Explaining high-dimensional text classifiers

Odelia Melamed
Microsoft R&D, Israel
t-omelamed@microsoft.com

Rich Caruana
Microsoft Research, Redmond
rcaruana@microsoft.com

Abstract

Explainability has become a valuable tool in the last few years, helping humans better understand AI-guided decisions. However, the classic explainability tools are sometimes quite limited when considering high-dimensional inputs and neural network classifiers. We present a new explainability method using theoretically proven high-dimensional properties in neural network classifiers. We present two usages of it: 1) On the classical sentiment analysis task for the IMDB reviews dataset, and 2) our Malware-Detection task for our PowerShell scripts dataset.

1 Introduction

In the last few years, Neural Networks have been commonly used for a variety of text-classification tasks (Zhang et al. [2016]; Yang et al. [2016]), from simple sentiment analysis to even fake news filtering (Karmakharm et al. [2019]). Unfortunately, those classifiers have trouble gaining human trust for two reasons: 1) The neural networks are considered black-box models, where one can see the inputs and output yet not understand the function in between; and 2) Textual inputs create non-continuous input space, which is hard to explore. A great effort has been devoted to developing methods to explain these models to increase trust and, therefore, their usability. Several methods were set to overcome the black-box barrier, creating good explanations for predictions on continuous data domains (Lundberg and Lee [2017], Poerner et al. [2018]). However, this is only sometimes successful with continuous high-dimensional datasets (e.g., images). Experimentally, it has been shown that the well-known explainability tools on high-dimensional data and neural network classifiers behave differently.

If we look at it from the point of view of adversarial examples research, these inadequate explanations are yet another effect of the adversarial mystery (Szegedy et al. [2013] and Biggio et al. [2013]). Experimentally, we witness this phenomenon in the case of neural network classifiers and high-dimensional data space. There, gradient calculations and other related methods experimentally lead us to adversarial examples: tiny, noise-looking changes in the input will mysteriously change the classifier decision ([Carlini and Wagner, 2017, Papernot et al., 2017, Athalye et al., 2018]). These small changes were experimentally shown as an out-of-distribution perturbation (Shamir et al. [2021] and many others). Indeed, even in our security domain and coding inputs, adversarial examples have been studied as potential security hacks (Schuster et al. [2021]): one can change a minor part of the code input (that does not influence its execution) and change the classifier's prediction.

In the explainability research, some have noticed the connection between these out-of-distribution adversarial examples and the inadequate explanations for high-dimensional data (Feng et al. [2018]). They note that the reason for the existence of these explanations is the high dimensionality of the input while implicitly relying on a lower-dimensional manifold (Anders et al. [2020]). Therefore, they suggest methods to ensure generating explanations within the data distribution (i.e., on the data manifold), using existing tools that are known to help avoid adversarial examples, such as Auto-Encoders (Alvarez-Melis and Jaakkola [2017]), Generative models (Chang et al. [2018]), using surrogate models (Anders et al. [2020]), and others. Such data manifold exploration

XAI in Action: Past, Present, and Future Applications @ NeurIPS 2023.

tools are not designed to be used when dealing with non-continuous input space such as text. Translating vector direction to a change in words is complex, so one cannot use simple vector manipulations. However, we can compromise for less - unlike adversarial robustness, detecting post-hoc the off-manifold examples is an easier task since no attacker is challenging the system.

In this paper, we are using a new perspective on adversarial examples to create informative explanations in the non-continuous input space. We use the theory from Melamed et al. [2023] regarding neural networks trained on data relying on a low dimensional linear subspace to analyze the gradients off the data subspace. First, we note that the off-manifold gradients tend to have a big norm and, therefore, can be filtered using a simple threshold. In addition, we prove that different classifiers trained on the same data distribution will result in highly uncorrelated off-manifold gradients with high probability (using cosine similarity). We then use these conclusions in continuous input space regarding correlations and norms to create an "on-manifold" explanation in the non-continuous input space of text.

Powershell is a common scripting language used by network administrators worldwide to perform anything from routine maintenance to complex admin tasks on a large number of machines. Unfortunately, attackers use these capabilities to develop malicious PowerShell scripts. As Powershell scripts are a widespread admin tool, we set out to classify malicious vs. benign scripts to detect malicious activity. As in many other domains, a Machine Learning based classification is accurate and frequently used. The explanation tools we provide are crucial for successful risk analysis and response. We utilize the many benefits of explanations, such as improved human trust, model and data evolution, and business intelligence. In addition, we can use predictions' explanations to identify the malicious areas in the code, mark them, and eventually prevent a comprehensive cyber-attack.

In our paper, we present a simple new method for creating on-manifold explanations. It will transfer easily between domains and even deployment environments. It only requires repeating the training for the same data distribution a few extra times, with no additional changes. This method is very approachable if the training set is very large or the training procedures are well-optimized and almost unchangeable, which is the case for many data-driven products today. We test it with two datasets: 1) the IMDB reviews dataset with the classical sentiment analysis task and 2) an industry Security PowerShell code dataset used to train models for malware detection. We present outstanding results in both settings compared to standard methods such as gradient max-norm, LIME, or SHAP.

