EFFICIENT SHAPLEY VALUES ESTIMATION BY AMORTIZATION FOR TEXT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

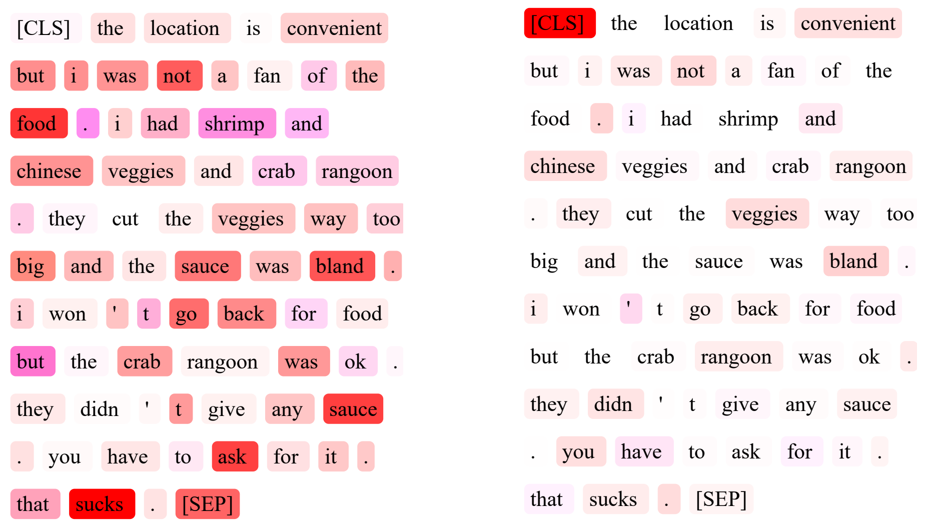
Despite the popularity of Shapley Values in explaining neural text classification models, computing them is prohibitive for large pretrained models due to a large number of model evaluations as it needs to perform multiple model evaluations over various perturbed text inputs. In practice, Shapley Values are often estimated stochastically with a smaller number of model evaluations. However, we find that the estimated Shapley Values are quite sensitive to random seeds—the top-ranked features often have little overlap under two different seeds, especially on examples with the longer input text. As a result, a much larger number of model evaluations is needed to reduce the sensitivity to an acceptable level. To mitigate the trade-off between stability and efficiency, we develop an amortized model that directly predicts Shapley Values of each input feature without additional model evaluation. It is trained on a set of examples with Shapley Values estimated from a large number of model evaluations to ensure stability. Experimental results on two text classification datasets demonstrate that, the proposed amortized model can estimate black-box explanation scores in milliseconds per sample in inference time and is up to 60 times more efficient than traditional methods.

1 INTRODUCTION

Many powerful natural language processing (NLP) models used in commercial systems only allow users to access model outputs. When these systems are applied in high-stake domains, such as healthcare, finance and law, it is essential to interpret how these models come to their decisions. To this end, post-hoc black-box explanation methods have been proposed to identify the input features that are most critical to model predictions (Ribeiro et al., 2016; Lundberg & Lee, 2017). A famous class of post-hoc black-box local explanation methods leverages the Shapley Values (Shapley, 1953) to identify important input features, such as Shapley Value Sampling (SVS) (Strumbelj & Kononenko, 2010) and KernelSHAP (Lundberg & Lee, 2017). These methods typically start by stochastically sampling a set of perturbations or permutations of the input ("*perturbation samples*") and summarize how the model outputs are changed. Then they will assign an *explanation score* for each input feature to indicate its contribution to the output.

Despite the widespread usage of the post-hoc interpretation methods, we observe that when they are applied to text data, the estimated explanation score for each token varies significantly based on different random seeds of the sampling process. This issue becomes more severe when the input sequence becomes longer and can only be partially mitigated by collecting a large number of samples, which would induce unaffordable computational and time burdens. This instability and sensitivity to randomness in the sampling process will lead to an unreliable interpretation of the model predictions and hinder developers from understanding model behavior.

Figure 1 shows an example of interpreting a BERT-based sentiment classifier (Devlin et al., 2019) on Yelp-Polarity dataset , a restaurant review dataset (Zhang et al., 2015) by KernelSHAP (Lundberg & Lee, 2017). The interpretation results vary significantly when using different random seeds (see details in the caption). They are stable only when the number of perturbed samples increases to more than 2,000. As KernelSHAP requires access to the output of the sentiment classifier for each perturbed sample, the computation cost is enormously high. For example, it takes about 183 seconds to interpret each instance on Yelp-Polarity using the KernelSHAP implementation in Captum (Kokhlikyan et al., 2020) on A100 GPU.



(a) Seed=1 (b) Seed=2

Figure 1: Heatmaps for two runs of KernelSHAP (Lundberg & Lee, 2017) that only differ in random seeds for a fine-tuned BERT model (Morris et al., 2020) on Yelp-polarity, using 200 samples (approx. 3.47s per instance on average using single A100 GPU, more than 150 times slower than the forward process of the target model). The darker each token is, the higher explanation score is. We observe that the interpretation results are significantly different when using different seeds.

To achieve a better trade-off between efficiency and stability, we propose a simple yet effective amortized model to approximate the model explanation scores. This is inspired by the observation that different instances share a similar set of important words (e.g., in sentiment classification, emotional words can be strong label indicators (Taboada et al., 2011)). Therefore, an amortized model can leverage similar interpretation patterns across instances when predicting the explanation scores. Specifically, we first collect pairs of inputs and their corresponding explanation scores based on the text classifier. These explanation scores are used as references to train an amortized model. Note that although we need to collect a large enough set to train a stable and precise amortized model, the collection needs to be done only once. At inference time, our amortized model directly outputs the explanation scores for new instances. We show that the amortized model generates consistent results similar to the ones from running massive target model evaluations.

