**IE643: Deep Learning : Theory and Practice** **July-Dec 2021**

# Automatic RNN Repair

*Team Name: Team Novel* *Team Members: Shreyansh Goyal, Hitvarth Diwanji, Shubham Sharma*

**Abstract**

This project report contains the details on our project which aims to repair RNNs by carrying out sample-level and segment level influence analysis. We describe the deep learning tools involved in our project. In particular, we explain about the deep neural network architectures, the loss functions and activation functions used and the training algorithm used. We have tried to come with some modifications to improve the results and we carry out various experiments to bring forth valuable insights.

# 1 Introduction

Deep neural networks are vulnerable to adversarial attacks. Due to their black-box nature, it is rather challenging to interpret and properly re- pair these incorrect behaviors. To fix an incorrect prediction, it is important to understand the root cause of the incorrect prediction. Once the root cause is identified, users may fix the errors by removing harmful training data or adding specific data to improve model accuracy. Recently, influence functions have been widely studied for interpreting the predictions of neural networks by estimating the effect of removing training samples on the accuracy of the model. Although proven effective for feed forward networks these methods have three major shortcomings on their usage with Recurrent Neural Networks(RNNs) :

1. Given Recurrent Neural Networks (RNNs), they usually suffer from the vanishing gradient and long-distance dependency.
2. Different from FNNs, RNNs often come with stateful structures for processing sequential inputs (e.g., audio, natural language). For a sequential test input, we need to study the effect of its segments more precisely. Existing methods mainly performed influence analysis for the whole test input but not at the segment level.
3. Existing influence analysis based methods inevitably introduce intensive computation, making the selection of useful samples inefficient.

To tackle these problems encounterd in RNNs the paper proposes a light-weight model-based influence analysis for RNNs, named RNNRepair.
**Approach Overview :** At a high level, we first adopt the clustering to capture the stateful behaviors of all training data. They first extract the abstract states by grouping state vectors (i.e., hidden representations of the RNN) of all training data and then based on the state abstraction, we can extract the trace for a given input. The influence function is constructed based on the transitions in the traces to further perform the segment-level and sample-level influence analysis. Lastly, based on the influence analysis, they deploy a remediation mechanism to analyze and offset the test errors.
We aim to give a comprehensive study of the novel method proopsed in the paper and carry out experiments that serve as possible extensions of the work. Our work gives more insight into the proposed method of influence analysis and its use in Recurrent Neural Networks.

We provide a survey of existing literature in Section 2. Our work for the project is described in Section 3. We give details on experiments in Section 5. A description of future work is given in Section 7. We conclude with a short summary and pointers to forthcoming work in Section 8.

## 2 Literature Survey

**Influence Analysis :** Although the paper presents a very original idea for building influence functions it tends to rely on the same insights that the past work on influence analysis has shown. Particularly Koh et al.[4] who studied the influence of training samples upon a given testing sample for DNNs. Specifically, they utilized the influence function to identify the most representative training samples for a given testing sample. They were the first to prove that under convex loss assumptions the computed inference resembles the actual influence of test samples upon retraining.

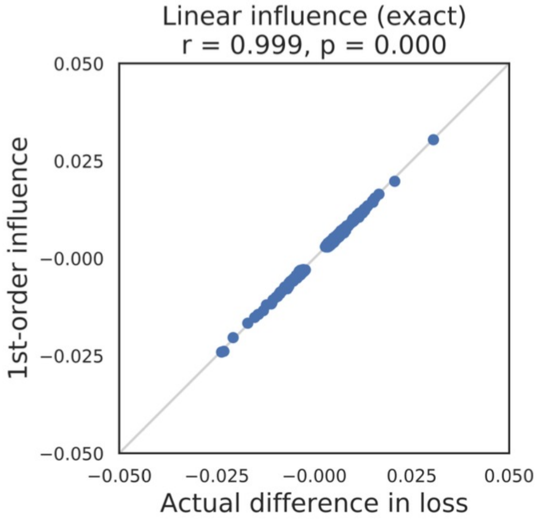


Figure 1: Koh et al. showcasing the influence analysis approximation

Recent efforts have been made to either enable the influence analysis for non-optimal models trained by using non- convex losses (Hara et al., 2019 [2]) or analyze the influence of a group of training samples upon a given prediction (Koh et al., 2019 [3])

