

Learning Obfuscations Of LLM Embedding Sequences: Stained Glass Transform

Jay Roberts
Protopia AI
jay@protopia.ai

Kyle Mylonakis
Protopia AI
kyle@protopia.ai

Sidhartha Roy
Protopia AI
sid@protopia.ai

Kaan Kale
Protopia AI
kaan.kale@protopia.ai

Abstract—The high cost of ownership of AI compute infrastructure and challenges of robust serving of large language models (LLMs) has led to a surge in managed Model-as-a-service deployments. Even when enterprises choose on-premises deployments, the compute infrastructure is typically shared across many teams in order to maximize the return on investment. In both scenarios the deployed models operate only on plaintext data, and so enterprise data owners must allow their data to appear in plaintext on a shared or multi-tenant compute infrastructure. This results in data owners with private or sensitive data being hesitant or restricted in what data they use with these types of deployments. In this work we introduce the Stained Glass Transform, a learned, stochastic, and sequence dependent transformation of the word embeddings of an LLM which information theoretically provides privacy to the input of the LLM while preserving the utility of model. We theoretically connect a particular class of Stained Glass Transforms to the theory of mutual information of Gaussian Mixture Models. We then calculate a-postiori privacy estimates, based on mutual information, and verify the privacy and utility of instances of transformed embeddings through token level metrics of privacy and standard LLM performance benchmarks.

1. Introduction

The rise of Transformers and Large Language Models (LLMs) has driven rapid advances across a wide range of technologies, including code completion, retrieval-augmented generation (RAG), and other systems enabling natural language interaction with vast amounts of data [1], [2]. While the privacy of training data of LLMs is well studied, enterprises looking to privately use pretrained LLMs for inference are limited to technologies such as tokenization, masking, and abstraction [3], [4]. Unfortunately such methods are insufficient as they can significantly degrade LLM performance when the sensitive data is critical to the inference and are hard to use when it is unclear what is sensitive in the prompt [5], [6], [7]. In some cases, the solutions used to identify sensitive data use LLM functionalities of their own that require the data owner to trust the tokenization/masking provider with access to their plain-text sensitive information in order to protect it from the original

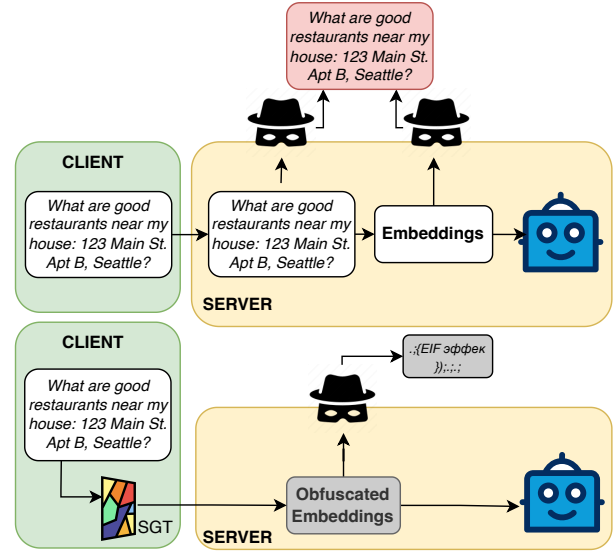


Figure 1. Client input is sent as plaintext (top) and intercepted by an adversary. The points of potential leak include not only the tokens but also the target model’s embedding which can be trivially inverted back to text with a nearest neighbor attack. With the Stained Glass Transform (bottom) the Client no longer transmits plaintext and instead the SGT converts the plaintext into a protected embedding that can be directly used by the LLM while still maintaining privacy.

provider (e.g. Amazon Comprehend, Azure AI Language, Google Cloud Sensitive Data Protection).

The inference attack vector we address is one where the the client’s data has left their trust zone to be used by a LLM where it necessarily must be in a unencrypted state. At this point we assume an unauthorized user is able to access the client’s unencrypted data by way of access to the server running the AI model (see Figure 1).

