Hybrid Statistical Modeling for Anomaly Detection in Multi-Key Stores Based on Access Patterns

Tiberiu Boros[1][a] and Marius Barbulescu[2]
[1]*Security Coordination Center, Adobe Systems, Bucharest, Romania*
[2]*Cloudops, CES Security Solutions, Adobe Systems, Bucharest, Romania*

Keywords: Machine Learning, Statistical Modeling, Multi-Key Store, Access Pattern, Anomaly Detection.

Abstract: Anomaly detection in datasets with massive amounts of sparse data is not a trivial task, given that working with high intake data in real-time requires careful design of the algorithms and data structures. We present a hybrid statistical modeling strategy which combines an effective data structure with a neural network for Gaussian Process Modeling. The network is trained in a residual learning fashion, which enables learning with less parameters and in fewer steps.

1 INTRODUCTION

Secrets management platforms (SMPs) are the industry standard for managing and protecting sensitive information such as passwords, private keys, and other secrets. Leveraging the features provided by these platforms helps eliminating secret sprawl and reducing the exposure of credentials. Since these platforms hold access keys to private services and systems, they are also a potential target for malicious users. Given the delicate nature of the content, it is exceedingly important that the data is kept secure and away from unauthorized access.

Relying on SMPs for storing and retrieving secrets has the following advantages: (a) one does not have to implement any custom secrets management strategy in order to avoid locally storing secrets on production machines; (b) SMPs come with everything you need to implement the least privilege access model and (c) SMPs keep records on what has been accessed when and how – which means that in the case of an incident you have a valuable source of logs that can be used in the investigation.

Our research explores if SMP logs are suitable to detect a compromise. In other words, we are focused on access pattern modeling for anomaly detection in secrets consumption. To be precise, we detect the unauthorized access to credentials, while detection of credential theft is out-of-scope for our research. While generally applicable, our constraints state that the object (secret) is retrieved based on a combination of categorical values (keys). In a typical scenario, the lookup multi-key is a combination of say customer, namespace, and path. Fine-grained modeling is obtained by adding additional attributes such as: (a) hosting-based information such as the autonomous system number (ASN), region (Europe, US, etc.), country, city, IP address, etc. and (b) time-lag features such as time of day (ToD), day of week (DoW), week of month (WoM), etc.

We start with a brief overview of the related work (Section 2), we describe the dataset we are using in our experiments (Section 3.1), and we introduce our methodology (Section 3.2). Finally, we provide an evaluation (Section 4) and deliver our conclusions in Section 5. The key contributions of this work touch the anomaly detection modeling process itself, and the hybrid algorithmic/machine-learning approach we had to take, to mitigate issues with processing a high volume of input data.

2 RELATED WORK

Our work closely relates to Security Information and Event Management (SIEM) log analysis for intrusion detection. Prior work is divided between static rules and machine learning-based methods. Both types of methods are used as a primary filtering mechanism for interesting events. The resulting subset of events is later analyzed and sorted by human experts.

[a] https://orcid.org/0000-0003-1416-915X

Boros, T. and Barbulescu, M.
Hybrid Statistical Modeling for Anomaly Detection in Multi-Key Stores Based on Access Patterns.
DOI: 10.5220/0012621300003705

In *Proceedings of the 9th International Conference on Internet of Things, Big Data and Security (IoTBDS 2024)*, pages 185-190
ISBN: 978-989-758-699-6; ISSN: 2184-4976

Anumol (2015) introduces a statistical ML model for intrusion detection based on network logs. Feng et al. (2017) present a ML user-centered model designed to reduce the number of false positive alerts generated by static rules. Du et al. (2017) employ a deep learning approach for anomaly detection inside system logs. Finally, many other works (Bryant and Saiedian, 2020; Noor et al., 2019; Zhou et al., 2020; Das et al., 2020; Gibert Llauradó et al., 2020; Piplai et al., 2020; Idhammad et al., 2018; Zekri et al., 2017; Osanaiye et al., 2016) employ some type of machine learning algorithm for solving security related problems, by analyzing some type of log data, whether we are talking about application or a network log. Our work is focused on a subset of security-related modeling problems that share the following items:

- **High Volume of Data.** We focus on operating with a high volume of input data, that is specific to application/system logs for large intranet infrastructures and network logs for medium intranet configurations.
- **Low Resourced Computational Environment.** The computational resources used by our approach are minimal, especially when compared with state-of-the-art deep learning models, that use billions of parameters. This is because we reduce the computational effort required to model an entities' behavior, by employing a hybrid ML/data-structure approach.
- **Mixed Numerical and Categorical Attributes.** Our input data is a mixture of categorical and numerical values. We mention that the target attributes (the attributes we want to model), should be preponderantly numerical. Otherwise, the data-structure we use in our hybrid approach is not efficient.
- **Skewed Numerical Ranges.** Finally, our approach addresses a corner case that poses serious issues to neural networks with small number of parameters, but also affects the training of large models – data with a non-uniform variance. To be precise, if one would cluster the input data based on the variance of the output targets, he would obtain high and low-variance clusters. When modeling the output for both types of data clusters at the same time, larger variance has a higher impact on the overall loss. Thus, the network is unlikely to balance automatically, unless some mathematical trick is applied, such as weighted loss.

3 PROPOSED METHODOLOGY

As previously mentioned, our journey starts with the dataset description (Section 3.1), to give a better understanding on how the input looks and how it can be interpreted. Then, we introduce our proposed hybrid method, and we discuss several issues and their mitigation using a residual learning scheme (Section 3.2).

3.1 Dataset Description

Our reference dataset is a collection of access logs for an SMP. The objects (secrets) in our dataset, are accessed via a namespace (categorical value) and a path. To identify the client, the logs contain the Customer ID, a hash of the token used to authenticate and the source IP address. Additionally, the log contains the status of the operation (successful/failed – with reason), and the type of the request, which takes one of the following discrete values: read (R), create (C), update (U), delete (D) and list (L).

Data preprocessing is crucial to any machine learning (ML) approach, and in our case, we first analyzed activity from adversary emulations that mimicked real-world scenarios, in order to have a better understanding on current attack patterns that malicious users follow. This revealed that:

(a) It is impossible to say if a single event is malicious or not, out of context.

(b) Adversary emulations showed activity that is associated with the discovery phase of any attack (listing of secrets on specific paths).

(c) Automations, such as secret cycling, resemble discovery and lateral movement phases of an attack, but there are certain cues and patterns that highlight its benign nature.

(d) Operations performed by an attacker are likely list and read. Update operations are avoided, since they could potentially cause a critical system outage (CSO), and thus the attack would not go undetected.

(e) Malicious activity might come as a spike of operations in certain cases.

Based on this, we converted our dataset into hourly aggregations over the read, create, update, and delete operations. Also, we compute spikes for all 4 operations as the maximum number of events of the same type in a single minute.

Figure 1 represents a histogram of the number of read operations, with the timespan of 1 hour, and the resolution of 1 minute. The total number of operations is 590 with an average of 9.83. As can be easily observed, there is a spike of 50 operations at the

32-minute mark. Similarly, this is computed for the other 4 types of operations. Thus, a single entry in our dataset contains the following attributes:

Categorical values		
Customer	-	A unique identifier for the current customer
Cluster	-	Region-based segregation for secrets
Path	-	A full path to the object (secret) that is being accessed
Namespace	-	A namespace used to segregate the secret landscape. Can be any value chosen by the customer.
Source IP	-	IP address (internal or external) from which the request originated
Numerical		
μ_R, μ_L, μ_C, μ_U, μ_D	-	The average number of operations performed in the timespan for (R) read, (L) list, (C) create, (U) update and (D) delete.
max_R, max_L, max_C, max_U, max_D	-	Maximum number of operations (i.e., spikes), performed in the timespan (1h) for read, list, create, update and delete.
Time lag features (also categorical values)		
ToD	-	Time of Day – a numeric identifier between 0 and 23, specifying the exact timespan within a day, for which the statistics are being computed
DoW	-	Day of Week – a numeric identifier between 0 and 6, with 0 representing Monday and 6 representing Sunday
WoM	-	Week of Month – a numeric identifier between 0 and 4 – like ToD and DoW (not used in the current implementation).
Month	-	Like the above – a numeric identifier between 0 and 11 (not used in the current implementation).

