Support Vector Generation: Kernelizing Zero-Shot Classifiers from Pre-Trained Language Models

Shohei Ohsawa
shohei.ohsawa@iyp.co.jp

Abstract

We introduce Support Vector Generation (SVG), a kernel-based framework that converts a frozen language model into an interpretable, training-free classifier for zero- and few-shot learning. SVG operates by combining Metropolis-Hastings sampling with support vector machine optimization in the reproducing kernel Hilbert space (RKHS) induced by the language model's embedding. Each classification decision is based on a weighted combination of at most 32 natural-language sentences, which serve as explicit support vectors and provide faithful rationales. Our theoretical analysis proves that SVG minimizes the empirical hinge loss over the span of the supports and admits a generalization bound independent of the language model size. Experiments on the GLUE benchmark show that SVG matches or surpasses prompting-based zero-shot baselines in accuracy across multiple tasks—without any fine-tuning or GPU acceleration. Notably, our CPU-only implementation completes training in under three minutes per task, and maintains competitive inference speed. These results suggest that SVG offers a viable path toward efficient, interpretable NLP systems under compute constraints.

1 Introduction

Large-scale pre-trained language models (PLMs) [3, 6] achieve impressive zero-shot accuracy on many natural-language understanding tasks. Despite this empirical success, two practical obstacles hinder their deployment at scale. First, the decision process is *opaque*: predictions emerge from billions of latent parameters, and token-level saliency methods provide only indirect, often disputed explanations [1, 2]. Second, inference is *computationally costly*: autoregressive decoding grows linearly with output length and typically requires GPU-class hardware, whereas many edge or privacy-sensitive applications must run on CPUs.

Fig. 1 contextualises these obstacles. Panel (a) depicts the canonical pipeline in which a labelled corpus is used to train a parametric classifier; panel (b) reinterprets this as a teacher-student communication game. Panel (c) situates our contribution: a frozen PLM (*teacher*) communicates with a non-parametric kernel machine (*student*) over the Internet by exchanging natural-language prompts, thereby obviating gradient-based training while retaining interpretability[1].

Let $p_\theta(x)$ denote the probability a frozen PLM with parameters θ assigns to a sentence x. The negative log-likelihood induces a positive-definite *language kernel*

$$k_\theta(x, x') \;=\; \exp\!\big[-\log p_\theta(x) \;-\; \log p_\theta(x')\big],$$

embedding all sentences in a reproducing-kernel Hilbert space (RKHS) whose dimensionality is *independent* of $|\theta|$. Within this RKHS the optimal large-margin classifier is representable as a finite

[1] I have conducted a series of studies on multi-agent communication in distributed environments [17, 25, 26, 27, 28], and the present paper can be regarded as one of them.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

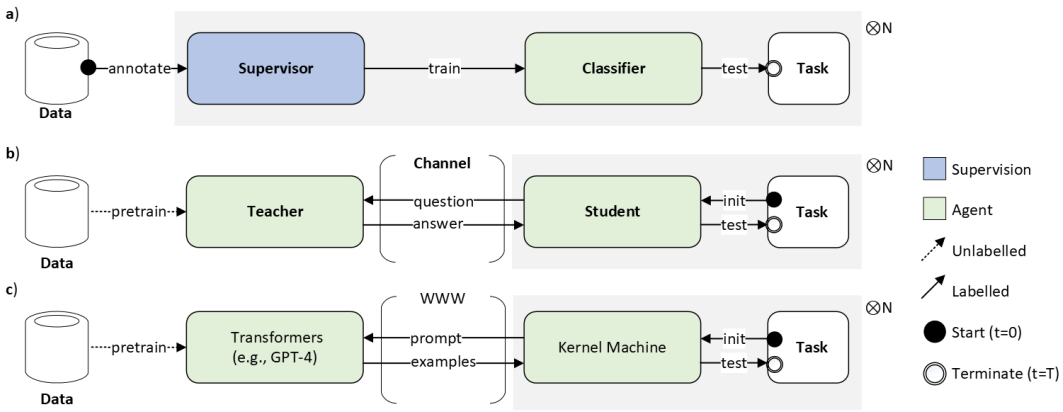

Figure 1: The communication model of our approach. A teacher tells students possibly taking on muliple tasks by addressing difference questions The different agents communicates over channels such as Internet and local networks. **a**) Typical machine learning **b**) Teacher-student model **c**) Support vector generation (ours)

weighted sum of kernel evaluations, suggesting that explicit, compact, and interpretable classifiers can be extracted from PLMs without fine-tuning.

We introduce *SVG*, a training-free, two-stage procedure (Fig. 1c). Stage 1 employs a Metropolis-Hastings sampler queried against the PLM to synthesise sentences near the decision surface. Stage 2 solves the dual support-vector-machine problem restricted to these candidates, retaining only those with non-zero dual weights; the surviving sentences form an explicit *support set*. At test time classification reduces to a weighted vote over at most 32 supports, and the dual coefficients provide exact influence scores—yielding faithful sentence-level rationales.

Contributions. (i) We formalise the PLM-induced language kernel and prove that SVG minimises empirical hinge loss within the span of its supports, obtaining a generalisation bound that scales with the support count rather than with $|\theta|$. (ii) We devise an efficient CPU implementation whose entire training phase finishes in under three minutes per GLUE task without any gradient updates. (iii) On seven zero-shot tasks from the GLUE benchmark, SVG matches or surpasses strong prompting baselines while providing exact post-hoc explanations. (iv) We release anonymised code and scripts to facilitate independent verification.

Collectively, these results demonstrate that interpretable and resource-efficient classifiers can be distilled from frozen language models, paving the way for PLM-level accuracy in compute-constrained settings.

