Towards Building Accountable Deep Neural Model by Interpreting Classification Accuracy

Rangeet Pan, Giang Nguyen, and Shibbir Ahmed

Iowa State University, Ames, IA, 50011, USA,
{rangeet,gnguyen,shibbir}@iastate.edu

Abstract. Deep neural networks have gained large popularity in the past decade. Compared to the non-ML based system, there is not much existing verification mechanism for these learning-based models. One such verification technique includes the contract between the advertised trustworthiness and the actual performance of these models. Our study identifies that due to the random initialization, accuracy i.e., trustworthiness changes even if the experimental setup remains unchanged. In order to address this issue, we have proposed a mining-based approach to gain the domain knowledge of the input and the learning operations to generate an interval of the output metrics. We have also proposed a specification language to restrict the learning process that helps a DNN model to achieve accountability in the aspect of classification accuracy.
Keywords: Accountability, Deep Neural Network, Interpretability , Model Verification.

1 Introduction

With the increasing popularity of Machine Learning (ML) based systems, the verification and validation of such systems are necessary. Although there is a vast amount of research carried out on the non-ML validation framework, there is a few validation based research done on the ML-based systems due to the nature of probabilistic and complexity. Without a proper validation framework, one might ask about the way to hold ML-based systems being accountable to the expectation. For instance, the contract made with the end-user by exposing the trustworthiness of these systems in terms of accuracy. But this metric of trustworthiness is variant even if the whole experimental setup remains the same. In this study, we address such problems and have proposed a programming language infrastructure to measure the accuracy in a closed interval.

The recent works on this field can be primarily categorized into two sections, verifying a model to be accountable for the assigned task [1,2,3,4,5] and holding the accountability by making ML models more robust [6,7,8]. The prior works have focused on validating the input influence [9], explaining the models to make the black-box system grayer. However, these systems either hold domain knowledge or model operation knowledge as the key to increase the explainability. In this study, we have combined these two type of knowledge and propose a system that can takes the input dataset, model operations and their distributions as input and produce a metric of accuracy in a closed interval rather than a single value that changes everytime an ML model has been trained with same dataset and same experimental setup. We leverage the information to propose a specification language *ADNN* that restricts the model to learn and produce output accuracy in an interval provided by the end-user.

Problem Statement. Our proposed work is focused on the image-based deep neural network (DNN) classifier. We have identified multiple pieces of evidence that with the same experimental setup, one DNN model can produce different output evaluation metrics e.g., accuracy. In the learning process, these models begin the operation with random initialization of several parameters e.g., seed, weight, bias. With this randomly initialized value, the learning process continues that ends up producing different results for a different execution. In order to address such a problem, we are proposing a mining-based distribution learning and specification language-based approach that helps the user to hold a model structure accountable for the output.

Our contribution to this study has been the following:

- We have proposed a framework that takes the domain knowledge and DNN operations to understand the DNN image classification-based system.
- Our approach leverages the knowledge of the input and DNN operations' distribution to restrict the learning process through a proposed specification language.

2 Motivation

In Figure 1, we illustrate an example from the Stack Overflow [1].

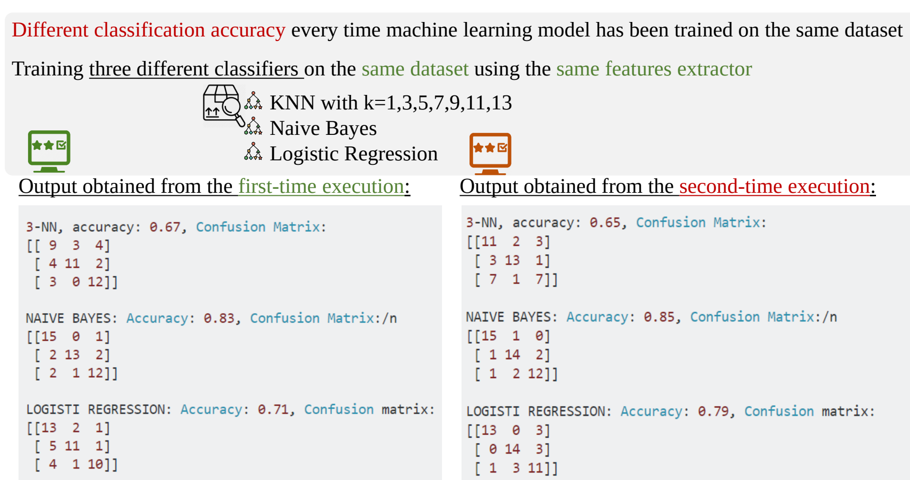

Fig. 1: Motivating example of accountable assertion to interpret accuracy for accountable machine learning classifier