From a quantitative aspect, the timespan of the dataset is 6 weeks and contains a total number of 80398543 records (i.e. hourly aggregations). The unique number of clusters is 7, for 1248 customers, with 49 namespaces and 71832 IP addresses. To get a better understanding on how heterogenous customer behavior is, the mean number deviation of 231.40, while the minimum and maximum values are 0 and 69239 respectively. Similarly, the average number of list operations is 0.47 with a standard deviation of

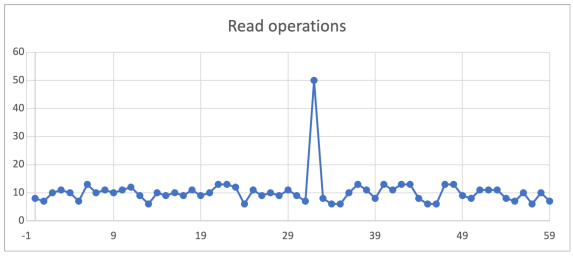

Figure 1: Example histogram of # read operations per minute over the 1-hour timespan.

322.31, with values from 0 to 227907. These numbers suggest that threshold-based alerting is not a good candidate for incident detection, and this was proved in our initial assessment. Furthermore, our analysis showed that normal operations are consistent with the generation of spikes in list and read operations (see Figure 2). This is because customer automations are scheduled periodically, and when triggered, they quickly generate list, read, update, delete and create events. Also, the set operations that are going to be triggered is not straight forward to determine. In Figure 2, images (a) and (e) represent two different automations, one that only performs read operations on the secrets, and another that only lists secrets from the repository. Image (f) is a mixture of both operations, where we can see some linearity between list and read operations, while (b), (c) and (d) have a blend of behaviors (automations that only perform reads and automations that perform list and read operations at the same time). Observing these automations over longer periods of time, we noticed sporadic behaviors that seem to fall out of line with normal operations. This happens in two cases: (a) when multiple automations, that are scheduled with different frequencies end up being triggered at the same time (in one cycle) and then drift apart again and (b) whenever scheduled secret rotation takes place the distribution between the different types of operations gets skewed in favor of update, delete, and create.

3.2 Residual Learning Model

Anomaly detection can be achieved by directly scoring datapoints, or by modeling normal behavior and by checking how divergent is the observed behavior from what is modeled. Our approach implements the second option and, to target attacker behavior, we model read and list operations based on the categorical attributes and on the numerical attributes left behind (update, create, and delete). The reason for using these numerical attributes as input instead of output, is that attackers avoid causing CSOs (updating or deleting secrets would likely generate this). Instead,