2 Related Work

With the vast use of machine learning in text classification arises the demand for explanations of these classification decisions (Ribeiro et al. [2016], Wu et al. [2022]). For a few years now, scientists have been looking beyond correct classification to correct feature importance or silency maps (Simonyan et al. [2013], Ross et al. [2017], Ancona et al. [2017], Sundararajan et al. [2017]). As many traditional explainability methods (Lundberg and Lee [2017], Poerner et al. [2018] and many others) frequently empirically failed to explain text classification tasks and particularly neural network classifiers, this subject has gained research interest.

In the last few years, researchers noticed some methods yield out-of-distribution explanations, sometimes referred to as explanations off the data manifold, and have tried to instead pursue on-manifold importance or explanations. (Wallace et al. [2018]) tried to explore the data manifold using the nearest neighbor. Some had manifold explorations solutions that apply to images or continuous data only (Chang et al. [2018],Agarwal and Nguyen [2020],Frye et al. [2020],Zhang et al. [2016]) starting from encoding and decoding, explanations generators, and even more complicated models for explainability that require interaction (Arous et al. [2021]). Few trained robust models with some extra loss, sometimes require extra explanation information (Ross et al. [2017], Anders et al. [2020],Liu et al. [2018]).

In our paper, we start with a theoretical analysis of the problem and solutions and present a straightforward post-training explanation generation. Using no extra losses or special networks, which we believe is more accessible to the industry needs.

3 Theoretical background

In Melamed et al. [2023], the authors define the data distribution in a simplified way, as sampled from a low dimensional linear subspace of the input space. Using this simplification, they showed that the off-manifold gradients of the trained neural network have a large norm, hinting at the existence of nearby adversarial examples off the data distribution. The analysis of these gradients also yields a vital realization - they deviate very little from their initialization.

In this section, we analyze the gradients according to these simplified settings. we denote the full input dimension as d, and the data subspace dimension as $d - \ell$. We denote an input sample by $x_0 \in \mathbb{R}^d$, where $x_0 \sim M \subset \mathbb{R}^d$ for some linear subspace M of dimension $d - \ell$. Note that we cannot calculate ℓ exactly in real-life datasets. In general, when using simple methods for linear subspaces such as PCA, several off-manifold dimensions can be approximated. Yet, especially in cases when the data manifold is not exactly linear, other off-manifold dimensions might be accidentally considered as on-manifold dimensions.

3.1 Gradient Norms

In Melamed et al. [2023], the only restriction on the dataset is to lie on M, with no constraints on the distance between the data points. Surprisingly, under reasonable assumptions, the authors prove a lower bound of the off manifold gradient norm of $\Omega(1)$, which hints at a close-by adversarial example (i.e., an example from the opposite class).

3.2 Cosine Similarity between Same-Input-Gradients

It was shown in Melamed et al. [2023] that the off-manifold gradients are affected mainly by the initialization. Therefore, when we train two different neural networks N_1, N_2 with the same training method on the same training data distribution, one can expect that the two off-manifold gradients would be very non-correlated (with cosine-similarity of some sub-exponential with $\approx \frac{1}{\ell}$ variance). Formally, we use the simplified setting of a two-layer, fully connected neural network to show exponential concentration bound for the inner product between the gradients. We define:

$$N_1(x, \mathbf{w}^1_{1:m}) = \sum_{i=1}^{m} u_i \sigma({w_i^1}^\top x), \;\; N_2(x, \mathbf{w}^2_{1:m}) = \sum_{i=1}^{m} v_i \sigma({w_i^2}^\top x) \ .$$

Theorem 3.1. *Let an input sample $x_0 \in M \subset \mathbb{R}^d$. For neural network N_1, let $S_1 = \{i \in [m] : \langle w_i^1, x_0 \rangle \geq 0\}$, and let $k_1 := |S_1|$. Similarly for N_2, $S_2 = \{i \in [m] : \langle w_i^2, x_0 \rangle \geq 0\}$, and let $k_2 := |S_2|$. We denote by g_i the gradient of the network N_i with respect to the input at x_0, i.e. $g_i = \frac{\partial N_i(x_0)}{\partial x}$, and its projection on M by $\tilde{g}_i$ i.e. $\tilde{g}_i = \Pi_{M^\perp}\left(\frac{\partial N_i(x_0)}{\partial x}\right)$. Then:*

$$\Pr\left[|\langle \tilde{g}_1, \tilde{g}_2 \rangle| \geq \frac{\sqrt{2\ell}}{d} \right] \leq e^{-\ell/16} + 2e^{-m/2} \ .$$

The full proof can be found in appendix A. In short, we note that:

$$\tilde{g}_1 = \sum_{i \in S_1} u_i \hat{w}_i^1 \ , \ \tilde{g}_2 = \sum_{i \in S_2} v_i \hat{w}_i^2$$

and therefore,

$$|\langle \tilde{g}_1, \tilde{g}_2 \rangle| = |\langle \sum_{i \in S_1} u_i \hat{w}_i^1, \sum_{i \in S_2} v_i \hat{w}_i^2 \rangle| = \frac{1}{m^2} |\langle \sum_{i \in S_1} sign\left(u_i\right) \hat{w}_i^1, \sum_{i \in S_2} sign\left(v_i\right) \hat{w}_i^2 \rangle| \ .$$

