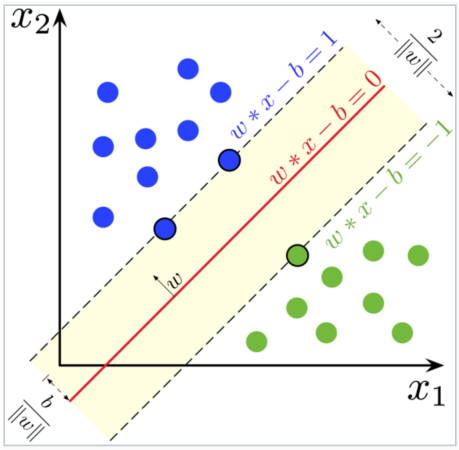
- RF: Μοντέλο που αποτελείται από πολλά δέντρα απόφασης. Η τελική ταξινόμηση προκύπτει συνδυάζοντας κατάλληλα τις αποφάσεις των απλούστερων (ensemble method) δέντρων απόφασης που το απαρτίζουν.

- Nearest Centroid: Από το σύνολο εκπαίδευσης υπολογίζουμε τα κεντροειδή (μέσος όρος των σημείων στον χώρο χαρακτηριστικών) κάθε κλάσης και στη συνέχεια κατατάσσουμε το κάθε νέο σημείο στην κλάση που έχει το πλησιέστερο centroid.

- KNN: Για κάθε νέο σημείο, εξετάζουμε τους k πλησιέστερους γείτονές του (ως προς κάποια μετρική π.χ. ευκλείδεια) και το κατατάσσουμε στην κλάση που περιέχει τους περισσότερους.

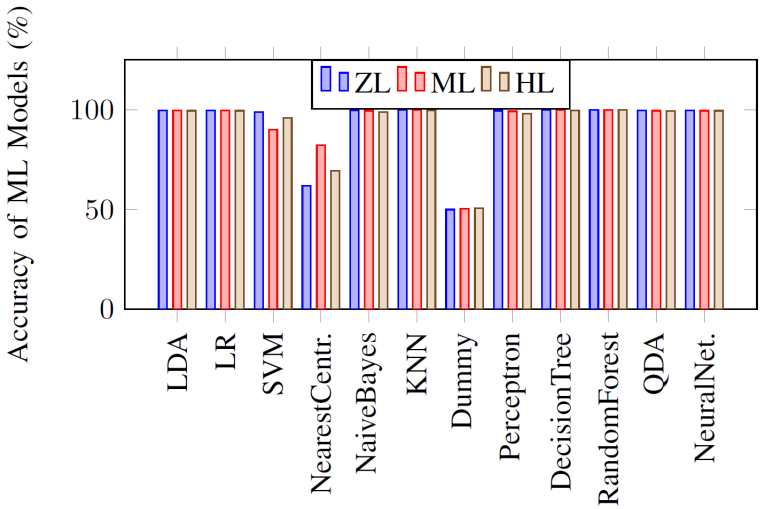
- Neural Networks: Όπως αναπτύχθηκαν στην εισαγωγή.

- Dummy: Πιθανώς, το απλούστερο μοντέλο μηχανικής μάθησης. Εστιάζει σε κάποια συγκεκριμένη ιδιότητα μιας κλάσης, υπολογίζοντας πόσα δείγματα την εμφανίζουν στο σύνολο εκπαίδευσης αυτής της κλάσης. Αν για παράδειγμα εντοπίσει ότι παρουσιάζεται στο 70% των δειγμάτων της κλάσης 1 (στο σύνολο εκπαίδευσης), τότε για κάθε νέο στοιχείο που την περιέχει το κατατάσσει στην 1 με πιθανότητα 0.7.

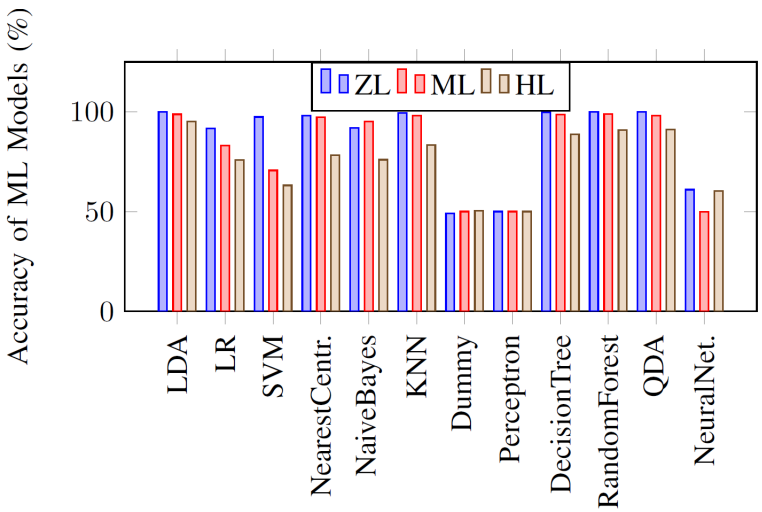


Σχήμα 3.6: Γραφική αναπαράσταση SVM. Παρατηρήστε ότι οποιαδήποτε ευθεία στο κίτρινο περιθώριο διαχωρίζει τις δυο κλάσεις, αλλά προτιμάται αυτή που βρίσκεται στη μέση [50]

Τα αποτελέσματα των πειραμάτων (ως προς την ευστοχία εντοπισμού) αναγράφονται παρακάτω:



Σχήμα 3.7: Σύγκριση μοντέλων Μηχανικής Μάθησης ως προς ευστοχία εντοπισμού F+R επίθεσης στο RSA [51]



Σχήμα 3.8: Σύγκριση μοντέλων Μηχανικής Μάθησης ως προς ευστοχία εντοπισμού F+F επίθεσης στο AES [51]

Αρχικά παρατηρούμε ότι ο εντοπισμός γίνεται δυσκολότερος καθώς μεταβαίνουμε από ZL σε HL, αφού τα επιπλέον προγράμματα που τρέχουν παράλληλα προσθέτουν θόρυβο στις κατανομές των χαρακτηριστικών. Επιπλέον, η πεποίθηση ότι η F+F είναι μια πιο κρυφή τεχνική επίθεσης ([54], [51]) ενισχύεται και από τα πειραματικά αποτελέσματα, όπου η ευστοχία για κάθε μοντέλο είναι γενικά χαμηλότερο από ότι στο F+R.

Θυμηθείτε ότι τελικός σκοπός αυτής της εφαρμογής είναι η σύγκριση και απομόνωση των μεθόδων που πετυχαίνουν τον εντοπισμό της επίθεσης γρήγορα, με ακρίβεια και χωρίς σημαντική επιβάρυνση μνήμης. Υπό αυτή την σκοπιά, εστίασαν αρχικά στην ευστοχία από την οποία έκριναν ότι τα Dummy, Perceptron, Neural Networks παρουσιάζουν ανεπαρκή επίδοση και επομένως κρίνονται αναποτελεσματικές. Στη συνέχεια, βάσει πειραμάτων που διεξήγαν, παρατήρησαν ότι τα RF και τα δέντρα απόφασης που πετύχαιναν μεγάλη ευστοχία είχαν μεγάλο ύψος και πολλά φύλλα, με αποτέλεσμα να υστερούν χρονικά των υπολοίπων. Επιπλέον, το μεγάλο μέγεθος δέντρων συνεπάγεται πολλούς βρόγχους $if - then$ οι οποίοι έκριναν ότι επιβάρυναν αρκετά την μνήμη. Όμοια επιβάρυνση μνήμης παρατήρησαν και στο KNN, αφού απαιτεί την αποθήκευση όλων των σημείων εκπαίδευσης (για τον εντοπισμό των γειτόνων). Τελικά, απομόνωσαν τα LDA, LR, SVM, QDA ως προτεινόμενα μοντέλα ανίχνευσης των δυο ειδών επιθέσεων, στα οποία εντόπισαν 2% επιβάρυνση μνήμης.

Αυτό που πιστεύουμε ότι αξίζει να συγκρατήσει ο αναγνώστης από αυτό το υποκεφάλαιο είναι η ύπαρξη μοντέλων ανίχνευσης σύνθετων CSCA, χωρίς να εστιάσει απαραίτητα σε ποια απομονώθηκαν. Ενδεχομένως, ένα Νευρωνικό Δίκτυο με διαφορετική, κατάλληλη υλοποίηση να παρουσίαζε παρόμοια επίδοση ως προς τις μετρικές που όρισαν. Τέλος, ίσως πιο αξιόλογη είναι η απόδοση των μεθόδων στην F+F επίθεση, αφού οι ίδιοι ισχυρίζονται ότι είναι η πρώτη

δημοσίευση που καταφέρνει τον εντοπισμό της ([51], 2018).

3.2 Προστασία Νευρωνικών Δικτύων από adversarial επιθέσεις

Ένα από τα κρισιμότερα ζητήματα που καλούμαστε να απαντήσουμε πριν προχωρήσουμε στην επόμενη ενότητα είναι η δυνατότητα χρήσης των νευρωνικών δικτύων για κρυπτογράφηση και αποκρυπτογράφηση. Επομένως, οφείλουμε να εξετάσουμε την ασφάλεια των κρυπτοσυστημάτων, όταν υποκαθιστούν τμήμα ([57]) ή ολόκληρο το κρυπτοσύστημα ([3], [4]). Μια βασική αρχή που διέπει την κρυπτογραφία είναι ότι ένα κρυπτοσύστημα οφείλει να είναι ασφαλές ακόμα και όταν όλα του τα χαρακτηριστικά (δομή, συναρτήσεις), πλην του κλειδιού, γνωστοποιηθούν (δεύτερος νόμος του Kerckhoffs). Αυτή η υπόθεση είναι ιδιαίτερα ισχυρή και προβληματική για μια αμελή και απροστάτευτη υλοποίηση νευρωνικού δικτύου. Υποθέστε, για παράδειγμα, ότι αξιοποιούμε ένα νευρωνικό δίκτυο για κρυπτογράφηση και αποκρυπτογράφηση. Πιθανότατα, το output του θα απαρτίζεται από 0 ή 1 που θα αντιστοιχούν στα bits εξόδου. Στην πραγματικότητα, μπορεί κανείς να δει ένα νευρωνικό με έξοδο k bits ως σύνθεση k νευρωνικών δυαδικής ταξινόμησης (0 ή 1 για το κάθε bit/νευρώνα εξόδου) που διαφέρουν μόνο στο output layer (parameter sharing). Έστω ότι ο αντίπαλος επιδιώκει να αλλάξει το πρώτο bit εξόδου. Εστιάζοντας στο αντίστοιχο νευρωνικό, για δεδομένη είσοδο x (π.χ. $x = m$ bits μηνύματος) ο αντίπαλος είναι σε θέση να βρει κατάλληλο x^t, x^t κοντά στο x, εκμεταλλευόμενος την παραγωγισιμότητα του δικτύου:

$$x^t = x + \varepsilon \nabla_x J(x, w, t)$$

όπου t η τιμή του *bit* που θέλει να αλλάξει. Η ιδέα είναι πολύ απλή:

- Το νευρωνικό κατατάσσει στην κλάση (bit) που ελαχιστοποιεί το σφάλμα
- Για να βρει το επιθυμητό x^t αρκεί απλά να μεταβεί στην κατεύθυνση που αυξάνει την συνάρτηση κόστους.

Οι [58] μελέτησαν περαιτέρω επιθέσεις σε deep learning μοντέλα και τις κατηγοριοποίησαν ως προς τον στόχο τους και τις γνώσεις που διαθέτει και εκμεταλλεύεται ο αντίπαλος για το νευρωνικό δίκτυο. Βάση της πληροφορίας που έχει στη διάθεση του χωρίζονται σε:

1. White box: έχει πλήρη επίγνωση των χαρακτηριστικών του μοντέλου και του συνόλου εκπαίδευσης και αξιολόγησης.
2. Black box: δεν διαθέτει καμία απολύτως πληροφορία για την αρχιτεκτονική και τις παραμέτρους του μοντέλου. Μπορεί, ωστόσο, να παρατηρήσει την έξοδο του δικτύου για οποιαδήποτε είσοδο χρήσει θεμιτή (μαντείο).

3. Gray Box: Έχει πρόσβαση στο παραπάνω μαντείο, γνωρίζει την αρχιτεκτονική του μοντέλου καθώς και τα σύνολα εκπαίδευσης και αξιολόγησης. Δεν έχει καμία πρόσβαση ή πληροφορία για τις παραμέτρους ασφαλείας του μοντέλου.

Η εφαρμογή που θα σας παρουσιάσουμε εστιάζει στο Gray Box σενάριο, το οποίο είναι ίσως και το πιο ρεαλιστικό.

Ανάλογα με τον στόχο της επίθεσης, εντάσσεται σε δυο κατηγορίες:

1. Στοχευμένη (targeted): στην οποία επιδιώκει για δοθείσα κλάση-στόχο t και είσοδο x ώστε $D(x) \neq t$ ($D(.)$ το αποτέλεσμα-κλάση του ταξινόμητη), να τροποποιήσει την είσοδο x κατά προσδιοριστέο z, ώστε $D(x+z) = t$ ([59]).

2. Μη στοχευμένη (Non-targeted): όπου αποσκοπεί στην τροποποίηση της εισόδου ώστε ο ταξινομητής να προβεί σε λανθασμένη ταξινόμηση. Δηλαδή, αναζητά z ώστε $D(x+z) \neq D(x)$ ([59]).

Όπως αντιλαμβάνεστε, οι μη στοχευμένες επιθέσεις είναι ασθενέστερες από τις στοχευμένες. Και οι δύο, όμως, εμφανίζουν πολυάριθμες εφαρμογές. Στο παράδειγμα που εισάγαμε προηγουμένως, μια μη στοχευμένη επίθεση μπορεί να καταφέρει να διαστρεβλώσει το μήνυμα c (τροποποιώντας τα bit του), ώστε όταν αποκρυπτογραφηθεί από τον παραλήπτη: $D(c) = m' \neq m$. Η στοχευμένη είναι ακόμα πιο επικίνδυνη, αφού παρέχει την δυνατότητα ακριβούς τροποποίησης του c, με σκοπό, πιθανώς, αποκρυπτογράφησης κατάλληλου $m' = D(c)$.

Η πλειοψηφία των επιθέσεων βασίζονται στην διαφορισιμότητα του δικτύου. Η πρώτη τέτοιου είδους επίθεση πραγματοποιήθηκε το 2014 από τους [60], εκμεταλλευόμενοι τον αλγόριθμο βελτιστοποίησης L-BFGS, ο οποίος παρουσιάζει παρόμοια φιλοσοφία με την γνωστή Newton Raphson (εντάσσεται στις Quasi-Newton μεθόδους) λαμβάνοντας υπ' όψιν επιπλέον χωρικούς περιορισμούς (λόγω πολλών διαστάσεων). Στη συνέχεια εμφανίστηκαν και άλλοι αλγόριθμοι, όπως οι Fast Gradient Sign Method (FGSM) [61], Basic Iterative Method(BIM) [62], Random FGSM [63] και Carlini and Wagner (CW) [59].

Ενδεικτικά, η FGSM επιδιώκει να βρει την κατάλληλη τροποποίηση x^t στην είσοδο, ώστε:

$$x^t = x + \varepsilon sign(\nabla_x J(x, w, t))$$

Η BIM επεκτείνει την FGSM, τρέχοντας την διαδοχικά πολλές φορές:

$$x_0^t = x,\ x_{n+1}^t = proj_S\{x_n^t + \varepsilon sign(\nabla_x J(x, w, t))\}$$

όπου $proj_S$ είναι η προβολή του σημείου στο σύνολο S το οποίο είναι συνήθως μια περιοχή ε κοντά του αρχικού x.

Η R-FGSM προσθέτει επιπλέον θόρυβο στην παραδοσιακή BIM:

$$x^t = x' + (\varepsilon - a)sign(\nabla_{x'} J(x', w, t)),\ x' = x + a * sign(N(0^d, I^d))$$

Τέλος, η CW αναζητά δ, $x^t = x + \delta$:

$$\min_{\delta\ s.\ t.\ 0 \le x_i + \delta_i \le 1} (\|\delta\|_p + c * f(x + \delta))$$

όπου $\|\delta\|_p = (\sum_i |\delta_i|)^{\frac{1}{p}}$, $f(x^t) = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -k)$, $Z()$ η τιμή του ταξινομητή στο τελευταίο στρώμα πριν την εφαρμογή της συνάρτησης ενεργοποίησης (συνήθως $softmax$), και k, c κατάλληλες μεταπαράμετροι. Παρατηρήστε ότι $D(x+\delta) = t \Leftrightarrow Z(x+\delta)_t \geq \max\{Z(x+\delta)_i : i \neq t\} \Leftrightarrow f(x+\delta) \leq 0$. Επομένως, ευελπιστούμε η ελαχιστοποίηση της παραπάνω παράστασης να οδηγήσει σε αρνητικές τιμές της f, για σχετικά μικρό δ (όπως επιβάλει η νόρμα του).

Η μεγαλύτερη αδυναμία των προαναφερθέντων μεθόδων είναι ότι βασίζονται σε αλγορίθμους βελτιστοποίησης πρώτης (απαιτείται υπολογισμός παραγώγου) ή δεύτερης τάξης (υπολογίζουν δεύτερες παραγώγους) οι οποίοι με τη σειρά τους απαιτούν την γνώση ορισμένων χαρακτηριστικών του δικτύου. Επομένως, δε μπορούν να αξιοποιηθούν σε Black Box επιθέσεις. Για αυτό αναπτύχθηκαν εναλλακτικές προσεγγίσεις, όπως η Zero Order Optimisation (ZOO) ([64]), η οποία επιχειρεί να επιλύσει το πρόβλημα βελτιστοποίησης αξιοποιώντας μόνο το μαντείο που έχουμε στην διάθεση μας. Για τον λόγο αυτό τροποποιεί την συνάρτηση σφάλματος του CW, ώστε

$$f(x) = \max(\max_{i \neq t} \log F(x')_i - \log F(x')_t, -k)$$

όπου $F(x)_i = \frac{e^{Z(x)_i}}{\sum_{j=1^K} e^{Z(x)_j}}$. Η F, επομένως, είναι η τιμή του τελευταίου στρώματος μετά την εφαρμογή της $softmax$ συνάρτησης ενεργοποίησης. Θυμηθείτε ότι έχουμε επίγνωση μόνο των τιμών εξόδου $F(x)\ \forall\ x$ (μαντείο) και καμία πληροφορία για το δίκτυο, όπως τα $Z(x)$ που αξιοποιεί η CW. Αξιοποιώντας το μαντείο, γνωρίζουμε $f(x) \forall\ x$ και μπορούμε να προσεγγίσουμε τις παραγώγους της ως:

$$g_i = \frac{\partial f}{\partial x_i} \simeq \frac{f(x + he_i) - f(x - he_i)}{2h}$$

$$h_i = \frac{\partial^2 f}{\partial^2 x_i} \simeq \frac{f(x + he_i) - 2f(x) + f(x - he_i)}{h^2}$$

όπου e_i το γνωστό μοναδιαίο. Εφοδιασμένοι, πλέον, με προσεγγίσεις των παραγώγων είμαστε σε θέση να τρέξουμε οποιονδήποτε αλγόριθμο βελτιστοποίησης που βασίζεται σε αυτές.

Οι τεχνικές προστασίας απέναντι σε τέτοιου είδους επιθέσεις είναι πολυάριθμες και μπορούν να καταταχθούν στις ακόλουθες κατηγορίες ([58]):

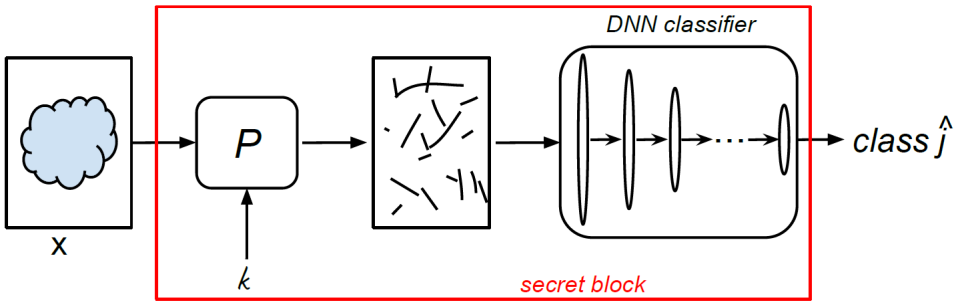
1. Defence via Retraining: Ίσως η απλούστερη και πιο φυσική μορφή άμυνας απέναντι σε τέτοιες επιθέσεις είναι αυτή που στηρίζεται στην επανεκπαίδευση. Πρωτεργάτες αυτής της προσέγγισης είναι οι [65] και βασίζεται στην παρεμβολή τέτοιων προβληματικών input στο στάδιο εκπαίδευσης του δικτύου.

2. Defence via detection and rejection: Η βασική ιδέα τέτοιων μεθόδων άμυνας είναι ο εντοπισμός και απόρριψη τέτοιων εισόδων. Μια πρώτη προσέγγιση είναι η χρήση ενός ξεχωριστού νευρωνικού δικτύου για τον εντοπισμό τέτοιων τροποποιημένων εισόδων [66]. Εναλλακτικά, οι [67] προτίμησαν να εστιάσουν στην μελέτη ορισμένων στατιστικών ιδιοτήτων του δείγματος και πώς αυτές τροποποιούνται με την εμφάνιση τέτοιων στιγμιοτύπων.