2 Related Work

Before the deep learning era, kernels (*i.e.* "similarity") were widely employed to measure the distance between a pair of inputs, especially sequences of information such as web documents or protein sequences, which can be of variable length. The representative one is the cosine similarity between TF-IDF vectors, which give good results for information retrieval [21], with a probabilistic interpretation given in [12]. String kernels [15, 33], which compare the number of substrings in common, have been used for amino acid sequences. The string kernel is known as a scalable Mercer's kernel, which can be computed in linear time for a length of string using suffix trees [18, 34, 38]. One of the string kernels which *consider* strings of a fixed length k is known as the k-spectrum kernel, and has been used to classify proteins into SCOP superfamilies [18]. The string kernel has been generalized to compare trees [9], which is useful for parse trees and evolutionary trees. In computer vision, a pyramid match kernel [14] is used to compare two images of feature vectors obtained from SIFT [20]. One of the drawbacks of these kernels is that they cannot capture the recurring structure of strings, because they merely assume the document as a bag-of-words (or bag-of-features), and only care about the frequency of the words.

The paradox between the number of parameters of a neural network and its generalisation performance has been one of the unsolved problems in the field of deep learning: traditional learning theories state that if the parameters of a model exceed the number of samples, the model is over-trained and cannot adapt sufficiently to unknown samples. One hypothesis for this is that neural networks acquire the ability to "interpolate" between any two samples after having fully memorised all samples, which also has experimental evidence in the form of the double descent phenomena [24]. Nevertheless, another problem with this paradox is the derivation of explicit optimal solutions, *i.e.* how to regularise a model with degrees of freedom that go beyond the train data. [41] map the MNIST samples into a dual space and optimise the hinge loss with a "kernel trick" to obtain a regularised explicit representation consisting of the weights of each sample and the Gram matrix, which is a similarity representation between each sample. This solution also showed higher performance than the expected value of the solution optimised simply by the stochastic gradient descent, suggesting a strong relationship between neural networks and kernels.

Self-attention [37] is a powerful building block that has enabled the development of large language models (LLMs) such as BERTs and GPTs, which can quickly capture the in-context relationships between tokens in a text sample. LLMs have been used for various tasks including text generation, question answering, and dialogue generation [6, 30, 31, 42]. Recent research has focused on controlled text generation, generating text that adheres to a set of constraints while being fluent and relevant to the given context [19, 44]. Zero-shot learning, a more challenging task, uses transfer learning or data augmentation to overcome data scarcity and computational constraints [13, 22]. Despite its computational advantage, self-attention requires the use of GPUs to accommodate the large number of parameters ($|\theta| \sim 10^{10}$) needed to memorize all examples. This can be computationally expensive and may not be feasible for certain applications.

3 Language Kernels

Suppose we are a data scientist in a company who has been assigned a task of sentiment analysis, checking whether the given text represents a negative or positive review for each film manufactured by the client. In a zero-shot learning scenario, instead of requesting disclosure of training data from the client, we can derive an inner product $\phi(x)^T\phi(w_y)$ between an input $x \in \mathcal{X}$ and a label $w_y \in \mathcal{X}$ for $y \in \{\pm 1\}$, where $\mathcal{X}$ is a set of strings, and ϕ represents text embeddings, and we can directly insert desired labels such as $\phi(x)^T[\phi(\text{"positive"}) - \phi(\text{"negative"})]$. Though it seems too straightforward, one can find out that the accuracy for SST-2 is 0.83[2]. The inner product $k(x, w_y) = \phi(x)^T\phi(w_y)$ measuring the text similarity of the sentence pair $k : \mathcal{X}^2 \to \mathbb{R}$ is called a *kernel*.

Here we use a simple string of "positive"/"negative" as an example, but there are countless such task description expressions, each with slightly different vector representations (*e.g.*, "good"/"bad"). Therefore, we generalize the above method into a set of n synonyms per label $\{x_1, \ldots, x_{2n}\} = \{w_y^1, \ldots, w_y^n\}_{y\in\{\pm 1\}}$ with a decision function $f_\alpha : \mathcal{X} \to \mathbb{R}$ as follows:

$$f_\alpha(x) = \sum_{i=1}^{2n} \alpha_i k(x, x_i) y_i, \tag{1}$$

where $\alpha_i \geq 0$ represents the sample-wise attention of each description.

The problem is how to find the solution of the system α with $2n$ degrees of freedom without auxiliary oracles such as real labeled samples. Surprisingly, the optimal solution is found by solving the dual objective of SVMs [10] as follows:

$$\max_\alpha J(\alpha) = \sum_{i=1}^{2n} \alpha_i - \frac{1}{2}\sum_{i=1}^{2n}\sum_{j=1}^{2n} \alpha_i\alpha_j k(x_i, x_j) y_i y_j \quad \text{s.t.} \quad \sum_{i=1}^{2n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \tag{2}$$

where $C > 0$ is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error. The number of support vectors $n_{\text{SV}} < 2n$ is bounded by the VC-dimension [4, 36], which measures the complexity of a binary classification problem with the maximum number of samples the classifier can shatter. For any positive definite function $\kappa(\cdot, \cdot) : \mathbb{R}^{2d} \to \mathbb{R}$ and text embedding $\phi(\cdot) : \mathcal{X} \to \mathbb{R}^d$, we define a *language kernel* as $k_\phi(x_1, x_2) := \kappa(\phi(x_1), \phi(x_2))$.

[2] Confirmed through text-embedding-ada-002. The chance rate is 0.50 and the state-of-the-art is 0.95 (fully-trained RoBERTa-large-FT, *i.e.,* supervised learning with GPUs) [13].