We also note that $\sum\limits_{i \in S_1} sign\left(u_i\right) \hat{w}_i^1 \sim \mathcal{N}\left(\mathbf{0}, \frac{k_1}{d} I_\ell\right)$, and $\sum\limits_{i \in S_2} sign\left(v_i\right) \hat{w}_i^2 \sim \mathcal{N}\left(\mathbf{0}, \frac{k_2}{d} I_\ell\right)$. We then use concentration bounds provided in the paper to conclude the proof.

Corollary 3.1. *In the settings of Theorem 3.1, assume that $\ell = \Theta(d)$ and $k = \Theta(m)$. Then, with probability $\geq 1 - (2e^{-\Theta(d)} + 2e^{-m})$:*

$$|S_C(\tilde{g}_1, \tilde{g}_2)| \leq \Theta\left(\frac{1}{\sqrt{\ell}}\right) \ .$$

See Appendix A for details. As expected, under the original assumptions, we showed that the gradient vectors w.r.t. the same input for two differently initialized networks are highly non-correlated, by giving an upper bound for their cosine similarity.

4 Our Method - Theory to Practice

In text, a pre-processed and embedded input sample is a 2D matrix $x_0 \in \mathrm{R}^n \times \mathrm{R}^p$, where n is the number of words in the input (padded or clipped if needed), and p is the embedding dimension chosen for each word. One main limitation exists when using gradient-based tools on a textual dataset rather than continuous data - we cannot explore the input space by changing an input word in the direction of the gradient. Therefore, we usually look at each word's gradient norm to determine its significance rather than its direction. In this section, we use the theoretical observations to find the "on-manifold" gradients in settings where the data manifold is not easily found or even defined (e.g., in the case of a non-continuous data manifold).

Separating the input gradient into n words' gradients of dimension p, we wish to understand if a word's gradient is mostly on or off the data manifold. Let C be the classifier we are explaining. We denote by g_C the gradient of C with respect to the input at x_0 (i.e., $g_C = \frac{\partial C(x_0)}{\partial x}$). Note that $g_C \in \mathrm{R}^n \times \mathrm{R}^p$. We denote by g_C^j the gradient of C with respect to the j-th word of the input at x_0. We say that the word's gradient is mostly off-manifold if most of its coordinates are off-manifold coordinates. We wish to detect the gradients that are mostly off-manifold and keep the ones that are mostly on the manifold.

4.1 Expected Gradient Norms

We assume that our data approximately lie within a low dimensional linear subspace. In addition, assuming the inputs lie within a $O(\sqrt{d})$ distance from each other is standard in real-life data. In this case, one can expect an average gradient of $O(\frac{1}{\sqrt{d}})$ along the shortest path between two different input samples (this path also lies in the linear subspace). It is easy to see that an $\Omega(1)$ norm of the off manifold gradient is very surprising. Consequently, one can expect the off-manifold gradient to have a relatively large norm, also when divided into p-dimensional vectors. Specifically, for any $j \in [n]$, we look at $\left\| g_C^j \right\|$.

4.2 Expected Cosine Similarity

Let $\{N_i\}_{i=1}^t$ be our surrogate classifiers ensemble, where each classifier has been trained on the same training distribution as C with different initialization weights. We denote by g_i the gradient of the surrogate network N_i with respect to the input at x_0, i.e. $g_i = \frac{\partial N_i(x_0)}{\partial x}$. Note, $g_i \in \mathrm{R}^n \times \mathrm{R}^p$. Similarly to g_C^j, we denote g_i^j as the gradient of N_i with respect to the j-th word of the input at x_0. Note, $g_C^j, g_i^j \in \mathrm{R}^p$. Now, for any $j \in [n]$, we look at $\alpha_{g_C^j} = \frac{1}{t}\sum_{i=0}^{t} |S_C(g_C^j, g_i^j)|$.

4.3 The Method

Our method aims to find the words' gradients that are mostly on-manifold. If the gradient was mostly off-manifold, we expect it to have a relatively large norm and slight cosine similarity with the corresponding gradients of the other networks, i.e., small $\alpha_{g_C^j}$ (as explained in 3). Therefore, looking for the on-manifold gradient, we take the maximal $\alpha_{g_C^j}$, between those with relatively small gradient norms. The relevant norm threshold for each dataset should be of size $O(\frac{1}{\sqrt{p}})$. Since the O notation can hide any constant, one should test the dataset's gradient's norms distribution to easily understand the relevant norm threshold, see Section 5 and Appendix B and C for examples.