3. Defence via input pre-processing: Υπό μια έννοια αποτελεί επέκταση της παραπάνω προσέγγισης, όπου δεν αρκούμαστε μόνο στον εντοπισμό, άλλα και στην επαναφορά των τροποποιημένων εισόδων στα αρχικά (απαλοιφή του ε από το $x^t = x + \varepsilon$) ([68]).

4. Defence via regeneration: Μια πιο σύνθετη προσέγγιση όπου επιδιώκουμε να εντοπίσουμε μια απεικόνιση των τροποποιημένων εισόδων στα αρχικά. Οι [69] ενσωμάτωσαν αυτήν την επιπλέον λειτουργία στο Νευρωνικό τους Δίκτυο τροποποιώντας κατάλληλα την συνάρτηση σφάλματος ώστε να είναι ανθεκτικότερη σε τέτοιου είδους εισόδους.

Η εφαρμογή που θα σας παρουσιάσουμε προτείνει μια μέθοδο άμυνας που εντάσσεται τυπικά στην Defence via input pre-processing κατηγορία, αφού αποσκοπεί στην κατάλληλη επεξεργασία της εισόδου, δανειζόμενη τεχνικές και ιδέες από την κρυπτογραφία. Η βασική λειτουργία και αρχιτεκτονική συνοψίζεται ως εξής:



Σχήμα 3.9: Σχηματική αναπαράστασης ενισχυμένου νευρωνικού δικτύου. Η είσοδος x, η έξοδος $\widehat{j}$ και η αρχιτεκτονική είναι γνωστή. Οι τιμές παραμέτρων που εμπεριέχονται στο secret block παραμένουν άγνωστες [58]

Η P θεωρούμε ότι αποτελεί έναν κατάλληλο μετασχηματισμό (ιδανικά μη αντιστρέψιμος και μη διαφορίσιμος) των χαρακτηριστικών ο οποίος εξαρτάται εξ' ολοκλήρου από την τιμή του κλειδιού k και όχι από το x. Η ανεξαρτησία μετασχηματισμού-δεδομένων δεν επιτρέπει στον

αντίπαλο να μάθει οποιαδήποτε πληροφορία για αυτόν αξιοποιώντας το δείγμα. Τέλος, όπως είναι σύνηθες στην κρυπτογραφία, αποφεύγεται η επαναχρησιμοποίηση του ίδιου κλειδιού k στον μετασχηματισμό P.

Τα πειράματα που διεξήχθησαν βασίστηκαν σε ταξινόμηση ψηφίων (MNIST) ([70]) και αντικειμένων μόδας (Fashion-MNIST) ([71]) από εικόνες.

Σχήμα 3.10: Ενδεικτικά παραδείγματα δείγματος: Πάνω: MNIST, Κάτω:Fashion-MNIST [58]

Και για τα δυο προβλήματα ταξινόμησης θεώρησαν δείγμα $55000 - 5000 - 10000$ (train-validation-test) ασπρόμαυρων εικόνων μεγέθους 28×28. Μελέτησαν την ανθεκτικότητα του μοντέλου τους απέναντι σε FGSM (υλοποίηση παρμένη από την CleverHans βιβλιοθήκη της python [72]) και CW (όπως την υλοποίησαν οι ίδιοι [73]) επιθέσεις. Θεώρησαν ως P μια ψευδοτυχαία μετάθεση, ενώ οι αρχιτεκτονικές των νευρωνικών που αξιοποίησαν παρουσιάζονται παρακάτω:

Layer	# filters	kernel size
Conv2D	64	8×8
ReLu		
Conv2D	128	6×6
ReLu		
Conv2D	128	5×5
ReLu		
Flatten		
Dense		10

Layer	# filters	kernel size
Conv2D	32/64	3×3
ReLu		
Conv2D	32/64	3×3
ReLu		
MaxPool		2×2
Conv2D	64/128	3×3
ReLu		
Conv2D	64/128	3×3
ReLu		
MaxPool		2×2
Flatten		
Dense		200/256
ReLu		
Dropout		
Dense		200/256
ReLu		
Dense		10

Σχήμα 3.11: Αρχιτεκτονική νευρωνικών δικτύων: Αριστερά: για την FGSM επίθεση, Δεξιά: για την CW επίεση [58]

Τα αποτελέσματα των πειραμάτων τους συνοψίζονται στον παρακάτω πίνακα:

Attack	Classical classifier		Classifier on permuted data	
	original	attacked	original	attacked
MNIST				
CW ℓ_2	1.00	100.00	3.00	8.64
CW ℓ_0	1.00	100.00	3.00	14.53
CW ℓ_∞	1.00	99.99	3.00	12.24
FGSM	1.00	92.10	1.40	18.00
Fashion MNIST				
CW ℓ_2	7.50	100.00	11.50	12.12
CW ℓ_0	7.50	100.00	11.50	13.48
CW ℓ_∞	7.50	99.90	11.50	12.55
FGSM	8.60	60.60	11.20	27.50

Σχήμα 3.12: Σύγκριση ποσοστιαίου σφάλματος απροστάτευτου Νευρωνικού Δικτύου (αριστερά) και ενισχυμένου (δεξιά) [58]

Παρατηρήστε την δραματική μείωση του σφάλματος, όπου σε σχεδόν όλες τις περιπτώσεις υποδεκαπλασιάζεται.

Κλείνοντας αυτήν την υποενότητα θέλουμε να σας επιστήσουμε την προσοχή σε αυτό που κρίνουμε πιο ουσιώδες. Το βασικό ερώτημα που διατυπώσαμε στην αρχή αυτού του κεφαλαίου είναι το κατά πόσο τα νευρωνικά δίκτυα είναι σε θέση να αξιοποιηθούν σε τομείς ασφάλειας (π.χ. ως τμήμα κρυπτοσυστήματος). Ευελπιστούμε ότι η θεωρία και οι εφαρμογές που αναφέραμε και αναπτύξαμε να υποδεικνύουν την δυνατότητα ενίσχυσης τους με σκοπό την ανθεκτικότητα τους απέναντι σε κρυπτογραφικές επιθέσεις που εκμεταλλεύονται την δομή τους. Μολονότι το πρόβλημα μελετήθηκε υπό την σκοπιά ταξινόμησης εικόνων - ένα πρόβλημα που φαντάζει αρκετά απόμακρο με οποιοδήποτε κρυπτοσύστημα - δε παύει να αποτελεί μια πρώτη ισχυρή θετική ένδειξη.

Ο λόγος που αποφασίσαμε να σας παρουσιάσουμε αυτή την εφαρμογή, έναντι των υπολοίπων που υπάρχουν στην βιβλιογραφία, είναι διπλός. Αρχικά, δανείζεται κομψά έννοιες και από τα δυο επιστημονικά πεδία στα οποία εστιάζουμε σε αυτή την εργασία. Πρόκειται για μια βασική και ουσιαστική εφαρμογή η οποία μπορεί να αναλυθεί περαιτέρω, είτε εξετάζοντας πιο σύνθετες και σύγχρονες επιθέσεις νευρωνικών δικτύων, είτε μελετώντας την επίδραση που έχει στην ασφάλεια του συστήματος η επιλογή μιας πιο σύνθετης συνάρτησης μετασχηματισμού P από τον κλάδο της Κρυπτογραφίας. Έπειτα, η ενίσχυση του νευρωνικού επιτυγχάνεται με απλό και ανεξάρτητο τρόπο από την αρχιτεκτονική του νευρωνικού δικτύου, αφού ο μετασχηματισμός P που επιφέρει την επιπλέον προστασία προσαρμόζεται εύκολα πάνω του (modular).

3.3 Ανακεφαλαίωση

Σε αυτήν την ενότητα ασχοληθήκαμε με την εφαρμογή μοντέλων μηχανικής μάθησης για την προστασία κρυπτοσυστημάτων και αναλύσαμε την ανθεκτικότητα των Νευρωνικών Δικτύων απέναντι σε στοχευμένες επιθέσεις που επιχειρούν την δυσλειτουργία τους. Μελετήσαμε αναλυτικά την επίδοση τους στον εντοπισμό CSCA, όπου παρατηρήσαμε εντυπωσιακά αποτελέσματα, αφού κατάφεραν να εντοπίσουν επιτυχώς για πρώτη φορά F+F επιθέσεις. Έπειτα, εξετάσαμε την ανθεκτικότητα τους απέναντι σε κατάλληλα τροποποιημένα δείγματα που αποσκοπούν στην στοχευμένη αστοχία τους. Η μη προσεκτική τους υλοποίηση και ενίσχυση τα καθιστά ευάλωτα σε τέτοιες επιθέσεις, γεγονός που τα χρήζει επικίνδυνα για την χρήση τους ως τμήματα κρυπτοσυστήματος (π.χ. να αντικαταστήσουν τα S-Box του AES [57]).

Στο επόμενο κεφάλαιο θα μελετήσουμε την δυνατότητα χρήσης Deep Learning μοντέλων για την προσέγγιση σύνθετων κατανομών (capacity) οι οποίες μπορούν να χρησιμοποιηθούν αυτούσιες ως κρυπτοσυστήματα.

Κεφάλαιο 4

Υποκατάσταση Κρυπτοσυστημάτων με Μηχανική Μάθηση

Το παρόν κεφάλαιο θα μελετήσει την δυνατότητα χρήσης μοντέλων Μηχανικής Μάθησης για την υποκατάσταση κρυπτοσυστημάτων. Πιο συγκεκριμένα, θα εστιάσει στα Νευρωνικά Δίκτυα, τα οποία, λόγω της αυξημένης πολυπλοκότητας τους, φαίνεται να παρουσιάζουν το μεγαλύτερο capacity και επομένως είναι σε θέση να προσεγγίσουν πιο σύνθετες κατανομές. Μέσω των εφαρμογών που θα σας παρουσιάσουμε, θα διερευνηθεί η δυσκολία αυτού του προβλήματος και θα αναλογιστούμε τον βαθμό στον οποίο είναι εφικτή η προσέγγιση τόσο σύνθετων συναρτήσεων.

Στην πρώτη ενότητα, θα σας παρουσιάσουμε μια εφαρμογή που εστιάζει στην κρυπτανάλυση ενός απλού Simon Cipher κρυπτοσυστήματος, υιοθετώντας ένα μοντέλο Νευρωνικού Δικτύου ([3]). Αποφασίσαμε να την εντάξουμε σε αυτό το κεφάλαιο, έναντι του δευτέρου, επειδή η προσέγγιση που ακολουθεί φανερώνει τον βαθμό δυσκολίας της υποκατάστασης ενός κρυπτοσυστήματος με ένα τέτοιο μοντέλο. Ορμώμενοι από αυτά τα αποτελέσματα, στην δεύτερη ενότητα θα σας παρουσιάσουμε μια πιο σύνθετη προσέγγιση η οποία εκμεταλλεύεται την τεχνική του adversarial training, αποσκοπώντας στην δημιουργία κατάλληλου GAN που θα προσομοιάζει ένα μοντέλο ασφαλούς επικοινωνίας ([4]).

4.1 Μελέτη δυσκολίας υποκατάστασης κρυπτοσυστήματος από την κρυπτανάλυση Simon Cipher με Νευρωνικό Δίκτυο

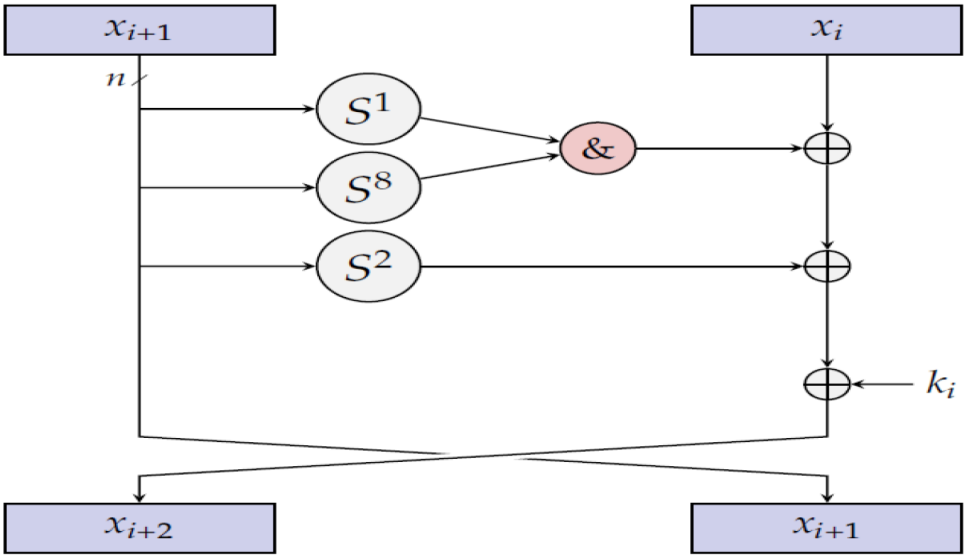
Η εφαρμογή που θα σας παρουσιάσουμε υιοθετεί μια ενδιαφέρουσα οπτική για την κρυπτανάλυση του Simon Cipher. Παραδοσιακά, οι επιθέσεις βασίζονται στην εξόρυξη πληροφορίας για το αρχικό μήνυμα ή το κλειδί κρυπτογράφησης βασιζόμενοι αποκλειστικά σε ζεύγη (m, c) που αντιστοιχούν στο ίδιο προσδιοριστέο κλειδί. Ήδη, στο δεύτερο κεφάλαιο, μελετήσαμε την επίθεση του DES σε συνθήκες Known Plaintext Attack, όπου έχουμε στην διάθεση μας πολλά ζεύγη (m, c). Η επίθεση που θα σας παρουσιάσουμε επεκτείνει το Known Plaintext Attack, θεωρώντας ότι είμαστε εφοδιασμένοι με πολλά ζεύγη (m, c) που αντιστοιχούν σε ένα **σύνολο**

από κλειδιά $D = \{k_1, ..., k_n\}$. Τελικός σκοπός της εφαρμογής είναι η εκπαίδευση ενός νευρωνικού δικτύου που θα μπορεί, δοθέντος του ζεύγους (m, c), να κατασκευάσει το κλειδί $k \in D$ που του αναλογεί.

Στο σημείο αυτό, πιθανώς, ο αναγνώστης να αναρωτιέται πώς αυτή η εφαρμογή κρυπτανάλυσης συνδέεται με το κεντρικό εγχείρημα αυτού του κεφαλαίου. Ίσως γίνει πιο ξεκάθαρο αν εστιάσουμε περαιτέρω στον σκοπό του Νευρωνικού Δικτύου. Το μοντέλο, δεχόμενο ως είσοδο (m, c), παράγει μια αλληλουχία από bits που αντιστοιχούν σε κάποιο κλειδί του χώρου κλειδιών K, $D \subseteq K$. Η έξοδος, δηλαδή, δεν είναι απλά ένας δείκτης $i \in \{1, ..., n\}$ που υποδηλώνει ποιο από τα n κλειδιά στο D αξιοποιήθηκε, άλλα μια σειρά από bits. Υπό αυτή την έννοια, επιχειρεί να προσεγγίσει την σύνθετη κατανομή των κλειδιών, περιορισμένη στο σύνολο D. Ιδανικά, αν για $D = K$ (είχαμε δηλαδή (m, c) για κάθε κλειδί) πετυχαίναμε μεγάλη ακρίβεια, τότε θα είχαμε προσεγγίσει μια πολύ σύνθετη κατανομή, όμοια με αυτή που θέλουμε να υποκαταστήσουμε σε αυτό το κεφάλαιο. Αντιθέτως, αν η εφαρμογή παρουσιάζει περιορισμένη ακρίβεια σε αυτό το αισθητά απλούστερο πρόβλημα, τότε θα είχαμε μια ένδειξη για την δυσκολία του.

Το Simon Cipher αποτελεί ένα lightweight block cipher ([74]). Τα lightweight ciphers είναι μια ειδική κατηγορία κρυπτοσυστημάτων τα οποία υιοθετούν απλούστερες συναρτήσεις κρυπτογράφησης για να μειώσουν την υπολογιστική τους απαίτηση. Η δυσκολία υπόκειται στην εύρεση και σύνθεση τέτοιων συναρτήσεων για την δημιουργία αποδοτικών και ασφαλών αλγορίθμων κρυπτογράφησης-αποκρύπτογράφησης τα οποία είναι σε θέση να τρέξουν και να προστατεύσουν συσκευές περιορισμένης υπολογιστικής ισχύος, όγκου και ενέργειας. Ίσως η πιο άμεση και σπουδαία εφαρμογή τέτοιων lightweigh αλγορίθμων είναι στο Internet of Things (IoT), όπου συσκευές της καθημερινότητας θα συνδέονται, θα επικοινωνούν και θα αλληλεπιδρούν μεταξύ τους μέσω του διαδικτύου (προς διευκόλυνση του χρήστη). Συνεπώς, θα διαχειρίζονται ευαίσθητα προσωπικά δεδομένα, αξιοποιώντας αισθητά λιγότερη υπολογιστική ισχύ.

Επέλεξαν, λοιπόν, το Simon Cipher, ελπίζοντας ότι οι απλούστερες συναρτήσεις που το απαρτίζουν θα οδηγήσουν σε μια λιγότερο σύνθετη κατανομή πάνω στα κλειδιά.



Σχήμα 4.1: Ένας γύρος από το δίκτυο Feistel του Simon Cipher [3]

Στην πραγματικότητα, όπως φαίνεται και στο παραπάνω σχήμα, το Simon Cipher αποτελεί ένα δίκτυο Feistel (όπως και το DES). Η είσοδος $x_{i+1}||x_i$ bits χωρίζεται στη μέση, ενώ οι S^1, S^8, S^2 αντιστοιχούν σε κυκλικές μεταθέσεις των bits προς τα αριστέρα κατά $1, 8, 2$, θέσεις αντίστοιχα. Το &, $\oplus$ αντιστοιχούν σε AND, XOR bit πράξεις. Εν γένει, ο αλγόριθμος για κάθε ένα από τους T γύρους που τρέχει, παράγει T το πλήθος k_i από το αρχικό κλειδί k, τα οποία και αξιοποιεί σε κάθε γύρο Feistel, όπως φαίνεται στο σχήμα. Μολονότι υπάρχουν διάφορες υλοποιήσεις του Simon Cipher, ανάλογα με τα μεγέθη των m, c, k, σε αυτήν την εφαρμογή θεώρησαν $|m| = |c| = 32$, $|k| = 64$. Επιχείρησαν μια απλοποιημένη εκδοχή του η οποία απαρτιζόταν από 1 ή 2 γύρους, έναντι των 32 που προτάθηκε από τους πρωτεργάτες του ([74]). Αξιοποιήθηκε η έτοιμη υλοποίηση από τους [75], προσαρμόζοντας το για 1 ή 2 γύρους.

Το Νευρωνικό Δίκτυο που εκπαίδευσαν απαρτιζόταν από πολλά Fully Connected Hidden Layers μέγιστου πλάτους 1024 νευρώνων. Το input και output layer είχε 64 νευρώνες (για τα bits m, c, k αντίστοιχα), ενώ ως συνάρτηση ενεργοποίησης προτιμήθηκε η ReLU. Η binary crossentropy χρησιμοποιήθηκε ως συνάρτηση σφάλματος, ενώ ο αλγόριθμος βελτιστοποίησης ήταν ο adam. Ο adam (Adaptive Moments) ([77]) αποτελεί μια επέκταση της στοχαστικής μεθόδου των κλίσεων η οποία στην επανάληψη t εκμεταλλεύεται κατάλληλα την πληροφορία των τιμών των παραγώγων των προηγούμενων επαναλήψεων $\{0, 1, ..., t-1\}$. Πειραματικά, παρουσιάζει, εν γένει, λιγότερη ευαισθησία ([14]) ως προς την επιλογή των υπερπαραμέτρων (π.χ. βήμα μάθησης ϵ) με αποτέλεσμα να απαιτεί λιγότερες δοκιμές για τον προσδιορισμό τους.

Algorithm 8.7 The Adam algorithm

Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
  Initialize time step $t = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
    Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

Σχήμα 4.2: Αλγόριθμος Adam [14]

Τα βάρη στο i-οστό hidden layer αρχικοποιήθηκαν από την $U(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}})$, όπου m, n είναι το πλάτος (πλήθος νευρώνων) του στρώματος $i-1$ και i αντίστοιχα ([76]).

Το δείγμα τους αποτελούνταν από 5000 χιλιάδες ζεύγη (m, c) για κάθε ένα από τα 1000 εξεταζόμενα κλειδιά. Σε κάθε επανάληψη, αξιολογείται η επίδοση του νευρωνικού για όλα τα ζεύγη που αντιστοιχούν στο ίδιο κλειδί (batch size= 5000), ενώ η εκπαίδευση διήρκεσε 35 epochs.

