GDALR: An Efficient Model Duplication Attack on Black Box Machine Learning Models

Nikhil Joshi
Security Researcher
Payatu Software Labs LLP
Pune, India
nikhilj@payatu.com

Rewanth Tammana
Security Consultant
Payatu Software Labs LLP
Pune, India
rewanth@payatu.com

***Abstract*—Trained Machine learning models are core components of proprietary products. Business models are entirely built around these ML powered products. Such products are either delivered as a software package (containing the trained model) or they are deployed on cloud with restricted API access for prediction. In ML-as-a-service, users are charged per-query or per-hour basis, generating revenue for businesses. Models deployed on cloud could be vulnerable to Model Duplication attacks. Researchers found ways to exploit these services and clone the functionalities of black box models hidden in the cloud by continuously querying the provided APIs. After successful execution of attack, the attacker does not require to pay the cloud service provider. Worst case scenario, attackers can also sell the cloned model or use them in their business model.**

Traditionally attackers use convex optimization algorithm like Gradient Descent with appropriate hyper-parameters to train their models. In our research we propose a modification to traditional approach called as GDALR (Gradient Driven Adaptive Learning Rate) that dynamically updates the learning rate based on the gradient values. This results in stealing the target model in comparatively less number of epochs, decreasing the time and cost, hence increasing the efficiency of the attack. This shows that sophisticated attacks can be launched for stealing the black box machine learning models which increases risk for MLaaS based businesses.

***Index Terms*—Machine Learning, Model Stealing**

I. Introduction

Machine learning and Deep learning is getting advanced and increasingly popular these days. Deep learning has proved its effectiveness in solving problems like classification, regression and clustering. Deep learning models are currently used by businesses to solve problems in a broad range of domain like image/video processing [10], speech recognition [11], Cyber security [12], medicine [13] and finance [14]. Businesses invest considerable resources while training Machine Learning models and hosting them. Hence, trained models possess high business value and are treated as intellectual property.

Generally, trained models are made available to end users via two major ways (a) With the installation package: where the use case demands to run the models locally and need to work without any network connectivity (b) Cloud based models: The models are deployed on cloud and API access is provided to end user, for which he/she will be charged. Models provided with installation packages can be easily stolen by an adversary. We assume that in both the cases 'a' and 'b' there is very limited access to prediction APIs and are outputting confidence scores of best class or best n classes.

Although, these APIs provide a restricted access to trained models, outputs from APIs can still leak information about the model. Adversary can interpret the output as Labels and use them to train its own model as discussed in [1].

II. Related Work

Stealing black box machine learning models is gaining popularity these days. To replicate a black box model we need several parameters like architecture, hyper-parameters, hyper-surfaces, training data and so on. Attackers started cloning these black box models solely based on their functionalities.

In 2005, D.Lowd and C.Meek first introduced these kind of reverse engineering attacks [16] on linear classifiers of machine learning models. They proposed efficient algorithms to steal the parameters of the linear classifiers. Their proposed approach requires a lot of querying to the API service which is considered as disadvantageous in real world.

Later in 2016, F. Tram'er proposed a different approach but even that presumes the attacker knows the model type of the target model. Efficient model duplication attacks are possible if the service provider discloses the confidence score of the labels [1]. Researchers were able to clone the model without any discernible difference after 1,485 queries and just 650 queries in case of digit-recognition [1]. Attackers leveraged this vulnerability and exploited several service providers and started selling the cloned models at lower costs which made the service providers lose a fortune.

Later in 2017, N Papernot discussed about these kind of attacks on service providers like MetaMind but their technique also presumes the attacker to have minimal information about the target model like its model type [9].

Recently in 2018, Binghui Wang and Neil Zhenqiang Gong proposed an intuitive way to steal the hyper-parameters from black box models for different model types like ridge regression, logistic regression, support vector machine, and neural Network using cross validation [8]. Attackers can use this improved technique to clone the functionality of the models.

