Kurzlich hat KI und DNN viel entwickelt, gleichzeitig gibt es doch einige Probleme über seine Sicherheit. Backdoor ist eine typische Attack zur DNN Model. Es schädet die Leistung eines Model. In meine Bachlorarbeit, verwende ich TrojanNet als Attack Methode. Es gibt noch Verteidigungsmethode. ANP ist kraftig gegen Backdoor zu verteidigen und das DNN Model zu schützen.

Hier will ich mich zuerst für TrojanNet vorstellen. TrojanNet wird in einen Model eingebettet und beim Triggers activtiert, um das Model falsh zu entscheiden. Außdem kann es doch auf dem clean data ruhig bleiben. Z.B. ein Foto wird als Berge sortieren. Und wenn mann eine kleine Trigger in der Zentral des ganz Foto eingefüget hat, die wird beim Model als Goldfish erkennen. TrojanNet hat auch seine eigene besonders Eigenschaft. Die wird nicht das Origial Model verändert. Und nach dem Einbetten, braucht TrojanNet nicht Retraining. Auf dem Struktur Ebene, kann man sehen. Das ganz Model hat eine unüblich teil. Das Data vom Inputlayer wird beim zwer Teile paraell verarbeiten. Beides stellen sich seine eigene Output und passen zum nächste Layer. Das heißt, die 2 Teil sind nicht abhängig voneinander und hat keine Einfluss aufeinander. Folgend Layer entscheidet, welch Output wird als Ergebis ausgegeben.

Dann will ich auch für ANP eine vorstellen machen. ANP hat 3 Schritten, um das Backdoor im Model zuerkennen. Zum 1.Schritt gibt es eine Vorschlag : TrojanNet is sensitiv to perturbations . Mann kann Störung im Model eingeführen. Um eine Störung einzuführen, kann man das Gewicht und Bias jedes Neurons verändern. Mann soll auch die Störung optimieren. Wir können es durch Error Rate erreichen. Die Störung hat die Leistung des Models beeinflusst. Wenn die Error Rate eines Model auf Maximal erreichen, ist die Störung auch auf Besten. Außerdem gibt es noch eine wichtig Parameter. Robust wird beim hier definiert. Das beschreibt: wie viel Prozent hat die Ausgabe eines Model auf dem test data set nicht verändert, vor und nach der Störung. Es bewertet, ob eine Model leicht bei eine Störung beeinfluss ist. Hier gibt es eine Beispiel. Die erste Zeile ist die richtig Labels der test data. 2. Zeile ist die Ergebnise vor der Störung. 3.Zeile ist die Ergbnise nach der Störung. Vergleichen die 1 und 2 Zeile kann man das Error Rate vor der Störung bekommen. Vergleichen die 1 und 3 Zeile kann man das Error Rate nach der Störung bekommen. Wenn wir uns nur auf 2 und 3 Zeile konzentieren. Das können wir das Robust erhalten. Hier soll mann aufpassen. Eine veränderung eines Ausgabe bringt nicht unbedingt die veränderung des Error Rate mit.

Weiter soll man die Störung noch ausnehmen. Das ist ähnlich zur Störung einleiten. TrojanNet is not sensitive to clean data. Für die andere Richtung, dort soll das Error Rate des Model wider auf minimal erreichen. Und das Parameter Sensitivity definieren wir hier: wie viel Prozent hat die Ausgabe des Model auf dem test set nicht gleich sind, vor und nach der Ausnehmen des Störung.

Zum Schluss, werden wir das Backdoor durch 2 Paramters aus Model abschneiden. Für 2 Vorschlage: TrojanNet is sensitive to perturbations. Das heißt, TrojanNet hat eine niderig Robust. TrojanNet is not sensitive to clean data. Das heißt, TrojanNet soll eine niderig Sensitivity haben. Für jedes Neuron gibt es eine Mask Wert. Das benutzt für die Bewertung, dass ob dieses Neuron mit Backdoor sein. Die wird mit Threshold vergleichen. Und a als eine paramter zu kontrolieren, dass mann sich auf welche Paramter konzentieren soll. Und welche Paramter hat mehr beeinfluss auf der Prozess von Entscheiden zur backdoor Neurons erkennen. Wenn das Mask Wert kleiner als Threshold, dieses Neuron soll abgeschneidet werden.