We must then determine what data can be sent out of the trust zone that will maintain the client’s privacy, but also be effectively used by the LLM. Transmitting the tokens instead of text provides no protection. The next representation that could be sent are the embeddings of the tokens rather than the tokens themselves. These are not immediately human readable and would not change the model performance at all, however, they can be trivially inverted back into

plaintext, even in the presence of Gaussian noise, using nearest neighbor searches (see Figure 1) [8].

In this paper we seek an answer to the following question: *Is it possible to information theoretically protect the input embeddings sent to a fixed LLM at inference time without greatly impacting the LLM’s utility?*

Our affirmative answer to this is supported by the introduction of the Stained Glass Transform (SGT) for LLMs, a stochastic, input, and sequence dependent learned transformation of an LLM’s token embeddings (Figure 2). Working with vector token embeddings avoids the pitfalls of tokenization based privacy mentioned above by allowing us to apply continuous optimization techniques rather than limiting ourselves to discrete or combinatorial optimization.

We organize the remaining sections as follows: Section 2 provides background, relevant related works, and notation for the rest of the paper. Then Section 3 introduces the Stained Glass Transform. Section 4 formulates the learning problem used to define the SGT. Privacy metrics are introduced in Section 6, and experimental results are reported in Section 7.

2. Background

2.1. Large Language Models

Though the Stained Glass Transform (SGT) is modality agnostic (see Definition 3.2), in this work we focus on how the SGT is used to protect the inputs of decoder-only LLMs (the most common architecture of state-of-the-art autoregressive LLMs [9], [10]). This subsection provides background on how these LLMs function.

Broadly, current state-of-the-art LLM architectures can be broken into four stages: tokenization, embedding, decoding, and sampling. The tokenization stage converts text into a pre-defined vocabulary of tokens. These tokens are then mapped through an embedding function, which is a one-to-one function mapping token ids to embedding vectors. Often we refer to the embedding function as the embedding table or embedding matrix. Once the sequence of text has been converted to sequences of embeddings, the embeddings are fed through attention based decoder layers to generate a conditional probability distribution of the next token given the input embeddings. A sampling method such as beam search is then used to produce a sample of the predicted next token, which is then appended to the original input tokens and the process continues auto-regressively until a termination criteria is hit [10], [11].

Importantly, unless the tokenizer and embedding tables are secret, *knowledge of clean token embeddings is equivalent to knowledge of text*. The goal of the SGT for LLMs is to protect these clean token embeddings and corresponding original prompts while maintaining the performance of the LLM.

2.2. Related Works

In this section we cover relevant related works regarding privacy technologies for LLMs and privacy techniques which are similar in formulation to the Stained Glass Transform.

Tokenization and Masking One of the most straightforward ways to protect a prompt is simply to replace any token which is sensitive with a placeholder token. When the sensitive token is replaced with a meaningless token unrelated to the sensitive data, the technique is known as masking (e.g replacing "Jon Doe: 555-55-5555" with "[MASK]: [MASK]"). If instead the sensitive token is replaced with an obfuscated hash which is reversible to the original token after processing, the technique is known as tokenization (e.g replacing "Jon Doe: 555-55-5555" with "[USER12345]:[VALUE12345]" prior to processing). Both of these techniques rely on applying a series of rules or possibly other language models to replace the sensitive tokens with their masked or tokenized counterparts [12], [13].

Tokenization and masking both can damage LLM utility when applied too liberally, and, when the sensitive data is necessary for the LLM to properly, reason and respond to the query [6]. Moreover, they can only be applied when it is easy to identify sensitive data from non-sensitive data. This is an extremely difficult task in the case of unstructured text inputs to an LLM as pieces of data which are independently non-sensitive can become sensitive when combined together in a prompt. Due to this, tokenization and masking generally suffer from either being *not enough* to provide privacy in a meaningful way or *too much*, resulting in a non-trivial loss of LLM performance.

Abstraction Another option to rules or LLM based prompt modification is abstraction [14], [15]. This technique relies on replacing or withholding specific information in a prompt which may be sensitive. Any information which is replaced is done so with more general information that is not sensitive, but which will not dramatically affect the response of the LLM. Optionally withheld information can then be used in conjunction with the LLM response on the redacted information complete the inference [4]. For example in the first case, the question "What is the weather like in London, SW1P 4GH?" could be replaced with "What is the weather in central London?" and in the second it could be replaced with "What is the weather in London SW1P 4GH, San Diego 92128, and Austin 78701".