Our experiments on both MNLI and Yelp-Polarity datasets using BERT demonstrate the following: 1) Our amortized model achieves a better trade-off between stability and efficiency. It reduces the computation time for a typical instance from about 3.47s per instance to less than 50ms[1], which is 60 times faster. This comes with the one-time efforts to collect training data from stable interpretation methods. 2) We further show that with only a few thousand training samples, the amortized model achieves a stable estimation of reference scores. 3) Our model is robust to training time randomness (e.g., initialization, random seeds used for generating reference explanation scores in training dataset) in reference explanation scores. The Spearman's correlation between different runs is as strong as 0.77 (for more expensive explanation methods KernelSHAP-2000, it is only 0.52 across different runs); 4) In two downstream applications, our model is more faithful to the target model behavior compared with much more computationally expensive baselines.

2 Related Works

Post-Hoc Local Explanation Methods Post-hoc local explanations are proposed to understand the prediction process of neural models (Simonyan et al., 2014; Ribeiro et al., 2016; Lundberg & Lee, 2017; Shrikumar et al., 2017) . They work by assigning an explanation score to each feature in each individual instance ("local") to indicate its contribution to the model predictions. In this paper, we focus on studying KernelSHAP (Lundberg & Lee, 2017), an *additive feature attribution method* that approximate the Shapley Value (Shapley, 1953) for each attribution. KernelSHAP explains

[1]On Yelp-Polarity dataset and using A100 GPU, we compare with typical KernelSHAP explanation methods with 200 samples

the model output by first sampling perturbations on inputs, then using the output difference on perturbed inputs as contributions for perturbed features. The process repeats until the budget (the number of perturbation samples) is used out. As we will show in the experiments, when considering text classification involving long inputs, KernelSHAP requires a large number of model queries on different perturbation samples to obtain reliable estimation.

There are other methods in NLP for interpretability. For example, gradient-based methods (Simonyan et al., 2014; Li et al., 2016), which use the gradient w.r.t. a single point for saliency computations. Reference-based methods (Shrikumar et al., 2017; Sundararajan et al., 2017), that consider the model output difference between the original point and a reference point. Our amortized model is a model-based method to approximate the Shapley Values.

Shapley Value Estimation While Shapley Value is a theoretical way to calculate input attributions, there are limitations when applying it in practice given its prohibitively high cost for computation, especially when explaining prediction of long document in NLP. KernelSHAP works as an efficient way to approximate Shapley Value. Previous work on estimating Shapley Value mainly focus on accelerating the sampling process (Jethani et al., 2021; Covert & Lee, 2021) or removing redundant features (Aas et al., 2021; Covert et al., 2021). In this work, we propose to combat this challenge by training an amortized model for Shapley Value estimation.

Robustness of Local Explanation Methods Despite being widely adopted, there has been a long discussion on the actual quality of explanation methods. Recently, people find that explanation methods can assign substantially different attributions to similar inputs (Alvarez-Melis & Jaakkola, 2018; Ghorbani et al., 2019; Kindermans et al., 2019; Yeh et al., 2019; Slack et al., 2021; Yin et al., 2022), i.e., they are not robust enough, which adds to the more essential concerns on how faithful those explanation methods are (Doshi-Velez & Kim, 2017; Adebayo et al., 2018; Jacovi & Goldberg, 2020). In addition to previous work focusing on robustness against input perturbations for explanations, we demonstrate that even by just changing seeds, stochastic approximation of Shapley Values can give weakly-correlated interpretations among different runs, unless spending a huge amount of computational resources.

Amortized Explanation Methods Our method is similar to recent works on amortized explanation models including CXPlain (Schwab & Karlen, 2019) and FastSHAP (Jethani et al., 2021)), where they also aim to improve the computational efficiency of explanation methods. The key differences here are: 1) We do not make strong causal assumption between explanation scores and model behaviors; 2) We focus on text domains, where each feature is a discrete token and the same dimension of the inputs no longer have the same semantics (More details in Section 5).

3 Background

In this section, we will briefly review Shapley Values basics, with a special focus on its application to the text classification task. Usually, Shapley Values papers work on tabular data / well-structured data with a fixed set of features and each dimension has its specific semantics (e.g., ages, salaries).

Local explanation of black-box text classification models. In text classification tasks, inputs are usually sequences of discrete tokens $X = [w_1, w_2, \ldots, w_L]$. Here L is the sequence length for X and can vary across data. w_j is the j-th token for X. The classification model M_{CLF} takes the input X and predict the label as $\hat{y} = \arg\max_{y \in \mathcal{Y}} M_{\text{CLF}}(X)[y]$. Local explanation methods treat each data instance independently and compute an explanation score $\phi(j, y)$ for each token w_j, representing the contribution of w_j to a predicted label y. Usually, we care about the explanation scores when $y = \hat{y}$.