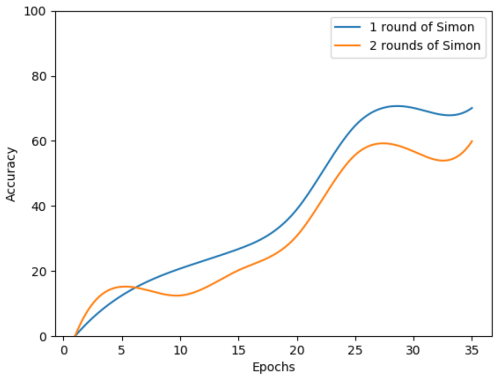


Fig. 9. Plot of Accuracy vs. Epochs

Σχήμα 4.3: Ευστοχία Νευρωνικού Δικτύου συναρτήσει epochs [3]

Ευστοχία θεωρήθηκε το ποσοστό των bits που προσδιορίζει επιτυχώς το δίκτυο. Όπως είναι λογικό, παρατηρείται μεγαλύτερη ακρίβεια όταν εξετάζεται ένας μόνο γύρος ($\sim 70\%$) του

simon cipher, παρά δυο ($\sim 60\%$). Ενδεικτικά, για 4 γύρους η ευστοχία κυμαινόταν μεταξύ $50\% - 60\%$. Ιδανικά, θα θέλαμε ευστοχία αισθητά μεγαλύτερη από 50% το οποίο αναλογεί στο να μαντεύουμε τυχαία bits.

```
Record: 00000010001001100111000001111111 01000001101101100000001000100110
Original Key:  10011001111011100000010001011111110101010111010100111101010010001
Predicted Key:
Epoch 2:       10101000101001011101010001010101101101111110111101111011101010101
Epoch 3:       10101000101001011101010001010101101101111110111101111011101010101
Epoch 15:      10101000101001101001010001010111101101011111011110111011101010101
Epoch 30:      10001000101001101001010001010111101101011111011110111011101010101
```

Σχήμα 4.4: Παράδειγμα εξόδου Νευρωνικού Δικτύου για 1 γύρο simon cipher [3]

Θεωρούμε ότι η εφαρμογή φανερώνει έναν δείκτη δυσκολίας για το πρόβλημα που επιχειρούμε να προσεγγίσουμε σε αυτό το κεφάλαιο. Προφανώς, δεν ισχυριζόμαστε ότι με μια διαφορετική, πιο προσεγμένη αρχιτεκτονική Νευρωνικού Δικτύου δε θα μπορούσε να επιτευχθεί μεγαλύτερη ακρίβεια. Άλλωστε, μια από τις μεγαλύτερες δυσκολίες αυτού του κλάδου (που του προσδίδει τόση ομορφιά) είναι το γεγονός ότι η εύρεση των αποτελεσματικότερων, για το πρόβλημα, παραμέτρων ενός σύνθετου δικτύου βασίζεται κυρίως στην διαίσθηση. Παρόλα αυτά, το συγκεκριμένο αποτέλεσμα μας εμπνέει να εισάγουμε πιο σύγχρονες αρχιτεκτονικές και μεθόδους εκπαίδευσης. Ανατρέξτε ότι η συγκεκριμένη υλοποίηση τείνει να προσομοιάζει το τυχαίο μάντεμα ήδη από τον τέταρτο γύρο, ενώ ο αλγόριθμος προτάθηκε για 32. Επιπλέον, η κατανομή που προσπάθησε να προσεγγίσει είναι αρκετά απλούστερη από την πραγματική, καθώς δεν εστίαζε σε όλο το σύνολο κλειδιών. Στην επόμενη ενότητα, θα σας παρουσιάσουμε μια εναλλακτική οπτική, η οποία αξιοποιεί το μοντέλο του adversarial training και των GAN.

4.2 Δημιουργία κρυπτοσυστήματος με χρήση GAN

Η ιδέα των GAN (Generative Adversarial Networks) πρωτοσυστάθηκε από τους [1] και αποτελεί, υπό μια έννοια, μια λογική επέκταση του Deep Learning. Θυμηθείτε, ότι έναυσμα της υιοθέτησης τέτοιων μοντέλων αποτέλεσε η επίλυση σύνθετων προβλημάτων όπου η συσχέτιση μεταξύ εισόδου και εξόδου δεν είναι εύκολο να προσδιοριστεί εκ των προτέρων. Επομένως, προτιμήσαμε να συνθέσουμε αλγορίθμους οι οποίοι θα την εξάγουν, βασιζόμενοι στα δεδομένα που τους τροφοδοτούμε. Η ορθή εκπαίδευση ενός Νευρωνικού Δικτύου, δεν είναι ιδιαίτερα τυποποιημένη και η βέλτιστη επιλογή υπερπαραμέτρων, αρχιτεκτονικής και αλγορίθμου βελτιστοποίησης δεν είναι πάντοτε εύκολα προσδιορίσιμη. Τις περισσότερες φορές, για την επίλυση μιας εφαρμογής, ανατρέχουμε στην βιβλιογραφία και ενσωματώνουμε χαρακτηριστικά Νευρωνικών Δικτύων που έχουν εμφανίσει θετικά αποτελέσματα σε παρεμφερή προβλήματα. Το ζήτημα

είναι πώς προσεγγίζουμε νέα προβλήματα των οποίων οι προσδιοριστέες κατανομές είναι ιδιαίτερα σύνθετες·

Υποθέστε, ότι θέλουμε να κατασκευάσουμε ένα Νευρωνικό Δίκτυο το οποίο θα προσεγγίζει μια σύνθετη κατανομή, ώστε να μπορεί να παράξει στοιχεία που ανήκουν σε αυτήν. Για παράδειγμα, μπορεί να ευελπιστούμε να κατασκευάσουμε εικόνες προσώπων ή ψηφίων (όπως μελετήθηκε στο [1]). Σκοπός μας, λοιπόν, είναι η αξιοποίηση του συνόλου εκπαίδευσης για την σύνθεση κανόνων που περιγράφουν σε ικανοποιητικό βαθμό την κατανομή, ώστε να μπορούμε να δημιουργήσουμε νέα δείγματα που υπόκεινται σε αυτήν. Ιδανικά, δοθέντος ενός στιγμιότυπου x του προβλήματος μας, θα θέλαμε να μη μπορούμε να διακρίνουμε αν αυτό παράχθηκε από το Νευρωνικό μας Δίκτυο ή προήλθε από την ζητούμενη κατανομή.

Έστω G (generator) το Νευρωνικό Δίκτυο που επιχειρεί να συνθέσει δείγματα από την προσδιοριστέα κατανομή. Αυτό, δεχόμενο ως είσοδο ένα διάνυσμα χαρακτηριστικών z (συνήθως τυχαίος θόρυβος) συνθέτει μια τιμή $f(z;\theta_G) = G(z)$. Όπως προείπαμε, ευελπιστούμε τα $G(z) \sim p_G, x \sim p_{data}$ να είναι δυσδιάκριτα. Για την ποσοτικοποίηση και τον έλεγχο αυτής της ομοιότητας εισάγουμε ένα δεύτερο Νευρωνικό Δίκτυο D (Discriminator), το οποίο αποτελεί έναν απλό δυαδικό ταξινομητή. Ο D, δοθέντος ενός στιγμιοτύπου x, προσπαθεί να μαντέψει αν αυτό προήλθε από τον G ή από την πραγματική κατανομή, δηλαδή $D(x;\theta_D) = P[x \in p_{data}]$. Πρόκειται, λοιπόν, για ένα παίγνιο μεταξύ των G και D, στο οποίο ο G προσπαθεί να γίνει καλύτερος (να παράξει πιο πειστικά στιγμιότυπα) για να ξεγελάσει τον D. Ο D με την σειρά του βελτιώνεται, απαιτώντας ολοένα και πιο ρεαλιστικά δείγματα από τον G για να ξεγελαστεί. Στην ισορροπία του παιγνίου $D(x) = 0.5$ δηλαδή, το D θα αδυνατεί να διακρίνει την προέλευσή του και θα καταφεύγει σε τυχαίο μάντεμα.

Πιο τυπικά, ο D θέλει να μεγιστοποιήσει την πιθανότητα $D(x)$ για $x \sim p_{data}$, ενώ ο G θέλει να τον ξεγελάσει, επομένως να ελαχιστοποιήσει την πιθανότητα ανίχνευσης του, δηλαδή το $1 - D(G(z))$. Άρα, έχουμε ένα min-max πρόβλημα:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_G(z)}[\log(1 - D(G(z)))]$$

Εν γένει, η τιμή της V μπορεί να ποικίλει, ανάλογα με τις υποθέσεις και τους στόχους του κάθε νευρωνικού δικτύου. Στην εφαρμογή που θα σας παρουσιάσουμε στη συνέχεια, θα δείτε μια διαφορετική μορφή της V η οποία όμως έχει παρόμοια φιλοσοφία (ένα κατάλληλο min-max παίγνιο μεταξύ των αντιπάλων).

Οι αλγόριθμοι βελτιστοποίησης δεν διαφοροποιούνται από τους παραδοσιακούς που έχουμε δει έως τώρα. Στην πράξη, τα δυο Νευρωνικά Δίκτυα εκπαιδεύονται παράλληλα, όπως φαίνεται και στον παρακάτω αλγόριθμο:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

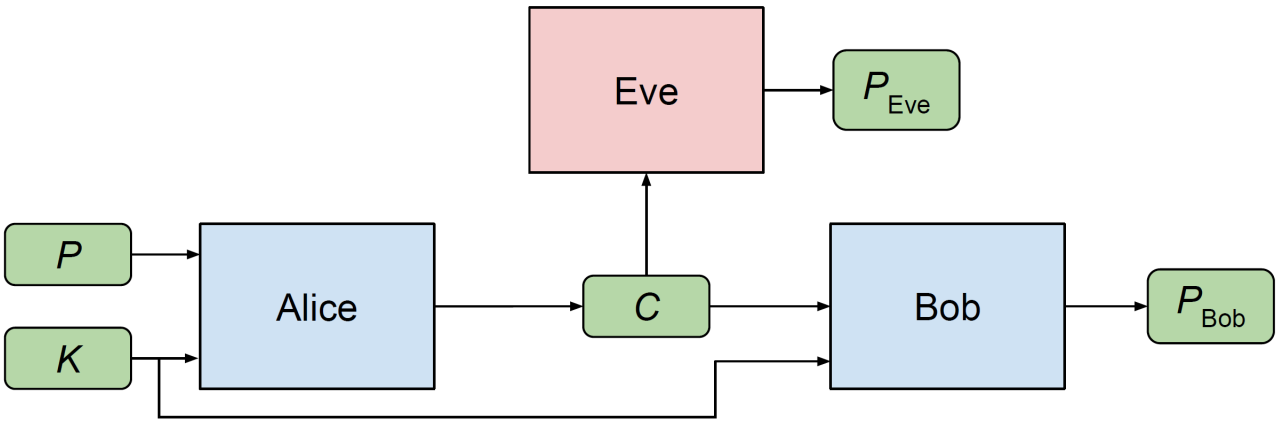
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Σχήμα 4.5: Αλγόριθμος εκπαίδευσης GAN [1]

Η καινοτομία αυτής της προσέγγισης είναι το γεγονός ότι ο G εκπαιδεύεται βάσει της εξόδου του D.

Η εφαρμογή που θα σας παρουσιάσουμε σε αυτήν την ενότητα αξιοποίησε την φιλοσοφία του adversarial training και την επέκτεινε, υπό μια έννοια, εισάγοντας στο παίγνιο έναν τρίτο παίκτη. Θυμηθείτε ότι η κρυπτογραφία αποσκοπεί στην δημιουργία ασφαλούς επικοινωνίας μεταξύ δύο χρηστών, παρουσία ενός ανεπιθύμητου τρίτου. Δηλαδή, η Alice, θέλει να επικοινωνήσει ένα κρυφό μήνυμα P (plaintext) στον Bob, χωρίς να αποκαλύψει οποιαδήποτε πληροφορία για τη δομή και το περιεχόμενο του στην Eve.



Σχήμα 4.6: Επικοινωνία Alice, Bob παρουσία Eve αξιοποιώντας συμμετρική κρυπτογραφία [4]

Η ιδέα της εφαρμογής είναι η αντικατάσταση των Alice, Bob, Eve με Νευρωνικά Δίκτυα στα οποία θα εφαρμοστεί adversarial training. Σε αυτό, οι Alice, Bob συνεργάζονται ενάντια στην Eve για να εξασφαλίσουν ένα ασφαλές μοντέλο επικοινωνίας μεταξύ τους. Μετά την εκπαίδευση, το Νευρωνικό-Alice αντιστοιχεί στον αλγόριθμο κρυπτογράφησης, ενώ αυτό του Bob στην αποκρυπτογράφηση. Η Eve θέλει να ανακατασκευάσει το P, δηλαδή να ελαχιστοποιήσει την απόσταση μεταξύ του P_{eve} και του P. Οι Alice και Bob αποσκοπούν στην διαυγή μεταξύ τους επικοινωνία, επομένως, στον περιορισμό της διαφοράς μεταξύ του P_{Bob} και P.

Έστω $A(\theta_A, P, K), B(\theta_B, C, K), E(\theta_C, C)$ οι έξοδοι των νευρωνικών δικτύων που αναλογούν στους Alice, Bob, Eve αντίστοιχα. Θεωρούμε ότι τα P, K απαρτίζονται από bits, δηλαδή 0 ή 1. Παρόλα αυτά, προτίμησαν, βασισμένοι σε πειραματικά αποτελέσματα, να θεωρήσουν το πρόβλημα στο $[-1, 1]$ όπου πλέον $0 \leftrightarrow -1$. Ως απόσταση μεταξύ των plaintext μεγέθους N προτιμήθηκε η L_1, όπου $d(P, P') = \sum_{i=1}^{N} |P_i - P'_i|$. Η συνάρτηση σφάλματος ανά στιγμιότυπο για την Eve ορίστηκε τελείως φυσικά ως:

$$L_E(\theta_A, \theta_E, P, K) = d(P, E(\theta_E, A(\theta_A, P, K)))$$

ενώ το κόστος πάνω στην κατανομή των P, K:

$$L_E(\theta_a, \theta_E) = E_{P,K}[d(P, E(\theta_E, A(\theta_A, P, K)))]$$

Εντελώς ανάλογα για τον Bob:

$$L_B(\theta_A, \theta_B, P, K) = d(P, B(\theta_B, A(\theta_A, P, K), K))$$

$$L_B(\theta_A, \theta_B) = E_{P,K}[d(P, B(\theta_B, A(\theta_A, P, K), K))]$$

Αν $O_E(\theta_A) = argmin_{\theta_E}(L_E(\theta_A, \theta_E))$, τότε όρισαν ως από κοινού σφάλμα επικοινωνίας των Alice, Bob:

$$L_{AB}(\theta_A, \theta_B) = L_B(\theta_A, \theta_B) - L_E(\theta_A, O_E(\theta_A))$$

Παρατηρήστε ότι η ελαχιστοποίηση της παραπάνω παράστασης ζητά (λόγω προσήμων) ευστοχία του B και αστοχία της E. Τελικά $(O_A, O_B) = argmin_{(\theta_A, \theta_B)}(L_{AB}(\theta_A, \theta_B))$.

Οι μέσες τιμές (που εμφανίζονται στα $L_E(\theta_A, \theta_B), L_B(\theta_A, \theta_B)$) εκτιμήθηκαν με την μέση τιμή σφάλματος πάνω σε mini-batch. Όπως και στον αλγόριθμο που είδαμε στο σχήμα 4.5, η εκπαίδευση γίνεται παράλληλα σε γύρους. Αρχικά, οι τιμές των παραμέτρων αρχικοποιούνται τυχαία. Έπειτα, για την δεδομένη «περιορισμένη» Eve, οι Alice, Bob υπολογίζουν την κατάλληλη τιμή των παραμέτρων τους (O_A, O_B) για να εξασφαλίσουν ασφαλή μυστική επικοινωνία. Στη συνέχεια, έρχεται η σειρά της Eve να ελαχιστοποιήσει το σφάλμα της $L_E(O_A, \theta_E)$. Αυτή η διαδικασία επαναλαμβάνεται, μέχρι να επέλθει ισορροπία στο παίγνιο.

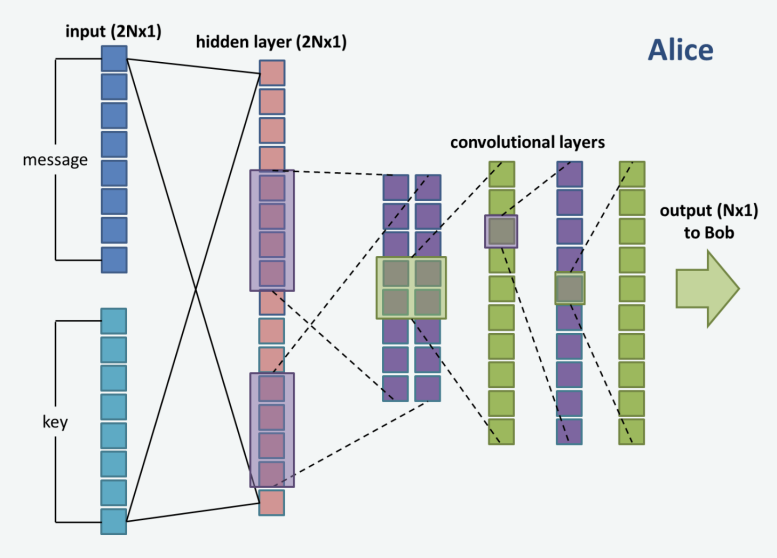
Ένας παρατηρητικός αναγνώστης, πιθανώς να εντόπισε μια αδυναμία σε αυτήν την προσέγγιση. Πιο συγκεκριμένα, στο L_{AB} δε θα θέλαμε η ελαχιστοποίηση του σφάλματος να βασίζεται

στην μεγιστοποίηση του L_E (σφάλμα ανακατασκευής). Αν οι Alice, Bob βασίζονται σε αυτή την ενέργεια, τότε η Eve μπορεί να ελαχιστοποιήσει το σφάλμα της, απλά αντιστρέφοντας κάθε έξοδο της ($0 \leftrightarrow 1$). Στην πραγματικότητα, αυτό που επιδιώκουμε είναι η πρόβλεψη της Eve να είναι δυσδιάκριτη από ένα τυχαίο μάντεμα. Έτσι, το L_E στην έκφραση L_{AB} αντικαταστήθηκε με την έκφραση:

$$\frac{(\frac{N}{2} - L_E)^2}{(\frac{N}{2})^2}$$

Παρατηρήστε ότι η παραπάνω έκφραση ελαχιστοποιείται όταν η Eve πετυχαίνει τα μισά bits που ισοδυναμεί με την αναμενόμενη ευστοχία ενός τυχαίου μαντέματος. Η τετραγωνική μορφή προτιμήθηκε για να δώσει μεγαλύτερη έμφαση στο σφάλμα (έναντι του L_B που συνυπάρχει στην έκφραση του L_{AB}) όταν η Eve αποκλίνει αισθητά από το τυχαίο μάντεμα. Τα L_E, L_B προκύπτουν από τους ορισμούς που δώσαμε προηγουμένως θεωρώντας για μετρική d την L_1 απόσταση.

Η αρχιτεκτονική των Νευρωνικών Δικτύων συνίσταται από fully connected και convolutional layers. Η Alice συνδυάζει τα N bits των P, K σε ένα διάνυσμα μεγέθους $2N$ που απαρτίζεται αποκλειστικά από τιμές $\{-1, 1\}$ που διοχετεύονται στο input layer. Στη συνέχεια έχουμε ένα FC στρώμα, ακολουθούμενο από 3 μονοδιάστατα convolutional στρώματα. Ως συνάρτηση ενεργοποίησης στα hidden layers προτιμήθηκε η sigmoid, ενώ στο output layer χρησιμοποιήθηκε η $tanh$ (ανατρέξτε ότι θέλουμε η έξοδος να βρίσκεται στο $[-1, 1]$). Τα νευρωνικά των Bob, Eve είναι πανομοιότυπα με μόνη διαφορά το input layer της Eve το οποίο δέχεται μόνο ένα διάνυσμα μήκους N που αντιστοιχεί στο κρυπτοκείμενο C.

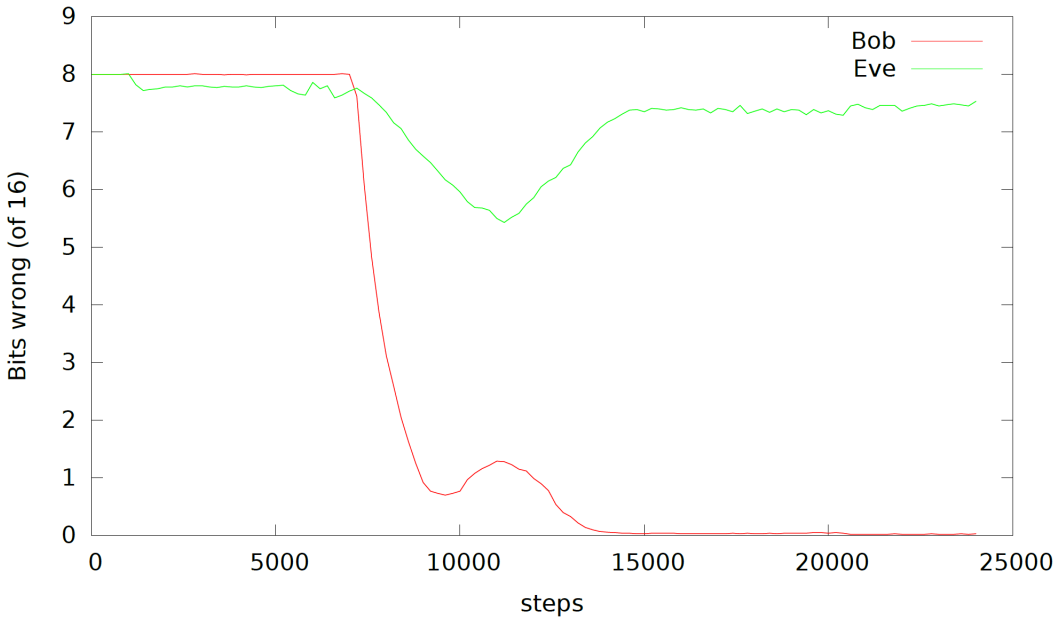


Σχήμα 4.7: Αναπαράσταση Νευρωνικού Δικτύου Alice [78]

Μολονότι δεν εστιάζουν πολύ στην αρχιτεκτονική, θεωρούμε ότι η υλοποίηση που επιχε-

ίρησαν παρουσιάζει κάποιο ενδιαφέρον, αφού φαίνεται να ενισχύει την πιθανότητα εμφάνισης δυο επιθυμητών ιδιοτήτων που συνηθίζονται στις συναρτήσεις κρυπτογράφησης: της διάχυσης (diffusion) και της σύγχυσης (confusion). Το FC σε συνδυασμό με τα convolutional layers δημιουργούν μια σύνθετη σχέση εισόδου-εξόδου επιτρέποντας σε κάθε bit εισόδου να επηρεάζει πολλά bits εξόδου (diffusion), ενώ παράλληλα κάθε bit εξόδου επηρεάζεται από πολλά bits του κλειδιού.