Algorithm 1 Choose k top words for explanation

```
Input: Classifier $C$, surrogate classifiers $\{N_i\}_{i=1}^{t}$, norm treshold $T$
$g_C \leftarrow \frac{\partial C(x_0)}{\partial x} \in \mathrm{R}^n \times \mathrm{R}^p$,
$\forall i \in [t]$, $g_i \leftarrow \frac{\partial N_i(x_0)}{\partial x} \in \mathrm{R}^n \times \mathrm{R}^p$
$small - norms \leftarrow \{j : \left\| g_C^j \right\| < T\}$
$avarage - cosine - similarities \leftarrow \{\alpha_{g_C^j} : j \in small - norms\}$
return Max-k $(avarage - cosine - similarities)$
```

5 Experiment - IMDB dataset

5.1 Data Manifold analysis Results

We use a Sentiment Analysis classifier for IMDB reviews. On this dataset, the input dimension is $d = 32,000$, $n = 500$ words are taken for each review, and the embedding size is $p = 64$. Figure 1 shows the implicit low-dimensional linear subspace for this dataset using PCA decomposition. One can see that we can separate many "zero"-data dimensions (zero up to a rotation) from the data subspace. The accumulated variance reaches 1 after considering only the first 6000 features. Note that for the explanation we disregard the padding, so there are fewer off-manifold dimensions expected for the text only. This figure hints that the separation between off and on-manifold gradients will be beneficial.

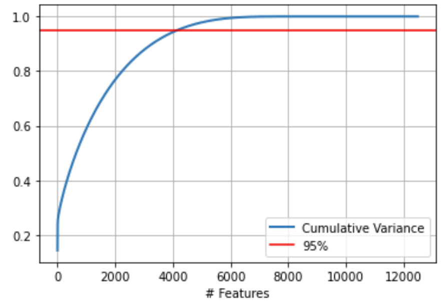


Figure 1: PCA for IMDB Sentiment Analysis Dataset. Showing cumulative variance for input dimension $d = 32,000$ we can see that the first 4000 already gets 95% of the variance, and the first 6000 are enough. So, here $d - l \leq 6000$.

5.2 Choosing norm threshold

In Figure 2 we plot in a histogram the different norms, normalized using L_∞ norm for convenience, of the words in the input sentence. In the figure, one can see a clear gap between words above and below the threshold of 0.1. The histogram shows $1)$ many words with norms smaller than 0.1, $2)$ most norms in the middle section (which makes sense after observing many off-manifold dimensions in the previous section), and $3)$ a few with larger norms. For visualization, the negligible norms (less than e^{-3}) are filtered. Altogether, This histogram hint on 0.1 as a good candidate for norm threshold for this dataset. This simple test can be done for each input separately or for the entire dataset together, according to user preferences and application.

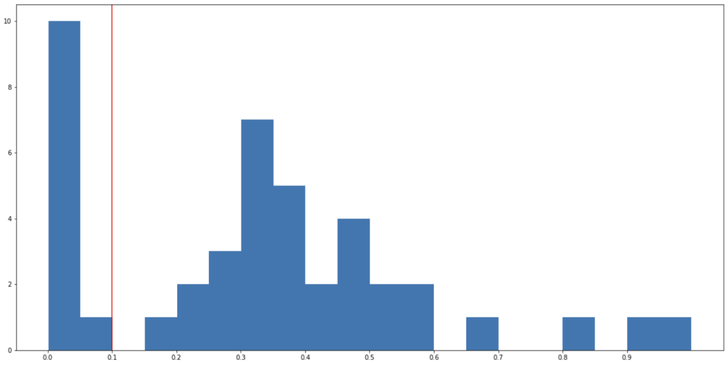


Figure 2: Word Gradient Norm for IMDB Sentiment Analysis Dataset. One can see a clear gap around 0.1.

5.3 Explanation Results

After choosing a norm threshold for this dataset, we take the top ten words j with the maximal $\alpha_{g_C^j}$ among those with $\left\|g_C^j\right\|$ smaller than the norm threshold. Appendix B provides more details about this experiment. Table 1 compares our new explanation method to the classical gradient-based method of taking top-norm words. One can see that the top norm words do not relate to negative sentiment and are neutral even in context. Similar explanations of this text using LIME and SHAP methods can be found in Appendix B. The words picked by our method are clearly related to negative sentiment, e.g., "poor story", "impossible plot", etc..

Table 1: IMDB Sentiment Explanation

By Norm	Seems Sensei Seagal is getting more and more moralising and less "action packed". To date this has to be his worse movie... no action, a poor story line, an impossible plot and to make things worse, one of the CHEEZI-EST endings I have ever seen. Seagal films are like seeing a Dirty-Harry, you do not go see it for the great social causes or impeccable acting... you want a good action flick. On a scale of 1 to 10, this one gets a 1...
Ours	Seems Sensei Seagal is getting more and more moralising and less "action packed". To date this has to be his worse movie... no action, a poor story line, an impossible plot and to make things worse, one of the CHEEZI-EST endings I have ever seen. Seagal films are like seeing a Dirty-Harry, you do not go see it for the great social causes or impeccable acting... you want a good action flick. On a scale of 1 to 10, this one gets a 1...

Table 2: IMDB movie review labeled **negative** and Neural Network sentiment model explained by vector norm and by our method. Orange words are the top 10 for each method.