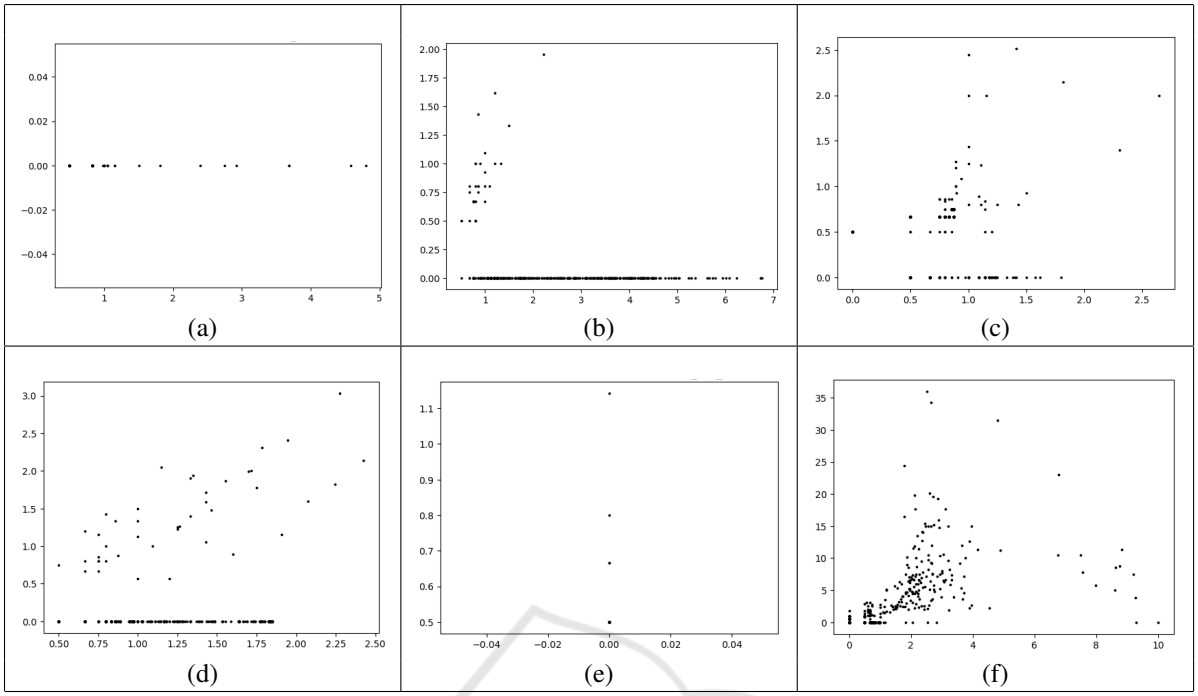

Figure 2: Spikes for read and list operations computed for different automations/customers. The vertical axis represents list operations and the horizonal represents read operations. Images (a) and (e) perform a single type of operation - (a) read operations and (e) list operations. Image (f) is a mixture of both read and write operations and judging by the distribution they are linearly dependent. Images (b) (c) and (d) show both type of behaviours: independent read or list operations combined with dependent ones.

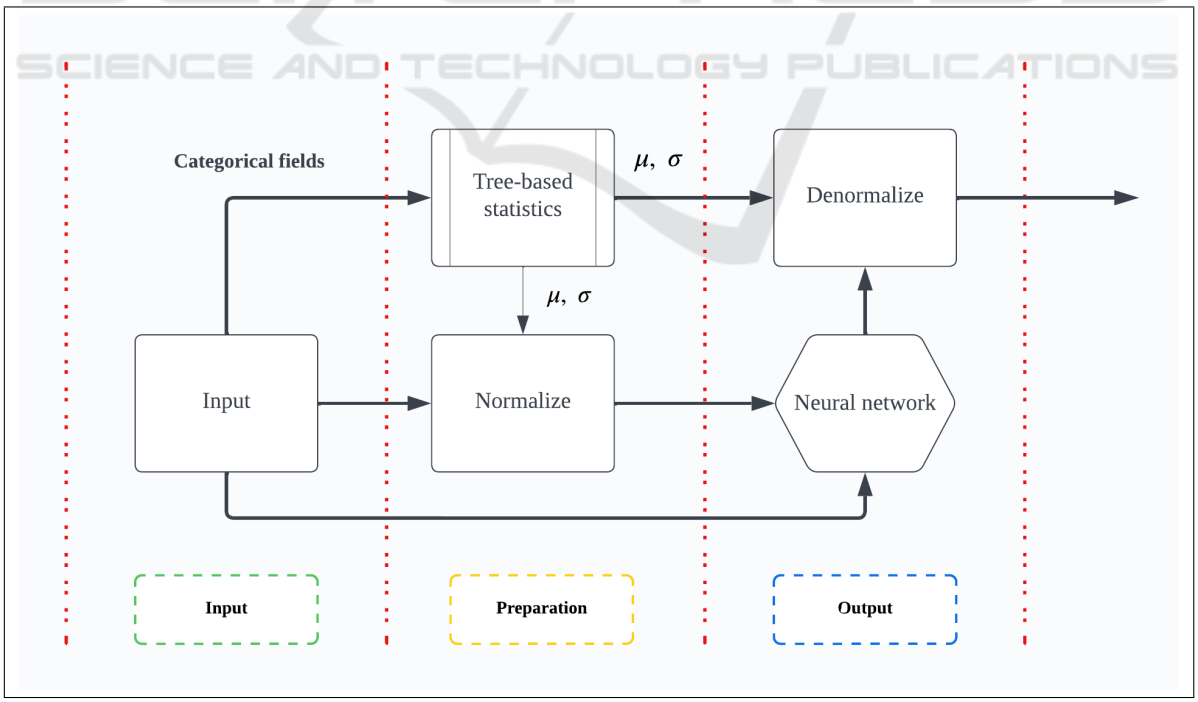

Figure 3: Overview of our proposed architecture: the input is transformed using a tree-like data-structure for statistics and the network is trained to output the normalized values. In the end, the output is de-normalized, so that the values are in the original range of their cluster.