Shapley Values. Shapley Values are concepts from game theory originally developed to assign credits in cooperative games (Shapley, 1953; Strumbelj & Kononenko, 2010; Lundberg & Lee, 2017; Covert et al., 2021). Let $s \in \{0,1\}^L$ be a masking of the input and define $X_s \overset{\text{def}}{=} \{w_i\}_{i:s_i=1}$ as a *perturbed inputs* or *perturbation* under the masking s. In this paper, we follow the typical practice (Ye et al., 2021; Ye & Durrett, 2022; Yin et al., 2022) to replace the masked token with

[PAD]. Let $|s|$ represent the number of non-zero terms in s. Shapley Values $\phi_{\text{SV}}(i, y)$ for token w_i (Shapley, 1953) can be computed by:

$$\phi_{\text{SV}}(i, y) = \frac{1}{L} \sum_{s: s_i \neq 1} \binom{L-1}{|s|}^{-1} \left(M_{\text{CLF}} \left(X_s \cup \{w_i\} \right) [y] - M_{\text{CLF}} \left(X_s \right) [y] \right) \tag{1}$$

Intuitively, Shapley Values $\phi_{\text{SV}}(i, y)$ computes the marginal contributions to the output of the predictive model. The computational complexity for original Shapley Values is known to be NP-hard (Deng & Papadimitriou, 1994) because we need to sum over all $O(2^L)$ masks and is thus intractable to compute. In practice, we can only do approximation to efficiently compute Shapley Values. Shapley Values Sampling (SVS) (Castro et al., 2009; Strumbelj & Kononenko, 2010) is a widely-used Monte-Carlo estimation of Shapley Values:

$$\phi_{\text{SVS}}(i, y) = \frac{1}{m} \sum_{\sigma \in \Pi(L)} \left[M_{\text{CLF}} \left(X_{\mathbb{S}([\sigma]_{i-1} \cup \{i\})} \right) [y] - M_{\text{CLF}} \left(X_{\mathbb{S}([\sigma]_{i-1})} \right) [y] \right] \tag{2}$$

Here $[\sigma]_{i-1}$ represents the set of indices ranked lower than i in the ordering $\sigma \in \Pi(L)$. $\mathbb{S}([\sigma])$ maps the set of indices $[\sigma]$ to masking $s \in \{0,1\}^L$ as $s_i = \mathbf{1}[i \in [\sigma]]$. m is the number of *perturbation samples* used for computing SVS. SVS is an unbiased estimator and converges asymptotically at a rate of $O(\frac{1}{\sqrt{m}})$ according to Central Limit Theorem (Mitchell et al., 2022).

KernelSHAP. Although SVS have successfully reduced the exponential time complexity to polynomial, it still requires sampling permutations and needs to do sequential updates to traverse the features following sampled orderings and compute the explanation scores, which is an apparent efficiency bottleneck. On this side, Lundberg & Lee (2017) introduce a more efficient approximation named KernelSHAP, which allows doing updates in parallel and computing explanation scores for all tokens at once using linear regression. That is achieved by showing that computing Shapley Values is equivalent to solving the following optimization problem:

$$\phi_{\text{KernelSHAP}}(\cdot, y) \approx \underset{\phi(\cdot, y)}{\arg\min} \frac{1}{m} \sum_{s(i) \sim p(s)} \left[M_{\text{CLF}} \left(X_{s(i)} \right) [y] - \vec{s}(i)^T \phi(\cdot, y) \right]^2 \tag{3}$$

$$\text{s.t.} \quad \mathbf{1}^T \phi(\cdot, y) = M_{\text{CLF}} (X) [y] - M_{\text{CLF}} (\varnothing) [y] \tag{4}$$

where $\vec{s}$ is the vector corresponding to s mentioned above and $p(s)$ is the Shapley Kernel $p(s) = \frac{L-1}{\binom{L}{|s|}|s|(L-|s|)}$. m is again the number of perturbation samples. Equation (4) is often called as "sufficiency constraint". We will use "SVS-m" and "KernelSHAP-m" in the rest of the paper to indicate the sample size when we use SVS and KernelSHAP, respectively. In practice, the set of perturbation samples will usually be determined by the random seed.

Here we can see that, the larger the number of perturbation samples is, the more model evaluations are required for a single instance, which can be quite computationally expensive as we usually use a multi-layer transformer as the explained model in text classification nowadays. Therefore the main performance bottleneck is the number of model evaluations.

4 Stability of Local Explanation

One of the most common applications of Shapley Values is feature selection, which selects the most important features by following the ordering of Shapley Values. Usually, people use KernelSHAP with an affordable number of perturbation samples (the typical numbers of perturbation samples used are around 25, 200, 2000) to do this task. However, as we see in Figure 1, such ranking can be quite sensitive to random seeds when we use the stochastic estimation of Shapley Values. In this section, we systematically investigate this stability issue. We demonstrate stochastic approximation of Shapley Values can be unstable when applied in classification tasks with long text under common settings. In particular, when ranking the tokens in input based on the explanation scores, Spearman's correlation between rankings from different runs is low.

Setting	Spearman	Top-5 Inter.	Top-10 Inter.	MSE	Running Time
SVS-25	$0.84(\pm 0.00)$	$3.41(\pm 0.00)$	$7.02(\pm 0.00)$	$0.01(\pm 0.00)$	183.72s/it
KernelSHAP-25	$0.04(\pm 0.00)$	$0.43(\pm 0.01)$	$1.45(\pm 0.01)$	$0.00(\pm 0.00)$	1.92s/it
KernelSHAP-200	$0.16(\pm 0.00)$	$1.09(\pm 0.01)$	$2.47(\pm 0.00)$	$0.82(\pm 0.29)$	3.47s/it
KernelSHAP-2000	$0.37(\pm 0.00)$	$2.45(\pm 0.01)$	$4.38(\pm 0.05)$	$0.03(\pm 0.00)$	33.40s/it
KernelSHAP-8000	$0.63(\pm 0.00)$	$3.73(\pm 0.02)$	$6.93(\pm 0.01)$	$0.01(\pm 0.00)$	123.29s/it

Table 1: Ranking stability experiments on the Yelp-Polarity dataset. Each local explanation setting is evaluated across 5 runs with different random seeds. “Top-K Inter.” denotes top-K intersection. All values in this table are absolute values. Here we can see a clear trade-off between stability and computation cost.