Algorithm 1 The Support Vector Generation (SVG)

Require: $x_0, \alpha_0, \theta, T, \Omega = \{C, \phi\}$
for $t = 0$ to $T - 1$ **do**
  $x_{t+1} \sim q_\theta(\cdot|x_t)$
  $y_{t+1} \leftarrow f_\alpha(x_{t+1}; \mathcal{D}_t)$
  $\alpha' \leftarrow \texttt{train}(\mathcal{D}_{t+1}, \mathcal{L}, \Omega)$
  if $\tilde{A}_{t+1}(x_t, x_{t+1}) < \texttt{random}(0, 1)$ **then**
    $\alpha \leftarrow \alpha'$
  else
    $(x_{t+1}, y_{t+1}, \alpha_{t+1}) \leftarrow (x_t, y_t, 0)$
  end if
end for
return $f^* := f_\alpha(\cdot; \mathcal{D}_T) \in \mathcal{H}$

4 Support Vector Generation

The zero-shot decision function in Eq. (1) is represented as follows:

$$f_\alpha(x_{\text{new}}) = \mathbb{E}_{x,y\sim\pi}[\, k(x_{\text{new}}, x)\, y\,], \tag{3}$$

where the π is a probability distribution with countable spikes on $\mathcal{X} \times \{\pm 1\}$, each spike having an appropriate ordered index i. Here, if we denote the ordered set by $\mathcal{D}$, π can be expressed as follows:

$$\pi(x, y) = \begin{cases} \alpha_i / \|\alpha\|_1, & \text{if } (x, y) = (x_i, y_i) \\ 0, & \text{if } (x, y) \notin \mathcal{D} \end{cases} \tag{4}$$

where $\|\alpha\|_1 := \sum_i \alpha_i$ is the L^1 norm of α. Note that even if $(x, y) \in \mathcal{D}$, if x does not support the decision boundary *i.e.*, $\alpha_i = 0$, then the case is equivalent to $(x, y) \notin \mathcal{D}$. Thus, when we denote the set of support vectors, *i.e.*, the vectors with $\alpha_i > 0$, as $\mathcal{D}_{\text{SV}}$, the $\mathcal{D}$ is identified with $\mathcal{D}_{\text{SV}}$ (a.c.). As the optimal α is found by minimizing Eq. (2), it is sufficient to find the $\mathcal{D}_{\text{SV}}$ (neither $\mathcal{D}$ nor θ!) to represent the problem. This trick binds the optimal hyperplane to a VC-dimension ($\approx |\mathcal{D}_{\text{SV}}|$), which is finite and smaller than $\dim \theta$, especially for deep neural nets, not to mention that $|\mathcal{D}|$, and then makes the problem far easier than fine-tuning and training data generation.

The idea of SVG is to sample from the prior $p_\theta(x, y)$ and optimize the decision boundary with the kernel machines. However, directly sampling from p_θ is not easy due to intractability, as in many cases, we only know a scaled and/or conditional $q_\theta(x|y)$ [3]. To address this, we employ Markov chain Monte Carlo (MCMC), particularly Metropolis-Hastings (MH) sampling [8], assuming $\mathcal{D}$ is an ergodic process $x_1 \to x_2 \to \cdots x_t \to \cdots \mid \theta$ whose empirical distribution $p(x, y|\mathcal{D}^{(<t)})$ converges to $\pi(x, y)$ ($t \to \infty$, a.c.). MH aims to achieve the *detailed balance* on the state transition at the fixed point holding $\pi(x_t, y_t) q_\theta(x_{t+1}|x_t) = \pi(x_{t+1}, y_{t+1}) q_\theta(x_t|x_{t+1})$, which is tractable because the normalization of both sides are canceled. Given a sample x_t at each step t, the sampling step of MH proposes a new sample from a distribution $q_\theta(x_t \to x_{\text{new}}) = q_\theta(x_{\text{new}}|x_t)$, and decides whether to update x_{t+1} with x_{new} or x_t based on the following acceptance probability [8] [4]

$$A(x_t, x_{\text{new}}) = \min\left[1, \frac{\pi(x_{\text{new}}, y_{\text{new}})\, q_\theta(x_t|x_{\text{new}})}{\pi(x_t, y_t)\, q_\theta(x_{\text{new}}|x_t)}\right]. \tag{5}$$

We estimate the backward using $q_\theta(x_t|x_{\text{new}}) = q_\theta([x_{\text{new}}; x_t])/q_\theta(x_{\text{new}})$, where $[x_{\text{new}}; x_t]$ is the concatenation. As π is unknown, we approximate it with a scaled $\tilde{\pi}_{t+1}(x_{\text{new}}, y_{\text{new}}) =$

[3] Though completion models provide the transition $p_\theta(x_{\text{completion}}|x_{\text{prompt}}) = \prod_{i=1}^m p_\theta(w_i|x_{\text{prompt}}, w_{<i})$, the stable probability $p_\theta(x_{\text{completion}})$ is intractable: $\int_{\mathcal{X}} p_\theta(x_{\text{completion}}|x_{\text{prompt}}) dp_\theta(x_{\text{prompt}})$. This problem, concerning the "initial token" $p_\theta(x)$, is related to symbol grounding and multi-modality.

[4] Intuitively, the weight $q_\theta(x_{t+1} \to x_t)/q_\theta(x_t \to x_{t+1})$ penalizes when the backward path $x_{t+1} \to x_t$ is too low. For example, in text generation, "lions are $\to$ mammals" is correct, but "mammals are $\to$ lions" are not always correct or depends on contexts, so q_θ("mammals"| "lions")$/q_\theta$("lions"|"mammals") should be low. We can confirm that $A = 0$ if $f_t(x_{t+1})$ is misclassified or $A = 1$ if the process satisfies the detailed balance for backward/forward sampling $q_\theta(x_{t+1}|x_t) = q_\theta(x_t|x_{t+1})$.