Our proposed work, efficiently produces a clone of target model than methods discussed in this section, without knowing the hyper-parameters of a black box model.

III. BUILDING MACHINE LEARNING MODELS

Machine learning models can be represented as $F : X \rightarrow Y$. Where, F is a function that maps X_i from a sample space X to Y_i from set of target categories Y. Every X_i is a d dimensional feature vector represented as $X_i = \{x_1, x_2, x_3, ..., x_d\}$ and Y_i is c dimensional vector $Y_i = \{y_1, y_2, y_3, ..., y_c\}$ where y_i is confidence score for respective class from c number of classes. Convex optimization algorithms like [2] [6] [7]. or swarm based optimizations [3] [4] [5] etc, can be used to train F. Optimization algorithm minimizes the cost function $F_c(Y, T)$ where T is set of target class for samples in X and Y is predicted class. Finally, we get an F that accurately maps samples from X to its respective class c_i for $i \in [0, c-1]$. These models are deployed on cloud and are made available to end user via an API.

IV. MODEL DUPLICATION ATTACK

Attacker's aim is to replicate F as F' such that $F'(X) \approx F(X)$. Attacker possesses a data set X_a with n samples $\{X_{a1}, X_{a2}, X_{a3}, ..., X_{an}\}$. X_a is sent to F via prediction APIs to produce $Y_a = \{Y_{a1}, Y_{a2}, Y_{a3}, ..., Y_{an}\}$. Remember, every query costs an amount to attacker, therefore attacker's goal is also to minimize n. Depending on how the APIs were developed, Y_{ai} may or may not be a vector of size c. APIs may only provide the confidence score for top k classes $C_a = \{C_{a1}, C_{a2}, ..., C_{ak}\}$ or the best class. In this case confidence score for every class not in C_a is considered as 0 ($zero$). If API returns just the best class, then Y_{ai} for predicted class is considered as 1 (one) and rest as 0 ($zero$).

$$Y_{ai} = \begin{cases} C_{ai}, & \text{if score of } c_i \in C_a \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Y_a resulting from (1) can be used by attacker to train F'.

A. Traditional Model Stealing Approach

Traditional approach involves leveraging optimization techniques discussed [2] [3] [4] [5] [6] to minimize the cost mentioned in (2) to train F'.

$$F_{ac} = Loss(F'(X_a), Y_a) \tag{2}$$

In our experimentation we have used Mean Squared Error (3) and Log Loss (4) as $Loss$ functions. But it can be replaced by any of the standard cost function like Root mean squared error, Mean Absolute Error, Mean Absolute Percentage Error, etc.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(F'(X_a) - Y_a)^2 \tag{3}$$

$$Logloss = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c}(Y_{aij} \log F'(X_a)_{ij}) \tag{4}$$

Following steps are executed by attacker to steal the model.

1) Select an appropriate model architecture with hyper-parameters.
2) Query F using provided API and generate Y_a for all samples in X_a.
3) Use Y_a as class labels and optimization algorithm to train F' such that $Loss$ (3) (4) is minimum.
4) Continue previous step until the stopping criterion is met.

Above method is proven to be effective in [1] while stealing models with black box access. But assumptions made in (1) can be a drawback. Our modification to training method overcomes the demerits of assuming (1) which is described in next section.

V. GDALR: PROPOSED MODEL STEALING APPROACH

Though the assumption (1) is made while building Y_a, in real world, model F always has some confidence score for classes C'_a that are not visible in C_a, output from prediction APIs. C'_a and C_a follow these properties

$$C'_a \cap C_a = \emptyset \tag{5}$$

$$C'_a \cup C_a = Y_a \tag{6}$$

If the target model F is designed to predict large number of classes (consider a well known image classifiers Inception V3 and Resnet [17] classifies images to 1000 classes [15]) then number of elements in C'_a is naturally larger than that of in C_a. Also, $sum(C'_a)$ is significantly larger than $0(zero)$. The summation of the deviated values in the $Loss$ function defined in (3) and (4) causes noticeable increase in gradients which is defined as