Η εκμάθηση χρησιμοποίησε mini-batches μεγέθους $256 - 4096$, ενώ ως αλγόριθμος βελτιστοποίησης προτιμήθηκε ο adam. Κάθε βήμα εκπαίδευσης των Alice, Bob ακολουθήθηκε από δυο της Eve, με το σκεπτικό ότι το πρόβλημα που καλείται να λύσει είναι αρκετά πιο σύνθετο (δεν έχει επίγνωση του κλειδιού). Για κριτήριο τερματισμού προτιμήθηκε η ευστοχία, επιδιώκοντας τα L_1 σφάλματα των Alice, Bob να μην υπερβαίνουν την τιμή 0.05, ενώ παράλληλα η επίδοση της Eve να μην αποκλίνει πάνω από 2 bits από το τυχαίο μάντεμα.



Σχήμα 4.8: Μέσο σφάλμα ανακατασκευής των Alice, Bob, Eve συναρτήσει των βημάτων εκπαίδευσης για $N = 16$ [4]

Στο παραπάνω επιτυχές πείραμα, βλέπουμε ότι μετά από 15000 βήματα παίρνουμε το επιθυμητό κριτήριο τερματισμού. Από τα 20 πειράματα που διεξήχθησαν, τα 14 στέφθηκαν με επιτυχία. Μια πιθανή εξήγηση για αυτό το φαινόμενο είναι η μορφή της συνάρτησης που καλούμαστε να ελαχιστοποιήσουμε. Ένα από τα κρισιμότερα ζητήματα του Deep Learning είναι η μαθηματική θεμελίωση της ορθότητας των αλγορίθμων βελτιστοποίησης που χρησιμοποιούμε. Οι συναρτήσεις που μελετάμε είναι ιδιαίτερα σύνθετες, μη κυρτές συναρτήσεις με πολλά τοπικά ακρότατα. Εμπειρικά, σε απλά Νευρωνικά Δίκτυα, αυτά τα τοπικά ακρότατα φαίνεται να

προσδίδουν ικανοποιητικά μικρές τιμές στην αντικειμενική μας συνάρτηση κόστους ([14]). Αντιθέτως, οι συναρτήσεις που προκύπτουν σε adversarial training δεν παρουσιάζουν ανάλογες ιδιότητες με αποτέλεσμα η εκπαίδευση να είναι λιγότερα ευσταθής ([79]).

Ιδιαίτερο ενδιαφέρον παρουσιάζει το γεγονός ότι η αρχιτεκτονική παρουσίασε ανάλογα αποτελέσματα για $N = 32$ και $N = 64$, τα οποία αποτελούν πιο ρεαλιστικά μεγέθη κλειδιών για κρυπτογραφική χρήση. Επιπλέον, το C εξαρτάται, όπως και οφείλει, τόσο από το K όσο και από το P, με την τροποποίηση ενός μόνο bit του K να επιφέρει αλλαγές σε $3 - 6/16$ bits του C.

Στο δεύτερο μισό της εφαρμογής, ορμώμενοι από τα θετικά αποτελέσματα που σας παρουσιάσαμε προηγουμένως, μελετήθηκε η δυνατότητα επιλεκτικής κρυπτογράφησης (selective protection), στην οποία δημοσιοποιούμε πληροφορία που σχετίζεται με το αρχικό μας μήνυμα P, με τρόπο που δεν παραβιάζει την συνολική του μυστική ακεραιότητα. Σε αυτό το σενάριο, μπορούμε να θεωρήσουμε ότι το P αποτελείται από πολλές τιμές $P =< x_1, ..., x_n >$ οι οποίες είναι συσχετισμένες μεταξύ τους. Υποθέστε λοιπόν, ότι θέλουμε να δημοσιοποιήσουμε την x_1. Μια πρώτη φυσιολογική προσέγγιση είναι να γνωστοποιηθεί το $\{x_1, c_2, ..., c_n\}$, όπου c_i είναι κάποια κατάλληλη κρυπτογράφηση του x_i. Το πρόβλημα με αυτήν την σκέψη είναι ότι η x_1 προσδίδει πληροφορία για τα υπόλοιπα x_i (λόγω της μεταξύ τους συσχέτισης).

Τέτοιου είδους εφαρμογές εμφανίζονται συχνά όταν θέλουμε να βγάλουμε κάποιο συμπέρασμα για ένα υποσύνολο του $D \subseteq P =< x_1, ..., x_n >$, διατηρώντας την μυστικότητα του $P \setminus D$. Για παράδειγμα, μπορεί να έχουμε στην διάθεση μας δυάδες $< A, B >$, όπου A :όνομα εταιρίας και B : τζίρος της. Μπορεί να θέλει ένας συνεργάτης μας να υπολογίσει τον συνολικό τζίρο όλων των εταιριών που έχουμε στην βάση δεδομένων μας, χωρίς όμως να μάθει την αντιστοιχία τζίρου-εταιρίας. Παρατηρήστε ότι η αφελής προσέγγιση που παρουσιάσαμε παραπάνω θα αποτύγχανε, αν ο συνεργάτης μας γνώριζε κάποια εκ των προτέρων διάταξη μεταξύ των εταιριών ως προς το κέρδος τους (π.χ. το μεγαλύτερο κέρδος ανήκε στην Google). Παραδοσιακά στην κρυπτογραφία, τέτοια προβλήματα αντιμετωπίζονται με ομομορφική κρυπτογραφία, όπου οι πράξεις μεταξύ των κρυπτογραφημένων c_i μεταφράζονται με φυσικό τρόπο στις αντίστοιχες μεταξύ των x_i, δηλαδή $E(k, P_1 *_M P_2) = E(k, P_1) *_C E(k, P_2)$. Η εφαρμογή διερευνά μια εναλλακτική προσέγγιση βασιζόμενη σε adversarial training Νευρωνικών Δικτύων.

Στο πείραμα που διεξήγαν θεώρησαν $P =< A, B, E, D >$, όπου τα A, B, E, D είναι κατάλληλα συσχετισμένα. Σκοπός τους είναι η εκπαίδευση ενός συστήματος το οποίο θα παράγει, δοθέντος των $< A, B, E >$ δυο τιμές για το D: D_{true} και D_{public}. Θυμηθείτε την τριάδα Alice, Bob, Eve. Η Alice, εφοδιασμένη με ένα μυστικό κλειδί K, τροφοδοτείται με την τριάδα $< A, B, E >$ και παράγει ένα κρυπτοκείμενο C (όπως προηγουμένως) και το D_{public}, το οποίο αποτελεί μια ασφαλή δημόσια μορφή του D που δεν προσδίδει ουσιαστική πληροφορία για το E. Με είσοδο τα C, D_{public}, η Eve προσπαθεί να ανακατασκευάσει το E, ενώ ο Bob (χρησιμοποι-

ώντας το κοινό κλειδί K) το D_{true}, το οποίο αποτελεί μια όσο το δυνατόν καλύτερη προσέγγιση του αρχικού D της πλειάδας $< A, B, E, D >$. Αποσκοπούμε, δηλαδή, στην δημιουργία κατάλληλου D_{public} από το οποίο μπορεί να ανακατασκευαστεί επαρκώς το αρχικό D, χωρίς όμως να παραβιάσουμε την μυστικότητα του E.

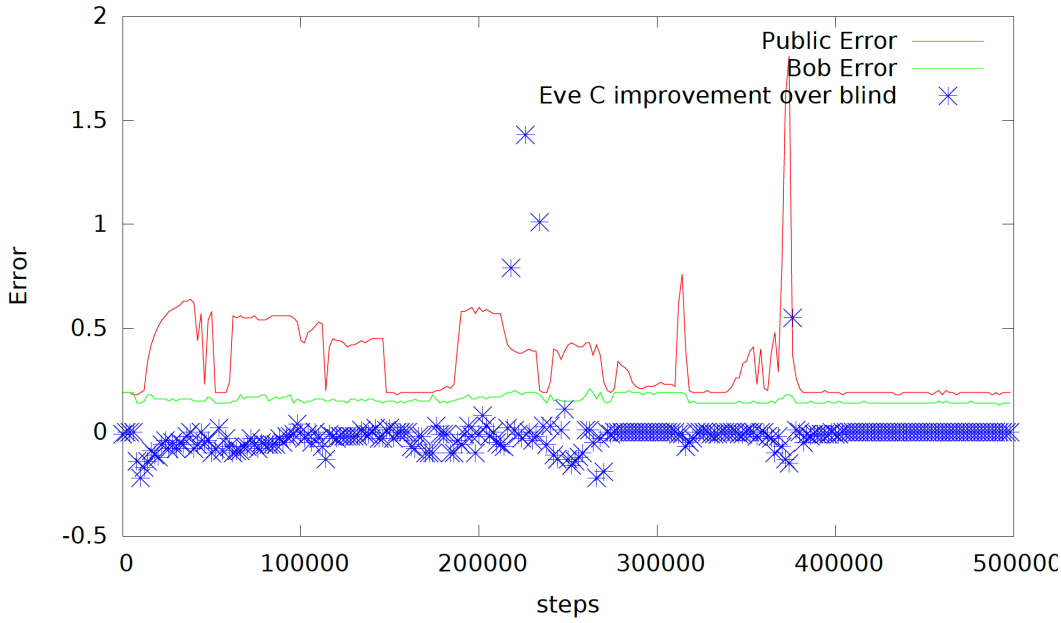
Η αρχιτεκτονική διαφοροποιείται μόνο στην αρχή του κάθε δικτύου με την προσθήκη ενός επιπλέον 12×12 FC στρώματος. Η είσοδος απαρτίζεται, πλέον, από 12 νευρώνες: 8 για τα bits του κλειδιού και 4 για τις τιμές. Συναρτήσεις κόστους θεωρήθηκαν οι:

$$L_{Eve} = MSE(E_{Eve}, E)$$

$$L_{Alice-Bob} = MSE(D, D_{true}) + MSE(D, D_{public}) + |Cov(E_{eve}, E)|$$

όπου τα $D, E, D_{true}, D_{public}$ είναι διανύσματα (batch training). Η φιλοσοφία είναι ότι οι Alice, Bob θέλουν να τροποποιήσουν το D κατάλληλα (να μην αποκλίνει αισθητά από το πραγματικό D), χωρίς όμως να επιτρέψουν στην Eve να εντοπίσει E_{Eve} ισχυρά συσχετισμένο με την πραγματική τιμή E.

Για την μελέτη του συστήματος εισήχθη ένας επιπλέον εκτιμητής, η «τυφλή» Eve, ο οποίος προσπαθεί να ανακατασκευάσει το E γνωρίζοντας μόνο την κατανομή του, αγνοώντας το D_{public} και το C. Αν η Eve έχει αισθητό πλεονέκτημα (μικρότερο σφάλμα ανακατασκευής) έναντι αυτού του εκτιμητή, τότε τα D_{public}, C δεν αποκρύπτουν επαρκώς την πληροφορία για το E.



Σχήμα 4.9: Γραφική παράσταση σφαλμάτων συναρτήσει βημάτων εκπαίδευσης [4]

Παρατηρήστε ότι το πείραμα παρουσιάζει λογικά και επιθυμητά αποτελέσματα. Όπως είναι αναμενόμενο, ούσα δημόσια πληροφορία, το D_{public} φαίνεται να υποχρεούται (για να μην διευ-

κολύνει την Eve) να αποκλίνει περισσότερο από ότι το D_{true}. Έπειτα, παρατηρούμε ότι η Eve παρουσιάζει σχεδόν μηδενικό προβάδισμα έναντι της «τυφλής», υποδηλώνοντας ότι τα C, D_{public} δε της προσφέρουν κάποιο πλεονέκτημα.

Θεωρούμε ότι η παρούσα εφαρμογή παρουσιάζει δυο ενδιαφέρουσες και καινοτόμες οπτικές για την αξιοποίηση μοντέλων Μηχανικής Μάθησης στην Κρυπτογραφία. Στο πρώτο σκέλος της, εστιάζει στην υποκατάσταση του παραδοσιακού συμμετρικού κρυπτοσυστήματος με δυο Νευρωνικά Δίκτυα (Alice και Bob), ενώ στο δεύτερο προτείνει μια εναλλακτική, έναντι της ομομορφικής κρυπτογραφίας, για το πρόβλημα της επίλεκτης κρυπτογραφίας. Λόγω της πολυπλοκότητας των κατανομών που κλήθηκαν να προσεγγίσουν, μοντελοποίησαν το πρόβλημα τους ως ένα κατάλληλο παίγνιο και εφήρμοσαν adversarial training. Μολονότι τα αποτελέσματα είναι ενθαρρυντικά, οφείλουμε να ξεκαθαρίσουμε ότι τα θετικά πειραματικά αποτελέσματα δεν επαρκούν ως απόδειξη ασφάλειας. Η προστασία των Alice, Bob απέναντι στο Νευρωνικό Δίκτυο της Eve (με την συγκεκριμένη αρχιτεκτονική) δεν εξασφαλίζει την αντοχή τους έναντι πιο σύνθετων επιθέσεων και αλγορίθμων. Ανακαλέστε, ότι στον ορισμό που δώσαμε στην εισαγωγή, θέλουμε ο αντίπαλος μας είναι PPT. Επομένως, θα χρειαζόμασταν κάποιου είδους απόδειξη ότι η Eve εμφανίζει επαρκές capacity για να εμπεριέχει όλη αυτήν την κλάση αλγορίθμων.

Παρόλα αυτά, πιστεύουμε ότι η εφαρμογή που σας παρουσιάσαμε εμφανίζει ενδιαφέρον από την σκοπιά της Κρυπτογραφίας και της Μηχανικής Μάθησης. Τα πειράματα που διεξήγαν οι [4] αποτελούν μια ισχυρή ένδειξη δυσκολίας και πολυπλοκότητας του κρυπτοσυστήματος. Στην πραγματικότητα, πολλά κρυπτοσυστήματα (π.χ. AES) χρησιμοποιούνται ευρέως, χωρίς να έχουν κάποια μαθηματικά θεμελιωμένη απόδειξη ασφάλειας. Έπειτα, παρουσιάζει ιδιαίτερο ενδιαφέρον το εγχείρημα γενίκευσης αυτής της ιδέας και για ασύμμετρα κρυπτοσυστήματα. Ανάλογο ενδιαφέρον εμφανίζει η αρχιτεκτονική και η γενίκευση του adversarial training ως παίγνιο μεταξύ τριών ατόμων. Η μετατροπή του Discriminator από ταξινομητή σε regressor πραγματοποιείται με ιδιαίτερα κομψό τρόπο και θεωρούμε ότι ο πειραματισμός αυτού του μετασχηματισμού και της επιλογής κατάλληλων συναρτήσεων κόστους θα αποτελέσει ενεργό πεδίο έρευνας.

4.3 Ανακεφαλαίωση

Στο παρόν κεφάλαιο μελετήσαμε την δυνατότητα υποκατάστασης ενός κρυπτοσυστήματος από ένα μοντέλο Μηχανικής Μάθησης. Στην πρώτη ενότητα μελετήσαμε και αιτιολογήσαμε μερικώς, την δυσκολία αυτού του εγχειρήματος αξιολογώντας τα αποτελέσματα μιας εφαρμογής που επιχειρούσε, σε ένα παρεμφερές πρόβλημα, έμμεσα, την προσέγγιση μιας σύνθετης κατανομής κλειδιών. Στην δεύτερη ενότητα, ορμώμενοι από το προηγούμενο αποτέλεσμα, σας παρουσι-

άσαμε μια απόπειρα προσέγγισης ενός κρυπτοσυστήματος υιοθετώντας μιας αρκετά πιο σύνθετη μορφή εκπαίδεσης, την adversarial training. Τα θετικά αποτελέσματα μοιάζουν ενθαρρυντικά για το εγχείρημα, και φαίνεται να ενισχύουν την πεποίθηση μας ότι στην προσέγγιση τέτοιων πολύπλοκων προβλημάτων αξίζει ο περαιτέρω πειραματισμός με πιο σύνθετα μοντέλα Μηχανικής Μάθησης (π.χ. GAN).

Κλείνοντας αυτό το κεφάλαιο, ολοκληρώνουμε τον κεντρικό κορμό της εργασίας μας. Οι εφαρμογές που παρουσιάστηκαν επιλέχθηκαν για την όσο το δυνατόν πληρέστερη και συνολική παρουσίαση του κλάδου. Οι αρχιτεκτονικές και οι ιδέες πίσω από τα μοντέλα που αναπτύχθηκαν, μπορούν να αξιοποιηθούν σε παρεμφερή εφαρμογές (π.χ. επιθέσεις άλλων κρυπτοσυστημάτων) από τους αναγνώστες που ενδιαφέρονται να επεκτείνουν το συγκεκριμένο επιστημονικό πεδίο.

Στο επόμενο κεφάλαιο, θα αναλυθούν μοντέλα που θα αξιοποιήσουμε σε εφαρμογές που σκοπεύουμε να αναπτύξουμε σε μελλοντικές εργασίες.

Κεφάλαιο 5

Νέες κατευθύνσεις και εφαρμογές

Κατά την συγγραφή αυτής της εργασίας, ορμώμενοι από τις πολυάριθμες εφαρμογές που μελετήσαμε και σας παρουσιάσαμε, αναλογιστήκαμε τρόπους με τους οποίους μπορούμε να επεκτείνουμε τον συγκεκριμένο κλάδο. Σε αυτό το κεφάλαιο θα σας παρουσιάσουμε τρεις εφαρμογές τις οποίες σκοπεύουμε να μελετήσουμε και να αναλύσουμε σε μελλοντικές εργασίες. Πιο συγκεκριμένα, η πρώτη εφαρμογή αφορά την χρήση GAN για την προσέγγιση γνωστών κατανομών κρυπτογράφησης. Η δεύτερη αποπειράται την αξιοποίηση παρόμοιας αρχιτεκτονικής για mining, ενώ η τρίτη αφορά την δημιουργία απόκρυφου κώδικα για χρήση στην κρυπτογραφία.

5.1 Προσέγγιση σύνθετων κατανομών και mining

Η κρυπτογραφία στεγάζει πολυάριθμες σύνθετες συναρτήσεις και κατανομές. Μια από αυτές είναι το DES, με το οποίο ασχοληθήκαμε στο δεύτερο κεφάλαιο. Μολονότι το αποτέλεσμα που είδαμε είναι ήδη αξιόλογο, καθώς κατάφερε να προσεγγίσει την κατανομή του DES για ένα δεδομένο κλειδί, εμείς επιδιώκουμε κάτι πιο σύνθετο: την προσέγγιση της κατανομής του για κάθε κλειδί. Παράλληλα, ένα ζήτημα που οφείλει περισσότερη μελέτη, είναι το capacity της αρχιτεκτονικής του περασμένου κεφαλαίου. Πιο συγκεκριμένα, αν ο συνδυασμός 1-D Convolutional και Fully Connected στρωμάτων, εκπαιδευμένα σε adversarial αρχιτεκτονική, εμφανίζουν επαρκές capacity για την προσέγγιση σύνθετων κρυπτογραφικών κατανομών.

Υπό αυτή την οπτική, επιχειρήσαμε να κατασκευάσουμε ένα GAN δύο νευρωνικών για την προσέγγιση της κατανομής του DES. Ο G υιοθετεί την αρχιτεκτονική της Alice, ενώ ο D της Eve. Ο G, με είσοδο έναν θόρυβο $z \in [0,1]^{196}$, εξωτερικεύει $G(z)$. Σκοπός του G είναι η προσέγγιση της κατανομής (k, m, c) του DES. Τα output layers των G και D αξιοποιούν σιγμοειδείς συναρτήσεις ενεργοποίησης, ώστε το σύνολο τιμών τους να είναι το $[0,1]^{192}$ και $[0,1]$ αντίστοιχα ($D(x) = P[x \in data]$). Ως συνάρτηση σφάλματος, χρησιμοποιήθηκε η binary-cross entropy. Ο D με είσοδο ένα στιγμιότυπο $x \in [0,1]^{196}$ προσπαθεί να συμπεράνει αν αυτό προέρχεται από την κατανομή του DES ή από τον G. Ένα θετικό αποτέλεσμα σε αυτό το

πείραμα θα ενίσχυε σημαντικά τα συμπεράσματα της εφαρμογής που σας παρουσιάσαμε στο προηγούμενο κεφάλαιο.