[1] https://stackoverflow.com/questions/55775450/

In this scenario, different classification accuracy has been obtained with every execution of a machine learning model that has been trained on the same dataset. Three different classifiers e.g., k-nearest neighbors, Naive Bayes, and Logistic Regression have been compared using the accuracy of each model. With every execution, these models have been trained with the same experimental setup, different values of accuracy for each model have been obtained as illustrated in Figure 1. These results in a lack of trustworthiness of an ML-based model. The obvious question would be how we can interpret the accuracy or how the classifier can become accountable with the evaluation metric? This problem motivates us to propose a mining and specification-based language to hold an ML model accountable.

3 Related Work

3.1 Study on accountable ML model.

Validating Accountability. There is a vast amount of research on validating and holding an ML model accountable. One of the earliest validation technique proposed by Pulina and Tacchella [1] that utilizes abstraction to explain special types of a neural network, multi-layer perceptrons (MLP). [1] proposed a refinement approach to study the safety of MLP and repair them. However, this work can only be utilized if a network has six neurons. Gehr et al. [2] has proposed an infrastructure AI^2 that converts a neural network with convolution and fully connected layers to address the safety and robustness of the ML models. This work has addressed some issues, e.g., the trade-off between precision and scalability, presenting an abstract representation of robustness in convolutional operation. However, this infrastructure does not work for a complex model, and it suffers from scalability issues. Du et al. [3] have surveyed models and research papers and have found that though there is a vast majority of work in machine learning has been done on increasing the explainability of the models. This work describes the clear categorization and complete overview of prevalent techniques to increase the interpretability of machine learning models aiming to help the community to understand the capabilities and weaknesses of different accountable approaches better. This study concludes that the prior works do not answer some question developers have e.g., *"why Q, not R"* or in our case why the accuracy changes with the same experimental setup and how we can believe an ML model if it does not provide a concrete contract to the end-user. Another survey [4] has been conducted on similar research artifacts and have concluded the overall theme for HCI researchers to hold an ML model accountable.

Holding Accountability. Jia et al. [8] proposed a programming language to prune the neural network to explain and interpret the model behavior and find bugs in the graph-based operations. RELUVAL [6] proposed a symbolic interval analysis to provide a formal guarantee of a deep neural network-based model. This work proposed a technique to formalize the dependency information of a network while the network operations propagate. Input dependency and output estimation related problems are addressed by providing an interval that is similar to our work where we demonstrate the output accuracy in terms of an interval based on the input dependencies propagating through the network. Similar to the RELUVAL, [7] built a framework based on SMT solver to verify neural networks. Another area of research includes increasing robustness through crafting attacks and creating a defense against adversarial attack [10] that posses a threat against an ML model. Other studies [11,12] produce a verification process to defend against such attacks.

3.2 Accuracy validation

Ribeiro et al. [13] focused on the trustworthiness validation on an ML model and proposed LIME, a modular and interpretable approach to explaining the predictions of any model in an accountable manner.

In another study [14], the authors proposed an explainable DNN model that validates the prediction outcome. This work demonstrated that the abstract representation of DNN-based models can diagnose and interpret the working mechanism of the prediction task.

Zhang et al. [15] proposed an alternate convolutional neural network (CNN) based models that holds more informations than the traditional ones which makes the CNN model more accountable to the classification accuracy even though in some cases, accuracy may declined. So, accuracy validation is crucial while making an interpretable CNN model.

3.3 Accountability

According to the study [16], accountability in decision making represents the explanation about the "ongoing strategy". From the §2, we have found that a single model structure can provide different decision-making capabilities due to the assertion of the probabilistic distribution in the initialization process. The prior work [17] has already demostrated that assertion based specification language can achieve accountability by reasoning about programs that behave randomly. In our proposed approach, we learn the unknown distribution of the initialization parameter. Then, we interpret the learning process by providing an interval of output metric rather than a single value that changes with every iteration of the learning process. This approach will help the end-users to hold a model accountable in terms of the contract made between the model structure and the predicted output.

4 Background

In this section, we have discussed the traditional operations carried out in validating the accountability of a DNN model.