$$\frac{d}{d\theta}Loss$$

Increased gradients is undesired while replicating exact boundaries learnt by F in hyperspace, it also makes the Gradient Descent [6] to abruptly change the parameters θ of F' for constant learning rate l, which hinders in effectively training the cloned model F'. Considering values of C'_a to be zeros leads to inefficient duplication of F. In this case, one can suggest smaller value of learning rate l, but that will take more number of epochs and larger time to train F'. To solve this problem we propose a modification to Gradient Descent that dynamically decreases learning rate l when gradients are higher, causing the optimizer to less intensively adjust the parameters θ of F'

$$g'_i = \tanh(g_i) \tag{7}$$

$$fact_i = \text{abs}\left(g'_i 2\pi \log_{10}\left(\text{abs}\left(g'_i\right)\right)\right) \tag{8}$$

$$l'_i = l_i \cdot fact_i \tag{9}$$

Where l_i, l'_i and g_i are learning rate, modified learning rate and gradients at epoch i respectively. $\tanh$ transforms the gradients g_i to be in range (-1,1). Equation (8) generates a factor $fact_i$ in range (0,1.003851) that is used to scale originally set learning rate l_i to obtain new learning rate l'_i and l'_i is then used by optimizer to optimize θ for respective

epoch i. Figure 1 shows the relationship between gradients g_i and $fact_i$.

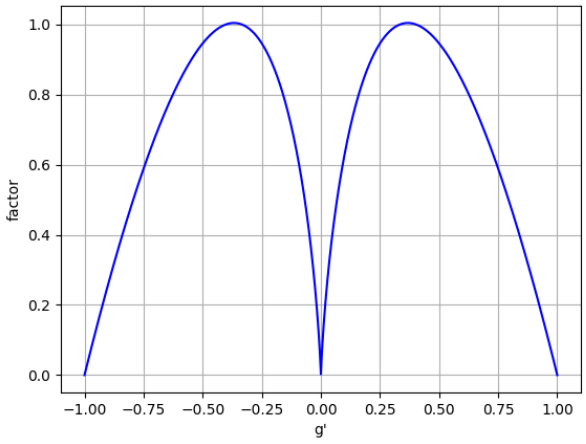

Fig. 1. $fact$ vs g'

We have implemented proposed modification (9) to steal models trained on multiple data sets and found that the modified optimizer is converging significantly better than original Gradient Descent.

VI. EXPERIMENTAL SETUP AND RESULTS

To test our proposed method, we have considered three of the well known classifiers: Logistic Regression, Multi Layer Perceptrons (MLP) and Convolutional Neural Networks (CNNs). Below mentioned datasets are used to train the classifiers and generate our target model F. Wrapper was developed on model F to return only the predicted class label. This results in a simulated black box environment. X_a is a set of random samples from X that was used to train F. Y_a is obtained by querying target model F. Later X_a and Y_a are used to train F'.

The model F initializes the weights and bias on its creation. To demonstrate proof of work (PoW) we explicitly initialize the seed before generating random values. Pytorch [23] has been used to demonstrate the attack. We initiated the duplication attack on the target model F using both traditional and proposed methods. Loss values after every epoch are recorded.

In the results discussed below, T_{Loss} and P_{Loss} refers to Loss value at final epoch for Traditional and Proposed approaches (GDALR) respectively.

A. Iris Data set

We chose to test our proposed algorithm on logistic regression classifier with Iris dataset [18]. The total samples in the Iris dataset are 150 and 33.3% of the entries are taken into the testing set randomly. The data is labelled into 3 classes representing different species of Iris flower.

To prove the working efficiency of our proposed method, we have experimented with different learning rates with values 0.01, 0.05 for 400 epochs. Mean Squared Error (MSE) (3) was selected as loss function.