Table 1: Complexity analysis of SVG and PLMs. n: the number of samples, m: the maximum length of texts in token, d: the dimension of the embeddings, n_{SV}: the number of support vectors ($\leq n$).

	Train	Predict
Kernel machines (ours)	$O(n^2 \cdot d)$	$O(n_{\mathrm{SV}} \cdot d)$
Transformers [37]	$O(n \cdot m^2 \cdot d)$	$O(m^2 \cdot d)$

Table 2: Results from zero-shot learning of GLUE benchmark with CPUs and without any GPUs or GPU memories. The values of † are from [13]. The experiment was repeated three times, and the average and standard deviation are listed. **Bold** indicates the best score. The rightmost column shows the average elapsed time for a single experiment.

	Single Sentence		**Sentence-pair**					
	SST-2 (Acc.)	**CoLA** (Matt.)	**QQP** (F1)	**MRPC** (F1)	**RTE** (Acc.)	**QNLI** (Acc.)	**MNLI** (3-Acc.)	(sec)
Chance rate	*50.0*	*0.0*	*50.0*	*50.0*	*50.0*	*50.0*	*33.3*	-
SVG (ours)	$\mathbf{91.7}_{0.9}$	$\mathbf{9.1}_{3.2}$	$\mathbf{72.9}_{1.5}$	$\mathbf{63.7}_{12.1}$	$\mathbf{57.9}_{2.8}$	$\mathbf{64.0}_{5.2}$	$51.8_{1.2}$	48.1
without MCMC	$88.9_{1.5}$	$4.3_{1.5}$	$65.7_{2.1}$	$58.1_{8.0}$	$55.1_{1.2}$	$55.9_{4.8}$	$44.9_{1.2}$	**1.9**
Prompting†	$83.6_{0.0}$	$2.0_{0.0}$	$49.7_{0.0}$	$61.9_{0.0}$	$51.3_{0.0}$	$50.8_{0.0}$	$51.7_{0.0}$	-

$\max(0, \alpha'_{t+1} y_{\mathrm{new}} f_{\alpha'}(x_{\mathrm{new}}))$, where α' is the dual coefficient learnt assuming $(x_{t+1}, y_{t+1}) = (x_{\mathrm{new}}, \mathrm{sgn} f_{\alpha}(x_{\mathrm{new}}))$. As by definition, y_{new} takes either binary value, we can write $y_{\mathrm{new}} = \mathrm{sgn} f_{\alpha'}(x_{\mathrm{new}})$ a.c. without loss of generality. With $\tilde{\pi}_{t+1}$, the above equation is approximated as,

$$\tilde{A}_{t+1}(x_t, x_{\mathrm{new}}) = \min \left[1, \frac{\alpha'_{t+1}\ q_\theta([x_{\mathrm{new}}; x_t])\, q_\theta(x_t)}{\alpha'_t\ q_\theta([x_t; x_{\mathrm{new}}])\, q_\theta(x_{\mathrm{new}})} \right]. \tag{6}$$

The proposed SVG algorithm is outlined in Algorithm 1. This method generates additional support vectors that help shape decision boundaries achieved by sampling from the prior distribution and subsequently updating the parameters of the kernel machine. The advantage of this method is that it allows artificial expansion of the training data, which is particularly useful in scenarios where available data is scarce. By encapsulating the data distribution more effectively, the SVG ensures improved performance of zero-shot learning models, even under computational constraints. Table 1 illustrates the complexity analysis. Even though the QP solver for Eq. (2) takes $O(n^3)$ times, we assume the practical algorithm for SVMs such as Sequential Minimal Optimization (SMO) [29], which takes $O(n^2)$ times. In *few*-shot learning, SVG is faster than the Transformers because $n \ll m^2$.

5 Numerical Experiment

To evaluate the effectiveness of our proposed approach, we conducted experiments on the General Language Understanding Evaluation (GLUE) benchmark [39]. The GLUE benchmark comprises a set of sentence or sentence-pair language understanding tasks, providing a comprehensive evaluation of the performance of language models in various natural language understanding scenarios. We have compared the performance of our proposed method, SVG to a baseline methodology: 'Prompting' [6]. In this comparison, we employed a conventional prompting technique, a zero-shot learning approach that adopts a set of manually-constructed prompts as an exemplar for the labels. For context, we used a non-fine-tuned PLM [13] comprising our baseline.

The experimental configuration used two CPU-only virtual machines on a public cloud as the computational environment and an OpenAI pay-as-you-go account as the trained language model. Three executions per task were carried out in a multi-process manner, with one CPU (not GPU) of 3 GHz and 1 GB memory are assigned to each process. The training was completed in three minutes, which is far faster and more economical than networks with GPUs.