Τα πειράματα διεξήχθησαν στην γλώσσα Python, αξιοποιώντας την βιβλιοθήκη Keras (Tensorflow Backend). Για το DES χρησιμοποιήσαμε την βιβλιοθήκη Crypto.Cipher[98]. Η λειτουργία του DES ήταν σε ECB-mode (όπως και στην εφαρμογή του δευτέρου κεφαλαίου). Η αξιολόγηση του μοντέλου πραγματοποιήθηκε στρογγυλοποιώντας την έξοδο του νευρωνικού (στο κοντινότερο bit) και συγκρίνοντας το τμήμα που αντιστοιχούσε στο κρυπτοκείμενο c με το πραγματικό. Δηλαδή, αν $G(z) = (r_1, r_2, r_3)$ πρόβλεψη για τα (k, m, c) τότε στην ευστοχία υπολογίζουμε το μέσο πλήθος κοινών bits μεταξύ των $round(r_3)$ και $c = E_{DES}(round(r_1), round(r_2))$.

Λόγω των ανεπιτυχών πειραμάτων (ευστοχία $\sim$ 50%), τροποποιήσαμε τις συναρτήσεις ενεργοποίησης. Έτσι, αντικαταστήσαμε σταδιακά τις sigmoid των ενδιάμεσων στρωμάτων με $ReLU$ και παραλλαγές της, όπως η soft ReLU, όπου

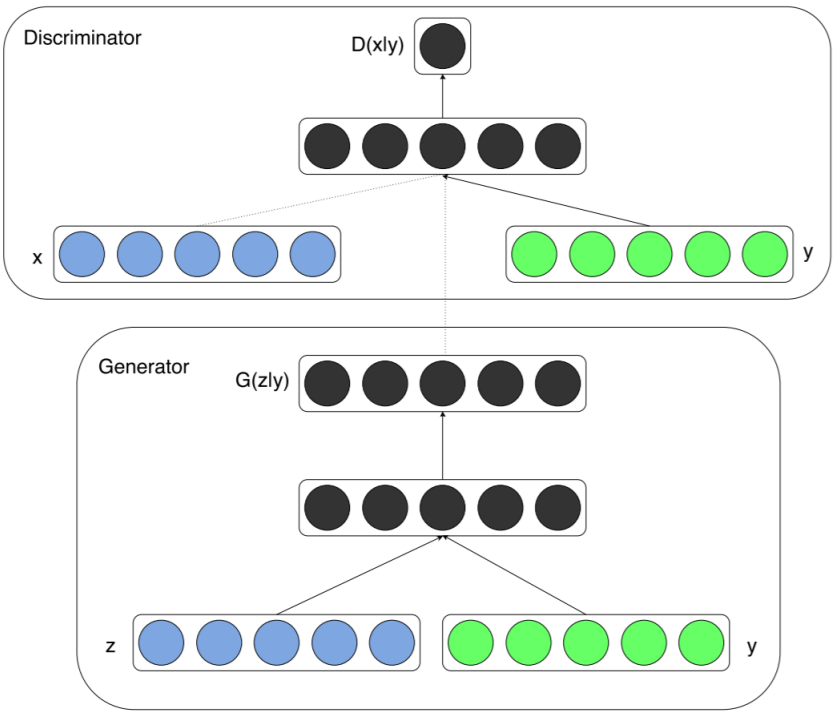
$$softReLU = \begin{cases} x & \text{αν } x > 0 \\ ax & \text{διαφορετικά} \end{cases}$$

με a κατάλληλη μεταπαράμετρος. Η SoftReLU εμφανίζει παρόμοια χαρακτηριστικά με την $ReLU$ και έχει παρουσιάσει αξιόλογα αποτελέσματα τα τελευταία χρόνια. Η επιτυχία αυτή οφείλεται στην απλή της μορφή, η οποία όμως συγκρατεί περισσότερη πληροφορία για αρνητικές τιμές του x απ΄ ότι η κανονική $ReLU$.

Για την αποφυγή overfitting αξιοποιήσαμε τεχνικές dropout. Το dropout έχει παρουσιάσει εντυπωσιακά πειραματικά αποτελέσματα και η λειτουργία του είναι αρκετά απλή. Όταν εφαρμόζουμε dropout σε ένα στρώμα, τότε δίνουμε σε κάθε του νευρώνα μια πιθανότητα p (το οποίο παρέχεται ως παράμετρος) να μην λειτουργήσει. Επομένως, για κάθε είσοδο $x \in$ train-set, εκπαιδεύεται ένα υποσύνολο του νευρωνικού (που αποτελείται από τους νευρώνες που δεν έγιναν dropout). Το κάθε στρώμα είναι πλέον πιο ανεξάρτητο, καθώς αυτή η τυχαιότητα δεν το επιτρέπει να βασίζεται εξ΄ ολοκλήρου στο προηγούμενο ή επόμενο στρώμα. Στην πραγματικότητα αποτελεί ένα είδος bagging, καθώς η τελική απόφαση στο στάδιο αξιολόγησης στηρίζεται στις επιμέρους εξόδους των νευρωνικών-υποσύνολα (τα οποία προέκυψαν λόγων του dropout) που εκπαιδεύτηκαν παράλληλα.

Σκοπεύουμε να συνεχίσουμε τα πειράματα μας μέχρι να βρούμε ικανοποιητική ακρίβεια. Επειδή γνωρίζουμε την δυσκολία του εγχειρήματος μας, σε πρώτο στάδιο ευελπιστούμε να πετύχουμε ακρίβεια που αποκλίνει $1 - 2\%$ από το τυχαίο μάντεμα (50%) σε ένα υποσύνολο $K' \subseteq K$ των κλειδιών. Επιπλέον, επιδιώκουμε την επέκταση της αρχιτεκτονικής και μετατροπής του GAN σε Conditional Generative Adverserial Network [99]. Τα cGAN ενσωματώνουν στην παραδοσιακή GAN αρχιτεκτονική ένα επιπλέον διάνυσμα y στην είσοδο των G και D, το

οποίο χαρακτηρίζει την επιθυμητή έξοδο του G. Για παράδειγμα, ας υποθέσουμε ότι επιθυμούμε την παραγωγή εικόνων ψηφίων από 0 έως 9. Αν υιοθετήσουμε την παραδοσιακή GAN αρχιτεκτονική, τότε ο G με είσοδο έναν τυχαίο θόρυβο z παράγει μια εικόνα $G(z)$. Αν θέλουμε την παραγωγή εικόνας ενός συγκεκριμένου ψηφίου, τότε πρέπει να το εισάγουμε ως είσοδο στο G, μέσω ενός διανύσματος y (π.χ. $y \in [0,1]^9$). Έτσι, το $G(z|e_1)$ παράγει μια τυχαία εικόνα του 1. Η αρχιτεκτονική συνοψίζεται στην παρακάτω εικόνα:



Σχήμα 5.1: Αρχιτεκτονική cGAN [99]

Το αντίστοιχο παίγνιο, μοντελοποιείται μέσω του παρακάτω min-max:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_G(z)}[\log(1 - D(G(z|y)))]$$

Στην εφαρμογή μας θα ενσωματώσουμε την παραπάνω αρχιτεκτονική, απομονώνοντας την κατανομή για ένα συγκεκριμένο κλειδί ($y = k$).

Η δεύτερη εφαρμογή με την οποία θα ασχοληθούμε σε μελλοντική εργασία είναι η εκπαίδευση νευρωνικού δικτύου με στόχο το mining σε κρυπτονομίσματα. Στα κρυπτονομίσματα οι συναλλαγές καταγράφονται σε blocks. Όπως στα παραδοσιακά χρηματικά συστήματα συναλλαγής, η αξία των χρημάτων γεννάται από τον φόρτο εργασίας που καλείσαι να εκτελέσεις για να τα αποκτήσεις, έτσι και στα κρυπτονομίσματα, η αξία τους προέρχεται από την δυσκολία δημιουργίας τέτοιων έγκυρων block. Εμείς εστιάζουμε στο δυσεπίλυτο πρόβλημα που εμφανίζεται

στην δημιουργία ενός block στο Bitcoin [2], το οποίο οφείλει να ενσωματώνεται κατάλληλα στην έως τώρα block chain αλυσίδα.

Το Proof of Work πρόβλημα απαιτεί την εύρεση του κατάλληλου αλατιού (nonce), ώστε:

$$sha256^2(id||nonce) < 2^{256-T}$$

Το id αντιστοιχεί στην hash αναπαράσταση του προηγούμενου block, ενώ το T αναπαριστά τον δείκτη δυσκολίας, καθώς όσο μεγαλύτερη είναι η τιμή του, τόσο δυσκολότερο γίνεται το πρόβλημα. Λόγω της σύνθετης δομής της $sha256^2$ η παραπάνω αναζήτηση επιτυγχάνεται τυχαία. Επομένως, για την επίλυση του, οι miners δοκιμάζουν τυχαίες τιμές του nonce μέχρι να εντοπίσουν τιμή της $sha256^2$ που εμπεριέχεται στο ζητούμενο διάστημα. Αν θεωρήσουμε την δυαδική αναπαράσταση του $sha256^2$, τότε το παραπάνω πρόβλημα συνοψίζεται στον εντοπισμό μιας δυαδικής ακολουθίας που απαρτίζεται από T διαδοχικά μηδενικά στην αρχή. Συνεπώς, αν υποθέσουμε ότι $P[digit_i = 0] = P[digit_i = 1] = \frac{1}{2}$, τότε $P[\cap_{i=1}^{T}\{digits_i = 0\}] = \frac{1}{2^T}$ και επομένως το αναμενόμενο πλήθος δοκιμών μέχρι την επιτυχή εύρεση είναι $E[\cap_{i=1}^{T}\{digits_i = 0\}] = 2^T$.

Για την προσέγγιση του παραπάνω προβλήματος υπό την οπτική της Μηχανικής Μάθησης, επιθυμούμε να εκπαιδεύσουμε ένα Νευρωνικό Δίκτυο για την προσέγγιση της κατανομής των nonce. Σε πρώτο στάδιο, θα θεωρήσουμε την απλοποιημένη εκδοχή του, όπου το id αγνοείται, $T = 1$ και η δυαδική αναπαράσταση του nonce βρίσκεται σε συμπαγές χωρίο. Δηλαδή:

$$sha256^2(nonce) < 2^{255} \ , |nonce| \leq q$$

Θα υιοθετήσουμε adversarial αρχιτεκτονική, όμοια με αυτή της προηγούμενης εφαρμογής. Ο G με είσοδο έναν τυχαίο θόρυβο στο $[0,1]^q$ θα εξωτερικεύει ένα διάνυσμα στο $[0,1]^q$ το οποίο θα αντιστοιχεί (μετά από στρογγυλοποίηση) στην δυαδική αναπαράσταση του nonce. Ο D καλείται να εντοπίσει αν το στιγμιότυπο που δέχεται ως είσοδο προέρχεται από τον G ή από την κατανομή των nonce που επιλύουν το παραπάνω πρόβλημα. Για την παραγωγή του δείγματος, θα αξιοποιήσουμε το εργαλείο Cryptii [100]. Μετά την εκπαίδευση, επιδιώκουμε $P[sha256^2(round(G(z))) < 2^{255}] > \frac{1}{2}$, παρουσιάζοντας, έτσι, προβάδισμα έναντι του τυχαίου μαντέματος. Εφόσον λάβουμε θετικά αποτελέσματα, μπορούμε να επεκτείνουμε το προαναφερθέν GAN για να συμπεριλάβει το id και την δυσκολία, θεωρώντας το αντίστοιχο cGAN με $y = (id, T)$. Πλέον, στοχεύουμε $P[sha256^2(id||round(G(z|(id,T)))) < 2^{255-T}] > \frac{1}{2^T}$.

Εμπνευστήκαμε την πρώτη εφαρμογή, ορμώμενοι από τα αποτελέσματα της επίθεσης που αναλύσαμε στο δεύτερο κεφάλαιο, του adversarial μοντέλου του τετάρτου κεφαλαίου και την δυνατότητα γραμμικής προσέγγισης της κατανομής (για δεδομένο κλειδί), όπως πραγματοποιείται στο linear cryptanalysis. Η δεύτερη εφαρμογή είναι αρκετά πιο σύνθετη, καθώς η sha256 είναι μια πολύπλοκη κρυπτογραφική συνάρτηση σύνοψης η οποία χρησιμοποιείται εκτενώς, ακόμα και σήμερα. Για τον λόγο αυτό, δεν ευελπιστούμε να επιλύσουμε το Proof of Work

πρόβλημα γρήγορα, αλλά αρκούμαστε στην οσοδήποτε μικρή, στατιστικά σημαντική βελτίωση στον μέσο αριθμό κλήσεων της συνάρτησης. Ενδεικτικά, βελτίωση της τάξης του 1% στο c του DES και στο αναμενόμενο πλήθος αναζητήσεων του nonce θα ήταν επαναστατικό.

Οι επόμενες δυο ενότητες θα ασχοληθούν με την παραγωγή απόκρυφου κώδικα (obfuscated code). Στην πρώτη ενότητα, θα σας παρουσιάσουμε την θεωρητική θεμελίωση της χρήσης απόκρυφου κώδικα στην Κρυπτογραφία, όπως την ανέλυσαν οι [81]. Στην δεύτερη, θα σας αναλύσουμε ένα μοντέλο RNN το οποίο σκοπεύουμε να αξιοποιήσουμε για την παραγωγή απόκρυφου κώδικα.

5.2 Code obfuscation στην Κρυπτογραφία

Το code obfuscation ασχολείται με την παραγωγή κώδικα από τον οποίο είναι αδύνατη η εξόρυξη οποιασδήποτε πληροφορίας για την λειτουργία και σημασιολογία του και αποτελεί ενεργό κλάδο της Επιστήμης Υπολογιστών. Πιο συγκεκριμένα, η απόκρυψη (obfuscation) O ενός προγράμματος P οφείλει να πληροί δυο λειτουργίες:

1. Αξιοποιώντας το O, μπορούμε να υπολογίσουμε την τιμή του προγράμματος P
2. Το O δεν προσφέρει περισσότερη πληροφορία από όση παρέχει ένα μαντείο πάνω στο P

Στην πραγματικότητα, το O αποτελεί ένα μαντείο της P (μας δίνει απλά την δυνατότητα υπολογισμού του προγράμματος για διάφορες τιμές εισόδου). Έτσι, μολονότι το O είναι αισθητά διαφορετικό (οπτικά και δομικά) από το αρχικό πρόγραμμα P, η λειτουργία του παραμένει ίδια.

```
1  var a = "ale";
2  a += "rt(";
3  a += "'hello'";
4  a += ");";
5  eval(a);
```

Σχήμα 5.2: Απλό παράδειγμα code obfuscation [80]

Η απόκρυψη ενός προγράμματος (ή γενικότερα μιας συνάρτησης F) έχει μελετηθεί εκτενώς στην βιβλιογραφία, γεννώντας μια πληθώρα διαφορετικών ορισμών. Η διαφοροποίηση υπόκειται στην υιοθέτηση ισχυρότερων ή ασθενέστερων υποθέσεων για την λειτουργία του O, ανάλογα με τον σκοπό που οφείλει να επιτελέσει. Θεωρείστε ότι θέλουμε να αποκρύψουμε μια κλάση συναρτήσεων $F = \{F^n\}_{n\in\mathbb{N}}$, όπου $F^n = \{F_k\}_{k\in K_n}$. Τότε, στο μοντέλο μας, ο αλγόριθμος μας αποτελεί μια δυάδα PPT αλγορίθμων (O, G), όπου το O αποκρύπτει την ευαίσθητη πληροφορία (π.χ. κλειδί), ενώ το G δρα ως μαντείο πάνω στο O. Πιο συγκεκριμένα, αν υποθέσουμε ότι κάθε συνάρτηση F^n παίρνει ως είσοδο ένα $k \in K_n$ και ένα μήνυμα $x \in X_n$, τότε θεωρούμε O :

$\cup_n K_n \longrightarrow \cup_n K'_n$, όπου K'_n χώρος μετασχηματισμένων (απόκρυφων) κλειδιών, ενώ $G_{O(k)}(x) = F_k(x)$.

Μια πρώτη προσέγγιση, όπως παρουσιάζεται από τους [81], είναι αυτή της οικουμενικής απόκρυψης (universal obfuscation):

Ορισμός 4. Ένας PPT αλγόριθμος O αποτελεί universal obfuscator μιας κλάσης ντετερμινιστικών συναρτήσεων F αν ισχύουν τα ακόλουθα

- Προσεγγιστική λειτουργία (Approximate Functionality): Για κάθε $n \in \mathbb{N}$, $k \in K_n$, $x \in X_n$ $F_k(x) \neq G_{O(k)}(x)$ με αμελητέα πιθανότητα.
- Virtual Black-Box: Ένας αντίπαλος, έχοντας επίγνωση του $O(k)$ μιας F_k, αποκτά όση πληροφορία θα του παρείχε η χρήση ενός μαντείου πάνω στην F_k.

Στην πραγματικότητα, η δεύτερη συνθήκη μπορεί να αναλυθεί περαιτέρω:

Ορισμός 5. Simulation-Based Virtual Black-Box property: Μια συνάρτηση απόκρυψης O πάνω στην κλάση συναρτήσεων F έχει την Simulation-Based Virtual Black-Box ιδιότητα αν και μόνο αν για κάθε PPT αλγόριθμο D (distinguisher), υπάρχει ένας αλγόριθμος S (Simulator) ώστε

$$|P_{k\in K_n}[D^{F_k}(1^n, O(k)) = 1] - P_{k\in K_n}[D^{F_k}(1^n, S^{F_k}(1^k)) = 1]| \leq negl(n)$$

Ο αλγόριθμος S προσομοιάζει την σημασιολογία και την πληροφορία που φέρει το $O(k)$. Έτσι, αν η παραπάνω ποσότητα είναι αμελητέα, ο διακριτής D επωφελείται εξίσου από την γνώση του S, ο οποίος έχει πρόσβαση μόνο σε ένα μαντείο της F_k.

Σε αυτό το σημείο αξίζει να σημειωθεί, ότι υπάρχουν θεωρητικά αποτελέσματα για την επιτευξιμότητα του παραπάνω εγχειρήματος, δηλαδή της εύρεσης μιας οικουμενικής συνάρτησης απόκρυψης. Οι [82] έδειξαν ότι για κάθε αλγόριθμο O, υπάρχει μια οικογένεια συναρτήσεων που δεν αποκρύπτεται από αυτό. Αυτές, όμως, οι οικογένειες είναι ειδικά και τεχνικά κατασκευασμένες. Έτσι, η απόκρυψη πιο επίλεκτων και συγκεκριμένων συναρτήσεων (προγραμμάτων) εξακολουθεί να παρουσιάζει ενδιαφέρον.

Υποθέστε ότι έχουμε στην διάθεση μας ένα σχήμα συμμετρικής κρυπτογράφησης, όπου η κρυπτογράφηση και η αποκρυπτογράφηση πραγματοποιείται με το ίδιο κρυφό κλειδί. Τότε, μπορούμε να το μετασχηματίσουμε σε ένα ασύμμετρο σχήμα, όπου πλέον θα έχουμε ένα ζεύγος κλειδιών (s_k, p_k), στο οποίο $s_k = k_{symmetric}$, $p_k = O(E_k)$, όπου E_k η συνάρτηση κρυπτογράφησης, με είσοδο το μυστικό κλειδί k (ίδιο με αυτό του αρχικού συμμετρικού σχήματος κρυπτογράφησης). Επομένως, ο καθένας πλέον θα έχει στην διάθεση του το μυστικό του κλειδί και θα δημοσιοποιεί ως δημόσιο κλειδί το $O(E_k)$. Πλέον, θα κρυπτογραφούμε το μήνυμα μας τρέχοντας $G_{O(E_k)}(m)$, το οποίο αποκρυπτογραφείται $D(G_{O(E_k)}(m), k) = m$.