The results for Duplication Attack on Logistic Regression model are as follows -

For $l = 0.01$, $epochs = 400$

- $T_{Loss} = 0.0849$
- $P_{Loss} = 0.0317$, as shown in 2

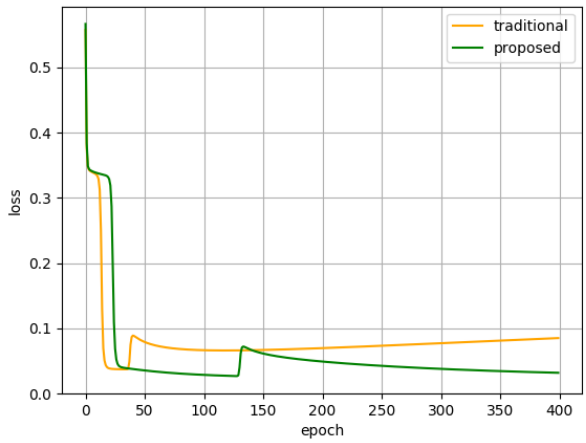

Fig. 2. learning rate = 0.01

For $l = 0.05$, $epochs = 400$

- $T_{Loss} = 0.1233$
- $P_{Loss} = 0.0342$, as shown in 3

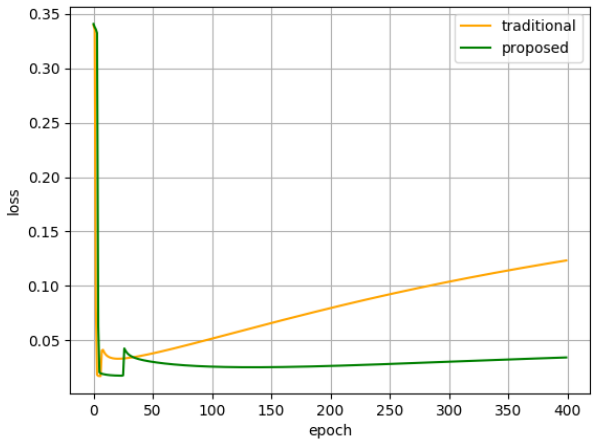

Fig. 3. learning rate = 0.05

B. Liver Disease Data set

Multi Layer Perceptron (MLP) was trained on liver disease dataset. The total samples in liver disease dataset [21] are 345 and 10% of the entries are taken into the testing set.

We have performed classification with different learning rates: 0.001, 0.0001, 0.00001 for 400 epochs with Mean Squared Error (MSE) (3) as loss function. Below mentioned Layer configuration is used to build F'

- layer 1: Linear Input layer with 6 nodes and relu activation function
- layer 2: Linear layer with 16 nodes and relu activation function
- layer 3: Linear layer with 2 nodes and softmax activation function

Results for MLP on Liver disease dataset are as follows - For $l = 0.001$, $epochs = 50$

- $T_{Loss} = 0.0014$
- $P_{Loss} = 5.444 \times 10^{-5}$, as shown in 4

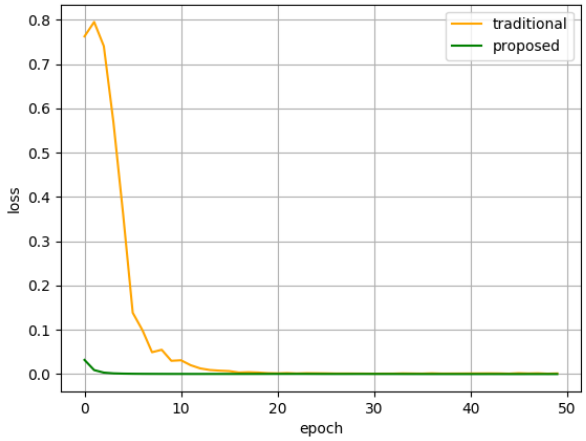

Fig. 4. learning rate = 0.001

For $l = 0.0001$, $epochs = 400$

- $T_{Loss} = 5.934 \times 10^{-5}$
- $P_{Loss} = 3.913 \times 10^{-5}$, as shown in 5