5.1 Zero-shot Learning

The experimental results are shown in Table 2. We report the accuracy and F1 score for each task, comparing the performance of SVG with the baseline methods. The results show that SVG outper-

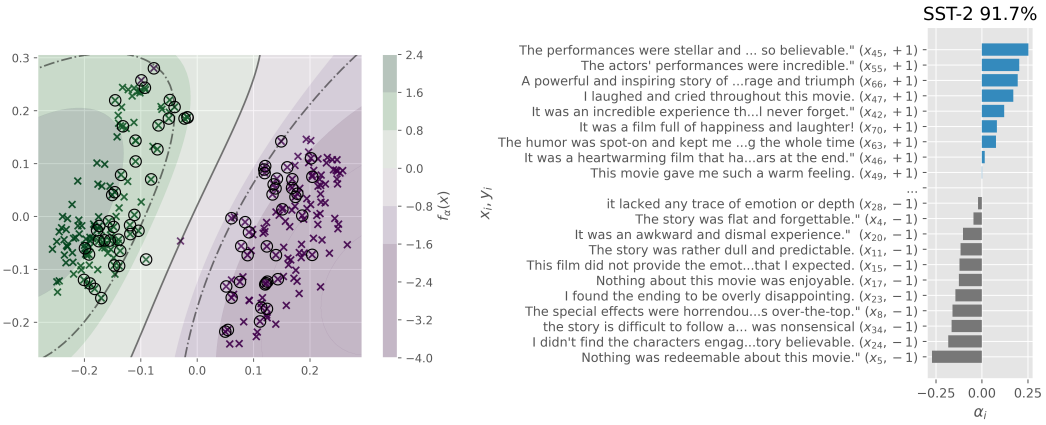

Figure 2: **Left:** Generated samples and the decision boundary, with the points in circles representing support vectors and contours with $f_\alpha(x) = +1, 0, -1$ from the left. SST-2 (Accuracy: 0.917), text-curie-001. **Right:** The spikes — high-performed support vectors generated from the SVG algorithm. The vectors are visualized from a chain out of five parallel MCMC search.

forms the baseline methods in terms of both accuracy and F1 score, demonstrating the effectiveness of SVG in improving the performance of zero-shot learning tasks, even in resource-constrained environments. Fig. 2 shows the generated samples and the decision boundary.

The superior performance of SVG can be attributed to the combination of the generative capabilities of PLMs and the computational efficiency of kernel machines. By generating support vectors from the PLMs, SVG is able to augment the training data for zero-shot learning tasks without the need for fine-tuning or additional computational resources. This allows SVG to achieve comparable or even superior performance to the baseline methods, while using only CPU resources.

The experimental results overall corroborate the effectiveness of SVG in zero-shot learning, specifically in resource-constrained contexts. Merging PLMs with kernel machines introduces new opportunities for elaborated language processing assignments. Such a combination facilitates the creation of precise and efficient natural language understanding systems.

Performance of kernel machines highly depends on the hyperparameter C, the upper bound of each entry of α, and is on a trade-off between over- and underfitting [11]. If C is too large, the model yields overfitting, and *vice versa*. One of the common search algorithms is the combination of grid search and cross-validation [35]: we divide $\mathcal{D}$ into K discrete batches randomly and test the each with $K-1$ others, and choose the best candidate maximizing the metrics, as shown in Fig. 3. We optimized $C = C_0/n$ for the number of samples n and select the best candidate C_0 from a discrete log space $\{10^{-2}, \ldots, 10^{10}\}$ at every $t_0 = 10$ step.

A proposal q_θ as close as π improves the acceptance ratio of MH. Other than completion API as we have employed in this paper, there are several heuristics of data augmentation such as back translation and text attack [5, 23].

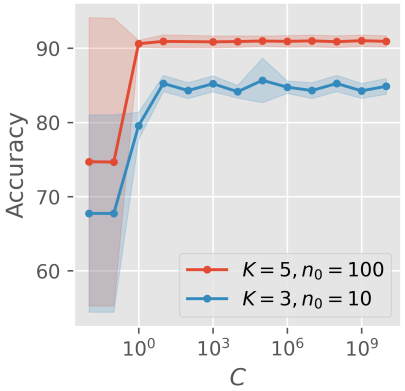

Figure 3: Results of performance by fixed C on SST-2.

5.2 Few-shot Learning

Although we can simply initialize $\mathcal{D}$ with the given labeled samples from the training set and optimize α with Eq. (2), SVG also retains the ability to generate additional support vectors that resemble these labeled samples. This is primarily because these generated support vectors are expected to be located around the decision boundary, indicating that they should closely mirror the given labeled samples. Where this approach deviates from data augmentation is in its ability to generate samples

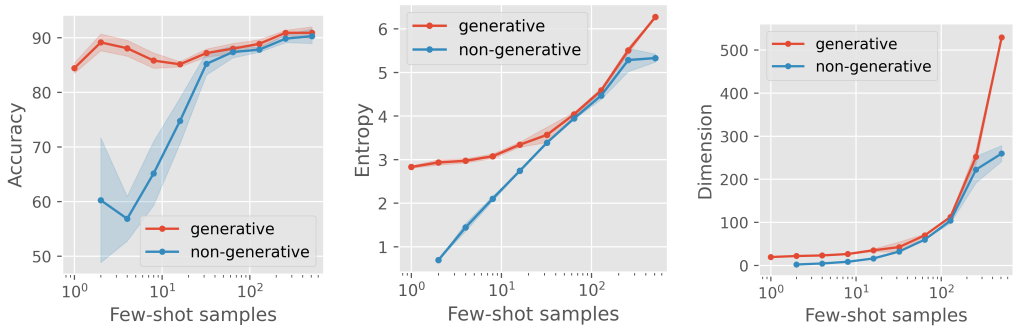

Figure 4: The result of few-shot learning of SVG through SST-2 in comparison to the conventional kernel machines, such as non-generative SVMs. The accuracy shows that even for the scarce dataset, SVG can complement the lack of data points. Entropy $\mathbb{H}[\tilde{\pi}_n] := \log \|\alpha\|_1 - \sum_{i=1}^{n} (\alpha_i / \|\alpha\|_1) \log \alpha_i$ shows that SVG successfully obtains the complexity for the scarce dataset. The right figure shows the number of support vectors acquired by each method.

which are more semantically similar to the given labeled samples than the samples generated by data augmentation. This can be attributed to the fact that SVG is based on a kernel machine, a method that effectively captures the similarity between samples by transforming the input data into a high-dimensional space, enabling a more nuanced similarity measure.