Abstraction can suffer from similar problems to tokenization in masking with regards to sensitive feature identification and LLM utility. It is also unclear how practical these methods are for scaling out to enterprises which may be dealing with many different types of sensitive data.

Split Learning Split learning learning aims to protect sensitive data by decomposing an existing neural network (e.g a foundational model) into two components, a client side network which is located in the trust zone of the user, and a service side component which lives in an untrusted environment [16], [16], [17], [18], [19], [20], [21]. The

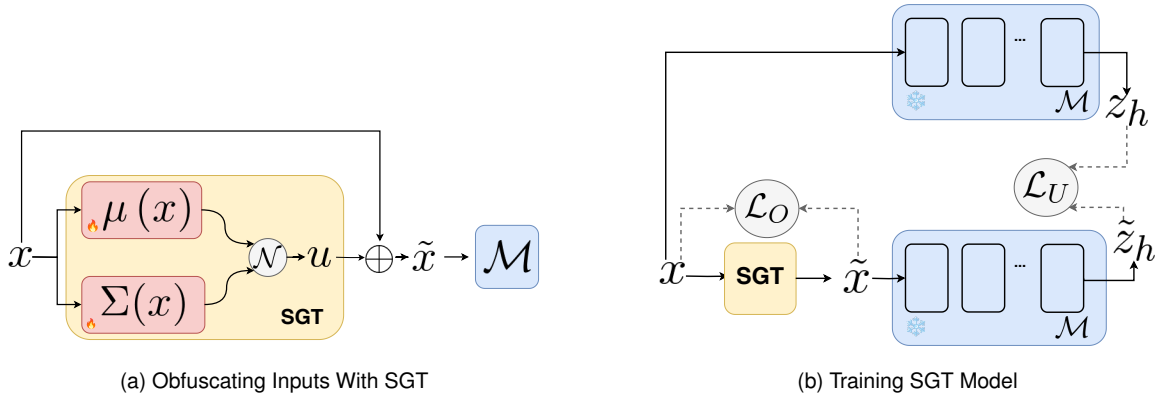


Figure 2.

A high-level diagram of inference, (a), and training, (b), for SGT models with sequence and learnable-parameters notation suppressed for ease of reading. Subfigure (a): During inference the input sequence, x , is used to generate a mean, $\mu(x)$, and a covariance matrix, $\Sigma(x)$. These define a multivariate Gaussian, \mathcal{N} , which is sampled from and added to the input to produce its obfuscations \tilde{x} . The obfuscation is then fed through the model, \mathcal{M} , as a usual embedding. Subfigure (b): The SGT is trained by applying the obfuscation loss, \mathcal{L}_O , to the input and the obfuscation (dotted). To compute the utility loss, \mathcal{L}_U , the input, x , and its obfuscations, \tilde{x} , are passed through the frozen model to compute logits, z_h and \tilde{z}_h , respectively, which are then compared with a cross entropy loss. In this way the SGT can learn to generate protected embeddings that maintain the utility of the target model without modifying the model at all.

output of the client side model is transformed, either by fine-tuning the client side model, or by directly applying transformations to client’s output activations. Information theoretic quantities such as mutual information or distance correlation (or bounds on those quantities), can be used as losses to generate obfuscations to be transmitted to the server outside the client’s trust zone [22], [23].