Setting	Spearman	Top-5 Inter.	Top-10 Inter.	MSE	Running Time
SVS-25	$0.75(\pm 0.00)$	$3.54(\pm 0.02)$	$7.46(\pm 0.02)$	$0.02(\pm 0.00)$	128.07s/it
KernelSHAP-25	$0.06(\pm 0.00)$	$0.97(\pm 0.01)$	$3.41(\pm 0.03)$	$0.01(\pm 0.00)$	0.33s/it
KernelSHAP-200	$0.24(\pm 0.00)$	$1.79(\pm 0.01)$	$4.37(\pm 0.03)$	$0.07(\pm 0.00)$	2.04s/it
KernelSHAP-2000	$0.52(\pm 0.00)$	$3.19(\pm 0.00)$	$6.09(\pm 0.00)$	$0.03(\pm 0.00)$	20.39s/it
KernelSHAP-8000	$0.76(\pm 0.00)$	$4.08(\pm 0.02)$	$7.74(\pm 0.02)$	$0.01(\pm 0.00)$	89.48s/it

Table 2: Ranking stability experiments on the MNLI dataset.

Measuring ranking stability. Given explanation scores produced by different random seeds using a stochastic Shapley Values estimator, we want to measure the difference between these scores. Further, we are interested in the ranking difference as this is what matters in applications. To measure the ranking stability among different runs, we mainly use Spearman’s correlation between rankings. As Spearman’s correlation is calculated between two different ranking results, we run with multiple random seeds and compute Spearman’s correlation between any two of them and use the averaged Spearman’s correlation to measure the ranking stability. Besides, we also follow Ghorbani et al. (2019) to report Top-K intersections between two rankings, since in many applications only the top features are of explanatory interest. It computes the size of the intersection of Top-K features ranked by two different runs of the same Shapley Values estimator.

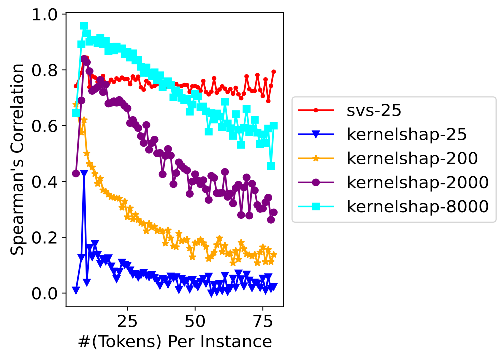
Setup. We conduct our experiments on the Yelp-Polarity dataset (Zhang et al., 2015) and MNLI dataset (Williams et al., 2018). Yelp-Polarity is a binary sentiment classification task and MNLI is a three-way textual entailment classification task. Due to computational resource limitations, we only conduct experiments on 500 random samples (we refer to these datasets as “Stabilty Evaluation Sets” subsequently) with 5 different random seeds on the validation set of both Yelp-Polarity and MNLI[2]. We use the publicly available fine-tuned BERT-base-uncased checkpoints[3] (Morris et al., 2020) as the target models to interpret and use the implementation of Captum (Kokhlikyan et al., 2020) to compute the explanation scores for both KernelSHAP and SVS. For each explanation method, we test with widely-used or recommended numbers of perturbation samples[4] used to compute the explanation scores for every instance. For Top-K intersections, we use $K = 5$ and $K = 10$.

Trade-off between stability and computation cost. The ranking stability experiment results are listed in Table 1 and Table 2 for Yelp-Polarity and MNLI datasets, respectively. We observe that when we follow the common setting of interpretation methods to generate 25 to 200 perturbation samples, the stability of ranking and Top-K intersection is low. By drawing more masked samples for each instance, the interpretation becomes more stable. However, the computational cost and running time explode at the same time, especially when the size of the input text is large. To reduce the sensitivity to an acceptable level (i.e., the Spearman’s correlation between two different runs is

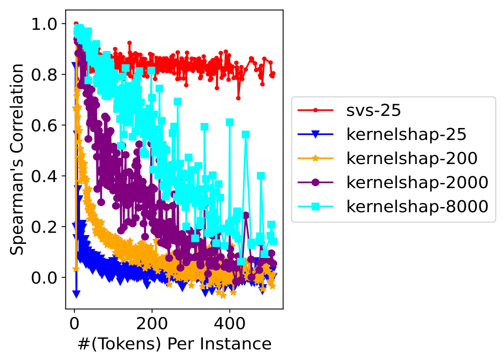
[2]We already take way more than 2,000 hours on a single A100 GPU for all experiments in this section.

[3]Yelp-Polarity: `https://huggingface.co/textattack/bert-base-uncased-yelp-polarity`
MNLI: `https://huggingface.co/textattack/bert-base-uncased-MNLI`

[4]For SVS, the recommended number of perturbation samples is 25 in Captum. For KernelSHAP, to our best knowledge, the typical numbers of perturbation samples used in previous works are $25, 200, 2000$. We also include KernelSHAP-8000 to see when given much longer running time, how stable KernelSHAP can be.



(a) MNLI



(b) Yelp-Polarity

Figure 2: Ranking stability over different input lengths. We observe that longer input suffers more from instability.

higher than 0.60, which indicates strong correlation), we usually need to spend thousands of model evaluation.

Low MSE does not imply stability. Mean Squared Error (MSE) is commonly used to evaluate the distance between two explanation scores. We follow this trend and also list MSE between two explanation scores. We observe that MSE weakly correlates with ranking stability. Even when the difference of MSE for different settings is as low as 0.01, the correlation between rankings produced by explanations can still be low.