Fig. 4 shows the results of few-shot learning. In machine learning theory, including few-shot learning, that accuracy increases with the number of data. Interestingly, in the case of SVG, accuracy reaches a maximum when the number of data is two (*i.e.,* one positive example and one negative example each), then drops and steadily increases again when the number of train data is sufficient. One possible explanation for this result is that while the generalisation performance is improved by incorporating different distributions as long as the number of 'external' data to be added is small, the quality of the external data is inferior to the quality of the train data generated by the SVG itself, which in turn hurts the accuracy of the model. This phenomenon is referred to as 'few-shot double descent' in SVG, though the reasons for this are not analysed further in this paper.

6 Discussion

6.1 Applications

While we have mainly discussed the case of single sentences with two classes, in tasks of natural language understanding, we often have to deal with more than two classes and more than one sentence. GLUE benchmark has sentence-pair classification tasks such as paraphrase identification (QQP, MRPC) and inference (RTE, QNLI), and multi-class tasks (MNLI) where the model has to predict one of three classes (entailment, neutral, contradiction) for a sentence pair. Each of the tasks also has a training data which has not been covered yet in this paper, though application to few-shot learning is also possible. In this section, we discuss how to extend the proposed method to such tasks.

6.1.1 Sentence-pairs

For sentence-pair classification, we can adopt the same methodology as in the case of single-sentence classification, albeit with a few alterations. Initially, our task is to represent the features of the sentence-pair. We can accomplish this by utilizing a language kernel $k_\phi : \mathcal{X}^4 \to \mathbb{R}$, which accepts two sentences as input and is defined as follows[5]

$$k_\phi([x_1; x_2], [x_3; x_4]) = \kappa(\phi(x_2) - \phi(x_1), \phi(x_4) - \phi(x_3)), \tag{7}$$

where $\phi : \mathcal{X} \to \mathbb{R}^d$ denotes a text embedding and $\kappa : \mathbb{R}^{2d} \to \mathbb{R}$ represents a positive definite kernel.

[5] We have assumed a recurring topology $\mathcal{X}^2 \subset \mathcal{X} = \mathcal{V}^m$, *i.e.*, pairings of sentences constitute a language just as individual sentences do, hence $k_\phi : (\mathcal{X}^2)^2 \to \mathbb{R}$ also constitutes a language kernel. Following this assumption, the language kernel can be alternatively written as $k_\phi : \mathcal{X} \to \mathbb{R}$. This topic is slated for discussion in future work.

Table 3: The model-agnostic multi-class task descriptions $x_1, \ldots, x_M$ which yield high performance in SVG. The placeholder of quote, labels and sample are obtained from the GLUE original paper [39], which can also be scraped at https://tensorflow.org/datasets/catalog/glue. Instead of text-curie-001 for MCMC, the initial samples will be inferred once via larger completion models such as text-davinci-003 .

(a) Template <quote> 1: <label$_1$>, 2: <label$_2$>, ..., M: <label$_M$> The possible ten examples of the <sample> of "i: <label$_i$>" are:
(b) SST-2 (single sentence, 2 classes) "The Stanford Sentiment Treebank consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence. We use the two-way (positive/negative) class split, and use only sentence-level labels." 1: positive, 2: negative The possible ten examples of the sentence of "2: negative" are:

First, we need to compute the acceptance probability $\tilde{A}_{i+1}(x_i, x_{\text{new}})$ for a sentence-pair. To do this, we can use the same approach as for single-sentence classification, but with a few modifications. Second, we need to compute the probability $q_\theta(x_{\text{new}}|x_i)$ for the sentence-pair. To do this, we can use a language model $q_\theta : \mathcal{X}^2 \to \mathbb{R}$ that takes two sentences as input defined as $q_\theta([x_1; x_2]) = \frac{1}{Z} \exp\left(\sum_{i=1}^{2} \log q_\theta(x_i)\right)$, where Z is a normalization constant and $q_\theta(x_i)$ is a language model for a single sentence. Finally, we can compute the acceptance probability $\tilde{A}_{i+1}([x_i^1; x_i^2], [x_{\text{new}}^1; x_{\text{new}}^2])$ for a sentence-pair using q_θ.

6.1.2 Multi-class

In multi-class classification tasks, we have to deal with more than two classes. Typically, we can use the one-vs-rest (OVR) or one-vs-one (OVO) approach. The former constructs a binary classifier for each class, and the class with the highest score is chosen as the predicted class, whereas the latter constructs a binary classifier for each pair of classes, and the class with the highest number of wins is chosen as the predicted class.

For the OVR approach, we can use the same algorithm as for the binary classification case, with the only difference being that the labels y_i are now one-hot vectors instead of the binaries $\{\pm 1\}$. For OVO, we can use the same algorithm, but with the labels y_i being a vector of length $M(M-1)/2$, where M is the number of classes. To calculate the acceptance ratio in the MCMC step, we need to compute the backward probability $q_\phi(x_i|x_{\text{new}})$. For OVR, we can simply use the same formula as for the binary classification case. On the other hand, for OVO we need to compute the backward probability for each pair of classes.