**Model Extraction :** Existing research has developed various approaches to extract DFA (Deterministic Finite Automaton) from the known RNN architectures. We follow an unsupervised learning-based approach and apply GMM for state partition.
Our project also shares ideas with the paper by Weiss et al [7]. In this paper, the authors aim to extract a Deterministic Finite Automaton (DFA) of a given trained RNN to get better understanding of the decisions of the RNN. The pipeline of RNNRepair borrows from this paper, the idea of partitioning RNN states and also minimizing the number of extract states.

**Network Repair :** A lot of successful network repair techniques that are available for repairing the error instances in Deep Neural networks (Wang et al [6]) have their shortcomings with RNNs. The method is different from the techniques about adversarial defenses (Boopathy et al., 2019[1]; Singh et al., 2018[5]) and noisy learning/data cleaning (Zhang et al., 2018) in that these techniques aim for improving the robustness against adversarial attacks or data poisoning attacks. Whereas, the remediation mechanism here locally offsets testing errors of an RNN.

# 3 Methods and Approaches

The process of RNN repair can be broken down into a three step process.

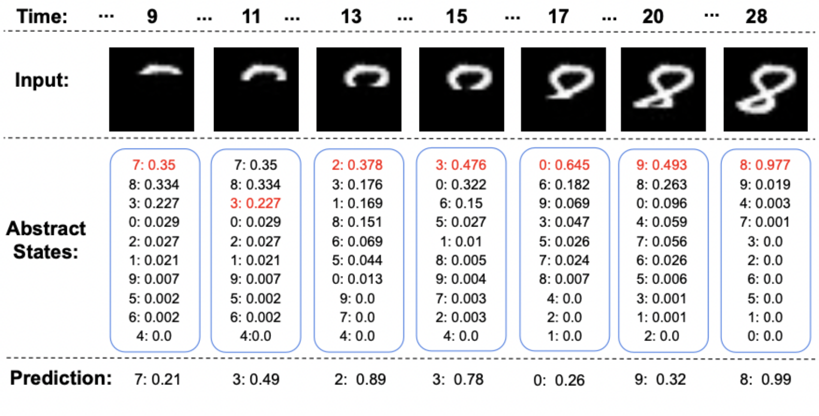
**Step 1 : Semantic-guided State Abstraction**
RNN repair first adopts clustering to capture the stateful behaviors of all training data. To efficiently represent a large number of state vectors, they use Gaussian Mixture Models (GMM), an unsupervised clustering method to group the state vectors. The metric developed to measure the semantics of the abstract state is *confidence score*, followed by the selection strategy.

Confidence Scores : Given an RNN classifier $R = (\mathcal{G}_R, d, m, \mathbf{h_0}, \mathcal{Y}_R)$ and a partitian result Q, the confidence score of each state q $\epsilon$ Q is defined as $C_q = [c_0, .., c_{n-1}]$

$$c_i = \frac{|\{\mathbf{h}|\mathbf{h}\epsilon SV_q \wedge \mathcal{Y}_R^n(\mathbf{h}) = i\}|}{|SV_q|} \tag{1}$$

The abstract state stability is another parameter based on the confidence scores that indicates a well-clustered state with regards to the concentration of output classes.

State Stability : For a state q and well as its confidence score $C_q = [c_0, .., c_{n-1}]$, the state is defined as $\delta$-stable, where $\delta = max(C_q)$