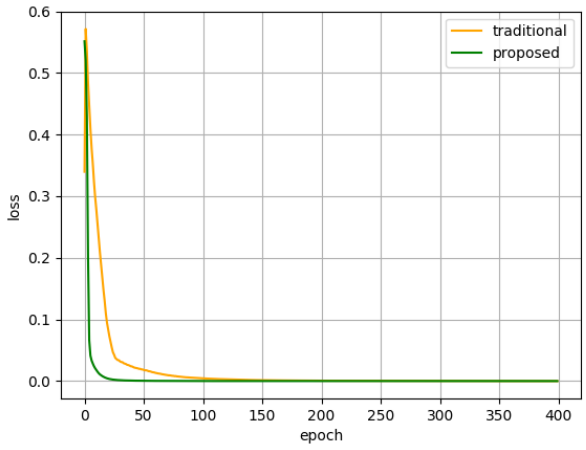

Fig. 5. learning rate = 0.0001

For $l = 0.00001$, $epochs = 400$

- $T_{Loss} = 0.0219$
- $P_{Loss} = 0.0007$, as shown in 6

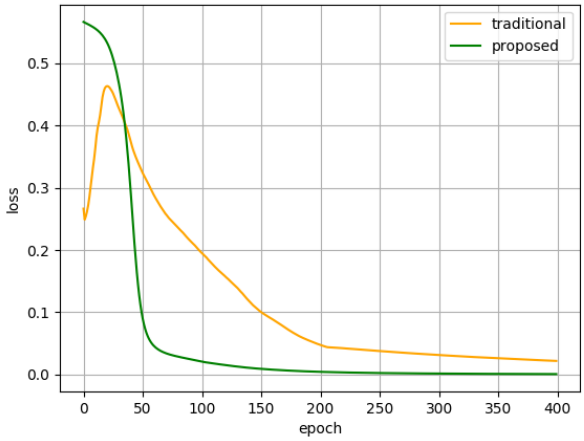

Fig. 6. learning rate = 0.00001

C. Statlog Land satellite Data set

Our proposed method GDALR was tested to steal Convolutional Neural Networks (CNNs) trained on land satellite dataset [22]. The total entries in land satellite dataset are 6435 and 3.1% of the entries are taken into the testing set. The data is labelled into 7 classes.

CNN was trained using different learning rates: 0.001, 0.0001, 0.00001 for 400 epochs with log loss or cross entropy as loss function (4). Following layer configurations is used for F' CNN.

- layer 1: Convolutional layer with 64 output channels and (2×2) kernel
- layer 2: 2D Avg Pooling layer with kernel size 2
- layer 3: Convolutional layer with 128 output channels and (2×2) kernel
- layer 4: 2D Avg. Pooling layer with kernel size 4
- layer 5: Linear layer with 7 output nodes