Longer input suffers more from instability. We also plot the Spearman's correlation decomposed at different input length in Figure 2. Here, we observe a clear trend that the ranking stability degrades significantly even at an input length of 20 tokens. The general trend is that when the longer input length, the worse the ranking stability. The same trend holds across datasets.

Discussion: why Shapley Values computation is instable in text domain? One of the most prominent characteristics for text domain is that, models often requires to perform high-level correlation analysis among input tokens (e.g, n-gram statistics), and missing even one token can drastically change the semantics of the input (e.g., the sentiment words, adjectives and the proper nouns). When the input length grows, the number of n-grams will grow fast. As shown in Section 3, the probability of certain n-gram get sampled is drastically reduced as each n-gram will be sampled with equivalent probability. Therefore, the observed model output will have large variance as certain n-grams may not get sampled. Kwon & Zou (2022) presented a related theoretical analysis about why the uniform sampling setting in Shapley Values computation can lead to suboptimal attribution results.

5 Amortized Inference for Shapley Values

Existing approaches for improving the efficiency of model interpretation methods mainly focus on enhancing sampling algorithms to use more representative perturbation samples thus reducing the number of model evaluations (Covert et al., 2021; Mitchell et al., 2022). As an orthogonal direction, we propose to train an amortized model to predict the local explanation scores given the inputs without any queries to the model. Such training is achieved by fitting pre-collected reliable explanation scores.

We build an amortized explanation model for text classification in a 2-stage way. In the first stage, we collect reliable explanation scores as the reference scores for training using the existing Shapley Values estimator. As shown in Section 4, SVS-25 is the most stable Shapley Values estimator and we use it to obtain reference scores. In the second stage, we train a BERT-based amortized model to take the text input and directly output the explanation scores by fitting the outputs to reference scores with MSE loss.

Specifically, given input tokens X, we will use a pretrained language model M_{LM} to encode words into d-dim embeddings $\vec{e} = M_{\text{LM}}(X) = [\vec{e}_1, \ldots, \vec{e}_{L(X)}] \in \mathbb{R}^{L(X) \times d}$. Then we use an linear layer to transform each $\vec{e}_i$ to the predicted explanation score $\phi_{AM}(i, \hat{y}_i) = W\vec{e}_i + b$. To train the model, we use MSE loss to fit $\phi_{AM}(i, \hat{y})$ to the pre-computed reference scores $\phi(i, \hat{y})$ over the training set $\mathbb{X}_{\text{Train}}$. This is an amortized model in the sense that there is no individual sampling and model queries for each test example X as in SVS and KernelSHAP. When a new sample comes in, all it needs are just the input tokens.

Better accuracy-efficiency trade-off via meta learning As the explanation scores are highly local, it is possible that the aforementioned simple L2 regression will only capture the globally shared patterns in explanation scores across instance, but still cannot generalize well on every instance. This naturally gives rise to a meta-learning based method: we can use the explanation scores generated by our models as good initialization for the Shapley Values for each instance, and then perform SVS update as in Equation (2) to achieve better local adaption. Actually, if we treat explaining each instance as a "task" and the SVS update as gradient descent steps based on L2 loss on perturbation samples, then this meta-learning methods is identical to the famous MAML framework in meta learning (Finn et al., 2017). This meta learning method inevitably introduce more computational overhead when inferring explanation scores for each instance, but it also provides more flexible control of accuracy-efficiency trade-off as we previously achieved using different number of perturbation instances in SVS and KernelSHAP. We present a more detailed analysis of this meta learning idea in Appendix B.

Other objective function We use L2 regression here to fit the amortized model for simplicity, but that is not the only modeling choice. We also experiment with using sorting network (Petersen et al., 2022), which is a continuous relaxation of sorting and can allow the model to learn sorting. However, we empirically find it does now work well as it is pretty unstable to train over discrete text data and the best performance observed cannot even beat FastSHAP baseline on Spearman's correlation. We leave further exploration for future works.

6 Experiments

In this section, we present experiments to demonstrate the properties of the proposed approach in terms of approximation precision, learning efficiency and sensitivity against reference scores. We conduct experiments on the validation set of Yelp-Polarity and MNLI datasets. To generate reference explanation scores, we leverage the Thermostat (Feldhus et al., 2021) dataset, which contains 9,815 pre-computed explanation scores of SVS-25 on MNLI. We also take two weeks to compute explanation scores of SVS-25 for 25,000 instances on Yelp-Polarity. For both datasets, we split them into 9:1:1 for training, validation, and test sets. The hyperparameters of amortized models are tuned on the validation set. We use Adam (Kingma & Ba, 2015) optimizer with a learning rate of 5e-5, train the model for at most 10 epochs and do early stopping to select best model checkpoints. We use BERT-base-uncased (Devlin et al., 2019) for M_{LM}.