6 Security Dataset Experiment

In the Security context, explainability tools can be a real game-changer for the malware detection task. As millions of lines of scripts are executed hourly on each computer in an organization, it's very hard to detect malware spreading and threatening the organization's computers and data. In addition, this classification task is constantly challenged by hackers. Potential attackers keep trying to manipulate their code to avoid getting caught by the detector while maintaining its harmful functionality. Therefore, trusting the detector to identify the harmful parts correctly is particularly crucial. For a simple example, we look at the code:

"foreach ($harmful_item in $harmful_set) {<harmful functionality>}".

If the detector recognizes malware-related activity thanks to the variable names "$harmful_item" and "$harmful_set", the attacker will simply change their names and bypass our detector (Schuster et al. [2021]). We want to ensure that the detector can identify harmful functionality given different variable names. Note that creating the correct explanation does not mean our detector is not still sensitive to adversarial examples, but a correct explanation will help us to adjust its functionality when it is mistaken.

6.1 Explanation Results

For this experiment, we chose the same norm threshold for this dataset as in the IMDB experiment. Similarly, we take the top ten words j with the maximal $\alpha_{g_C^j}$ among those with $\left\| g_C^j \right\|$ smaller than the norm threshold. More about the experiment details and threshold choice are in Appendix C.

Table 4 shows PowerShell code detected as malware, explained by the classical gradient-based by word norm, and using our method. The norm-based method mistakenly prefers the first few tokens that are just general security networking setups. In Appendix C, we show similar explanations using the LIME and SHAP algorithms. At the bottom, we show our method chooses several important expressions commonly related to malware: the "URI" is mandatory for functionality and commonly used to download malicious content from a remote server. "Convert", "FromBase64String", and "UTF8" terms are needed to make downloaded content into an executable script, and the "Invoke-Expression" is mandatory to execute that script, so this is an excellent explanation.

Table 3: PowerShell Detection Explanation

By Norm	[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;$xor = [System.Text.Encoding]::UTF8.GetBytes('**-**-**');$base64String = (Invoke-WebRequest -URI https://**.blob.core.**.**/**/**.txt -UseBasicParsing).Content;Try{ $contentBytes = [System.Convert]::FromBase64String($base64String) } Catch { $contentBytes = [System.Convert]::FromBase64String($base64String.Substring(3)) };$i = 0; $ decryptedBytes = @();$contentBytes.foreach{ $decryptedBytes += $ _ -bxor $xor[$i]; $i++; if ($i -eq $xor.Length) { $i = 0} };Invoke-Expression ([System.Text.Encoding]::UTF8.GetString($decryptedBytes))
Ours	[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;$xor = [System.Text.Encoding]::UTF8.GetBytes('**-**-**');$base64String = (Invoke-WebRequest -URI https://**.blob.core.**.**/**/**.txt -UseBasicParsing).Content;Try{ $contentBytes = [System.Convert]::FromBase64String($base64String) } Catch { $contentBytes = [System.Convert]::FromBase64String($base64String.Substring(3)) };$i = 0; $ decryptedBytes = @();$contentBytes.foreach{ $decryptedBytes += $ _ -bxor $xor[$i]; $i++; if ($i -eq $xor.Length) { $i = 0} };Invoke-Expression ([System.Text.Encoding]::UTF8.GetString($decryptedBytes))

Table 4: PowerShell script labeled **malware-related** and Neural Network classifier explained by vector norm and by our method. Orange-colored words are the top 10 for each method.

7 Conclusions and Future Work

In this paper we presented a novel method for creating on-manifold explanations, using a recent theoretical model from adversarial examples research. We presented a natural language use of this method on the IMDB sentiment analysis task, as well as on industrial scripts dataset for the malware detection task. One interesting research area using our theoretical work is understanding the gradient behaviour for implicit low-dimensional datasets for network architectures that are used in text-classification tasks. Another research area we find inspiring is the interdisciplinary point of view, i.e., the on manifold exploration in general. Currently, there is an extensive research effort to approximate and explore the data manifold, and the explainability could benefit from this research. One can use these efforts to help generate on-manifold post-hoc explanations in many different settings. In the other direction, An interesting future direction is how to enhance adversarial robustness using tips from explanations.

References

Chirag Agarwal and Anh Nguyen. Explaining image classifiers by removing input features using generative models. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

David Alvarez-Melis and Tommi S Jaakkola. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*, 2017.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2020.

Ines Arous, Ljiljana Dolamic, Jie Yang, Akansha Bhardwaj, Giuseppe Cuccu, and Philippe Cudré-Mauroux. Marta: Leveraging human rationales for explainable text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 5868–5876, 2021.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. *arXiv preprint arXiv:1807.08024*, 2018.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*, 2018.

Christopher Frye, Damien de Mijolla, Tom Begley, Laurence Cowton, Megan Stanley, and Ilya Feige. Shapley explainability on the data manifold. *arXiv preprint arXiv:2006.01272*, 2020.

Twin Karmakharm, Nikolaos Aletras, and Kalina Bontcheva. Journalist-in-the-loop: Continuous learning as a service for rumour analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 115–120, 2019.