The results for Duplication Attack on CNN model are as follows -

For $l = 0.001$, $epochs = 400$

- $T_{Loss} = 0.0019$
- $P_{Loss} = 3.576 \times 10^{-7}$, as shown in 7

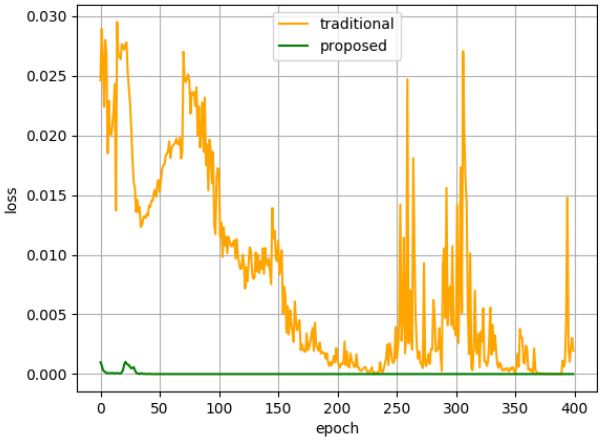

Fig. 7. learning rate = 0.001

For $l = 0.0001$, $epochs = 400$

- $T_{Loss} = 0.0011$
- $P_{Loss} = 3.993 \times 10^{-5}$, as shown in 8

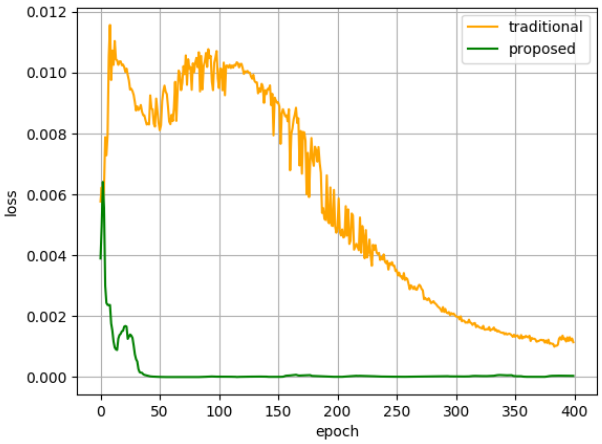

Fig. 8. learning rate = 0.0001

For $l = 0.00001$, $epochs = 400$

- $T_{Loss} = 0.0073$
- $P_{Loss} = 4.184 \times 10^{-5}$, as shown in 9

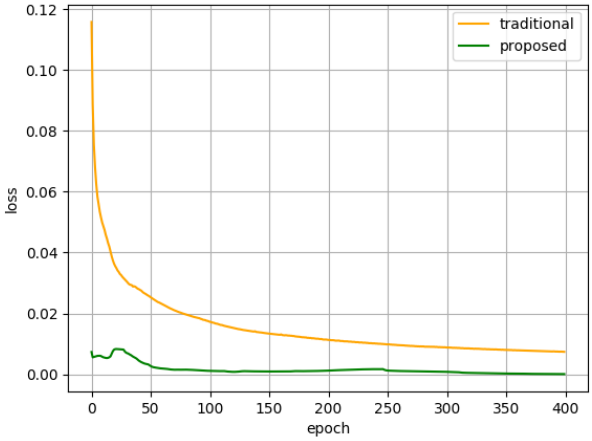

Fig. 9. learning rate = 0.00001

VII. CONCLUSIONS

In this paper, we analyze different approaches to steal black-box learning models. Considering the prior work from other researchers in this area, we demonstrated how an attacker can achieve better performance while stealing machine learning models. We proposed a new method, GDALR (Gradient Driven Adaptive Learning Rate) to perform attack with greater efficiency by modifying the learning rate dynamically after each epoch based on the gradient values.

To test the reliability of GDALR, we experimented with three different classifiers, Linear Regression, MLP (Multi Layer Perceptrons) and CNNs (Convolutional Neural Networks) on different learning rates.

Logistic regression is used for classification on Iris dataset, Fig 2 and 3 show that GDALR method converges better for all learning rates while an undesired increase in loss values is observed with traditional approach.

Faster convergence is observed when MLP classifiers are trained on Liver disease dataset. GDALR powered method achieves better loss than traditional method in equal number of epochs as shown in 4, 5 and 6.

Also, on complex classifiers like CNN trained on Statlog land satellite dataset, Fig. 7 and 8 show that optimization with GDALR not only achieves better loss values but also converges with nearly no fluctuations.

Through our experiments, we illustrated the significant increase in performance of attacks, such as low loss values and better convergence in less number of epochs. Our research, GDALR with its increased performance explains the serious need to rewrite the current countermeasures for MLaaS, an obligatory and interesting area for future work.