Forward Propagation. In this stage, a DNN model learns the features from the input. This stage includes random initialization, activation functions, feedforward, and loss function. A deep neural network can be represented as a fully connected graph as depicted in Figure 2. This graph consists of nodes and edges with associated values. These values are randomly initialized to start the training process. This step is very important because a wrong initialization can hamper the learning process and take a longer iteration to reach optimality. Also, it is never a good idea to initialize them with zero because the gradient will be computed as zero in the consecutive steps, and this will end up not learning anything from the input. The activation function has an impact on computation. However, it does not change the probability distribution of the computation. The main purpose of the activation function is to convert the representation of the value corresponds to the nodes according to the need. On the other hand, the loss function computes the difference between the expected and the actual result.

Backward Propagation. This is similar to the forward propagation, but only the difference is the way of computation. In this stage, the network operations are computed backward, and the derivative of each term has been computed to find the gradient that helps the learning process to localize the error and fix in the direction of the gradient.

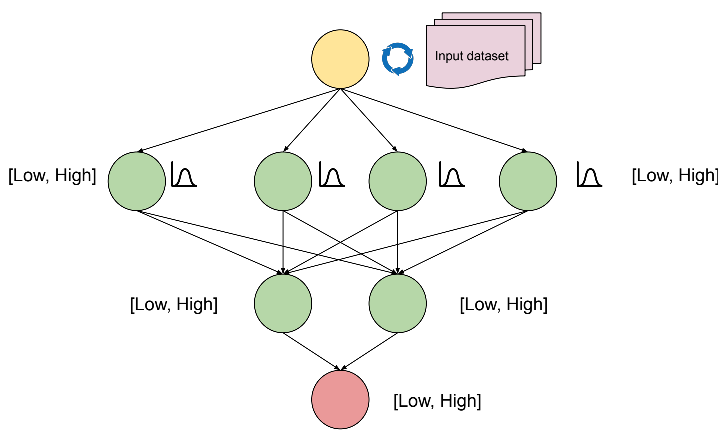

Fig. 2: Overview of a simple fully connected DNN model

5 Approach

In this section, we have proposed a comprehensive approach to address the discussed problem. In our approach, at first we have proposed a validation technique and utilized that to achieve accountability of a DNN model. In the §4, we have discussed the forward and backward propagation. In the two-step learning process, random initialization plays a crucial role in model validation. In Figure 2, a traditional DNN model iteratively chooses the input. The choice can be a single input at a time or a group of inputs. Three known parameters require random initialization i.e., seed, weight, and bias [18]. Our hypothesis in this study states that H_0*: Knowing the distribution of the random initialized parameter can provide the distribution of the output parameter, e.g., accuracy metric.* Based on this hypothesis, we select the operations that require random initialization.

Then, with the known distribution, we perform the model operations as follows.

$$f(\sum_{\chi} W_i X_i + B), \chi \sim D(\mu, \sigma^2) \tag{1}$$

In the equation above, the traditional dense operation has been depicted as an example, where, W_i, X_i, B_i, $f()$, and D represents the weight, input, bias, the activation function, and a unknown distribution of the operations respectively that require random initialization. The output of this learning process will be an interval of value rather than a single concrete value. In Figure 2, we have depicted a similar scenario that includes the learning of the distribution, which results in an interval reported as "[Low, High]". The interval is similar to any algorithmic evaluation with the best-case and worst-case scenario. In the diagram Figure 3, Our methodology has been discussed as follows,

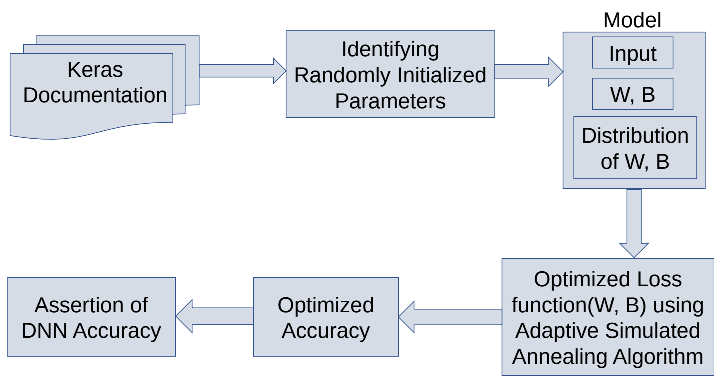

Fig. 3: Flow diagram of Proposed Approach

5.1 Optimized Loss Function

5.1.1. Identifying Randomly Initialized Parameter. In this phase, we have to understand the parameters that are subject to the random initialization. For this study, we have already mentioned that our focus has been on the deep learning-based image classifiers. To do so, we have mined the *Keras* documentation to understand the implementation of DNN API. In this process, 3 Ph.D. students have individually gone through the implementation of all the functions. Our goal is two-fold here, 1) Identify the places, where random initialization has taken place that describe the learning-based system, 2) If we can mine the type of distribution from the implementation, then we can understand how the values correspond to that particular parameter has propagated to the output metrics. Our mining technique has identified a class of operation in *Keras* that is responsible for initializing the parameters, called *Initializers.*

```
model.add(Dense(64,
kernel_initializer='random_uniform',
bias_initializer='zeros'))
```

