Automatic RNN Repair via Model-based Analysis

IE643 Course Project

Team Novel

Team Members: Hitvarth Diwanji, Shubham Sharma, Shreyansh Goyal

Outline

Description of Problem Statement

- Carry out Sample-level and Segment-level influence analysis of RNNs
- Repair RNN misclassifications by dealing with the influential samples
- Extend the idea of influence analysis to DNNs

Work done before midterm review

Applying Threshold on C

- Applied a threshold of **0.3** to the Confidence scores
- Used the setup for correctness of temporal features
- Ran both the original and the custom setup 10 times on MNIST and TOXIC datasets. The average accuracies are as 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

- Code

Work done before midterm review

Exploring and Comparing Clustering techniques

- Time comparison (Code)

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

- Implemented K-means and used it for state abstraction
- Ran the code 10 times and noted the average test accuracies (%).

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

Comments Given During the Mid-Term Project Review

Comments

- Time allotted to discussion of the framework could have been more
- A 10-min discussion of the experiments was an overkill and should have been trimmed to focus on details of the framework in the paper
- The future work proposed is quite extensive and focussing on particular aspects may yield in obtaining important insights.
- A reference slide citing the resources and papers is missing.

Minor Comments

- Contributions of each team member could have been included.
- A combined video should have been submitted instead of 3.
- Modifying the influence function by considering $I(q_{i-2}, x_{i-1})$
- Incorporating top-m samples for repair
- NLP-based sample level analysis can be worked upon

Addressing the comments given

- Reference and contribution slides added. A combined video is submitted this time.
- Influence function slightly modified for segment repair
- Challenges faced while attempting top m discussed
- Challenges faced while attempting NLP-based sample level analysis discussed
- Some other experiments that not mentioned in the mid term report were also done

Work done after midterm review

Modifying the Influence Function to tackle empty set

- Here, we suggest a modification to Section 3.3.2. Remediation With Segment-Level Influence Analysis
- Since, we randomly select m training samples from the set $I(q_{i-1}, x_i)$. If this set is empty, then the approach fails
- We suggest randomly choosing m training samples from $I(q_{i-2}, x_{i-1})$ if $I(q_{i-1}, x_i)$ is empty
- code

Work done after midterm review

Incorporating comparative analysis in the influence function

- Our idea: For the segment level influence analysis remediation instead of randomly selecting m samples from $I(q_{i-1}, x_i)$ we can select top m samples from the same set
- Should improve segment level accuracy after retraining
- Challenges: The code was incompatible for selecting top m samples while iterating over the layers

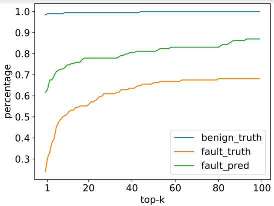
NLP-based sample level analysis

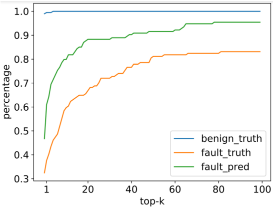
- Our idea: assemble the code modules provided by the authors
- Show that RNNRepair is far better than retraining on randomly chosen samples
- Challenge: either some code parts were missing or the code modules were not compatible with each other

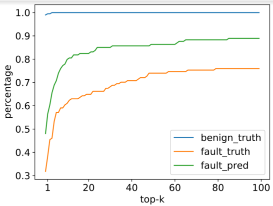
Work done after midterm review

Empirically verifying the authors' hypothesis for different number of GMM components

- Hypothesis: For a training sample with true label 't' misclassified as 'm', there are more influential samples with label m than with label t
- Metric-1: the percentage of failed inputs whose influence training set contains atleast one sample with truth label t (or m)
- Plotted Metric-1 vs n (number of top n samples considered for influence) and varied number of GMM components: 7, 16, 25, 43, 61