This approach is straightforward to apply in non-autoregressive tasks such as image and text classification, but becomes a major engineering challenge when applied to autoregressive LLMs as significant communication is required between the client and server at every step of the autoregression. However, when the split point between the client and the server is at the embedding layer of the LLM, communication between the client and the server is not required at every step of the autoregression, allowing for efficient generation of text for transformed input embeddings. This observation is what drives the central hypothesis of the paper: *it is possible to protect input embeddings to an LLM at inference time via information theoretic techniques without significantly degrading utility.*

2.3. Notation

For consistency in the rest of the paper, we introduce some notation in this subsection. The dataset of inputs used to train the SGT model is a random variable in $X \in L^2(\mathbb{R}^d; \mathbb{R}^d)$ with realizations written as x . The probability distribution of X is denoted as $\mathbb{P}[X \in B \subset \mathbb{R}^d] = P_X(B)$. The SGT of the dataset is a new random variable \tilde{X} with realizations \tilde{x} . The SGT model is a conditional probability model of the \tilde{X} conditioned on a realization of the

input, x , which we denote by $\mathbb{P}[\tilde{X} \in B | X = x] = P_{\tilde{X}|x}(B)$. The SGT model is trained with a utility loss, denoted \mathcal{L}_U , and an obfuscation loss, denoted \mathcal{L}_O . We use \mathcal{M} for the (frozen) target LLM of an SGT model.

The obfuscation loss (in part) depends on the (differential) entropy of various inputs and we will write $h(V)$ for the entropy of a random variable V . The conditional entropy of V given another random variable W , is written as $h(V|W)$. When using SGTs for LLMs we work with sequences of vectors (or matrices). We write a sequence of vectors as $V_{1:T} = [V_1, \dots, V_T]$. Finally, for a square matrix A , its determinant is denoted $|A|$, and the A -weighted L^2 norm is expressed as $|x|_A^2 = \langle x | A | x \rangle$. A Gaussian distribution of mean μ and covariance Σ is denoted $N(\mu, \Sigma)$, and we define Δ_d^+ as the set of diagonal d by d positive definite matrices. The set of non-negative real numbers is denoted \mathbb{R}^+ .

3. Stained Glass Transform: Theoretical Foundations

In this section we develop the theory of our obfuscation mechanism, the Stained Glass Transform (SGT). First we define Affine SGTs, a special class of SGTs that will be used in this paper. We then state the general definition of the SGT and provide a proposition which connects the two definitions. The general definition allows us to characterize the distribution of the full SGT of a dataset. This will serve as the cornerstone of our derivation of the information theoretic loss used to train SGT models.

Definition 3.1. Given a random variable X and a parameterized family of probability densities, $G(\cdot; \alpha_\theta(x)) : \mathbb{R}^d \rightarrow \mathbb{R}^+$,

the **Affine Stained Glass Transform** of X is defined as

$$\tilde{X} = X + U$$

where U is distributed according to $G(\cdot; \alpha_\theta(x))$ and α_θ represents learnable parameters of a distribution (e.g. mean and covariance).

Creating an (Affine) SGT for a random variable (i.e. dataset) X amounts to defining the family of probability densities. For LLMs the dataset consists of sequences of token embeddings from an LLM’s embedding table, $\{x_i\}_{i=1}^V \subset \mathbb{R}^d$. In this paper we choose the conditional densities to be sequence dependent Gaussian distributions. Specifically, these distributions are $N(\mu_\theta(X), \Sigma_\theta(X))$ where $\mu_\theta : \mathbb{R}^{d \times T} \rightarrow \mathbb{R}^{d \times T}$ and $\Sigma_\theta : \mathbb{R}^d \rightarrow (\Delta_d^+)^T$ and where $\mathbb{R}^{d \times T}$ are sequences of embeddings and $(\Delta_d^+)^T$ are sequences of diagonal positive definite matrices of length T . The sequence dependent mean and variance estimator functions are modeled as transformers to ensure that the SGT models are a function of the *sequence* of the embeddings rather than treating embeddings as independent of each other. Figure 2a gives a description of inference.