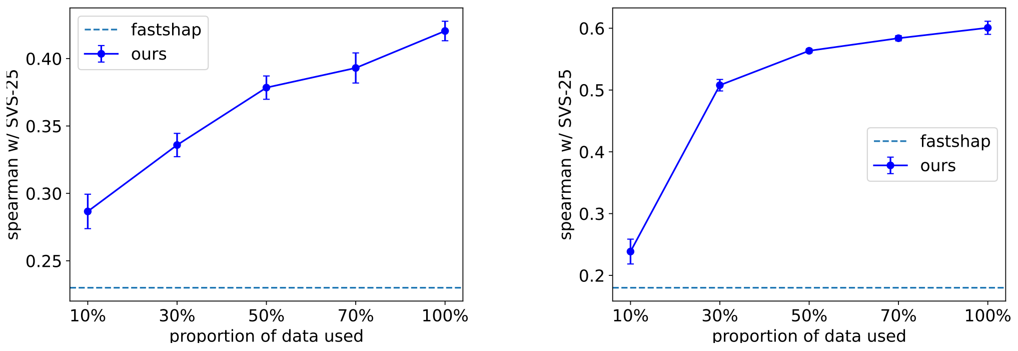
Baseline: FastSHAP We adapt FastSHAP (Jethani et al., 2021) to explain the text classifier and compare it with our approach. However, we find that it is non-trivial to adapt FastSHAP to the text domain. The original implementation can only be applied with tabular data or image data. As pre-trained language models occupy a large amount of GPU memory, we can only use a small batch size with limited perturbed samples (i.e., 32 perturbed samples per instance). This is equivalent to approximate KernelSHAP-32 and the corresponding reference explanation scores computed by FastSHAP are unstable. We also find it pretty hard and time-consuming to optimize FastSHAP. More details can be found in Appendix A.

6.1 Approximation and Computational Efficiency

To examine how well our model fits to the pre-computed Shapley Values (SVS-25), we compute both Spearman's correlation and MSE over the test set. As it is intractable to compute exact Shapley Values for ground truth, we use the SVS-25 as a proxy instead. We also include different settings for KernelSHAP results over the same test set. KernelSHAP is also an approximation to permutation-

Method	MNLI		Yelp-Polarity	
	Spearman	MSE	Spearman	MSE
SVS-25	0.75	1.90e-2	0.84	6.64e-3
KernelSHAP-25	0.17	9.95e-2	0.12	4.34e-2
KernelSHAP-200	0.35	7.73e-2	0.24	5.77e-2
KernelSHAP-2000	0.60	2.54e-2	0.51	1.86e-2
KernelSHAP-8000	**0.74**	**1.25e-2**	**0.70**	**6.25e-3**
FastSHAP	0.23	1.90e-1	0.18	7.91e-3
Our Amortized Model	**0.42**	**9.59e-3**	**0.61**	**4.46e-6**

Table 3: Approximation results for the Shapley explanation methods on MNLI and Yelp-Polarity datasets. Bold-faced numbers are the best in each column. Results are averaged over 5 runs. Spearman's correlation and MSE are computed against SVS-25, a proxy to exact Shapley Values. We can see our amortized model can achieve better approximation when compared to KernelSHAP-200 and our baseline FastSHAP, but not as good as much more time-consuming methods KernelSHAP-2000 and KernelSHAP-8000. We list SVS-25 results here as an upper bound.



(a) MNLI (b) Yelp-Polarity

Figure 3: Learning curves for the amortized model over MNLI datasets (left) and Yelp-Polarity datasets (right). The Spearman's correlations in this figure is computed against SVS-25. We can see our amortized model can learn more efficiently even if there is only 10% data used for training.

based Shapley Values computation (Lundberg & Lee, 2017). The approximation results are shown in Table 3.

First, we find that despite the simplicity of our amortized model, the proposed amortized models achieve a high correlation with the reference scores ($0.61 > 0.60$) on Yelp-Polarity given enough data. The correlation between outputs from the amortized models and references is moderate ($0.42 > 0.40$) on MNLI when data size is limited. During inference time, our amortized models can output explanation scores for each instance within 50 milliseconds, which is about 40-60 times faster than KernelSHAP-200 and 400-600 times faster than KernelSHAP-2000 on Yelp-Polarity and MNLI. Although the approximation results are not as good as other settings, such as KernelSHAP-2000 or KernelSHAP-8000, our amortized model achieves reasonably good results with far less computation cost.

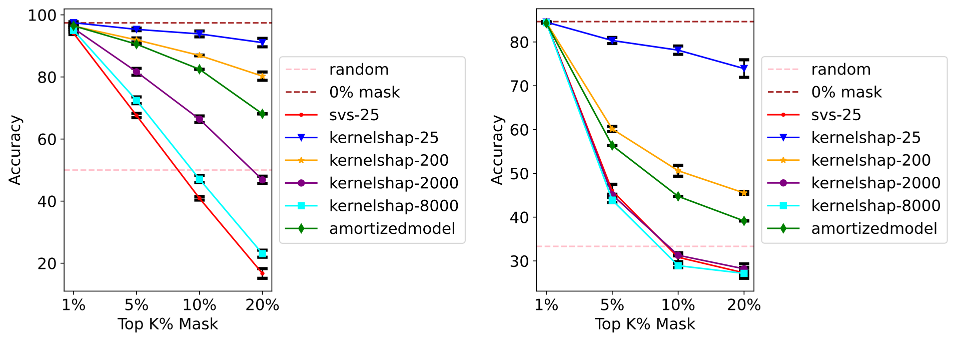
We also find that the amortized model achieves the best MSE score among all approximation methods. Note that the two metrics, Spearman's correlation and MSE, do not convey the same information. MSE measures how well the reference explanation scores are fitted while Spearman's correlation reflects how well the ranking information is learned. We advocate for reporting both metrics.

6.2 Inference and Learning Efficiency

Regarding the training, we need to pre-compute the explanation scores on a set of data so that we can use them to train the models. This one-time step can be time-consuming as we need to run the

Training Data Proportion	Spearman (MNLI)	Spearman (Yelp-Polarity)
10%	0.45	0.40
30%	0.57	0.65
50%	0.65	0.71
70%	0.65	0.72
100%	0.77	0.76

Table 4: Training time sensitivity study. To evaluate how much the amortized model will be influenced by randomness during training, we sample training data 5 times with different random seeds and then compute the averaged Spearman's correlation among all pairs of runs. The standard deviation is less than 1e-2. We can see our amortized model is stable against training time randomness.