| Time: | ··· 9 | ··· 11 ··· | 13 ··· | 15 ··· | 17 ··· | 20 ··· | 28 |
|---|---|---|---|---|---|---|---|
| Input: | | | | | | | |
| Abstract States: | 7: 0.35<br>8: 0.334<br>3: 0.227<br>0: 0.029<br>2: 0.027<br>1: 0.021<br>9: 0.007<br>5: 0.002<br>6: 0.002<br>4: 0.0 | 7: 0.35<br>8: 0.334<br>3: 0.227<br>0: 0.029<br>2: 0.027<br>1: 0.021<br>9: 0.007<br>5: 0.002<br>6: 0.002<br>4:0.0 | 2: 0.378<br>3: 0.176<br>1: 0.169<br>8: 0.151<br>6: 0.069<br>5: 0.044<br>0: 0.013<br>9: 0.0<br>7: 0.0<br>4: 0.0 | 3: 0.476<br>0: 0.322<br>6: 0.15<br>5: 0.027<br>1: 0.01<br>8: 0.005<br>9: 0.004<br>7: 0.003<br>2: 0.003<br>4: 0.0 | 0: 0.645<br>6: 0.182<br>9: 0.069<br>3: 0.047<br>5: 0.026<br>7: 0.024<br>8: 0.007<br>4: 0.0<br>2: 0.0<br>1: 0.0 | 9: 0.493<br>8: 0.263<br>0: 0.096<br>4: 0.059<br>7: 0.056<br>6: 0.026<br>5: 0.006<br>3: 0.001<br>1: 0.001<br>2: 0.0 | 8: 0.977<br>9: 0.019<br>4: 0.003<br>7: 0.001<br>3: 0.0<br>2: 0.0<br>6: 0.0<br>5: 0.0<br>1: 0.0<br>0: 0.0 |
| Prediction: | 7: 0.21 | 3: 0.49 | 2: 0.89 | 3: 0.78 | 0: 0.26 | 9: 0.32 | 8: 0.99 |

Figure 2: The prediction process of an image and the cor- responding abstract states

In addition to the stability, we also desire a small number of total abstract states for a better generalizability (Weiss et al., 2018[7]). To achieve these two goals at the same time, we define the semantics-guided abstraction to decide the number of abstract states.

**Step 2 : Building the Influence function**
After we have the state abstractions that can capture the semantics of the sequential data we define a *trace* of the the input taken.

Trace : Given an input x, $x = (x_1, x_2....x_n)$, the trace $\tau_x = (q_0, x_0, q_1, ....., x_n, q_n)$ is obtained from the state vector sequence $\mathcal{G}_R^x = (h_0.h_1, ...., h_n)$, where $q_i = p(h_i)$, p is the partitioning function Based on

the abstract states constructed above, we build the transitions as well as the influence function for the influence analysis. Specifically, given the $trace_x = (q_0, x_1, q_1, ..., x_n, q_n)$ of each training sample $x \in T$, the influecne function $I$ is updated as follows :

$$I(q_{i-1}, x_i) = I(q_{i-1}, x_i) \cup \; \mathbf{x} \tag{2}$$

where the influence function $I$ could capture the effect of training samples at each abstract state.

Now to further quantify the influence of training samples upon an entire testing sequence x, we define the temporal feature as follows.

Temporal Features : Given an RNN, R, an input $x$, and its trace $\tau_x$, the temporal features defined as $\mathcal{F}_x = (f_0, ...f_n)$, where $f_i = (ID(q_i), C_{q_i}, \mathcal{Y}_R^n(h_i)).q_i = p(h_i)$ is the abstract state to which $x_i$ belongs and $ID(q_i)$ is the unique identifier of state $q_i$.

With the definition of the temporal feature, we quantify the influence of a training sample on a test input. Specifically, given a training sample $x_{train}$ and a test sample $x_{test}$, the influence is quantified as the similarity between the temporal features of the training sample and the testing sample:

$$Influence_{score}(x_{test}, x_{train}) = similarity(\mathcal{F}_{x_{train}}, \mathcal{F}_{x_{test}}) \tag{3}$$

Different similarity metrics can be selected for different applications like vector norms or jaccarrd distance to name a few.

**Step 3 : Fault Localization and Remediation**
mainly focus on two kinds of misclassification: 1) misclassification caused by a whole input instance and 2) misclassification caused by an input segment.
For the first type of errors they first identify the responsible training samples. Then, they randomly generate new samples by manipulating the identified ones and apply the influence analysis to filter out the error-triggered training samples. Finally, they retrain the target RNN with the newly generated samples.