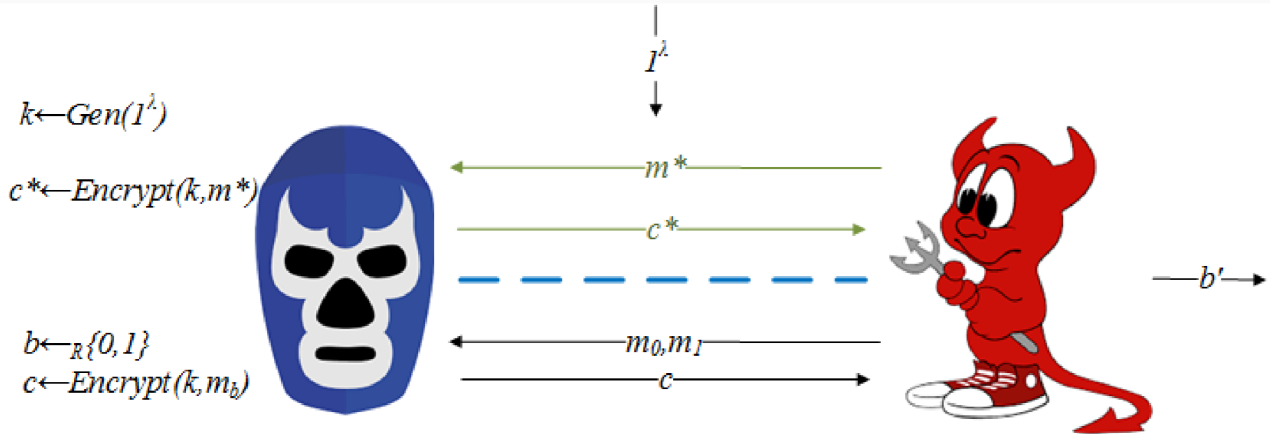
Τυπικά, το σχήμα συμμετρικής κρυπτογραφίας είναι μια τριάδα αλγορίθμων:

- Έναν PPT αλγόριθμο παραγωγής κλειδιού KeyGen το οποίο με είσοδο 1^n, $n \in N$ (παράμετρος ασφαλείας) παράγει ένα κλειδί k.
- Έναν (ντετερμινιστικό ή μη) αλγόριθμο κρυπτογράφησης E_k, ώστε $E_k(m) = c$.
- Έναν ντετερμινιστικό αλγόριθμο αποκρυπτογράφησης D_k, ώστε $D_k(c) = m$

Οι [81] έδειξαν το ακόλουθο:

Θεώρημα 2. *Έστω ένα IND-CPA σχήμα συμμετρικής κρυπτογράφησης και O μια συνάρτηση απόκρυψης που ικανοποιεί τον ορισμό 3 και 4. Τότε, το σχήμα δημοσίου κλειδιού που προκύπτει θεωρώντας ως $s_k = k_{symmetric}$, $p_k = O(E_k)$ είναι IND-CPA ασφαλές σχήμα δημοσίου κλειδιού*

Να υπενθυμίσουμε στον αναγνώστη, ότι ένα σχήμα είναι IND-CPA (chosen plaintext attack) ασφαλές, όταν ένας αντίπαλος, εφοδιασμένος με ένα μαντείο κρυπτογράφησης, αδυνατεί να διακρίνει, γνωρίζοντας c, m_0, m_1 και $c = E_k(m_b)$ για $b \in \{0, 1\}$, την τιμή του b.



Σχήμα 5.3: Παίγνιο IND-CPA. Ο αντίπαλος (δεξιά) είναι σε θέση να ζητήσει τις κρυπτογραφήσεις c^* πολυωνυμικών το πλήθος μηνυμάτων m^* πριν την επιλογή m_0, m_1 που καλείται να διακρίνει.[42]

Εντελώς ανάλογο αποτέλεσμα ισχύει και για την μετατροπή MAC (Message Authetication Code) σε σχήματα υπογραφών. Τυπικά, το MAC είναι μια τριάδα αλγορίθμων:

- $KeyGen(1^n) = k \in K$: συνάρτηση παραγωγής ιδιωτικού κλειδιού
- $Sign(1^n, k, m) = t$: συνάρτηση υπογραφής μηνύματος m.
- $Verify(1^n, k, m, t)$: συνάρτηση επαλήθευσης όπου η υπογραφή t αντιστοιχεί στο κλειδί k και το μήνυμα m, δηλαδή ελέγχει την αληθοτιμή $Sign(1^n, k, m) == t$.

Αν ένας αντίπαλος, εφοδιασμένος με ένα μαντείο πάνω στο $Verify$ (μαντείο επαλήθευσης), παρασκευάζει υπογραφή για ένα νέο μήνυμα m με αμελητέα πιθανότητα, τότε λέμε ότι το σχήμα είναι MAC-VOA (verificiation only attack) ασφαλές.

Το αντίστοιχο σχήμα υπογραφών προκύπτει διατηρώντας το ίδιο κρυφό κλειδί (με το οποίο γίνεται η υπογραφή) και παίρνοντας ως δημόσιο κλειδί (που αξιοποιείται για επαλήθευση της υπογραφής) το $O(Verify(1^n, k, \cdot, \cdot))$.

Θεώρημα 3. *Έστω ένα MAC που είναι MAC-VOA, O μια συνάρτηση απόκρυψης που ικανοποιεί τον ορισμό 3 και 4. Τότε, το σχήμα ψηφιακής υπογραφής που προκύπτει με την παραπάνω κατασκευή είναι SIG-VOA.*

Σε αυτό το σημείο αξίζει να σημειωθεί ότι σε περίπτωση ισχυρότερων αντιπάλων, τα παραπάνω αποτελέσματα παύουν να ισχύουν. Πιο συγκεκριμένα, οι παραπάνω κατασκευές όταν εφαρμόζονται σε IND-CCA (chosen ciphertext attack) (επιπλέον μαντείο αποκρυπτογράφησης) ή σε MAC-CMA (adaptive chosen message attacks) (επιπρόσθετο μαντείο υπογραφής), δεν διατηρούν κατά ανάγκη την ασφάλεια τους. Επιπλέον, ισχύει το ακόλουθο θεώρημα:

Θεώρημα 4. *Η απόκρυψη μιας ψευδοτυχαίας συνάρτησης μέσω μιας συνάρτηση O που υπακούει στον ορισμό 3 και 4 είναι αδύνατη*

Ύστερα, είναι σημαντικό να διαπιστώσουμε ότι τα θεωρήματα που διατυπώσαμε προϋποθέτουν την ύπαρξη IND-CPA σχημάτων κρυπτογράφησης στα οποία η συνάρτηση E_k είναι αποκρύψιμη (αντίστοιχα για τα MAC). Εύλογα διερωτάται κανείς αν υπάρχει τέτοιο σχήμα. Άλλωστε, η απαίτηση αποκρυψιμότητας σύμφωνα με τον ορισμό 3 επιβάλλει ορισμένους περιορισμούς. Υποθέστε, για παράδειγμα, ότι έχουμε στην διάθεση μας ένα συμμετρικό σχήμα κρυπτογράφησης το οποίο θέλουμε να είναι ασφαλές απέναντι σε ενεργητικό αντίπαλο. Ένας αντίπαλος είναι ενεργητικός, όταν έχει την δυνατότητα να τροποποιήσει ή να παρεμβάλει ένα δικό του κρυπτοκείμενο c στο κανάλι επικοινωνίας μεταξύ των Alice, Bob. Ένας τρόπος για να εξασφαλίσουμε την ακεραιότητα του μηνύματος είναι η ενσωμάτωση MAC υπογραφής που αναλογεί στο μήνυμα m (authenticated encryption). Επομένως, για να τροποποιήσει το c σε c' ο αντίπαλος υποχρεούται να βρει την ανάλογη υπογραφή. Ο ασφαλέστερος συνδυασμός αυτών των δυο σχημάτων επιτυγχάνεται με την λεγόμενη «Κρυπτογράφηση-μετά-Γνησιότητα», όπου $c = E_{k_{enc}}(m)$, $t = Sign(k_{mac}, c)$. Ακολουθώντας την παραπάνω κατασκευή, η απόκρυψη θα εμπεριέχει και τις δυο λειτουργίες. Επομένως, θα είναι σε θέση να δρα ως μαντείο για την κρυπτογράφηση και την υπογραφή. Όμως, αν το πρόγραμμα O ακολουθεί τον ορισμό 4, τότε μπορεί να υποκατασταθεί με έναν προσομοιωτή S ο οποίος δεν έχει καμία σημασιολογική επίγνωση του προγράμματος. Επομένως, θα μπορούσαμε, έχοντας πρόσβαση μόνο σε ένα μαντείο, να αναπαράξουμε υπογραφές, πράγμα που έρχεται σε αντίθεση με τον ορισμό ασφάλειας ακεραιότητας του μηνύματος μας.

Συνοψίζοντας, η θεωρία που ανέπτυξαν οι [81] μας δίνουν μια πρώτη ιδέα και δρα ως ένα έναυσμα για την απόπειρα απόκρυψης σύνθετου κώδικα. Είδαμε, πώς η εύρεση απόκρυψης μιας συνάρτησης κρυπτογράφησης E_k οδηγεί άμεσα στη κατασκευή ενός ανάλογου κρυπτοσυστήματος δημοσίου κλειδιού. Ίσως πιο ουσιαστική είναι η χρησιμότητα κατασκευής ψηφιακών υπογραφών, ένα πεδίο με τεράστιες εφαρμογές σε ηλεκτρονικές ψηφοφορίες. Εφοδιασμένοι με αυτή την γνώση, και λαμβάνοντας υπ' όψιν τους περιορισμούς αυτής της θεωρίας, θα επιχειρήσουμε στην επόμενη ενότητα να υιοθετήσουμε ένα μοντέλο κατασκευής κώδικα το οποίο σκοπεύουμε να αξιοποιήσουμε για την παραγωγή απόκρυφου κώδικα. Λόγω της αυξημένης δυσκολίας του ζητήματος θα ανατρέξουμε στην βιβλιογραφία για την αξιοποίηση ενός σύνθετου και κομψού μοντέλου νευρωνικού δικτύου που επιλύει ένα παρεμφερές πρόβλημα.

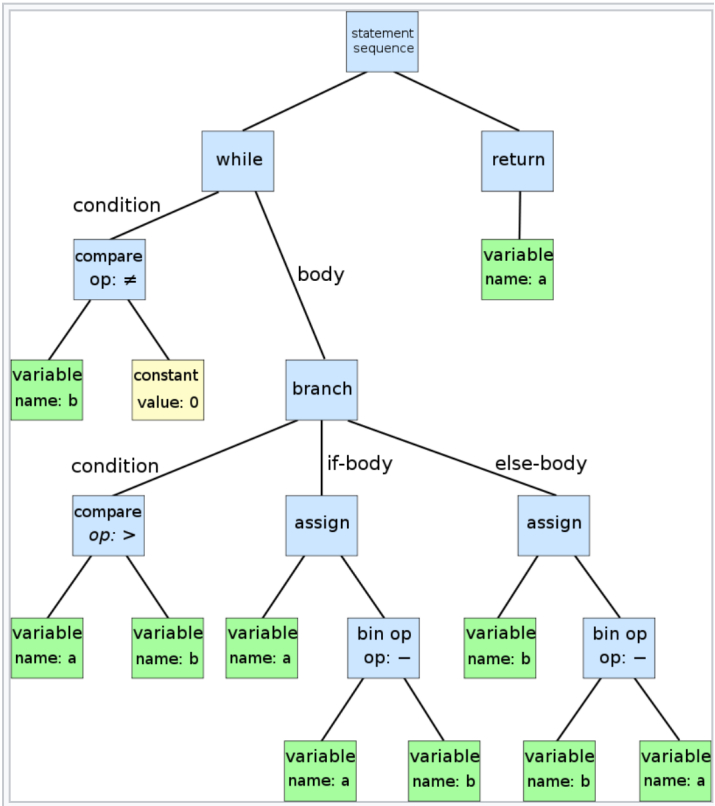
5.3 Αξιοποίηση Νευρωνικού Δικτύου για code obfuscation

Ένα πρώτο βήμα στην αξιολόγηση μεθόδων Μηχανικής Μάθησης σε αυτό τον κλάδο πραγματοποιήθηκε από τους [80], οι οποίοι μελέτησαν την δυνατότητα εντοπισμού (κατηγοριοποίηση) javascript κώδικα σε obfuscated, minified ή κανονικό (μη παραλλαγμένο) αξιοποιώντας SVM, RF, LDA. Λέμε ότι ένας κώδικας είναι minified, όταν έχει τροποποιηθεί κατά τρόπο που βελτιστοποιεί (ελαχιστοποιεί) την χωρική του απαίτηση (π.χ. αφαιρώντας κενά μεταξύ εντολών). Πρόκειται για μια συνήθη τακτική code optimization που αξιοποιείται ευρέως σε κώδικα διαδικτυακής χρήσης. Μολονότι οδηγεί σε τροποποιημένο κώδικα, πρώτο μέλημα παραμένει η βελτιστοποίηση της απόδοσης με αποτέλεσμα η σημασιολογία του προγράμματος να παραμένει, εν γένει, αντιληπτή και προσδιορίσιμη. Τα χαρακτηριστικά που μελετήθηκαν στην εφαρμογή (π.χ. πλήθος γραμμών) φανερώνουν ορισμένες ιδιότητες τις οποίες θα θέλαμε να ενσωματώσει στον απόκρυφο κώδικα που θα παράγει το Νευρωνικό μας Δίκτυο.

Το βασικότερο, ίσως, ζήτημα αυτής της απόπειρας αποτελεί η παραγωγή κώδικα, αξιοποιώντας Νευρωνικά Δίκτυα. Μάλιστα, θα προτιμούσαμε αυτή η παραγωγή να βασίζεται όσο το δυνατόν περισσότερο στην σημασιολογία του κώδικα, καθώς η απόκρυψη της αποτελεί το τελικό μας εγχείρημα. Επιπλέον, επιδιώκουμε να μπορούμε να περιγράψουμε στο Νευρωνικό Δίκτυο την λειτουργία του κώδικα που θα αποκρύψει. Οι [83] ασχολήθηκαν με την δημιουργία ενός καινοτόμου μοντέλου παραγωγής κώδικα βασισμένο στην σημασιολογία του. Δημιούργησαν ένα Νευρωνικό Δίκτυο, το οποίο, δεχόμενο μια περιγραφή της λειτουργίας του προγράμματος (σημασιολογία) παράγει το συντακτικό δέντρο που του αναλογεί (AST-Abstract Syntactic Tree).

Έτσι, δοθέντος μιας σημασιολογικής περιγραφής σε φυσική γλώσσα x, θέλουμε να εντοπίσουμε το βέλτιστο δέντρο $\widehat{y}$, ώστε:

$$\widehat{y} = arg\max_{y} p(y|x)$$



Σχήμα 5.4: Συντακτικό δέντρο Ευκλείδιου αλγόριθμου. Η ανάγνωση του γίνεται από αριστερά προς τα δεξιά και κατά βάθος (left-to-right depth first) [94].

Ο μετασχηματισμός του δέντρου $\widehat{y}$ στον αντίστοιχο κώδικα c είναι αρκετά απλός, καθώς, ανάλογα με την γλώσσα προγραμματισμού, μπορεί να αξιοποιηθούν αντίστοιχα έτοιμα εργαλεία (στην εργασία τους εργάστηκαν στη Python, όπου αξιοποίησαν την βιβλιοθήκη astor [84]).

Γενικότερα, η παραγωγή κώδικα προσεγγίζεται ως Natural Language Processing, όπου αποσκοπούμε στην δυνατότητα ανάλυσης και παραγωγής κειμένων φυσικής γλώσσας (π.χ. αγγλικών κειμένων). Η παραγωγή AST (και η εκπαίδευση πάνω σε αυτά) προτιμήθηκε έναντι της απευθείας παραγωγής πηγαίου κώδικα c για να δοθεί μεγαλύτερη έμφαση στο συντακτικό και τη γραμματική της γλώσσας προγραμματισμού. Το δέντρο παράγεται σε στάδια, όπου στο καθένα δύναται επέκτασης (προσθήκη κόμβου) εφαρμόζοντας κανόνες παραγωγής ή εξωτερικεύοντας τελικούς κόμβους. Οι κανόνες γραμματικής και οι τελικοί κόμβοι αναλογούν στα τερματικά

και μη τερματικά σύμβολα μιας τυπικής γραμματικής. Η γραμματική προσδιορίζεται πλήρως από τον ακριβή προσδιορισμό των παραπάνω δυο δυνατοτήτων, και έτσι, είναι ενσωματωμένη εκ των προτέρων στο μοντέλο μας, και δεν αποτελεί learnable παράμετρο.

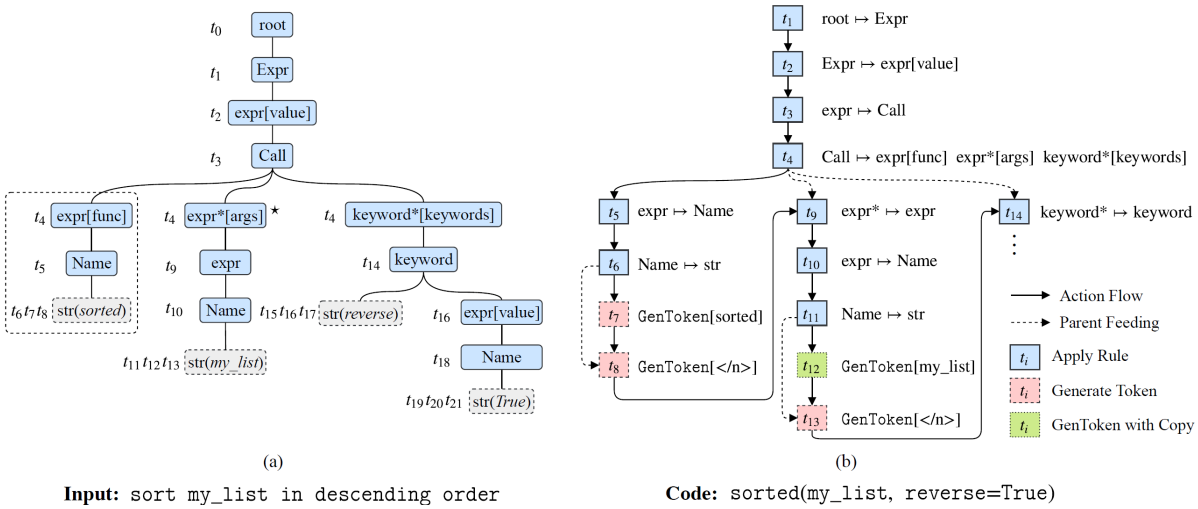
Αν υποθέσουμε ότι το δέντρο μας απαρτίζεται από T κόμβους-ενέργειες, τότε

$$p(y|x) = \prod_{t=1}^{T} p(a_t|x, a_{<t})$$

όπου a_t είναι η ενέργεια την χρονική στιγμή t (pre-order ιεραρχία), ενώ $a_{<t}$ η ακολουθία που εμπεριέχει όλες τις προγενέστερες ενέργειες. Στόχος μας είναι η εύρεση των a_t που μεγιστοποιούν την παραπάνω πιθανότητα.

Τα a_t είναι δυο κατηγοριών:

- APPLYRULE[r]: εφαρμόζει τον κανόνα r στο έως τώρα δέντρο.
- GENTOKEN[v]: εμπλουτίζει μια μεταβλητή, προσαρτώντας σε αυτήν το v.



Σχήμα 5.5: Παράδειγμα εφαρμογής (σε γλώσσα Python) της παραπάνω μεθόδου κατασκευής AST. (a) Το τελικό AST. (b) Η διαδικασία κατασκευής του (σε βήματα 1-21) [83]

Οι κανόνες παραγωγής στην Python είναι συγκεκριμένοι και οφείλουν να ακολουθήσουν δομή $Father \longmapsto Child_1, Child_2, ...$, όπου Father, Child είναι μη τερματικά σύμβολα-κανόνες. Για παράδειγμα:

Production Rule	Role	Explanation
Call ↦ expr[*func*] expr*[*args*] keyword*[*keywords*]	Function Call	▷ *func*: the function to be invoked ▷ *args*: arguments list ▷ *keywords*: keyword arguments list
If ↦ expr[*test*] stmt*[*body*] stmt*[*orelse*]	If Statement	▷ *test*: condition expression ▷ *body*: statements inside the If clause ▷ *orelse*: elif or else statements
For ↦ expr[*target*] expr*[*iter*] stmt*[*body*] stmt*[*orelse*]	For Loop	▷ *target*: iteration variable ▷ *iter*: enumerable to iterate over ▷ *body*: loop body ▷ *orelse*: else statements
FunctionDef ↦ identifier[*name*] arguments*[*args*] stmt*[*body*]	Function Def.	▷ *name*: function name ▷ *args*: function arguments ▷ *body*: function body

Σχήμα 5.6: Ορισμένοι βασικοί κανόνες παραγωγής στην Python [83]

Έστω n_{f_t} (frontier node) ο τελευταίος κόμβος στο έως τώρα παραγόμενο δέντρο. Τότε, το APPLYRULE[r] εφαρμόζει τον κανόνα r το οποίο αρμόζει (δύναται να έχει ως πατέρα-father node το n_{f_t}) στο n_{f_t}, επεκτείνοντας το, προσαρτώντας σε αυτό κόμβους-παιδιά. Για παράδειγμα, στο 5.4(b), την χρονική στιγμή 4, ο κανόνας $r = expr[func]\ expr^*[ars]\ keyword^*[keywords]$ επεκτείνει το $Call$, προσθέτοντας του τα ανάλογα παιδιά. Η έκφραση r μπορεί να περιέχει άλλους κανόνες παραγωγής, όπου αναδρομικά συνεχίζουμε την διαδικασία σύνθεσης ή ένα τερματικό σύμβολο (δήλωση βασικού τύπου μεταβλητής, όπως str για αλφαριθμητικό). Στα τερματικά σύμβολα αξιοποιούμε το GENTOKEN[v], το οποίο ενσωματώνει στο προϋπάρχων τερματικό σύμβολο n_{f_t} την τιμή v. Το v είναι μεταβλητή βασικού τύπου (π.χ. str, int, boolean). Όταν έχουμε ολοκληρώσει την επεξεργασία του τερματικού συμβόλου, προσαρτίζουμε το σύμβολο $< \backslash n >$. Έτσι, στο δοθέν παράδειγμα (σχήμα 5.4) βλέπουμε ότι το AST εντοπίζει ότι χρειάζεται να καλέσουμε μια συνάρτηση ($Call$) η οποία απαρτίζεται από το όνομα της ($func$), τα ορίσματα της ($args$) και τις τιμές ορισμένων από αυτών ($keywords$). Στο τέλος, κάθε μη τερματικό σύμβολο αντικαθίσταται από ένα τερματικό. Το όνομα με την λέξη $sorted$, τα ορίσματα με το my_list και το $keyword$ με το $reverse = True$. Το σύνολο των διαθέσιμων κανόνων είναι εκ των προτέρων καθορισμένο (καθορίζεται από την γλώσσα προγραμματισμού). Το λεξιλόγιο του GENTOKEN αποτελείται, ομοίως, από a-priori καθορισμένες λέξεις (π.χ. ονόματα συναρτήσεων), καθώς και νέες OOV(Out Of Vocabulary) λέξεις που εντοπίζει στην περιγραφή. Παρατηρήστε, πως στο σχήμα, το my_list είναι αντίγραφο (πράσινο χρώμα), της λέξης my_list που παρατηρεί στην περιγραφή που δέχεται ως είσοδο.