6.2 Tuning Technique

As we approximate the target probability π with $\tilde{\pi}_t$, there is a risk of exponential amplification of the model's approximation error as the generative progresses, especially if the dynamics $\mathcal{T} : \tilde{\pi}_t \to \mathcal{T}\tilde{\pi}_t$ is non-contractive in the measurable space on $\mathcal{X}$. To mitigate this drawback, we employ a "cross-validating" approach for posterior estimation in MCMC [16]. We run K independent chains in parallel, and use the posterior approximation $\tilde{\pi}_t^{i-1}$ from the previous chain as an approximation to π for the current chain, instead of using $\tilde{\pi}_t^i$ directly. Additionally, we have introduced the following heuristics into the implementation:

- A burn-in period t_0, assuming that the variance of $\tilde{\pi}_t$ stabilizes after t_0 iterations. We update the reference model $\tilde{\pi}_t$ every t_0 steps, instead of updating it at every step, to reduce the sensitivity of the initial samples and improve the overall stability of the approximation.

- We incorporate a probabilistic SVM [40] and multiply the probability $p(y_t|x_{\text{new}})$ to π_t. This helps to refine the estimation of the posterior by incorporating the predictive power of the SVM.

- If the newly generated samples x_{new} duplicates any of the previous ones, or if is one of the predefined stop words *e.g.,* ".", <EOS>, or <LF>, we set $\alpha'_{t+1} = 0$ and generate a new token.

To address the issue of sensitivity to initial samples, we employ transfer learning for initial sampling. We directly draw $2n_0$ seeds from the task description in the original paper of the target dataset [39]. We carefully select task descriptions that are model-agnostic and unbiased, enabling us to train our model in a zero-shot manner. While we use a mid-size completion model text-curie-001 during the process, we utilize text-davinci-003 for the initial sampling. We have observed that repeating this initial sampling process also yields high performance, but it increases the overall computational cost compared to MCMC. Therefore, using this heavy sampling approach once at $t = 0$ is the most practical option. In Table 3, we provide examples of the model-agnostic task descriptions used for transfer learning. These examples demonstrate the effectiveness of our approach in achieving high performance across different tasks. An exhaustive list of models and hyperparameters used in the experiments of this paper is presented in Table 4.

6.3 Limitation

While our proposed approach has yielded promising results, it presents certain limitations. First, SVG's effectiveness hinges on the quality of the representative samples that the PLMs generate. The interpretability of these generative models often lacks clarity, and they may inadvertently perpetuate undesirable biases inherent in the training data [43].

Second, in spite of superior memory efficiency compared to contemporary PLMs, the computational efficiency of kernel methods still tends to deteriorate significantly when dealing with high-dimensional embeddings or many-shot problems. As for computational speed $O(n^2)$, kernel methods frequently lag behind PLMs $O(nm^2)$ of $n \gg m^2$.

Table 4: Models and hyperparameters used for GLUE benchmark, shared by all the tasks after tuned with SST-2.

	Description	**Value**
ϕ	Text embedding	text-embedding-ada-002[†]
θ	Language model	text-davinci-003[†] (seeds)
		text-curie-001[†] (MCMC)
f_α	Kernel machine model	libsvm[‡]
n	Examples per class	100+1000 (seeds/MCMC)
d	Dimension of the embeds	1536[†]
m	Length in tokens	2048[†]
C_0	Upper bound of the duals α_i	$10^{-2}, 10^{-1}, \ldots, 10^{10}$
κ	Non-linearity	RBF, Linear
γ	Scaler of the RBF kernels	auto[‡]
K	Chains in parallel.	5
t_0	Burn-in period	10
	Multi-class settings	OVR, OVO
	Cross-validating bins	5
	Temperature	$.5 \pm .05$
	Prompt	x as _?
	Stop words	<EOS>|<LF>|.

[†]GPT-3 [6] [‡][7]

We anticipate addressing this limitation in our future work. One proposed direction of research involves the development of more informed and reliable strategies for generating representative samples. This could mean leveraging insights from active learning or using novel architectures that encourage diversity in the generated examples. We also aim to explore efficient kernel methods and applicable approximations, such as random Fourier features [32], to tackle large data scenarios. Pursuing these avenues, we believe, will bring us closer to creating efficient and effective zero-shot learning models with a wider range of applicability.

7 Conclusion

We presented *Support Vector Generation* (SVG), a training-free method that converts a frozen language model into an explicit, interpretable kernel classifier by (i) sampling candidate sentences near the decision boundary and (ii) solving a support-vector machine in the PLM-induced language kernel. SVG minimises empirical hinge loss within the span of its supports, completes training on a single CPU in under three minutes, andon seven zero-shot GLUE tasksmatches or slightly exceeds strong prompting baselines while providing faithful sentence-level rationales. These results demonstrate that large language models can be distilled into compact, explainable, and compute-efficient predictors, opening a path toward wider deployment of zero-shot NLP in resource-constrained settings.

References

[1] Sajjad Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 4190–4197, 2020. URL https://aclanthology.org/2020.acl-main.385.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 9505–9515, 2018. URL https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7d3c9e8a1e748-Paper.pdf.

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[4] A Blumer, A Ehrenfeucht, D Haussler, and M Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. of the ACM*, 36(4):929–965, 1989.

[5] R Brislin. Back-translation for cross-cultural research. *J. of Cross-cultural Psychology*, 1(3): 185–216, 1970.

[6] T Brown, B Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

[7] C Chang and C Lin. LIBSVM: a library for support vector machines. *ACM Trans. on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[8] S Chib and E Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[9] M Collins. Discriminative reranking for natural language parsing. In *ACL*, 2002.

[10] C Cortes and V Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[11] K Duan, S Keerthi, and A Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, 2003.

[12] C Elkan. Logistic regression, naïve Bayes, and the importance of proper feature scaling. *KDD*, 2005.

[13] T Gao, A Fisch, and D Chen. Making pre-trained language models better few-shot learners. *arXiv:2012.15723*, 2020.

[14] K Grauman and T Darrell. The pyramid match kernel: Discriminative classification with sets of image features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(6):1415–1430, 2007.