Inference with SGT models. Current state of the art LLMs take as input sequences of embeddings from a fixed embedding table. The affine structure of the SGT allows us to easily generate obfuscations of sequences of embeddings. Given an input sequence $x_{1:T} \in \mathbb{R}^{d \times T}$ we generate an obfuscation by computing the means $\mu_\theta(x_{1:T})_{1:T}$ and variances $\Sigma_\theta(x_{1:T})_{1:T}$. Next, we sample T independent multivariate standard normal vectors, $u_{1:T} \sim \mathcal{N}(0_d, I_d)$, creating the SGT:

$$\tilde{x}_{1:T} = x_{1:T} + \mu_\theta(x_{1:T})_{1:T} + \Sigma_\theta^{1/2}(x_{1:T})_{1:T} u_{1:T}. \quad (1)$$

Preparing for training. Equation (1) shows that obfuscation with an SGT is entirely defined by the conditional probability $P_{\tilde{X}|X=x}$. However, in order to learn the parameters of the model (μ_θ and Σ_θ), we need to understand how the SGT itself is distributed. This is most easily done using the general definition of the SGT:

Definition 3.2. Given a random variable X on \mathbb{R}^d , a **Stained Glass Transform** of X parameterized by density family $G(\cdot; \alpha(x)) : \mathbb{R}^d \rightarrow \mathbb{R}^+$, is defined as a random variable \tilde{X} whose (regular) conditional probability satisfies:

$$P_{\tilde{X}|X=x}(B; \theta) = \int_B G(y; \alpha_\theta(x)) dy.$$

The following proposition provides a description of the conditional densities of Affine SGTs.

Proposition 3.3. Let \tilde{X} be an Affine SGT of X with $\tilde{X} = X + U$ where $U \sim G(\cdot; \alpha)$, the the conditional probability of \tilde{X} given $X = x$ is:

$$P_{\tilde{X}|X=x}(B) = \int_B G(y - x; \alpha(x)) dy$$

Proof. Assume that U has conditional density $G(u; \alpha(x))$ and consider the random variable $\tilde{X} = X + U$. Then

$$\begin{aligned} P_{\tilde{X}|X=x}(B) &= \mathbb{P}[\tilde{X} \in B | X = x] \\ &= \mathbb{P}[\tilde{X} - X \in B - x | X = x] \\ &= \mathbb{P}[U \in B - x | X = x] \end{aligned}$$

Which in integral form becomes $\int_{B-x} G(u; \alpha(x)) du = \int_B G(y - x; \alpha(x)) dy$. \square

Proposition 3.3 characterizes the distribution of the SGT, \tilde{X} , over a dataset. The empirical distribution of our dataset is given by $P_X(x) = \sum \omega_i \delta(x - x_i)$, where $\delta(x - \cdot)$ is a point mass density. Proposition 3.3 then implies the following corollary.

Corollary 3.4. Let \tilde{X} be an Affine SGT of X of the form $\tilde{X} = X + U$ with $U \sim N(\mu_\theta(X); \Sigma_\theta(X))$, then Y is distributed as a Gaussian Mixture Model:

$$P_{\tilde{X}}(z) = \sum_{x_i \in X} \omega_i N(z; \mu_\theta(x_i) + x_i; \Sigma_\theta(x_i)) \quad (2)$$

where $\omega_i = P_X(x_i)$.

Knowing the precise form of the obfuscation allows us to derive information theoretic relationships between our the SGT and its input dataset. In particular it we can now rigorously incorporate the mutual information between the input dataset and the obfuscation into our learning objective.

4. Obfuscation Learning Problem

In this section we explain the loss function used to train the SGT for LLMs. Given a target model \mathcal{M} and a dataset X we want the obfuscation \tilde{X} to satisfy:

- 1) **Utility:** The obfuscation should maintain similar performance with respect to \mathcal{M} .
- 2) **Privacy:** The obfuscation should not be reconstructible.

To achieve both goals we take a multi-objective learning approach with two classes of losses: a utility loss, \mathcal{L}_U , that penalizes the obfuscation for degrading the performance of \mathcal{M} and a obfuscation loss, \mathcal{L}_O , which encourages the obfuscation to be difficult to reconstruct.

For our utility loss we use a self-supervised loss function at the logits layer of the frozen target LLM, \mathcal{M} between the logits arising from the SGT and untransformed input embeddings. Specifically, if z_h and \tilde{z}_h are the logits of the target model for inputs x and \tilde{x} , respectively, then the utility loss is: $\mathcal{L}_U(x, \tilde{x}) = d(z_h, \tilde{z}_h)$ where d is some metric on \mathbb{R}^d . In practice we choose \mathcal{L}_U to be cross entropy loss between the probabilities generated by the logits of the SGT and untransformed input embeddings. The obfuscation loss, \mathcal{L}_O , is to reward increasing the difficulty in reconstructing X from \tilde{X} . The full loss function is a sum of the utility and obfuscation losses:

$$\mathcal{L} = \mathcal{L}_U + \mathcal{L}_O. \quad (3)$$

In practice the obfuscation loss is made up of three subcomponents (detailed in Section 5.3).