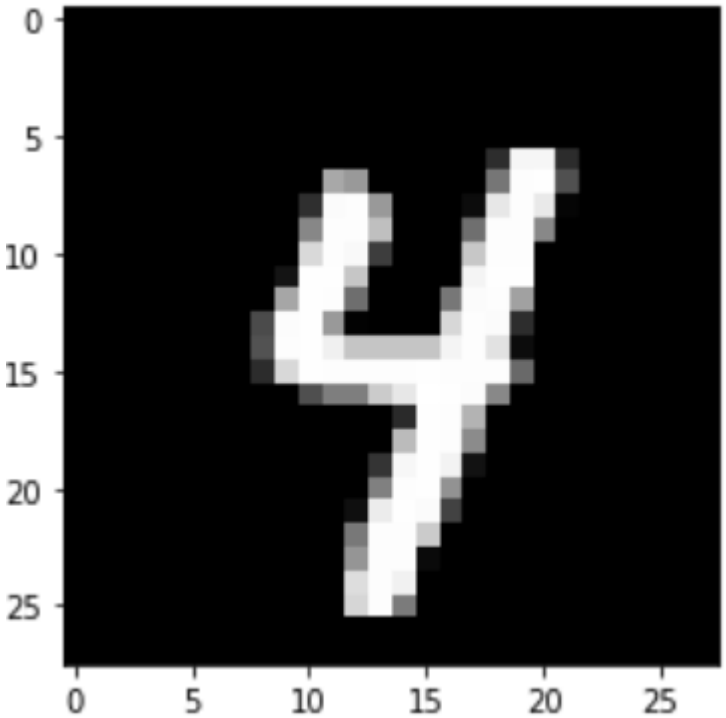
Work done after midterm review

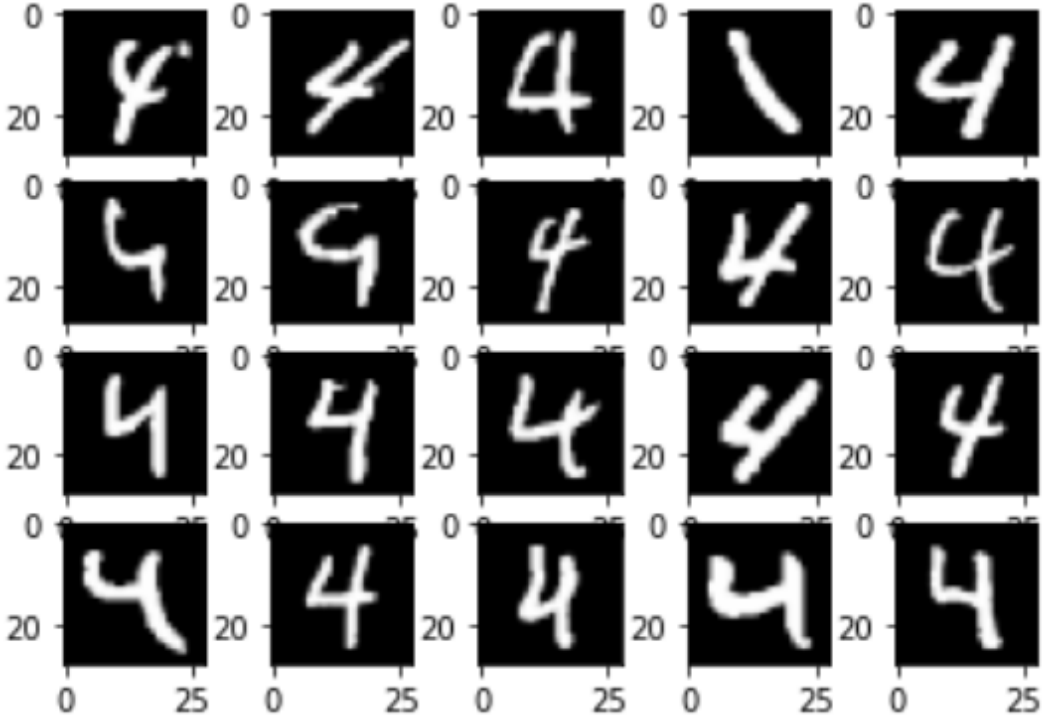
Extending the idea of influence analysis to DNNs