these operations are indicators of secret rotation running, which likely influences secret listing and reading behavior. As previously mentioned, there is a gap between customers or automations that generate large spikes in the data and those that have a consistent and predictable behavior. Modeling them at the same time proved difficult, because low-variance customers are underrepresented in the loss function (at least in our case) and data normalization and centering does not yield good results when one data cluster has a variance of say 100 and another cluster a variance of just 0.1. To mitigate these issues, we employ a hybrid approach (Figure 3), which is inspired by residual learning, but instead of training the network to model the residual signal, we use a non-uniform data normalization technique, where the mean and standard deviation are predicted using a tree. Each level of the tree handles a different attribute, and in our case, the order in which attributes are processed is static, based on our analysis of the data and their semantics: cluster, customer, namespace, IP address. However, this tree can be constructed automatically by using standard ML metrics, such as the entropy of the data. Every leaf and every node inside the tree contain statistics that model every datapoint that passed though the node or ended up in that leaf. This enables us to perform some type of backoff for previously unseen attributes. For instance, if an IP address is previously unseen, we backoff to the previous level in the tree and we model this entry based on the (cluster, customer, namespace). This ensures data and normalization consistency as best as possible, given previously observed datapoints.

4 EVALUATION

Direct evaluation of this type of modeling is hard in the absence of a proper dataset. Also, measuring the F-score for synthetically generated data is irrelevant, since in normal operations, where you don't have incidents daily, this score will be 0 regardless how good the system scored on the synthetic data. Instead, we measure (a) the modeling capacity of the network with and without the residual scheme, (b) the reduction in training time and (c) the reduction in False Positives (FPs). We mention that the output of the network is a normal distribution (μ, ρ), which brings major modeling advantages (i.e. we can directly use the z-rule to score anomalies) but requires us to use the Gaussian loss function during training (Equation 1) .

$$L = \frac{1}{2}\left(log\left(max(\rho, \varepsilon)\right) + \frac{(\mu - y)^2}{max(\rho, \varepsilon)}\right) \quad (1)$$

y - Is the target value
μ, ρ - Represent the mean and variance
ε - Is a constant used for numeric stability – in our experiments we used 1e-8

The results shown in Table 1 are reported using the following procedure: (a) each network is trained with and without residual learning on 80% of the dataset; (b) we compute the loss on previously unseen 10% of the dataset and we use the result as an early stopping criterion with a patience of 5 (epochs without improvement); (c) we report the loss computed on another 10% of the dataset, that is also unseen during training. Note, that the negative values for the loss function are accurate, because we are working with the gaussian loss function. Also, the network is a three-layer perceptron with tanh activation and gaussian output (μ, ρ). The input is composed of the concatenated embeddings for the categorical fields (128-sized partitions) and the log-normalized values for the numeric attributes. As can be seen, with one exception, the number of training epochs is reduced when relying on the tree-based statistics for normalization. Also, there is a constant reduction in the loss value, using this hybrid scheme. Finally, we tested the 1000-1000-1000 network in a production environment, and we got a 98% reduction in False Positives, which translates into 20-30 events per day, on about 1.9M datapoints.

Observation 1. With this information in hand, we can determine that using SMPs logs to identify unauthorized access to credentials is suitable, but the results must be correlated with other sources of events in order to properly classify the events as expected or not.

Observeration 2. When it comes to scalability of the proposed method, we observed that the memory consumed during training and analyze phases grows in direct ratio to the number of the datapoints in the

Table 1: Loss on the test set obtained by several network configurations with residual (R) and non-residual (NR) learning.