Figure 2b shows the procedure for training an SGT model for a fixed target LLM, \mathcal{M} . Broadly, we freeze the weights of \mathcal{M} during training and only update the SGT model’s weights. The input, $x_{[1:T]}$, is compared against its SGT realization, $\tilde{x}_{[1:T]}$, via the obfuscation loss. Then both the input embedding and its SGT are fed through \mathcal{M} . The resulting logits are compared with cross entropy. In this way an SGT model can be trained without modifying the target LLM.

4.1. Mutual information and PAC Privacy bounds

In this subsection we describe how we use mutual information in the obfuscation loss \mathcal{L}_O to provide information theoretic protection of embeddings. We begin by motivating its consideration.

Mutual information has been widely used as a measure of privacy [14], [24], [25], [26], [27]. Moreover, in [28] the authors develop a provable-approximately-correct theory of privacy (PAC-Privacy) which gives theoretical guarantees that bound the probability an adversary can reconstruct an input from its obfuscation. The key result is that, given an input X and obfuscation mechanism \tilde{X} , then the reconstruction advantage satisfies the bound

$$\Delta_{KL}\delta \leq MI(X; \tilde{X}), \quad (4)$$

where the $\Delta_{KL}\delta = \delta \log(\frac{\delta}{\delta_0}) + (1 - \delta) \log(\frac{1-\delta}{1-\delta_0})$, and is the Kullback-Leiber Divergence between the a-priori success rate, $1 - \delta_0$, of reconstruction (i.e. the likelihood an adversary can reconstruct an input knowing only the data distribution) and the the posterior advantage, $1 - \delta$, gained by observing the obfuscation. This result demonstrates that mutual information plays a key role in privacy protection.

However, Bound (4) was used by the authors to construct a specific obfuscation mechanism that was tractable to compute. Our situation is attempting to measure the protection of more general obfuscation mechanisms and, as was noted by the authors in [29], the bound shown in Equation (4) is not tight and in practice stronger privacy is observed.

This implies that the precise relationship between reconstruction difficulty and mutual information is more complex than in Equation (4). We believe that future work to improve the bound in Equation (4), even if only for homogeneous mixture models, would allow for a better understanding of the privacy offered by common noise mechanisms beyond the worst-case guarantees of differential privacy.

5. Controlling Mutual Information

The preceeding section provides a theoretical justification for using mutual information in our obfuscation loss. In this section we describe mutual information and shortcomings of common approximation method. We then use Corollary 3.4 to derive a mon

The mutual information between a dataset, X , and its obfuscation, \tilde{X} , can be written:

$$MI(X; \tilde{X}) = h(\tilde{X}) - h(\tilde{X}|X) \quad (5)$$

where $h(Z)$ and $h(Z|X)$ are the differential entropy of Z and of Z conditional on X respectively. Incorporating mutual information into our obfuscation loss requires approximating the entropy of the obfuscated dataset as well as the conditional entropy of the obfuscations and its input. For SGTs, Proposition 3.3 implies $h(\tilde{X}|X) = \mathbb{E}_X [h(N(y; X))]$ is the expected entropy of the learned family of Gaussians, which is straight forward to compute.

The entropy of the obfuscation itself, $h(\tilde{X})$, requires more care. Corollary 3.4 implies the obfuscations are distributed according to a high dimensional Gaussian Mixture Model (GMM). The entropy of a GMM does not have a closed form expression, so common approaches to compute it rely on bounds and approximations [30], [31], [32].

5.1. Avoiding Common Mixture Entropy Bounds

Before providing our approximation method for the mutual information loss, we first show why care must be taken it. We are not interested in just a close numerical approximation of the entropy, but rather we need an approximation which provides an adequate signal for the obfuscation mechanism to reduce the mutual information.