The above example of the *initializer* operation in *Keras* can initialize two parameters in the learning process, weight, and bias. The *kernel_initializer* is responsible for the weight value initialization, while *bias_initializer* is for bias value corresponds to that particular layer. This serves the purpose of finding the parameters that cause the value of the output to be variant with every execution.

To identify the distribution of these parameters, we have validated the implementation of the *initializer* class and have found that are a few different distributions that it supports and if not specifically given, which default distribution has been taken care of.

- Zeros: This initializes the parameter with 0. While, it may not be an issue for the bias, but if it has been initialized for weight, during the backpropagation, the derivative of the actual and computed value will be 1 for all cases that would make the learning process very hard to compensate the huge loss.
- Ones: This initializes the parameter with 1.
- Constant: Initializing with a constant is depended on the users' choice of the value.
- Normal Distribution: The parameter will be initialized with a normal distribution that takes input as mean (μ) and the standard distribution (σ) to describe the distribution.
- Uniform distribution: This initializes parameter with a uniform distribution where the range of a minimum and maximum value can be specified.
- Truncated Normal Distribution: The parameter is initialized with a truncated normal distribution where values greater than 2 standard deviations from mean are omitted. Here, the mean and standard deviation of the random values are used as arguments.
- LeCun uniform initializer: This initializes parameter using a uniform distribution where input unit numbers are specified as the limit for drawing samples from it.
- Glorot normal initializer: In this initializer, truncated normal distribution using the number of input and output units are used for drawing samples.
- Glorot uniform initializer: This initializes parameter using a uniform distribution where input unit numbers along with the output unit numbers are specified as the limit for drawing samples from it.
  We can observe how kernel and bias initializers are initialized from the following example,

  ```
  class Dense(Layer):
  def __init__(self, units,
  kernel_initializer='glorot_uniform',
  bias_initializer='zeros',
  ....
  **kwargs):
  ```

- LeCun normal initializer: In this initializer, truncated normal distribution using the number of input units are used for drawing samples.

5.1.2. Finding optimized initial Weight and Bias -Algorithm and its description

5.2 Assertion of DNN Accuracy

This approach includes the validation framework from the learned distribution. We have proposed an assertion mechanism that verify the operation of the distribution and the input to restrict the learning process to go beyond the desired interval of output metrics. We propose *ADNN*, a specification language that restricts a learning process from a pair (f, ν), where f and ν denotes the learning process and the specification provided by the user. An example of the programming language is depicted as below.

```
@adnn(0.95>accuracy>0.65)
f(input_image, ....)
\\Learning operations
```

Listing 1.1: Accountable specification language

6 Evaluation

6.1 Experimental Setup

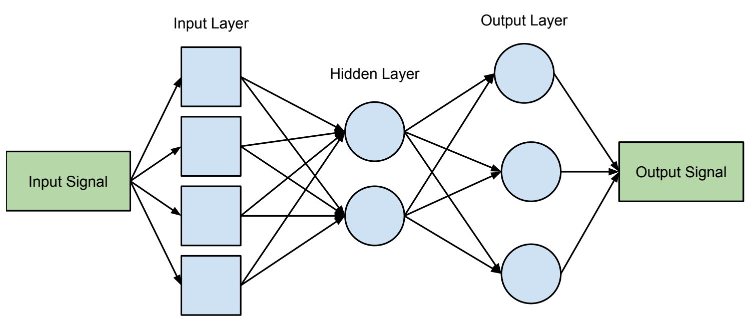

Fig. 4: MLP Structure

Model Mining We have mined **x** multilayer perceptron (MLP) models from 3000 Keras repositories. However, there are many different kinds of machine learning models in 3000 Keras repositories. Thus, we have to identify which model is MLP. To address this problem, we need to identify the general structure of MLP. Then, we filter the models which are not belong to this structure. Figure 4 represent a general structure of MLP. There are three components in MLP's structure which are input layer, hidden layer, output layer. Moreover, all of these layer are constructed by using Dense layer and activation layer. Therefore, if a mined model have another kinds of layer like convolutional layer or recurrent layer, it will be discarded. To detect and extract MLPs, we have used control flow graph (CFG). In particular, we manually collect a list of the Keras APIs used to build Keras models. Then, the CFG parses through each statement of the Python files and collect the API names. If the API names is belong to the APIs list, we will collect them. After that, we connect all the API calls together to acquire a complete model. If a model contains at least one API which are not used for constructing MLP, we will remove it. Moreover, we also filter the MLP models which miss some requirement information like input shape or output channel.

```
model.add(Dense(units=64, activation='relu',
    input_dim=100))

model.add(Dense(units=10, activation='softmax'))
```