Dann möchtei ich meine eigene Implementierung vorstellen. Zuerst soll ich für Experiment vorbereiten. Meine Implementierung wird als gray box gearbeitet. Das heißt, ich habe nur teilweise Informationen kennengelernt. Eine kleine data set mit Train set, Verificaition set, Clean test set und Poisoned data set. Außerdem weise ich nur das Stuktur des Model, sondern nicht das Gewicht und Bias. Dann fange ich an, die ANP am TrojanNet zu verwenden.

Um die Störung einzufügen, habe ich das Model wider mal trainiert. Retraining ist eine gut Methode, um Error Rate auf Maximal zu erreichen. Hier definiert ich eine neue loss funktion, also negativ cross entropy funktion. Hier vielleicht haben Sie eine Frage? Warum benutze ich hier Retrainning, sondern nicht eine angemeine Störung für jeder Neurons. 1. Das ist eine gray box. Ich kann nicht jeder Gewicht direkt verändern. Und die Störung könnte genauer und flexiber beim training eingefügt werden.

Dann ist der wichtiger Parameter Robust. Ich will die Robust für jeders Teil des Model annehmen. Das Data wird vom Input zur TrojanNet und target model gepasst. Die Beises Teile wird seine eigene Output Vektor ausgeben. Output Vektor ist eine andere Formulierung vom Ergebnis. Das könnte durch Index nach Ergebnis übergesetzt werden. Hier erwarte ich das Robust für TrojanNet bei der Unterschied zwischen die Ausgabe vor und nach der Störung. Und erwartet ich auch eine andere Robust für target model. Hier gibt es eine Punkt, was sollen wir aufpassen. In diesem Bild, ist die Werte auf y Achse Loss. Das heißt, wie viel Prozent hat die Ausgabe verändert werden. Wenn Neurons eine hoch Loss behalten, das bedeutet, dass diese Neuron eine schlechte Robust haben. Robust sollte hier mit 1-Loss definiert werden.

Für der ähnlich Fall, de-Perturbation, habe ich mit Retrain geschafft. Und das Loss Funktion benutze ich normale positive cross entropy Loss Funktion. Für das Parameter Sensitivity, soll ich hier aufpassen, dass Y Wert Loss kann direkt Sensitivity in diesem Prozess formuliert werden. Ein Neuron hat groß Loss bedeutet, das hat sehr gut Sensitivity auf dem Clean data set.

Weiter habe ich diese Prozess 5 mal widerholt. In der 1. Interation. TrojanNet und target model haben Störung bekommt. Und Am Anfang für grüne Linie hat TrojanNet dirkt schlecht Robust verhalten, aber für rot Linie, das target Model hat gut Robust am Anfang. Bis 22 Epoch hat die Störung groß genug, die Ausgabe des target Model hat verändert. Dann werden die Störung weg machen. Weil das TrojanNet ist nicht sensitiv auf clean data. Deshalb hat die Ausgabe dieses Teil immer sehr wenig verändert. In dieses Prozess wird das Model auf clean data getrainert. Das bedeutet, Model wird dabei verbessert könnte. Aufpassen. Nach der 1. Interation. Das TrojanNet verändert sich nicht mehr. Weil das TrojanNet hat schon zu viel Störung bekommen hat. Und die hat keine Verbesserung auf dem letzten Interarion. Das heißt, das Teil des TrojanNet könnte nicht mehr schlechter werden. Und außerdem kann man sehen. Für das Robust des target model in der weiter Interation wird die Ausgabe am Anfang dirkt verändert. Das bedeutet, die Störung hat auch beeinfluss auf dem Robust eines Model. Deshalb soll man das Backdoor Neuron von origia Model abschneiden, sondern nicht das Model, welche hat Störung Prozess viel mal erfährt hat.

Zum Schluss werde ich mich um Mask Wert kümmern, Für 2 verschiedne Teile Sequenial\_1 und Sequenial\_2, rechne ich seine eigene Robust und Sensitivity Wert. Mit dem Vergleichen kann man kennenlernen, dass Sequential\_2 hat schlechter Robust und Sensitivity. Wenn man sich hier mehr auf Robust konzentiert, ist es nur im 1.Interation sinnvoll. Sondern benutzen wir hier mit Sensitivity. Das Unterschied zwischen TrojanNet und target model ist mehr offensichtlich. Und das Threshold kann mann eine kleine Zahl wie 0.2 ersetzen.