Hui Liu, Qingyu Yin, and William Yang Wang. Towards explainable nlp: A generative explanation framework for text classification. *arXiv preprint arXiv:1811.00196*, 2018.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Odelia Melamed, Gilad Yehudai, and Gal Vardi. Adversarial examples exist in two-layer relu networks for low dimensional data manifolds. *arXiv preprint arXiv:2303.00783*, 2023.

Nicolas Papernot, 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*, pages 506–519, 2017.

Nina Poerner, Benjamin Roth, and Hinrich Schütze. Evaluating neural network explanation methods using hybrid documents and morphological agreement. *arXiv preprint arXiv:1801.06422*, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " 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*, pages 1135–1144, 2016.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1559–1575, 2021.

Adi Shamir, Odelia Melamed, and Oriel BenShmuel. The dimpled manifold model of adversarial examples in machine learning. *arXiv preprint arXiv:2106.10151*, 2021.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *Preprint, arXiv:1312.6199*, 2013.

Eric Wallace, Shi Feng, and Jordan Boyd-Graber. Interpreting neural networks with nearest neighbors. *arXiv preprint arXiv:1809.02847*, 2018.

Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381, 2022.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

Ye Zhang, Iain Marshall, and Byron C Wallace. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 795. NIH Public Access, 2016.

A Proof from Section 3

Proof. we noted that:

$$\Pi_{M^\perp}\left(\frac{\partial N_1(x_0)}{\partial x}\right)=\sum_{i\in S_1}u_i\hat{w}_i^1\ ,\ \Pi_{M^\perp}\left(\frac{\partial N_2(x_0)}{\partial x}\right)=\sum_{i\in S_2}v_i\hat{w}_i^2\ .$$

Therefore,

$$|\langle\tilde{g}_1,\tilde{g}_2\rangle|=|\langle\sum_{i\in S_1}u_i\hat{w}_i^1,\sum_{i\in S_2}v_i\hat{w}_i^2\rangle|=\frac{1}{m^2}|\langle\sum_{i\in S_1}sign\left(u_i\right)\hat{w}_i^1,\sum_{i\in S_2}sign\left(v_i\right)\hat{w}_i^2\rangle|\ .$$

Next, one can see that $\sum\limits_{i\in S_1}sign\left(u_i\right)\hat{w}_i^1\sim\mathcal{N}\left(\mathbf{0},\frac{k_1}{d}I_\ell\right)$, and $\sum\limits_{i\in S_2}sign\left(v_i\right)\hat{w}_i^2\sim\mathcal{N}\left(\mathbf{0},\frac{k_2}{d}I_\ell\right)$.

Therefore by Lemma C.3 in Melamed et al. [2023] we get that:

$$\Pr\left[|\langle\sum_{i\in S_1}sign\left(u_i\right)\hat{w}_i^1,\sum_{i\in S_2}sign\left(v_i\right)\hat{w}_i^2\rangle|\geq\frac{\sqrt{2\ell}}{d}m\sqrt{k_1}\right]\leq e^{-\ell/16}+2e^{-m^2/2k_2}\ .$$

And therefore,

$$\begin{aligned}&\Pr\left[|\langle\tilde{g}_1,\tilde{g}_2\rangle|\geq\frac{\sqrt{2\ell}}{d}\right]\\=&\Pr\left[\frac{1}{m^2}|\langle\sum_{i\in S_1}sign\left(u_i\right)\hat{w}_i^1,\sum_{i\in S_2}sign\left(v_i\right)\hat{w}_i^2\rangle|\geq\frac{\sqrt{2\ell}}{d}\right]\\\leq&\Pr\left[|\langle\sum_{i\in S_1}sign\left(u_i\right)\hat{w}_i^1,\sum_{i\in S_2}sign\left(v_i\right)\hat{w}_i^2\rangle|\geq\frac{\sqrt{2\ell}}{d}m\sqrt{k_1}\right]\\\leq&e^{-\ell/16}+2e^{-m^2/2k_2}\\\leq&e^{-\ell/16}+2e^{-m/2}\ .\end{aligned}$$

As $0\leq k_1,k_2\leq m$.

For the corollary to hold, we note that Lemma C.3 uses Lemma C.1 and the assumption that the norm of the normally distributed vector is not too big. For $w\sim\mathcal{N}(\mathbf{0},\sigma^2I_n)$:

$$\Pr\left[\|w\|^2\geq2\sigma^2n\right]\leq e^{-\frac{n}{16}}\ .$$

Therefore, to conclude that

$$|S_C(\tilde{g}_1,\tilde{g}_2)|\leq\Theta\left(\frac{1}{\sqrt{\ell}}\right)\ ,$$