Configuration	Type	Loss	Best
100-100-100	NR	1,23	5
	R	-3,40	8
300-300-300	NR	-1,59	6
	R	-4,76	3
700-700-700	NR	-6,21	8
	R	-7,45	3
1000-1000-1000	NR	-6,84	8
	R	-7,67	5
1500-1500-1500	NR	-6,26	7
	R	-7,58	4

dataset, which makes the statistics tree expand to many nodes. To overcome this limitation, we decided to implement some core data structures that the model uses to compute the statistics tree in C, by leveraging Cython to develop a Python extension. With this solution in place, we were able to lower the memory consumed by up to 65% when using a statistics tree with ∼40M nodes.

5 CONCLUSIONS AND FUTURE WORK

Our work introduced a hybrid modeling method for numerical data, based on a mixture of categorical and numeric inputs. As shown in the evaluation section, we achieved a 98% reduction in FPs with a significantly smaller number of parameters required for modeling and faster training times. The non-uniform normalization scheme, permitted using the tree, enables us to efficiently train models on datasets with skewed numerical outputs and provides a natural back-off method for the statistical estimation. Our future research plans will focus on (a) how the consumption order of the attributes can be efficiently computed, (b) on investigating more use-cases and (c) on further refining our results and reducing the number of False Positives.

With every SMP, it is expected to have frequent requests to fetch credentials, especially in enterprise grade, rapid changing environments. This translates in dozens of access logs being generated with every request, which could lead to introduction of new access patterns. To accommodate for the continuous shift in patterns, the model would have to be retrained often, to avoid data drift. This is one aspect that is part of our future work on improving the model to automatically detect that the model has drifted and decide when it's time to retrain.

Adding on future research plans, the current work focused on attack patterns observed while inspecting adversary emulations of real-world scenarios. As part of our future research plans, we will investigate how to improve the model's robustness against new and evolving attack patterns, to consider more viewing points when deciding if an event is anomalous or not.

Finally, although we have shown that our hybrid modeling method proved successful to reduce the number of False Positives and identify anomalous events in SMPs access patterns, we only had access to a single SMP to work with and test our approach against. To validate that the proposed solution is effective when confronted with access logs from other SMPs as well, we will conduct a new experiment using data gathered from multiple sources, and compare the results.

REFERENCES

Anumol, E. (2015). Use of machine learning algorithms with siem for attack prediction. In *Intelligent Computing, Communication and Devices: Proceedings of ICCD 2014, Volume 1*, pages 231–235. Springer.

Bryant, B. D. and Saiedian, H. (2020). Improving siem alert metadata aggregation with a novel kill-chain based classification model. *Computers & Security*, 94:101817.

Das, S., Ashrafuzzaman, M., Sheldon, F. T., and Shiva, S. (2020). Network intrusion detection using natural language processing and ensemble machine learning. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 829–835. IEEE.

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298.

Feng, C., Wu, S., and Liu, N. (2017). A user-centric machine learning framework for cyber security operations center. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 173–175. IEEE.

Gibert Llauradó, D., Mateu Piñol, C., and Planes Cid, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenge. *Journal of Network and Computer Applications, 2020, vol. 153, 102526*.

Idhammad, M., Afdel, K., and Belouch, M. (2018). Semi-supervised machine learning approach for ddos detection. *Applied Intelligence*, 48:3193–3208.

Noor, U., Anwar, Z., Amjad, T., and Choo, K.-K. R. (2019). A machine learning-based fintech cyber threat attribution framework using high-level indicators of compromise. *Future Generation Computer Systems*, 96:227–242.

Osanaiye, O., Cai, H., Choo, K.-K. R., Dehghantanha, A., Xu, Z., and Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016(1):1–10.

Piplai, A., Mittal, S., Abdelsalam, M., Gupta, M., Joshi, A., and Finin, T. (2020). Knowledge enrichment by fusing representations for malware threat intelligence and behavior. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE.

Zekri, M., El Kafhali, S., Aboutabit, N., and Saadi, Y. (2017). Ddos attack detection using machine learning techniques in cloud computing environments. In *2017 3rd international conference of cloud computing technologies and applications (CloudTech)*, pages 1–7. IEEE.

Zhou, H., Hu, Y., Yang, X., Pan, H., Guo, W., and Zou, C. C. (2020). A worm detection system based on deep learning. *IEEE Access*, 8:205444–205454.