(a) Yelp-Polarity (b) MNLI

Figure 4: Feature selection based on interpretations on Yelp-Polarity and MNLI dataset. The faster the curve drops, the explanation scores are more faithful to the model's actual behavior. We can see our amortized model is more faithful to the target model compared with KernelSHAP-200, but cannot lead to as a significant performance drop as other time-consuming methods.

original explanation approaches. However, as the learning curve shown in Figure 3, we observe that the model achieves good performance with about 5,000 instances.

During inference time, the amortized model is efficient and stable as it does not need to estimate the explanation scores from sampling.

6.3 Sensitivity Analysis

There is no randomness in generating explanation scores for new data samples when using the amortized model. However, there is still some randomness in the training process, including the random initialization of the output layer and the randomness in the model update. Therefore, similar to Sec. 6.1, we study the sensitivity of amortized model to the random seeds. Table 4 shows the results with different sample sizes. We observe that: 1) when using the same data (100%), random initialization does not affect the outputs of amortized models – the correlation between different runs is high (i.e., 0.77 on MNLI and 0.76 on Yelp-Polarity). 2) With more training samples, the model is more stable.

6.4 Downstream Applications of Amortized Models

Feature Selection The first case study is feature selection, which is a straightforward application of local explanation scores. The goal is to find decision-critical features via removing input features gradually according to the sorting rank given by the explanation methods. The more faithful the explanation method is to the target model, the more important tokens should be ranked higher and should result in more performance drops when the same number of features are masked. We gradually mask Top $\alpha\%$ tokens ($\alpha = 1\%, 5\%, 10\%, 20\%$) and compute the accuracy over corrupted results using the stability evaluation sets for MNLI and Yelp-Polarity datasets as mentioned in Section 4.

As the results shown in Figure 4, the amortized model is more faithful than KernelSHAP-200 but underperforms KernelSHAP-2000, KernelSHAP-8000 and SVS-25. However, amortized model is more efficient than these methods.

Explanation as Domain Calibrator Following recent findings that good explanations should be informative enough to help users to predict model behavior (Doshi-Velez & Kim, 2017; Chandrasekaran et al., 2018; Hase & Bansal, 2020; Ye et al., 2021). Ye & Durrett (2022) propose to combine the local explanation with pre-defined feature templates (e.g., aggregating explanation scores for overlapping words / POS Tags in NLI as features) to calibrate an existing model to new domains. The rationale behind this is that, if the local explanation truly connects to human-understandable model behavior, then following the same way how human transfer knowledge to new domains, the explanations guided by human heuristics (in the form of feature templates) should help calibrate the model to new domains. Inspired by this, we conduct a study using the same calibrator architecture but plugging in different local explanation scores.

For calibration experiments, we follow Ye & Durrett (2022) to calibrate a fine-tuned MNLI model[5] to MRPC. The experiment results are shown in Table 5. In the table, "BOW" means the baseline that uses constant explanation scores when building the features for the calibration model. Compared with the explanation provided by KernelSHAP-2000, the explanation given by the amortized model achieves better accuracy, suggesting that the amortized model learns robust explanation scores that can be generalized to out-of-domain data in downstream applications.

Model	Acc
BOW	67.3
ShapCal (KernelSHAP-2000)	67.4
ShapCal (Amortized)	68.0

Table 5: Calibration Experiments for Amortized Models. We can see that the explanation scores can help the calibration model achieves better accuracy on out-of-domain data than KernelSHAP-2000.

7 Conclusion

In this paper, we empirically demonstrate that it is challenging to obtain stable explanation scores on long text inputs. Inspired by the fact that different instances can share similarly important features, we propose to efficiently estimate the explanation scores through an amortized model trained to fit pre-collected reference explanation scores.

In the future, we plan to explore model architecture and training loss for developing effective amortized models. In particular, we will incorporate sorting-based loss to maintain the ranking order of features. Besides, we will investigate the transferability of the amortized model across different domains and different languages.

References

Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502, 2021.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9525–9536, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html`.

[5] `https://huggingface.co/textattack/bert-base-uncased-MNLI`

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*, 2018.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

Arjun Chandrasekaran, Viraj Prabhu, Deshraj Yadav, Prithvijit Chattopadhyay, and Devi Parikh. Do explanations make VQA models more predictable to a human? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1036–1042, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1128. URL https://aclanthology.org/D18-1128.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation using linear regression. In Arindam Banerjee and Kenji Fukumizu (eds.), *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3457–3465. PMLR, 2021. URL http://proceedings.mlr.press/v130/covert21a.html.

Ian Covert, Scott M Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22:209–1, 2021.

Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017. URL https://arxiv.org/abs/1702.08608.

Nils Feldhus, Robert Schwarzenberg, and Sebastian Moller. Thermostat: A large collection of nlp model explanations and analysis tools. In *EMNLP*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

Amirata Ghorbani, Abubakar Abid, and James Y. Zou. Interpretation of neural networks is fragile. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 3681–3688. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013681. URL https://doi.org/10.1609/aaai.v33i01.33013681.

Peter Hase and Mohit Bansal. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5540–5552, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.491. URL https://aclanthology.org/2020.acl-main.491.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4198–4205, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.386. URL https://aclanthology.org/2020.acl-main.386.

Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. Fastshap: Real-time shapley value estimation. In *ICLR*, 2021.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 267–280. Springer, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020. URL https://arxiv.org/abs/2009.07896.