Το Νευρωνικό Δίκτυο που αξιοποιείται, ακολουθεί μια encoder-decoder δομή. Οι encoder-decoder αρχιτεκτονικές χρησιμοποιούνται συχνά στην μηχανική μάθηση. Μας επιτρέπουν την προσέγγιση sequence-to-sequence προβλημάτων, στα οποία οι ακολουθίες παρουσιάζουν μεταβλητό μήκος. Επιπλέον, το encoder τμήμα αποτελεί ένα είδους learnable data preprocessing. Επεξεργάζεται τα δεδομένα-χαρακτηριστικά, αφομοιώνοντας τις σημαντικότερες πληροφορίες τους και μετασχηματίζοντας τα σε έναν κατάλληλο (συνήθως μικρότερων διαστάσεων) χώρο.

Το encoder τμήμα στη περίπτωση μας λαμβάνει ως χαρακτηριστικά-δεδομένα τις λέξεις $\{w_i\}_{i=1}^{n}$ της περιγραφής x, παράγοντας για κάθε μια από αυτές ένα μετασχηματισμένο χα-

ρακτηριστικό h_i. Πιο συγκεκριμένα η ακολουθία λέξεων διαβάζεται και με την φυσιολογική σειρά $\{w_i\}_{i=1}^{n}$, άλλα και αντίστροφα $\{w_i\}_{i=n}^{1}$:

$$\overrightarrow{h_i} = f_{\overrightarrow{LSTM}}(w_i, \overrightarrow{h_{i-1}})$$

$$\overleftarrow{h_i} = f_{\overleftarrow{LSTM}}(w_i, \overleftarrow{h_{i+1}})$$

Η μονάδα LSTM (Long short-term memory) είναι μια ειδική αρχιτεκτονική στρώματος που επιτρέπει την επεξεργασία και αξιοποίηση πληροφορίας από προγενέστερα στάδια εκπαίδευσης/παραγωγής. Αυτή και τα RNN γενικότερα αξιοποιούνται σε εφαρμογές όπου το νευρωνικό δίκτυο επωφελείται από την ύπαρξη μνήμης και επίγνωσης του παρελθόντος, όπως συμβαίνει στην φυσική επεξεργασία γλώσσας, όπου η λέξη την χρονική στιγμή t εξαρτάται από τις προηγούμενες (στην πρόταση ή ακόμα και στην παράγραφο).

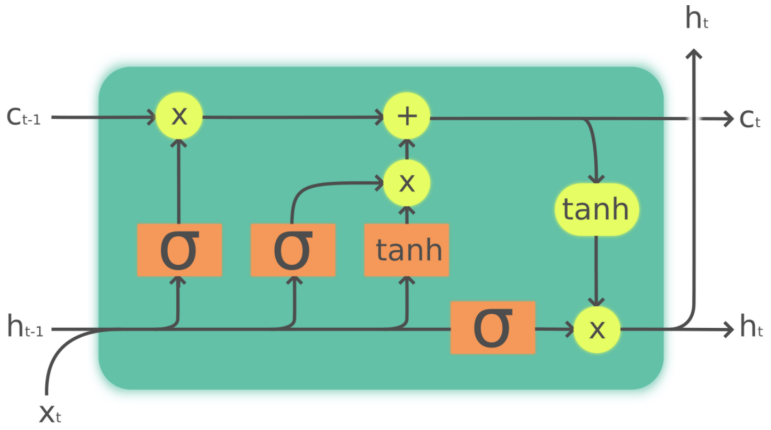
Τελικά:

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$$

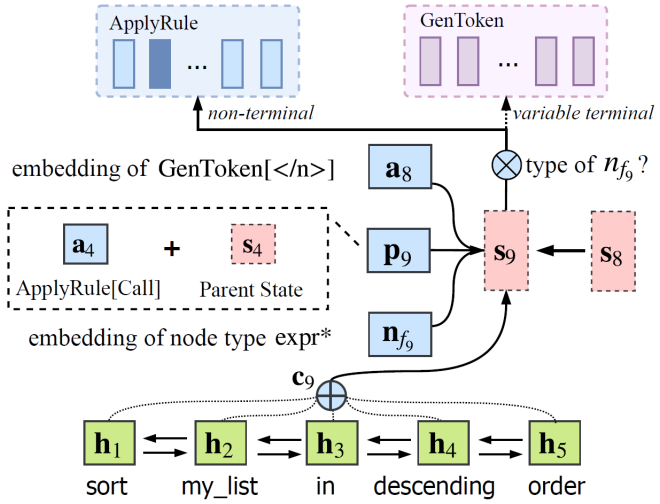
Το decoder τμήμα εκμεταλλεύεται παρόμοια RNN αρχιτεκτονική. Πιο συγκεκριμένα, η εσωτερική κατάσταση (μνήμη) s_t του Νευρωνικού δίνεται από την σχέση:

$$s_t = f_{LSTM}([a_{t-1}, c_t, p_t, n_{f_t}], s_{t-1})$$

όπου a_{t-1} η προηγούμενη ενέργεια, c_t είναι το διάνυσμα συμφραζομένων (context vector), το οποίο προκύπτει από τα h_i, ενώ τα p_t, n_{f_t} εμπεριέχουν την πληροφορία για τον γονέα κόμβο (αντίστοιχη ενέργεια και κατάσταση) και τον τελευταίο αντίστοιχα. Όλες οι ποσότητες μοντελοποιούνται ως διανύσματα πραγματικών αριθμών. Πιο συγκεκριμένα, η περιγραφή x τεμαχίζεται (tokenized) και η κάθε λέξη-token κωδικοποιείται σε ένα διάνυσμα πραγματικών αριθμών αξιοποιώντας την βιβλιοθήκη NLTK [101]. Η NLTK παρέχει διάφορες υλοποιήσεις (π.χ. word2vec [102]) για την εκμάθηση του μετασχηματισμού, ο οποίος μετατρέπει παρεμφερής λέξεις σε κοντινά (π.χ. Ευκλείδεια απόσταση ή cosine distance) διανύσματα. Η σχέση μεταξύ των λέξεων βασίζεται σε ένα πλήθος φράσεων-κειμένων (corpus) που τροφοδοτούνται στον αλγόριθμο. Στην παρούσα εφαρμογή τροφοδότησαν τον αλγόριθμο με τις περιγραφές x των κωδίκων. Οι υπόλοιπες κωδικοποιήσεις (π.χ. p_t, a_t,n_{f_t}) είναι learnable παράμετροι.



Σχήμα 5.7: Μονάδα LSTM. Η έξοδος του προηγούμενου σταδίου h_{t-1} συνδυάζεται με την τωρινή είσοδο x_t. Η έξοδος αυτού του κελιού είναι το h_t το οποίο συγχρόνως τροφοδοτείται αναδρομικά στο επόμενο. Το c_{t-1} αποτελεί μεταβλητή μνήμης το οποίο συγκρατεί πληροφορίες από όλα τα προγενέστερα στάδια εκπαίδευσης. Αναβαθμίζεται κατάλληλα (c_t) ανάλογα με το περιεχόμενο αυτού του κύκλου [95]



Σχήμα 5.8: Λειτουργία decoder-RNN για $t = 9$. Παρατηρήστε ότι στο p_9 εμπεριέχεται πληροφορία για την ενέργεια (a_4) και την κατάσταση (s_4) του father-κόμβου (ο οποίος δημιουργήθηκε την χρονική στιγμή $t = 4$) [83]

Γνωρίζουμε ότι το a_t αναλογεί είτε σε APPLYRULE[r] ή σε GENTOKEN[v].

Στην πρώτη περίπτωση θεώρησαν:

$$p(a_t = APPLYRULE[r]|x, a_{<t}) = softmax(W_R \cdot g(s_t))^T \cdot e(r)$$

όπου $g(s_t) = tanh(W \cdot s_t + b)$, W_R πίνακας κωδικοποίησης όλων των APPLYRULE ενεργειών (ανά γραμμή), ενώ το $e(r)$ αποτελεί ένα one-hot vector για τον κανόνα r. Παρατηρήστε ότι το $softmax$ τμήμα περιέχει την συνολική πληροφορία όλων των δυνατών κανόνων παραγωγής

W_R. Η δυαδική φύση του one-hot vector $e(r)$ καταφέρνει να εστιάσει στο τμήμα που αναλογεί στην δοσμένη ενέργεια r.

Στη δεύτερη περίπτωση έχουμε ότι η παραγωγή LSTM μπορεί να δημιουργήσει τελική μεταβλητή από το δοσμένο λεξιλόγιο ή μια μεταβλητή-αντίγραφο (όπως αναφέραμε προηγουμένως):

$$p(a_t = GENTOKEN[v]|x, a_{<t}) = p(gen|x, a_{<t})p(v|gen, x, a_{<t}) + p(copy|x, a_{<t})p(v|copy, x, a_{<t})$$

όπου $p(gen|\cdot), p(copy|\cdot) = softmax(W_S \cdot s_t)$ (W_S αντίστοιχο με το W_R). Επιπλέον, $p(v|gen, x, a_{<t}) = softmax(W_S \cdot g(s_t||c_t))^T e(v)$. Τέλος, $p(v|copy, x, a_{a<t}) = p(w_i|copy, x, a_{<t})$ (αφού αποφασίσαμε να αντιγράψουμε μια από τις λέξεις). Μοντελοποίησαν αυτή την πιθανότητα ως:

$$p(w_i|copy, x, a_{<t}) = \frac{e^{w(h_i, s_t, c_t)}}{\sum_{j=1}^{n} e^{h_j, s_t, c_t}}$$

όπου $w(\cdot)$ η έξοδος ενός νευρωνικού μονού κρυφού στρώματος.

Το c_t προκύπτει από τα h_i αξιοποιώντας soft attention. Το soft attention είναι μια τεχνική που μας επιτρέπει την επεξεργασία των διανυσμάτων h_i, συνδυάζοντας τα κατάλληλα σε ένα συνολικό διάνυσμα. Πιο συγκεκριμένα, αναθέτει σε κάθε βήμα εκπαίδευσης ένα βάρος $a_{t,i}$ στο καθένα (εκφράζει τον βαθμό προσοχής του κάθε h_i) και παίρνει το σταθμικό άθροισμα. Ενδεικτικά:

$$a_{t,i} = \frac{\exp e_{t,i}}{\sum_{j=1}^{n} \exp e_{t,j}}$$

$$e_{t,i} = v_a tanh(W_{state} s_{t-1} + W_h h_i)$$

$$c_t = \sum_{i=1}^{n} a_{t,i} h_i$$

Το soft attention προτιμάται, καθώς η διαδικασία εστίασης παραμένει διαφορίσιμη.

Τα δεδομένα απαρτίζονται από ζευγάρια (x_i, c_i), όπου x_i οι φυσικές περιγραφές των προγραμμάτων c_i. Τα c_i μετασχηματίζονται στα αντίστοιχα AST y_i, τα οποία αναλύονται ως μια αλληλουχία δράσεων $\{a_t\}_{t<T}$. Η εκπαίδευση του νευρωνικού βασίζεται στην μεγιστοποίηση της πιθανοφάνειας αυτής της αλληλουχίας, δηλαδή του $\log p(y|x), p(y|x) = \prod_{t=1}^{T} p(a_t|x, a_{<t})$.

Μετά την εκπαίδευση, η παραγωγή του αντίστοιχου δέντρου πραγματοποιείται αξιοποιώντας το παραπάνω νευρωνικό. Για κάθε περιγραφή x η εύρεση του βέλτιστου δέντρου $\widehat{y}$ πραγματοποιείται σε στάδια (για κάθε $\{a_t\}_{t\leq T}$), όπου εξετάζεται η καλύτερη δυνατή επέκταση των K έως τώρα καλύτερων δέντρων (K-beam search). Αναλυτικότερα, ο αλγόριθμος είναι ο ακόλουθος:

Algorithm 1: Inference Algorithm

Input : NL description x
Output: code snippet c

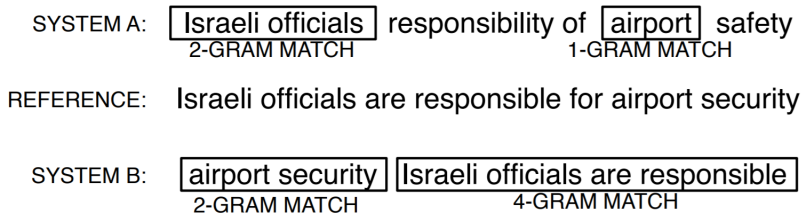
call Encoder to encode x
$Q = \{y_0(root)\}$ ▷ *Initialize a beam of size K*
for *time step t* **do**
  $Q' = \emptyset$
  foreach *hypothesis $y_t \in Q$* **do**
    $n_{f_t} =$ FrontierNode(y_t)
    $\mathcal{A} = \emptyset$ ▷ *Initialize the set of candidate actions*
    if *n_{f_t} is non-terminal* **then**
      foreach *production rule r with n_{f_t} as the head node* **do**
        $\mathcal{A} = \mathcal{A} \cup \{$ApplyRule$[r]\}$ ▷ ApplyRule *actions for non-terminal nodes*
    else
      foreach *terminal token v* **do**
        $\mathcal{A} = \mathcal{A} \cup \{$GenToken$[v]\}$ ▷ GenToken *actions for variable terminal nodes*
    foreach *action $a_t \in \mathcal{A}$* **do**
      $y'_t =$ ApplyAction(y_t, a_t)
      $Q' = Q' \cup \{y'_t\}$
  $Q =$ top-K scored hypotheses in Q'
$\hat{y} =$ top-scored complete hypothesis AST
convert $\hat{y}$ to surface code c
return c

Σχήμα 5.9: Αλγόριθμος εύρεσης βέλτιστου δέντρου $\widehat{y}$. Σε κάθε χρονική στιγμή t εξετάζουμε την επέκταση K πιθανών δέντρων, τα οποία έχουν λάβει το μεγαλύτερο σκορ (απέδωσαν την μεγαλύτερη τιμή στην πιθανοφάνεια $p(y|x)$) [83]

Προτιμήθηκαν δυο μετρικές: ακρίβεια (accuracy) και η BLEU-4. Για ένα ζευγάρι (x_i, c_i) από το δείγμα μας και το παραγόμενο c του νευρωνικού μας, η ακρίβεια συγκρίνει τα (c, c_i), αποδίδοντας θετικό αποτέλεσμα μόνο σε περίπτωση που συμπίπτουν τελείως (ίδιες λέξεις, ίδια σειρά). Το BLEU-4 αποτελεί ένα λογικότερο μέτρο, το οποίο λαμβάνει υπ' όψιν του και πιθανές μεταθέσεις των ενδεχομένως σωστών εντολών. Έτσι, αποτελεί πιο αξιόπιστο μέτρο της αποτελεσματικότητας του μοντέλου. Πιο συγκεκριμένα, εξετάζει τα c, c_i ως μια ακολουθία λέξεων. Στη συνέχεια, τεμαχίζει αυτές τις ακολουθίες σε φράσεις (υποακολουθία διαδοχικών λέξεων) μεγέθους $i_{1 \leq i \leq 4}$. Για κάθε μια από αυτές, υπολογίζει το $precision = \frac{correct}{|c|}$. Τελικά

$$BLEU - 4 = \min\left(1, \frac{|c|}{|c_i|}\right)\left(\prod_{i=1}^{4} precision_i\right)^{\frac{1}{4}}$$

Παρατηρήστε ότι αποδίδει ποινή στο μέγεθος της έκφρασης c (αν $\min\left(1, \frac{|c|}{|c_i|}\right) < 1 \Leftrightarrow |c| < |c_i|$), προτιμώντας, έτσι, εκτενέστερες και μεγαλύτερες αναπαραγωγές της φράσης c.



Metric	System A	System B
precision (1gram)	3/6	6/6
precision (2gram)	1/5	4/5
precision (3gram)	0/4	2/4
precision (4gram)	0/3	1/3
brevity penalty	6/7	6/7
BLEU	0%	52%

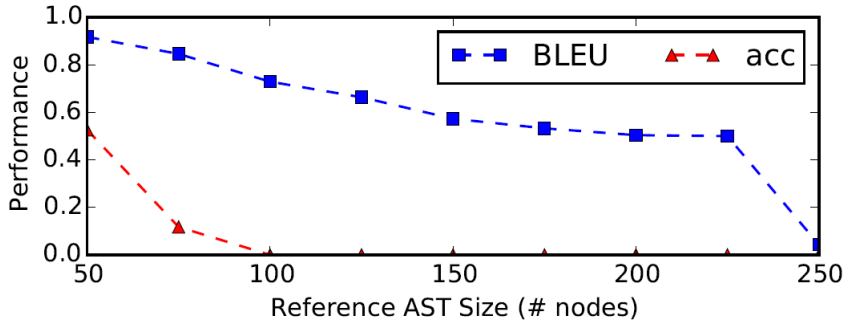
Σχήμα 5.10: Παράδειγμα αξιολόγησης παραγωγής φράσεων, μέσω της μετρικής BLEU-4 [85]

Ο λόγος που εξετάζουμε αυτό το μοντέλο είναι λόγω των θετικών αποτελεσμάτων που παρουσίασε (ως προς τις παραπάνω μετρικές) στα πειράματα που διεξήγαν. Ενδεικτικά, μελετήθηκε η επίδοση του μοντέλου πάνω στο Hearthstone (HS) [96] και DJANGO [97] δείγμα. Το Hearthstone (HS) αποτελεί ένα παιχνίδι καρτών και το δείγμα απαρτίζεται από τις υλοποιήσεις τους (σε python). Το Django περιέχει κώδικα από την ομώνυμη ιστοσελίδα. Το παρόν μοντέλο παρουσίασε καλύτερη επίδοση, έναντι άλλων προγενέστερων:

	HS		DJANGO	
	ACC	BLEU	ACC	BLEU
Retrieval System†	0.0	62.5	14.7	18.6
Phrasal Statistical MT†	0.0	34.1	31.5	47.6
Hierarchical Statistical MT†	0.0	43.2	9.5	35.9
NMT	1.5	60.4	45.1	63.4
SEQ2TREE	1.5	53.4	28.9	44.6
SEQ2TREE–UNK	13.6	62.8	39.4	58.2
LPN†	4.5	65.6	62.3	77.6
Our system	16.2	**75.8**	**71.6**	**84.5**

Σχήμα 5.11: Επίδοση του μοντέλου και σύγκριση του με άλλα παρεμφερή. [83]

Ένα ενδιαφέρον χαρακτηριστικό αποτελεί η ποιοτική μελέτη της συμπεριφοράς των μετρικών συναρτήσει του μεγέθους του AST (της πολυπλοκότητας του κώδικα). Άλλωστε, απώτερος σκοπός είναι η δημιουργία απόκρυφου κώδικα για πιο σύνθετα προγράμματα. Τα πειραματικά τους αποτελέσματα δείχνουν ενθαρρυντικά, καθώς το BLUE-4 σκορ στο HS παραμένει ικανοποιητικά υψηλό για μεγάλου μεγέθους δέντρα.



Σχήμα 5.12: BLUE-4 συναρτήσει του μεγέθους των AST για το HS δείγμα . [83]

Πληροφορίες για το δείγμα αναγράφονται στο παρακάτω πίνακα:

Dataset	HS	DJANGO
Train	533	16,000
Development	66	1,000
Test	66	1,805
Avg. tokens in description	39.1	14.3
Avg. characters in code	360.3	41.1
Avg. size of AST (# nodes)	136.6	17.2
Statistics of Grammar		
w/o unary closure		
# productions	100	222
# node types	61	96
terminal vocabulary size	1361	6733
Avg. # actions per example	173.4	20.3

Σχήμα 5.13: Χαρακτηριστικά δείγματος και γραμματικής [83]

Παρατηρήστε ότι τα δυο δείγματα (ιδιαίτερα το HS), έχουν σχετικά μικρό μέγεθος. Για την αποφυγή overfitting υιοθέτησαν recurrent dropout, τα οποία αφαιρούν με μια πιθανότητα p (όπως στο κανονικό dropout) τις αναδρομικές συνδέσεις-ακμές, διακόπτοντας τις συνδέσεις μεταξύ νευρώνων στον χρόνο.