(a) Original MLP

```
{'func': 'Dense', 'input_dim':100, 'units': 64}
{'func': 'relu'}
{'func': 'Dense', 'units': 64}
{'func': 'softmax'}
```

(b) ANN

Fig. 5: Original MLP vs ANN

To be easier in using the mined models, we have stored them in form of abstract neural network (ANN). Figure 5 shows the example of original MLP and ANN. ANN is a graph whose nodes represent for model layers and edges represent for the order of the layers. For example, the first line is Dense layer which includes three information which are 100 input channel, 64 output channel, and Relu activation. Actually, Relu activation is a layer; thus, we seperate this layer in two layer, Dense layer and activation layer.

Algorithm 1 Model Mining

```
procedure MODEL_EXTRACTION(pyFiles)
    CFG ← py
    return MLP
procedure MODEL_FILTERING(pyFiles)
    check = True
    for py ∈ pyFiles do
        check = True
        model = model_extraction(py)
        for layer ∈ model do
            if layer is False then
                check = False
                break
        if check = True then
            ANN ← model
```

Evaluation Metrics

6.2 Experimental Methodology

RQ1: What is the best value for δ. Using model mining, we will emprically evaluate the δ value.

RQ2: How efficient is this approach?

RQ3: How effective is this approach?

7 Future Plan

We have proposed a specification language and a framework to learn the distribution of the randomly initialized parameters so that we can generate an interval for output metrics. Thus, we can leverage the acquired domain knowledge to restrict a model from learning beyond a specified interval provided by the end-user. Our next steps will be:

- Accumulate the randomly initialized parameters.
- Mine models specifically image-based classification with dense operations to restrict the learning process and find the distribution of the parameters.
- Formalize the specification language with the obtained distribution.

Our goal is to deliver the first two steps for the next checkpoint of our project.

8 Acknowledgments

The author of this template is thankful to all those wonderful people who worked on this template.

References

1. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: International Conference on Computer Aided Verification, Springer (2010) 243–257
2. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP), IEEE (2018) 3–18
3. Du, M., Liu, N., Hu, X.: Techniques for interpretable machine learning. arXiv preprint arXiv:1808.00033 (2018)
4. Abdul, A., Vermeulen, J., Wang, D., Lim, B.Y., Kankanhalli, M.: Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In: Proceedings of the 2018 CHI conference on human factors in computing systems, ACM (2018) 582
5. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016)
6. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). (2018) 1599–1614
7. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: International Conference on Computer Aided Verification, Springer (2017) 97–117

8. Jia, Z., Padon, O., Thomas, J., Warszawski, T., Zaharia, M., Aiken, A.: Taso: Optimizing deep learning computation with automatic generation of graph substitutions. (2019)
9. Datta, A., Sen, S., Zick, Y.: Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In: 2016 IEEE symposium on security and privacy (SP), IEEE (2016) 598–617
10. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.: Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814 (2016)
11. Anderson, G., Pailoor, S., Dillig, I., Chaudhuri, S.: Optimization and abstraction: a synergistic approach for analyzing neural network robustness. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, ACM (2019) 731–744
12. Pan, R.: Static deep neural network analysis for robustness. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM (2019) 1238–1240
13. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM (2016) 1135–1144
14. Du, M., Liu, N., Song, Q., Hu, X.: Towards explanation of dnn-based prediction with guided feature inversion. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM (2018) 1358–1367
15. Zhang, Q., Nian Wu, Y., Zhu, S.C.: Interpretable convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 8827–8836
16. Veale, M., Van Kleek, M., Binns, R.: Fairness and accountability design needs for algorithmic support in high-stakes public sector decision-making. In: Proceedings of the 2018 chi conference on human factors in computing systems, ACM (2018) 440
17. Sampson, A., Panchekha, P., Mytkowicz, T., McKinley, K.S., Grossman, D., Ceze, L.: Expressing and verifying probabilistic assertions. In: ACM SIGPLAN Notices. Volume 49., ACM (2014) 112–122
18. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International conference on machine learning. (2013) 1139–1147