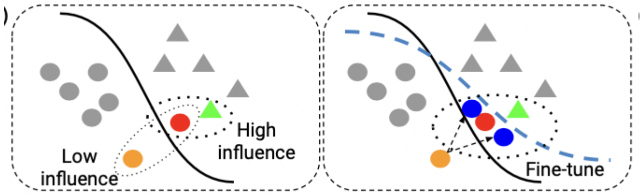


Figure 3: The overview of the fault localization and repair (Red circle represents the failed input)

Similar with repairing the sample-level error, they also follow a three-step procedure to repair the second type of errors that are caused by the rarely seen segments in the training data. Differently, they design a following method to identify the root cause input segment rather than identifying whole input samples.

## 3.1 Work done before mid-term project review

Before the mid term review,the following was done:

- Looked at the motivation behind the paper
- Looked at the problems addressed in the paper
- Analysed the impact of three major works on the paper: Koh and Liang, 2017, Hara et al. 2019, Weiss et al., 2018
- Explained the formal definitions used in the paper

- Re-conducted the author's experiments and verified the results in the paper
- Conducted some experiments ourselves by modifying some parameters (more on this in section 5)
- Suggested various improvements over the existing solution proposed in the paper

### 3.2 Work done after mid-term project review

- Work done after mid-term project review
- Modifying the influence function for Segment level influence analysis
- Incorporating comparative analysis in the influence function
- NLP-based sample level analysis
- Empirically verifying the authors' hypothesis for different number of components
- Extending the idea of influence analysis to DNNs

## 4 Dataset Details

The work selected two widely used public datasets (i.e., MNIST, and Toxic) to evaluate the influence analysis as done in the paper. MNIST (LeCun  Cortes, 1998) is selected for evaluating the sample-level influence analysis by comparing it with the existing baselines. We train an LSTM network with hidden size 100 for this task. At each time, the RNN reads one row (i.e., 28 pixels) from the image. Toxic Comment Dataset 2 is selected for evaluating the segment-level influence analysis. The task is to classify whether the comment is toxic or not. We train a GRU network with hidden size 300. In addition, we make use of another data set Standard Sentiment Treebank (SST) (Socher et al., 2013) for the segment-level repair and a LSTM network with hidden size 300 is trained. For influence analysis on DNNs, again MNIST is used.

## 5 Experiments

### 5.1 Applying a Threhsold on Confidence Scores

A threshold was applied to confidence scores. This was done as lower confidence scores are not relevant for representing an abstract state. They unnecessarily decrease the similarity score between the temporal features of two samples
The setup used for the experiment was the same as that by the authors for understanding the correctness of temporal features. A threshold of 0.3 was applied and the experiment was run in both the original and custom setup on MNIST and TOXIC data-sets. The average accuracies were recorded and the shown in Section 6 of this report. The code to this experiment can be found here

### 5.2 Exploring clustering techniques

In the original text, the authors used Gaussian clustering models clustering algorithm for state abstraction. In this experiment, we analysed the use of K-means and Mean-shift clustering methods. K-means often converges to a better local optimum[ref], but is subjective to initialization, slower than GMMs and also not very flexible[ref]. When using Mean-shift, there is no need to define the number of clusters but the method itself is very slow. First, time comparison was done for different number of samples for the three clustering methods discussed. K-means was implemented and used for state abstraction. It was run 10 times and the average test accuracies were recorded. The results for this experiment are shown in section 6. The code for time comparision can be found here and the code for implementation of K-means for our task can be found here

## 5.3 Verifying Author's Hypothesis

In the original text, the authors make the following Hypothesis: For a training sample with true label 't' misclassified as'm', there are more influential samples with label m than with label t. In this experiment we aim to empirically verify this hypothesis. For this, we plot metric-1 vs n, where metric-1 is the percentage of failed inputs whose influence training setcontains atleast one sample with truth label t (or m) and n is number of top n samples considered for influence). This was plotted for various number of GMM components: 7, 16, 25, 43, 61

## 5.4 Extending the Sample-level Influence Analysis to DNNs

In this experiment we implement ideas similar to sample-level influence analysis on CNNs. The code for this experiment was written completely from scratch. The dataset that we used for this experiment was MNIST. The solution proposed by the authors is limited to RNNs. They create states from the hidden states of RNN. We use the output after the flatten layer as the hidden state of DNN. All the 'hidden states'from the training samples are collected and clustered using GMM. Then the abstract states are used for influence analysis

### 5.4.1 Confidence Scores and Stability of Abstract States

For this experiment, we used 10 components and calculated the confidence scores with this strategy. The result was interpreted.