Ενδεικτικά αποτελέσματα κώδικα:

input *for every i in range of integers from 0 to length of result, not included* **pred.** `for i in range(0, len(result)):` ✓	**ref.** `for i in range(len(result)):`
input *call the function blankout with 2 arguments: t.contents and 'B', write the result to out.* **pred.** `out.write(blankout(t.contents, 'B'))` ✓	**ref.** `out.write(blankout(t.contents, 'B'))`
pred. `code_list.append(foreground[v])` ✓	**ref.** `code_list.append(foreground[v])`
input *zip elements of inner_result and inner_args into a list of tuples, for every i_item and i_args in the result* **pred.** `for i_item, i_args in zip(inner_result, inner_args):` ✓	**ref.** `for i_item, i_args in zip(inner_result, inner_args):`
input *activate is a lambda function which returns None for any argument x.* **pred.** `activate = lambda x: None` ✓	**ref.** `activate = lambda x: None`
input *if elt is an instance of Choice or NonCapture classes* **pred.** `if isinstance(elt, Choice):` ✗	**ref.** `if isinstance(elt, (Choice, NonCapture)):`
input *get translation_function attribute of the object t, call the result with an argument eol_message, substitute the result for result.* **pred.** `translation_function = getattr(t, translation_function)` ✗	**ref.** `result = getattr(t, translation_function)( eol_message)`
input *for every s in strings, call the function force_text with an argument s, join the results in a string, return the result.* **pred.** `return ''.join(force_text(s))` ✗	**ref.** `return ''.join(force_text(s) for s in strings)`
input *for every p in parts without the first element* **pred.** `for p in p[1:]:` ✗	**ref.** `for p in parts[1:]:`
input *call the function get_language, split the result by '-', substitute the first element of the result for base_lang.* **pred.** `base_lang = get_language().split()[0]` ✗	**ref.** `base_lang = get_language().split('-')[0]`

Σχήμα 5.14: Παραγωγή κώδικα από το django δείγμα . [83]

input *<name> Burly Rockjaw Trogg </name> <cost> 5 </cost> <attack> 3 </attack> <defense> 5 </defense> <desc> Whenever your opponent casts a spell, gain 2 Attack. </desc> <rarity> Common </rarity> ...*

pred.
```
class BurlyRockjawTrogg(MinionCard):
    def __init__(self):
        super().__init__('Burly Rockjaw Trogg', 4, CHARACTER_CLASS.ALL, CARD_RARITY.COMMON)
    def create_minion(self, player):
        return Minion(3, 5, effects=[Effect(SpellCast(player=EnemyPlayer()),
            ActionTag(Give(ChangeAttack(2)), SelfSelector()))]) ✓
```

input *<name> Maexxna </name> <cost> 6 </cost> <attack> 2 </attack> <defense> 8 </defense> <desc> Destroy any minion damaged by this minion. </desc> <rarity> Legendary </rarity> ...*

pred.
```
class Maexxna(MinionCard):
    def __init__(self):
        super().__init__('Maexxna', 6, CHARACTER_CLASS.ALL, CARD_RARITY.LEGENDARY,
          minion_type=MINION_TYPE.BEAST)
    def create_minion(self, player):
        return Minion(2, 8, effects=[Effect(DidDamage(), ActionTag(Kill(),
            TargetSelector(IsMinion()))())]) ✓
```

input *<name> Hellfire </name> <cost> 4 </cost> <attack> -1 </attack> <defense> -1 </defense> <desc> Deal 3 damage to ALL characters. </desc> <rarity> Free </rarity> ...*

pred.
```
class Hellfire(SpellCard):
    def __init__(self):
        super().__init__('Hellfire', 4, CHARACTER_CLASS.WARLOCK, CARD_RARITY.FREE)

    def use(self, player, game):
        super().use(player, game)
        for minion in copy.copy(game.other_player.minions):
            minion.damage(player.effective_spell_damage(3), self) ✗
```

ref.
```
class Hellfire(SpellCard):
    def __init__(self):
        super().__init__('Hellfire', 4, CHARACTER_CLASS.WARLOCK, CARD_RARITY.FREE)

    def use(self, player, game):
        super().use(player, game)
        targets = copy.copy(game.other_player.minions)
        targets.extend(game.current_player.minions)
        targets.append(game.other_player.hero)
        targets.append(game.current_player.hero)
        for minion in targets:
            minion.damage(player.effective_spell_damage(3), self)
```

Σχήμα 5.15: Παραγωγή κώδικα από το HS δείγμα. [83]

Για να αξιοποιήσουμε το μοντέλο τους, χρειαζόμαστε ένα δείγμα από AST απόκρυφου κώδικα. Για τον σκοπό αυτό προτείνουμε το pyminifier [90]. Έτσι, μπορούμε να πάρουμε οποιοδήποτε δείγμα προγραμμάτων στο οποίο έχουμε έτοιμο μια φυσική περιγραφή x (για παράδειγμα αυτό που αξιοποίησαν οι [83]) και να το μετασχηματίσουμε αξιοποιώντας ένα έτοιμο εργαλείο δημιουργίας απόκρυφου κώδικα. Επιπλέον, θα χρειαστεί να τροποποιήσουμε το δυνατό GENTOKEN λεξιλόγιο. Η συλλογή των token θα πραγματοποιηθεί σαρώνοντας τα δείγμα απόκρυφων προγραμμάτων, αποθηκεύοντας όσα από αυτά εμφανίζονται πάνω από ένα δεδομένο κάτω όριο (π.χ. συχνότητα εμφάνισης πάνω από 1%).

5.4 Ανακεφαλαίωση

Στο παρόν κεφάλαιο παρουσιάστηκαν τρεις εφαρμογές τις οποίες θα επεξεργαστούμε και θα αναλύσουμε σε μελλοντικές εργασίες. Θεωρούμε πως η προσέγγιση ακόμα πιο σύνθετων κατανομών, βασισμένοι σε αρχιτεκτονικές που έχουν παρουσιάσει αξιόλογα αποτελέσματα αποτελεί

ένα εύφορο πεδίο έρευνας αυτού του κλάδου. Στην πρώτη ενότητα σας περιγράψαμε δυο GAN αρχιτεκτονικές για την προσέγγιση της κατανομής του DES και των λύσεων του Proof-Of-Work προβλήματος που εμφανίζεται στο crypto-mining. Στην δεύτερη ενότητα, σας παρουσιάσαμε τα θεωρητικά θεμέλια που συνδέουν την κρυπτογραφία με την παραγωγή απόκρυφου κώδικα, τα οποία μας ώθησαν στην αξιοποίηση μοντέλων Deep Learning για την παραγωγή του. Στην τρίτη ενότητα, προτείνουμε ένα τέτοιο μοντέλο, το οποίο παρουσίασε αξιόλογα αποτελέσματα και περιγράφουμε τον τρόπο με τον οποίο θα μπορέσουμε να το αξιοποιήσουμε.

Τα πρώτα τέσσερα κεφάλαια εφοδίασαν τον αναγνώστη με μια σφαιρική και κατάλληλα δομημένη αναφορά σε σπουδαίες εφαρμογές της Μηχανικής Μάθησης στην Κρυπτογραφία. Τα μοντέλα που παρουσιάστηκαν, πέρα από νέα ερεθίσματα και ιδέες που ευελπιστούμε να γέννησαν, μπορούν να αξιοποιηθούν από ερευνητές του κλάδου ως αρχική αρχιτεκτονική ενός μοντέλου πάνω σε ένα παρεμφερές πρόβλημα (όπως συνηθίζεται στην επιστήμη της Μηχανικής Μάθησης). Το τελευταίο κεφάλαιο προτείνει μοντέλα τα οποία σκοπεύουμε να αξιοποιήσουμε σε καινοτόμες εφαρμογές, με σκοπό την περαιτέρω ανάπτυξη του κλάδου.

Βιβλιογραφία

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative adversarial nets". In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (eds.), Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 2672–2680, 2014a. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.

[2] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.

[3] Kowsic Jayachandiran. *A Machine Learning Approach for Cryptanalysis*. 2018.

[4] Martin Abadi, David G. Andersen. *Learning To Protect Communications With Adversarial Neural Cryptography*. 2016.

[5] Dan Boneh, Victor Shoup. *A Graduate Course in Applied Cryptography*, September 2017.

[6] Benjamin Timon, UL Transaction Security, Singapore, *Non-Profi*

led Deep Learning-Based Side-Channel Attacks. 2018.

[7] Liran Lerman, Stephane Fernandes Medeiros,Gianluca Bontempi, and Olivier Markowitch. Quality and Security of Information Systems, D´epartement d'informatique, Universit´e Libre de Bruxelles, Brussel, Belgium. Machine Learning Group, D´epartement d'informatique, Universit´e Libre de Bruxelles, Brussel, Belgium. *A Machine Learning Approach Against a Masked AES*. 2018.

[8] Mohammed M. Alani. Department of Computing, Middle-East College, Muscat, Sultanate of Oman. *Neuro-Cryptanalysis of DES and Triple-DES*. 2012.

[9] F. Rosenblatt, *THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN*, 1958

[10] B. Widrow, *An Adaptive 'Adaline' Neuron Using Chemical 'Memistors'*, Stanford Electronics Laboratories Technical Report 1553-2, October 1960.

[11] Cortes, Corinna, and Vladimir Vapnik. *PSupport-vector networks*, Machine learning 20.3 (1995): 273-297

[12] Ho, Tin Kam (1995). *Random Decision Forests*. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282

[13] Sepp Hochreiter, Jurgen Schmidhuber, *LONG SHORT-TERM MEMORY*, Neural Computation 9(8):1735-1780, 1997

[14] Ian Goodefellow, Yoshua Bengio, Aaron Courville, *Deep Learning-MIT*, 2017

[15] Kunihiko Fukushima, *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*, NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan , 1980

[16] Kumar Chellapilla, Sidd Puri, Patrice Simard, *High Performance Convolutional Neural Networks for Document Processing*, 2006

[17] Ruslan Salakhutdinov, Geoffrey Hinton, *Deep Boltzmann Machines*, 2009

[18] Kevin Jarrett, Koray Kavukcuoglu, Marc' Aurelio Ronzato, Yann LeCun, *What is the Best Multi-Stage Architecture for Object Recognition?*, The Courant Institute of Mathematical Sciences New York University, 715 Broadway, New York, NY 10003, USA, 2009

[19] Ciresan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). *Multi-column deep neural network for traffic sign classification*. Neural Networks , 32, 333–338.

[20] Le, Q., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., and Ng, A. (2012). *Building high-level features using large scale unsupervised learning*. In ICML'2012 .

[21] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). *ImageNet classification with deep convolutional neural networks*. In NIPS'2012 .

[22] Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B., and Andrew, N. (2013). *Deep learning with COTS HPC systems*. In S. Dasgupta and D. McAllester, editors, Proceedings of the 30th International Conference on Machine Learning (ICML-13), volume 28 (3), pages 1337–1345. JMLR Workshop and Conference Proceedings.

[23] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014a). *Going deeper with convolutions.* Technical report, arXiv:1409.4842.

[24] Hornik, K., Stinchcombe, M., and White, H. (1989). *Multilayer feedforward networks are universal approximators.* Neural Networks , 2, 359–366

[25] Cybenko, G. (1989). *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals, and Systems, 2, 303–314.

[26] Kurt Hornik. *Approcximation Capabilities of Multilayer Feedforward Networks*, Neural Networks, Vol. 4, pp. 251-257. 1991

[27] P. C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems* in Advances in Cryptology - CRYPTO'96 (N. Koblitz, ed.), vol. 1109 of Lecture Notes in Computer Science, pp. 104-113, Springer, 1996.

[28] NIST, textitData Encryption Standard (DES), FIPS PUB 46, Federal Information Processing Standards Publication 46, U.S. Department of Commerce (1977)

[29] Pascal Junod, *Linear Cryptanalysis of DES*

[30] Biham, E., Shamir, A.: *Differential Cryptanalysis of the Full 16-Round DES.* In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 487–496. Springer, Heidelberg (1993)

[31] Matsui, M.: *Linear Cryptanalysis Method for DES Cipher.* In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)

[32] NIST, *Data Encryption Standard (DES)*, FIPS PUB 46–3, Federal Information Processing Standards Publication 46–3 (1999)

[33] Stefan Lucks, *Attacking Triple Encryption*, 1998

[34] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 15, 1929–1958.

[35] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). *What is the best multi-stage architecture for object recognition?.* In ICCV'09 .

[36] Mangard, S., Oswald, E., Popp, T.: *Power Analysis Attacks- Revealing the Secrets of Smart Cards.* Springer, Heidelberg (2007)

[37] Schindler, W.: *Advanced stochastic methods in side channel analysis on block ciphers in the presence of masking.* J. Math. Crypt. 2(3), 291–310 (2008)

[38] Schindler,W., Lemke, K., Paar, Ch.: *A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B.* (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)

[39] Oswald, E., Mangard, S.: *Template attacks on masking—resistance is futile.* In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 243–256. Springer, Heidelberg (2006)

[40] DPAContest V4 (2013). http://www.dpacontest.org/home/

[41] Benedikt Gierlichs, Kerstin Lemke-Rust, Christof Paar. *Templates vs. Stochastic Methods –A Performance Analysis for Side Channel Cryptanalysis.* 2006

[42] E. Zachos, A. Pagourtzis, and P. Grontas. Computational Cryptography. E-book (in Greek), Kallipos Publications, 2015. Available Online at: http://hdl.handle.net/11419/5439.

[43] Grontas, A. Pagourtzis, A. Zacharakis, and B. Zhang. *Towards everlasting privacy and efficient coercion resistance in remote electronic voting.* In Proc. 3rd Workshop on Advances in Secure Electronic Voting (VOTING '18), in association with Financial Cryptography and Data Security 2018 (FC'18), Lecture Notes in Computer Science, vol 10958, pp. 210-231. Springer.

[44] Grontas, A. Pagourtzis, A. Zacharakis. *Coercion Resistance in a Practical Secret Voting Scheme for Large Scale Elections.* In Proc. ISPAN-FCST-ISCC 2017, June 21-23, 2017, Exeter, UK, pp. 514-519, IEEE Computer Society 2017.

[45] H. Maghrebi, T. Portigliatti, and E. Prouff, *Breaking cryptographic imple- mentations using deep learning techniques."* Cryptology ePrint Archive, Report 2016/921, 2016.

[46] E. Cagli, C. Dumas, and E. Prouff, *Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures* pp. 45-68. Cham: Springer International Publishing, 2017.

[47] E. Prouff, M. Rivain, and R. Bevan, *Statistical Analysis of Second Order Differential Power Analysis* IEEE Transactions on Computers, vol. 58, pp. 799-811, June 2009.

[48] Emmanuel Prouff and Remi Strullu and Ryad Benadjila and Eleonora Cagli and Cecile Dumas, *Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database.* Cryptology ePrint Archive, Report 2018/053, 2018. https://eprint.iacr.org/2018/053.

[49] ChipWhisperer Website."https://newae.com/tools/chipwhisperer/.

[50] Olga Taran, Shideh Rezaeifar, and Slava Voloshynovskiy. *Briding machine learning and cryptography in defence against adversarial attacks.* 2018

[51] Maria Mushtaq, Ayaz Akram, Muhammad Khurram Bhatti, Maham Chaudhry, Muneeb Yousaf, Umer Farooq, Vianney Lapotre, Guy Gogniat. *Machine Learning For Security: The Case of Side-Channel Attack Detection at Run-time.* 2018

[52] R.L Rivest, A. Shamir, L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* 1977

[53] Y. Yarom and K. Falkner. *"Flush+reload: A high resolution, low noise, l3 cache sidechannel attack,"* in USENIX Security 14, p. 719. 2013.

[54] D. Gruss, C. Maurice, K. Wagner, and S. Mangard. *"Flush+flush: A fast and stealthy cache attack,"* in DIMVA, pp. 279–299, 2016.

[55] https://www.spec.org/benchmarks.html, 2018.

[56] M. B. Christopher, *PATTERN RECOGNITION AND MACHINE LEARNING.* Springer-Verlag New York, 2016.

[57] K. Kalaiselvi , Anand Kumar.*Enhanced AES cryptosystem by using genetic algorithm and neural network in S-box.* 2016 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC).

[58] Yuan, X., He, P., Zhu, Q., Bhat, R.R., Li, X.:*Adversarial examples: Attacks and defenses for deep learning.* arXiv preprint arXiv:1712.07107 (2017)

[59] Carlini, N., Wagner, D.: *Towards evaluating the robustness of neural networks.* In: Security and Privacy (SP), 2017 IEEE Symposium on, IEEE (2017) 39-57

[60] Goodfellow, I.J., Shlens, J., Szegedy, C.: *Explaining and harnessing adversarial examples.* arXiv preprint arXiv:1412.6572 (2014)

[61] Dong, Y., Liao, F., Pang, T., Su, H., Hu, X., Li, J., Zhu, J.: *Boosting adversarial attacks with momentum.* 2017.

[62] Kurakin, A., Goodfellow, I., Bengio, S.: *Adversarial examples in the physical world.* arXiv preprint arXiv:1607.02533 (2016)

[63] Tramer, F., Kurakin, A., Papernot, N., Boneh, D., McDaniel, P.: *Ensemble adversarial training: Attacks and defenses.* arXiv preprint arXiv:1705.07204 (2017)

[64] Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: *Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models.* In: Proceedings of the 10th ACM Workshop on Arti

cial Intelligence and Security, ACM (2017) 15-26

[65] Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: *Distillation as a defense to adversarial perturbations against deep neural networks.* In: Security and Privacy (SP), 2016 IEEE Symposium on, IEEE (2016) 582-597

[66] Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: *On detecting adversarial perturbations.* arXiv preprint arXiv:1702.04267 (2017)

[67] Hendrycks, D., Gimpel, K.: *Early methods for detecting adversarial images.* (2017)

[68] Lee, S., Lee, J.:*Defensive denoising methods against adversarial attack.* (2018)

[69] Gu, S., Rigazio, L.: *Towards deep neural network architectures robust to adversarial examples.* arXiv preprint arXiv:1412.5068 (2014)

[70] Lecun, Y., Cortes, C., Burges, C.J.: *The mnist database of handwritten digits*, 2009. URL http://yann.lecun.com/exdb/mnist (2009)

[71] Xiao, H., Rasul, K., Vollgraf, R.: *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.* arXiv preprint arXiv:1708.07747 (2017)

[72] https://github.com/tensorflow/cleverhans

[73] https://github.com/carlini/nn_robust_attacks

[74] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. *"The simon and speck families of lightweight block ciphers,"*. Cryptology ePrint Archive, Report 2013/404, 2013, https://eprint.iacr.org/2013/404

[75] C. McCoy, *"Simon speck ciphers,"*. https://github.com/inmcm/SimonSpeckCiphers, 2016

[76] Glorot, X. and Bengio, Y. (2010). *Understanding the difficulty of training deep feedforward neural networks.* In AISTATS'2010.

[77] Kingma, D. and Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980

[78] https://nlml.github.io/neural-networks/adversarial-neural-cryptography/

[79] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. *Improved techniques for training GANs*. CoRR, abs/1606.03498, 2016. URL https://arxiv.org/abs/1606.03498.

[80] Simon Aebersold, Krzysztof Kryszczuk, Sergio Paganoniy, Bernhard Tellenbach, Timothy Trowbridge. *Detecting Obfuscated JavaScripts using Machine Learning*. Zurich University of Applied Sciences, Switzerland, yGovCERT.ch, Reporting and Analysis Centre for Information Assurance MELANI. 2016.

[81] Dennis Hofheinz, John Malone-Lee, Martijn Stam *Obfuscation for Cryptographic Purposes*. 2007

[82] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. *On the impossibility of obfuscating programs*. In Advances in Cryptology - CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 1–18. Springer-Verlag, 2001. Full version available at http://eprint.iacr.org/2001/069/.

[83] Pengcheng Yin, Graham Neubig. *A Syntactic Neural Model for General-Purpose Code Generation*. 2017

[84] Astor Library https://pypi.org/project/astor/

[85] http://www.statmt.org/book/slides/08-evaluation.pdf

[86] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tom´as Kocisk´y, Fumin Wang, and Andrew Senior. 2016. *Latent predictor networks for code generation*. In Proceedings of ACL.

[87] Maxime Nassar, Youssef Souissi, Sylvain Guilley, Jean-Luc Danger. *RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs*. 2012.

[88] Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. *Learning to generate pseudo-code from source code using statistical machine translation (T)*. In Proceedings of ASE.

[89] Olof Mogren.*C-RNN-GAN: Continuous recurrent neural networks with adversarial training*. Chalmers University of Technology Sweden. 2016

[90] Pyminifier: https://liftoff.github.io/pyminifier/

[91] Bogdan Vasilescu, Casey Casalnuovo, and Premkumar Devanbu. 2017. *Recovering Clear, Natural Identifers from Obfuscated JS Names*. In Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACMSIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17), 11 pages. https://doi.org/10.1145/3106237.3106289

[92] Diajing Liu, Yixuan Tan, Erfan Khoram, Zonfu Yu.*Training deep neural networks for the inverse design of nanophotonic structures*. 2017

[93] Ashima Dua, Amar Cupta. *Inversion of Neural Networks: A Solution to the Problems Encountered by a Steel Corporation*. MIT, 2000.

[94] https://en.wikipedia.org/wiki/Abstract_syntax_tree

[95] https://en.wikipedia.org/wiki/Long_short-term_memory

[96] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tom´as Kocisk´y, Fumin Wang, and Andrew Senior. 2016. *Latent predictor networks for code generation*. In Proceedings of ACL.

[97] Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. *Learning to generate pseudo-code from source code using statistical machine translation (T)*. In Proceedings of ASE.

[98] https://kite.com/python/docs/Crypto.Cipher

[99] Mahdi Mirza, Simon Osindero. *Conditional Generative Adversarial Nets*. 2014

[100] Cryptii website: https://cryptii.com/pipes/hash-function

[101] NLTK https://www.nltk.org/

[102] word2vec algorithm https://code.google.com/archive/p/word2vec/