REFERENCES

[1] Tramr, Florian, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. "Stealing machine learning models via prediction apis." In *25th USENIX Security Symposium (USENIX Security 16)*, pp. 601-618. 2016. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity16/

[2] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint *arXiv:1412.6980* (2014). [Online] Available: https://arxiv.org/pdf/

[3] Trelea, Ioan Cristian. "The particle swarm optimization algorithm: convergence analysis and parameter selection." *Information processing letters* 85, no. 6 (2003): 317-325.

[4] Dorigo, Marco, Luca Maria Gambardella, Mauro Birattari, Alcherio Martinoli, Riccardo Poli, and Thomas Sttzle, eds. *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings.* Vol. 4150. Springer, 2006.

[5] Davis, Lawrence. "Handbook of genetic algorithms." (1991).

[6] Bottou, Lon. "Large-scale machine learning with stochastic gradient descent." In *Proceedings of COMPSTAT'2010*, pp. 177-186. Physica-Verlag HD, 2010. [Online]. Available: http://khalilghorbal.info/assets/spa/papers/

[7] Boyd, Stephen, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004. [Online]. Available: https://pdfs.semanticscholar.org/a756/

[8] Wang, Binghui, and Neil Zhenqiang Gong. "Stealing hyperparameters in machine learning." In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 36-52. IEEE, 2018. [Online]. Available: https://arxiv.org/pdf/

[9] Papernot, Nicolas, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. "Practical black-box attacks against machine learning." In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506-519. ACM, 2017. [Online]. Available: https://arxiv.org/pdf/

[10] Young, Tom, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. "Recent trends in deep learning based natural language processing." *IEEE Computational IntelligenCe Magazine* 13, no. 3 (2018): 55-75. [Online]. Available: https://ieeexplore.ieee.org/iel7/10207/8416963/

[11] Zhang, Zixing, Jrgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Bjrn Schuller. "Deep learning for environmentally robust speech recognition: An overview of recent developments." *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, no. 5 (2018): 49. [Online]. Available: https://arxiv.org/pdf/

[12] HB, Barathi Ganesh, Prabaharan Poornachandran, and Soman KP. "Deep-Net: Deep Neural Network for Cyber Security Use Cases." *arXiv preprint arXiv:1812.03519* (2018). [Online]. Available: https://arxiv.org/pdf/

[13] Shickel, Benjamin, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. "Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis." *IEEE journal of biomedical and health informatics* 22, no. 5 (2018): 1589-1604.

[14] Heaton, J. B., Nicholas G. Polson, and Jan Hendrik Witte. "Deep learning in finance." *arXiv preprint arXiv:1602.06561* (2016).

[15] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818-2826. 2016.

[16] Lowd, Daniel, and Christopher Meek. "Adversarial learning." In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 641-647. ACM, 2005.

[17] Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[18] Fisher, Ronald. "Iris flower dataset.", 1936.

[19] Devasena, C. Lakshmi, T. Sumathi, V. V. Gomathi, and M. Hemalatha. "Effectiveness evaluation of rule based classifiers for the classification of iris data set." *Bonfring International Journal of Man Machine Interface* 1, no. Special Issue Inaugural Special Issue (2011): 05-09. [Online]. Available: http://www.journal.bonfring.org/papers/mmi/volume1/BIJMMI-01-1002.pdf

[20] Ramana, Bendi Venkata, M. Surendra Prasad Babu, and N. B. Venkateswarlu. "A critical study of selected classification algorithms for liver disease diagnosis." *International Journal of Database Management Systems* 3, no. 2 (2011): 101-114. [Online]. Available: http://www.academia.edu/download/38398988/3211ijdms07.pdf

[21] BUPA Liver Disorders Dataset. UCI repository of machine learning databases. Available from https://archive.ics.uci.edu/ml/machine-learning-databases/liver-disorders/bupa.data

[22] Statlog (Landsat Satellite) Dataset, UCI Machine Learning Repository.

[23] Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch." (2017). [Online]. Available: https://openreview.net/pdf?id=BJJsrmfCZ