[15] T Hastie, R Tibshirani, and J Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[16] L Held, B Schrödle, and H Rue. Posterior and cross-validatory predictive checks: a comparison of mcmc and inla. *Statistical modelling and regression structures: Festschrift in honour of ludwig fahrmeir*, pages 91–110, 2010.

[17] K Ito, S Ohsawa, and H Tanaka. Information diffusion enhanced by multi-task peer prediction. In *Proceedings of the 20th International Conference on Information Integration and Web-Based Applications & Services*, 2018.

[18] C Leslie, E Eskin, J Weston, and W Noble. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, pages 579–590, 2003.

[19] Y Liu, Z Yang, T Zhao, D Xiong, X Wang, and B Liu. Generating long and coherent paragraphs with topic-aware neural networks. *arXiv:1903.03051*, 2019.

[20] D Lowe. Object recognition from local scale-invariant features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(6):810–822, 1999.

[21] C Manning, P Raghavan, and H Schütze. Introduction to information retrieval. 2008.

[22] Y Meng, J Huang, Y Zhang, and J Han. Generating training data with language models: Towards zero-shot language understanding. *NeurIPS*, 2022.

[23] J Morris, E Lifland, J Yoo, J Grigsby, D Jin, and Y Qi. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *EMNLP*, 2020.

[24] P Nakkiran, G Kaplun, Y Bansal, T Yang, B Barak, and I Sutskever. Deep double descent: Where bigger models and more data hurt. *J. of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[25] S Ohsawa. Truthful self-play. In *ICLR*, 2023.

[26] S Ohsawa, Y Obara, and T Osogami. Gated probabilistic matrix factorization: learning users' attention from missing values. In *IJCAI*, 2016.

[27] S Ohsawa, K Akuzawa, T Matsushima, G Bezerra, Y Iwasawa, H Kajino, S Takenaka, and Y Matsuo. Neuron as an agent. In *ICLR Workshop*, 2018.

[28] H Okamoto, M Suzuki, I Higuchi, S Ohsawa, and Y Matsuo. Dual space learning with variational autoencoders. In *ICLR Workshop*, 2019.

[29] J Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[30] A Radford, K Narasimhan, T Salimans, I Sutskever, and J Martens. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

[31] A Radford, J Wu, R Child, D Luan, D Amodei, and I Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[32] A Rahimi and B Recht. Random features for large-scale kernel machines. In *NIPS*, 2008.

[33] C Rasmussen and C Williams. Gaussian processes for machine learning. *Adaptive Computation and Machine Learning*, pages 100–109, 2006.

[34] J Shawe-Taylor and N Cristianini. Kernel methods for remote homology detection. In *KDD*, 2004.

[35] I Syarif, A Prugel-Bennett, and G Wills. SVM parameter optimization using grid search and genetic algorithm to improve classification performance. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 14(4):1502–1509, 2016.

[36] V Vapnik and A Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264, 1971.

[37] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[38] S Vishwanathan, A Smola, and B Schölkopf. Fast kernel machines. *NIPS*, 2003.

[39] A Wang, A Singh, J Michael, F Hill, O Levy, and S Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[40] T Wu, C Lin, and R Weng. Probability estimates for multi-class classification by pairwise coupling. *NIPS*, 2003.

[41] C Zhang, S Bengio, M Hardt, B Recht, and O Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2016.

[42] Z Zhang, E Dinan, J Urbanek, A Sordoni, and A Trischler. Dialogpt: Large-scale generative pre-training for conversational response generation. In *NAACL*, 2019.

[43] J Zhao, T Wang, M Yatskar, V Ordonez, and K Chang. The limitations of deep learning in adversarial settings. *ACL*, 2018.

[44] X Zhu, J Dai, M Ye, Z Zhu, Y Liu, and L Song. Generative adversarial learning towards fast weakly supervised detection. In *CVPR*, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 state that SVG provides an interpretable, training-free kernel classifier and match the theoretical (Sec. 3) and empirical (Sec. 5) findings without over-claiming generalisation beyond the GLUE tasks considered.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 6.3 *Limitation* details reliance on LLM quality, scalability issues of kernel methods, and potential bias amplification.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Assumptions for Thms. 3.13.2 are stated in Sec. 3; complete proofs appear in Appendix A.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 5, Table 4, and the supplement list all hyper-parameters, data splits, and evaluation protocols; an anonymised GitHub repo with scripts is linked in the supplementary material.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Public GLUE datasets are cited; anonymised code (including inference scripts and instructions) is provided in the supplemental ZIP and will be made public upon acceptance.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 5 and Table 4 enumerate chain counts, burn-in, C-grid, kernel type, proposal temperature, and dataset splits.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 2 reports mean $\pm$ standard deviation over three independent runs with different random seeds; Sec. 5.1 explains the variability source.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 5.1 describes use of 3 GHz 1-core CPUs with 1 GB RAM on a public cloud; Table 2 (right column) reports per-task wall-clock time.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work follows the code; bias risks are acknowledged (Sec. 6.3) and no personal data is processed.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Sec. 6.3 discusses positive impacts (energy-efficient inference, interpretability) and negative ones (possible biased rationales, misuse of generated text).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No new high-risk model or dataset is released; SVG code only calls the OpenAI API under their existing usage policies.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: GLUE (MIT license) and LIBSVM (BSD-style) are cited; OpenAI GPT-3 API terms are referenced in Sec. 4.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new datasets or pretrained modelsonly code.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human-subject or crowdsourcing study was conducted.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human-subject research is involved.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

Answer: [Yes]

Justification: Sec. 4 details how frozen OpenAI completion models (text-davinci-003, text-curie-001) are used for MetropolisHastings proposals and initial seed generation, which is central to SVG.