### 5.4.2 Top m influential samples

For this experiment, we used 30 components and created temporal features similar to that in the paper. Then top m influential samples were seen for a test prediction.

# 6 Results

## 6.1 Experiment 5.1

The average test accuracy is shown below:

| Dataset | CSs | CSs(ours) | ID,R_L,CSs | ID,R_L,CSs(ours) | Original RNN |
|---|---|---|---|---|---|
| MNIST | 97.31 | 97.33 | 97.81 | 97.86 | 98.46 |
| TOXIC | 89.69 | 89.68 | 90.08 | 90.16 | 93.26 |

## 6.2 Experiment 5.2

Time comparison for different clustering techniques is shown below:

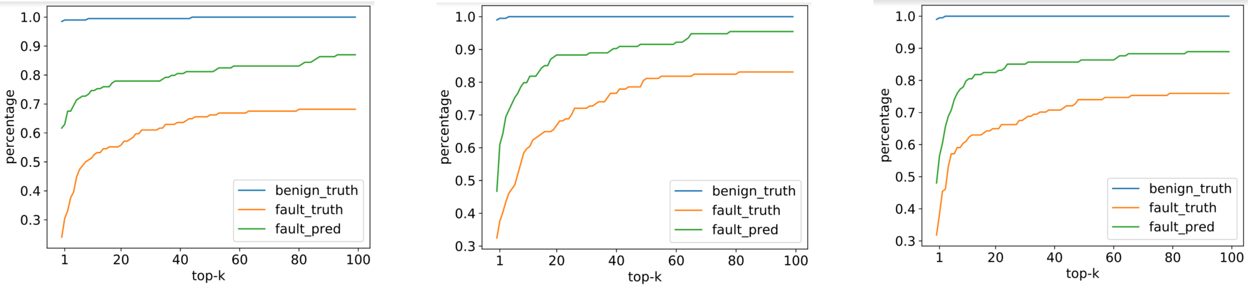
| Number of Samples | GMM | K-Means | Mean Shift |
|---|---|---|---|
| 100 | 0.02 | 0.05 | 0.52 |
| 200 | 0.02 | 0.04 | 1.56 |
| 300 | 0.02 | 0.04 | 1.73 |
| 400 | 0.01 | 0.04 | 2.38 |
| 500 | 0.02 | 0.06 | 6.42 |

Test accuracies for K-means and GMM are shown below:

| Features | GMM | K-Means |
|---|---|---|
| RL | 79.80 | 79.79 |
| ID | 59.35 | 58.35 |
| ID_RL | 87.31 | 85.82 |
| CS | 89.69 | 90.18 |
| ID_RL_CS | 90.08 | 89.92 |