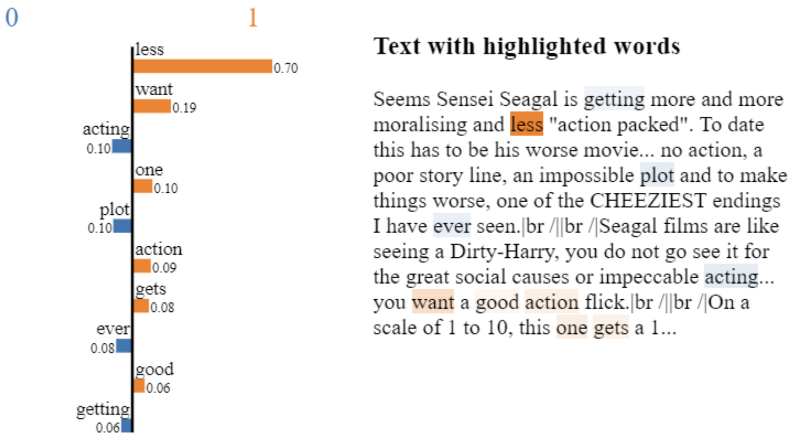
One should only note that under the corollary assumption, both $\tilde{g}_1$ and $\tilde{g}_2$ having $\Theta(1)$ norm with probability $\geq1-e^{-\Theta(d)}$. □

B Additional Information for IMDB Experiment

The classifier architecture found in https://www.kaggle.com/code/arunmohan003/sentiment-analysis-using-lstm-pytorch/notebook, and the dataset taken from https://www.kaggle.com/code/arunmohan003/sentiment-analysis-using-lstm-pytorch/input. We used the same hyper-parameters as in the cited notebook, trained for 50 epochs.

B.1 Other Explanation Algorithms

We add two more explanations using the popular LIME and SHAP methods in Figure 3 and Figure 4, respectively. One can see in both examples, that the explanations point out a few of the big-norm words picked also by the classical gradient algorithm chooses words by the norm. One can show an analysis for which, for simple cases, the LIME and SHAP methods are indeed also biased toward the adversarial directions from the input.



Text with highlighted words

Seems Sensei Seagal is getting more and more moralising and less "action packed". To date this has to be his worse movie... no action, a poor story line, an impossible plot and to make things worse, one of the CHEEZIEST endings I have ever seen.|br /||br /|Seagal films are like seeing a Dirty-Harry, you do not go see it for the great social causes or impeccable acting... you want a good action flick.|br /||br /|On a scale of 1 to 10, this one gets a 1...

Figure 3: LIME textual explanation for IMDB Sentiment Analysis Dataset. One can see for example that "gets" and "getting" are both colored as opposite sentiment related, while both are neutral. In addition, "plot" and "acting" are mistakenly colored as negative sentiment related, as opposed to "action", all quite neutral.

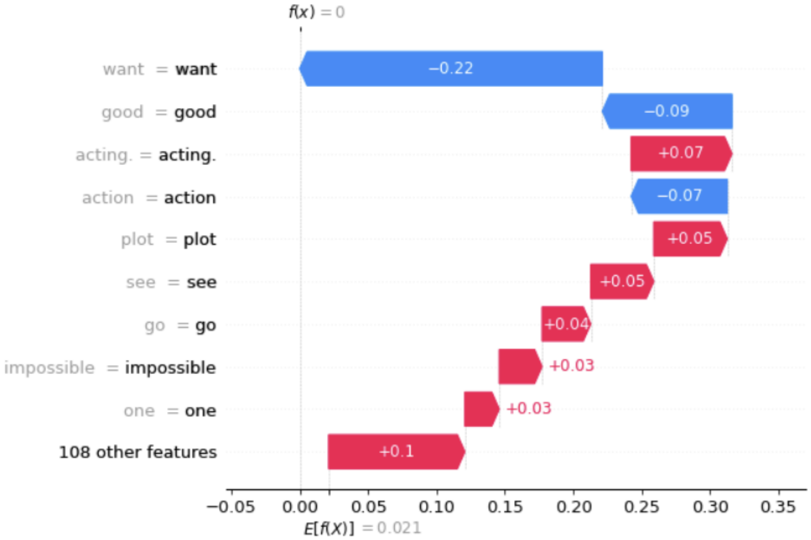


Figure 4: SHAP explanation for IMDB Sentiment Analysis Dataset. One can see here too, mostly neutral words picked by the algorithm.

C Additional Information for Security Detection Experiment

C.1 Choosing threshold

For this experiment, we used the same threshold 0.1 for the norms (normalized using L_∞ norm for convenience). In Figure 5 we show that this threshold still seems to separate well the big norms from the smaller ones, as the bars are dramatically shorter right to the 0.1 threshold. For visualization, the negligible norms (less than e^{-3}) are filtered. This short test can be done for each input separately or for the entire dataset together, according to user preferences and application.