Yongchan Kwon and James Zou. Weightedshap: analyzing and improving shapley based feature attributions. *NeurIPS*, 2022.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 681–691, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1082. URL https://aclanthology.org/N16-1082.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4765–4774, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 119–126, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.16. URL https://aclanthology.org/2020.emnlp-demos.16.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Monotonic differentiable sorting networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=IcUWShptD7d.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ”why should I trust you?”: Explaining the predictions of any classifier. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1135–1144. ACM, 2016. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

Patrick Schwab and Walter Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 10220–10230, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3ab6be46e1d6b21d59a3c3a0b9d0f6ef-Abstract.html.

Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, Princeton, 1953.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3145–3153. PMLR, 2017. URL `http://proceedings.mlr.press/v70/shrikumar17a.html`.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

Dylan Slack, Anna Hilgard, Sameer Singh, and Himabindu Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. *NeurIPS*, 34:9391–9404, 2021.

Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328. PMLR, 2017. URL `http://proceedings.mlr.press/v70/sundararajan17a.html`.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307, 2011. doi: 10.1162/COLI_a_00049. URL `https://aclanthology.org/J11-2001`.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL `https://aclanthology.org/N18-1101`.

Xi Ye and Greg Durrett. Can explanations be useful for calibrating black box models? In *ACL*, 2022.

Xi Ye, Rohan Nair, and Greg Durrett. Connecting attributions and qa model behavior on realistic counterfactuals. In *EMNLP*, 2021.

Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I. Inouye, and Pradeep Ravikumar. On the (in)fidelity and sensitivity of explanations. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 10965–10976, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/a7471fdc77b3435276507cc8f2dc2569-Abstract.html`.

Fan Yin, Zhouxing Shi, Cho-Jui Hsieh, and Kai-Wei Chang. On the sensitivity and stability of model interpretations in nlp. In *ACL*, pp. 2631–2647, 2022.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 649–657, 2015. URL `https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html`.

Method	MNLI Spearman	Yelp-Polarity Spearman
SVS-1	0.32	0.43
SVS-2	0.41	0.52
SVS-3	0.47	0.60
SVS-5	0.55	0.69
SVS-25	**0.75**	**0.84**
Our Amortized Model	0.42	0.61
Our Amortized Model (Adapt-1)	0.44	0.60*
Our Amortized Model (Adapt-2)	0.47	0.64
Our Amortized Model (Adapt-3)	0.53	0.69
Our Amortized Model (Adapt-5)	**0.57**	**0.71**

Table 6: Approximation results for the Shapley explanation methods on MNLI and Yelp-Polarity datasets. Bold-faced numbers are the best in each column. Results are averaged over 5 runs. Spearman's correlation and MSE are computed against SVS-25, a proxy to exact Shapley Values. Adapt-m means here how many sampled ordering σs we used here to do local adaption (m in Algorithm 1). We can see 1) by doing local adaption, we can further improve the approximation results using our amortized model, 2) using our amortized model as initialization, we can improve the sample efficiency of SVS significantly (by comparing the performance of SVS-X and Our Amortized Model (Adapt-X). $*$: not significant here compared with not using local adaption.

A Adaption for FastSHAP Baseline

As we mentioned in Section 6, we build our amortized models upon a pre-trained encoder BERT (Devlin et al., 2019). However, using the pre-trained encoder significantly increases the memory footprint when running FastSHAP. In particular, we have to host two language models on GPUs, one for the amortized model and the other one for the target model. Therefore, we can only adopt the batch size equals to 1 and 32 perturbation samples per instance. Following the proof in FastSHAP, this is equivalent to teaching the amortized model to approximate KernelSHAP-32, which is an unreliable interpretation method (See Section 6.3).

In experiments, we find that the optimization of FastSHAP is unstable. After an extensive hyperparameter search, we set the learning rate to 1e-6 and increased the number of epochs to 30. However, this requires us to train the model on a single A100 GPU for 3 days to wait for FastSHAP to converge.

Algorithm 1 Local Adaption

```
Require: $m$: the desired number of local adaption perturbation samples, $M_{\mathrm{AM}}$: the trained amortized explanation model, $X$: the target data instance that has length $L$, $\hat{y}$: the predicted label, $M_{\mathrm{CLF}}$: the target model
  $\phi \leftarrow M_{\mathrm{AM}}(X)$
  for $j = 1$ to $m$ do
      choose a random order $\sigma$ from permutation $\Pi(L)$
      $\phi \leftarrow \phi + \sum_i \left[M_{\mathrm{CLF}}\left(X_{\mathbb{S}([\sigma]_{i-1} \cup \{i\})}\right)[\hat{y}] - M_{\mathrm{CLF}}\left(X_{\mathbb{S}([\sigma]_{i-1})}\right)[\hat{y}]\right]$
  end for
  $\phi \leftarrow \frac{\phi}{m}$
```

B Amortized Inference via Meta Learning

As the Shapley Values are originally for each instance and thus it is possible that the amortized model may not achieve a good fit for every instance. Thus, a local adaption can be helpful to strengthen our model. Specifically, our meta learning framework is described in Algorithm 1. Note that here if we can recover the original SVS computation (Strumbelj & Kononenko, 2010) by replacing $\phi \leftarrow M_{\mathrm{AM}}(X)$ to be $\phi \leftarrow 0$. M_{AM} is trained using the method described in Section 5.

The experiment results are shown in Table 6. Here we can see that: 1) By comparing the results with using local adaption and not using, except the case that only using one sampled ordering to do update (maybe the update is not sufficient), we consistently observe a significant improvement in Spearman's correlation across datasets. 2) By comparing using our amortized model output as initialization and not using, we also see a consistent Spearman's correlation improvement over original SVS, indicating that our amortized model can help achieve better sample-efficiency by providing a good initialization.