- Implemented ideas similar to sample-level influence analysis on CNNs
- Wrote the complete code from scratch
- Dataset: MNIST
- Exp1:
  - 10 components
  - Calculated Confidence scores and stability of each state
  - Stability: 0.81, 0.99, 0.99, 0.44, 0.62, 0.98, 1.00, 0.99, 1.00, 0.46
  - Indicates the efficiency and understandability of state extraction
- Exp2:
  - 30 components
  - Created temporal features similar to the paper
  - Calculate top m influencing samples for a test prediction

Work done after midterm review

- Exp2 Results





Our Idea in Detail

Influence Analysis for DNNs

- Authors' solution is limited to RNNs
- They create states from the hidden states of RNN
- Our idea:
  - Use the output after the flatten layer as the hidden state of DNN
  - Collect all the 'hidden states' from the training samples
  - Cluster them using GMM
  - Use the abstract states for influence analysis

```
class CNN(nn.Module):
    def __init__(self):
      super(CNN, self).__init__()
      self.conv_block = nn.Sequential(
          nn.Conv2d(1, 10, kernel_size=5),
          nn.MaxPool2d((2,2),stride=(2,2)),
          nn.ReLU(),
          nn.Conv2d(10, 20, kernel_size=5),
          nn.MaxPool2d((2,2),stride=(2,2)),
          nn.ReLU()
      )

      self.dense_block = nn.Sequential(
          nn.Flatten(),
          nn.Linear(320, 50),
          nn.Linear(50, 10),
          nn.Softmax(dim=1)
      )

    def forward(self, input):
      return self.dense_block(self.conv_block(input))
```

Comparing our Idea

Detecting Adversarial Samples Using Influence Functions and Nearest Neighbors[1]

-Cohen et al.

- Detection of adversarial attacks
- Similarities:
  - Use of Influence functions and influence scores
  - Clustering internal representations of DNN
- Differences:
  - Different Influence function (Koh and Liang[2])
  - K-Nearest Neighbour classifier instead of GMM
  - Use of activations of all the layers

Conclusions

Hence, through our project we have:

- **Provided deeper insights**
  - comparing various clustering methods for the task
  - verifying the authors' hypothesis for different components
- **Made valuable improvements**
  - Thresholding C to improve accuracy
  - Tackling empty Influence set for more robust solution
- **Extended the idea of influence analysis to DNNs**
  - showed the effectiveness of abstract states
  - found the most influential training samples using temporal features
- **Pointed out some flaws in their solution**
- **Suggested gainful directions for future works**

Future Direction

- **Centroid-based clustering for improved temporal features**
  - Current temporal features just has the abstract state id
  - No consideration of the distance between two abstract states
  - Centroid-based clustering (e.g K-means)
  - Including the cluster center can improve the quality of the temporal features
- **Abstract states of DNN for Generating Images**
  - Deep encoder-decoder architecure
  - Abstract states of the bottleneck layer
  - Clustering using GMM
  - Sample from the distribution
  - Pass through the Decoder to generate new images
  - Similar to VAE

References I


Boopathy, A., W.-T.-W. C.-P.-Y. L. S. and Daniel, L. C.-c. (2019).
An efficient framework for certifying robust- ness of convolutional neural networks.
In *Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pp. 3240–3247, 2019.*


Cohen, G., Sapiro, G., and Giryes, R. (2019).
Detecting adversarial samples using influence functions and nearest neighbors.
CoRR, abs/1909.06872.


Hara, S., N.-A. and Maehara, T. (2019).
Data cleansing for models trained with sgd.
In *Advances in Neural Information Processing Systems*.

References II


Koh, P. W. W., A. K.-S.-T. H. and Liang, P. S. (2019).
On the accuracy of influence functions for measuring group effects.
In *Advances in Neural Information Processing Systems*.


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


Singh, G., G. T. M. M.-P.-M. and Vechev (2018).
M. fast and effective robustness certification.
In *Ad- vances in Neural Information Processing Systems, pp. 10802–10813, 2018.*

References III

Wang, H., U. B. and Calmon, F. P. (2019).
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)*.

Weiss, G., Goldberg, Y., and Yahav, E. (2018).
Extracting automata from recurrent neural networks using queries and counterexamples.
In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5247–5256. PMLR.