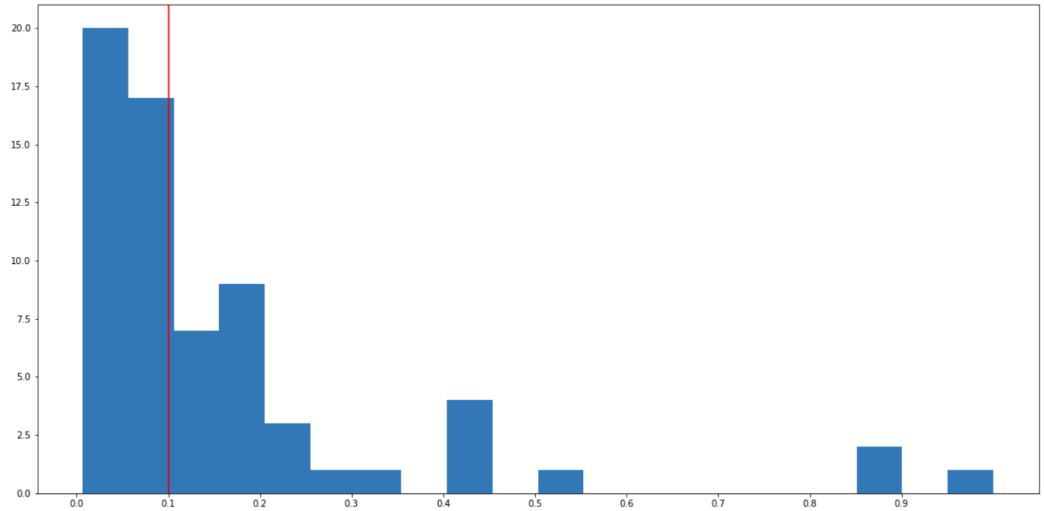
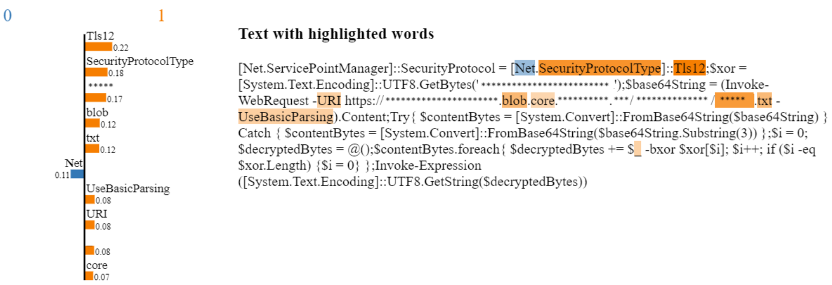


Figure 5: Word Gradient Norm for PowerShell Malware Detection Dataset. One can see a clear cut around norm 0.1.

C.2 Other Explanation Algorithms

Here we add two more explanations by the widely used LIME and SHAP algorithms for text inputs in Figure 6 and Figure 7, respectively. Here too, we can see that the two algorithms picked a word that had a big gradient norm, as seen in Section 6 in Table 4 in the max-norm row. We censor the user-sensitive information. In the pre-processing before training, we replace private user information with canonical saved words, here one can see two of them have been mistakenly chosen by the explanation methods.



Text with highlighted words

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;$xor =
[System.Text.Encoding]::UTF8.GetBytes('************************ ');$base64String = (Invoke-
WebRequest -URI https://**********************.blob.core.**********.***/ ************** / ***** .txt -
UseBasicParsing).Content;Try{ $contentBytes = [System.Convert]::FromBase64String($base64String) }
Catch { $contentBytes = [System.Convert]::FromBase64String($base64String.Substring(3)) };$i = 0;
$decryptedBytes = @();$contentBytes.foreach{ $decryptedBytes += $_ -bxor $xor[$i]; $i++; if ($i -eq
$xor.Length) {$i = 0} };Invoke-Expression
([System.Text.Encoding]::UTF8.GetString($decryptedBytes))
```

Figure 6: LIME textual explanation for PowerShell Malware Detection Dataset. One can see similar word choosing to the big-norm method: choosing of neutral coding phrases and canonical file names shared for all input data.

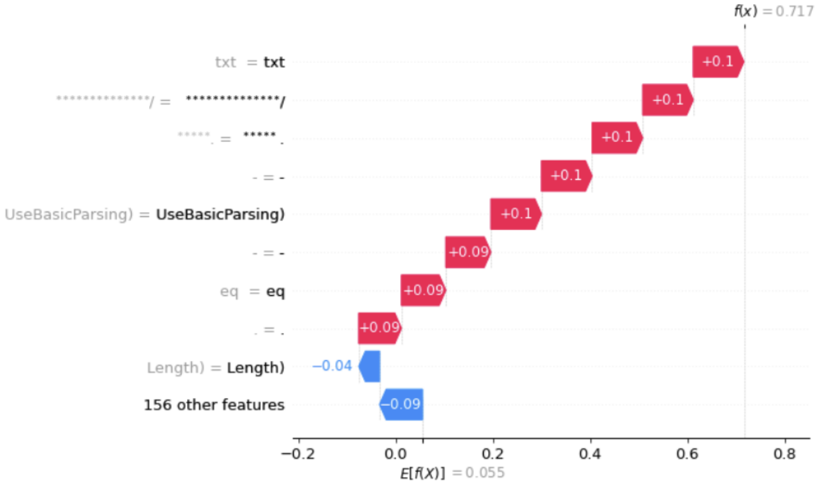


Figure 7: SHAP explanation for PowerShell Malware Detection Dataset. One can see again canonical phrases chosen as well as a hyphen, a dot, and neutral words.