## 6.3 Experiment 5.3

The following graphs were obtained for GMM components 7,25,43

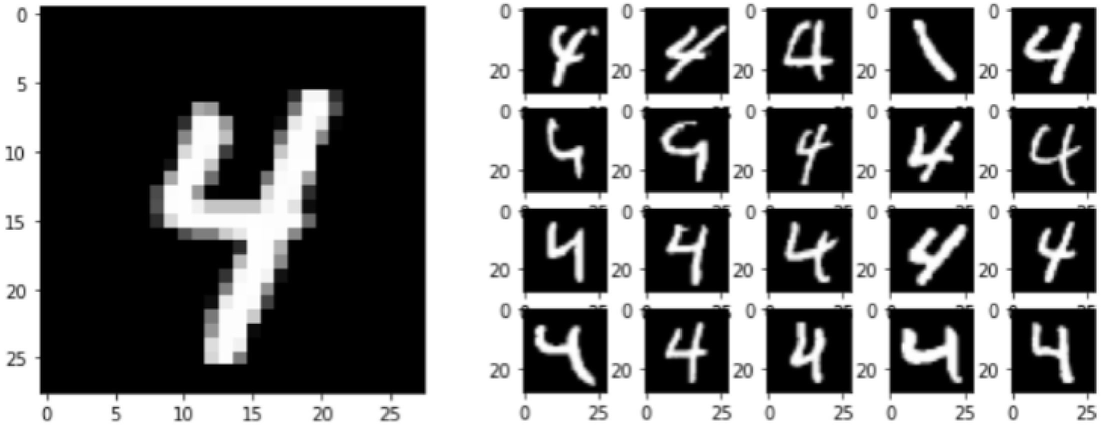


## 6.4 Experiment 5.4.1

The stability of the confidence scores was obtained to be the following 0.81, 0.99, 0.99, 0.44, 0.62, 0.98, 1.00, 0.99, 1.00, 0.46.
This indicates the stability and understandability of state abstraction.

## 6.5 Experiment 5.4.2

The top m samples for a given training sample were obtained and the result is shown below with the training sample on the left and the 20 most influential samples on the right.



## 7 Future Work

The influence analysis presented for RNNs is indeed very exiting and can be furthur extended in many ways. Some of the possible directions for the work could be :
**Centroid Based Clustering :** The current temporal features just have the abstract state id. Although proven effective it gives no consideration to the distance between two abstract states in the hyperspace. Centroid-based clustering (e.g K-means)methods can improve the quality of the temporal features
**Better Repair :** The GMM-based partitioning can be improved with more fine-grained refinement. We can also consider introducing more diverse types of data augmentation techniques (e.g., GAN, morphing) to generate candidate data for repairing.
The work on fault localization and repair can also be extended upon more complex errors such as the negative-to-positive cases.
**Abstract states of DNN for Generating Images:** A Deep encoder-decoder architecure can be used with the DNN-based influence analysis. And the abstract states calculated from the outputs of the bottleneck layer can be clustered using GMM. Now, we can sample from the distribution that was fitted by the GMM and pass the sampled vector through the (trained) decoder to generate new images. This idea is somewhat similar to VAE.

## 8 Conclusion

This paper presented a novel model-based technique for influence analysis of RNNs. Different from existing techniques that perform loss change estimation, the method here is less computation intensive and more efficient. We could identify the most influential training samples on given test inputs at both segment level and sample level. We showed that our techniques are effective in identifying important mislabeled training samples, and repairing RNNs.
We further explored aspects of the proposed method and tried out different variations in it. These experiments give insight into the whole repair process and validate the method proposed. We also extend the work into Deep Neural Networks and explore the relevance of the ideas proposed here for RNNs there.
The proposed influence based technique for RNN repair opens up new exiting possibilities in the field of fault localization and ramification for these black box networks.

## References

[1] Weng-T.-W. Chen-P.-Y. Liu S. Boopathy, A. and L. Cnn-cert Daniel. An efficient framework for certifying robust- ness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence, vol- ume 33, pp. 3240–3247, 2019.*, 2019.

[2] Nitanda-A. Hara, S. and T. Maehara. Data cleansing for models trained with sgd. In *Advances in Neural Information Processing Systems*, 2019.

[3] Ang K.-S.-Teo H. Koh, P. W. W. and P. S Liang. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems*, 2019.

[4] P. W. Koh and P Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017.*, 2017.

[5] Gehr T. Mirman M.-Püschel-M. Singh, G. and Vechev. M. fast and effective robustness certification. In *Ad- vances in Neural Information Processing Systems, pp. 10802–10813, 2018.*, 2018.

[6] Ustun B. Wang, H. and F. P. Calmon. Repairing without retraining: Avoiding disparate impact with counterfac- tual distributions. In *Chaudhuri, K. and Salakhutdinov, R. (eds.), Proc. of the 36th International Conference on Machine Learning (ICML)*, 2019.

[7] Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5247–5256. PMLR, 10–15 Jul 2018.