To demonstrate potential shortcomings of naive implementations, note that for a GMM the conditional entropy term in Equation (5) is the expected value of: $h(N(y; \mu(x), \Sigma(x))) \approx -\log(|\Sigma(x)|)$. The remaining term is the entropy of the GMM, \tilde{X} . A common upper bound for a Gaussian mixture model’s entropy is to use the entropy of the components themselves [33], [34], [35]. Specifically,

$$h(\tilde{x}) \leq \sum_i \omega_i h_y(N(y; \mu(x_i), \Sigma(x_i))) \quad (6)$$

where h_y denotes entropy computed with respect to the y variable.

Equation (6) provides a useful bound and can in some cases be a strong approximation of the entropy itself [32]. Unfortunately, it has a shortcoming that prevents it from providing a signal to minimize mutual information: there is no penalty for increasing information *between* the mixture components. This is because conditional entropy only depends on the variance of the components (i.e. $\log(|\Sigma|)$) and does not depend on the means. Using Equation (6) for our mutual information loss would result in a signal in which increasing the variance of the Gaussian components is the *only* way to decrease the mutual information. The resulting suboptimal performance is demonstrated empirically in Section 7.

5.2. Mutual Information Loss with Minibatch Monte Carlo

Algorithm 1 Minibatch Mixture Entropy Approximation

Require: Obfuscation Sampler $P(\tilde{X} | x)$, Batch sampler D , Density model $G(z; x)$, Batch size B

- 1: Sample two independent batches: $X_b \sim D$, $X'_b \sim D$
- 2: Initialize entropy estimate: $\mathcal{H} \leftarrow 0$
- 3: **for** $i = 1$ to B **do**
- 4: $x^1 \leftarrow X_b[i]$
- 5: Sample $\tilde{x}^1 \sim P(\tilde{X} | x^1)$
- 6: $x^2 \leftarrow X'_b[i]$ # Cross-sampling for mixture entropy estimation
- 7: $\mathcal{H} += -\log G(\tilde{x}^1; x^2)$
- 8: **end for**
- 9: **return** \mathcal{H}/B

We now present our mutual information based obfuscation loss for training SGTs. Though we only experiment with multivariate Gaussians for our SGT density families, the SGT is defined for any family of (regular) conditional distributions (Definition 3.2). For this reason, we present Algorithm 1 for a general family of conditional densities $G(z; x)$. The mutual information can then be written as:

$$MI(X; \tilde{X}) = \mathbb{E}_{z \sim \tilde{X}} [-\log(\mathbb{E}_X[G(z; x)])] - \mathbb{E}_X[h_z(G(z; x))]. \quad (7)$$

If we require that the h_z term have an analytic expression for the family of densities (e.g. Gaussian, Laplacian, Exponential, Cauchy), then it can be computed directly across minibatches. To approximate the remaining expectation in Equation (7) we use a minibatch Monte-Carlo approximation of the entropy, detailed in Algorithm 1.

Broadly, we sample a batch of inputs, $\{x_i\}_{i=1}^B$, and compute their obfuscations $\{\tilde{x}_i\}_{i=1}^B$. Then we *independently* sample another batch of inputs, $\{x'_i\}_{i=1}^B$, and compute the average negative log probability between the samples. The two entropy terms are then subtracted and averaged across the batch.

Note in Algorithm 1 the negative log probability term, $-\log G(\tilde{x}^1; x^2)$. This term is the negative log probability of an obfuscation with respect to an input that did not generate the obfuscation. *Thus, using Algorithm 1 ensures that the obfuscation loss provides a signal on how mixing of components contributes to the mutual information of the obfuscation.*

Since our density family are Gaussians there exists a closed form for the Monte-Carlo approximation of the mutual information loss between two batches X_b , X'_b , and the obfuscation \tilde{x}_b of X_b .

Proposition 5.1. *Let X_b be a batch sample of size, B , from our dataset X with SGT \tilde{X}_b , and let X'_b be an independently sampled batch. Further, denote $\Sigma_k = \Sigma(x_k)$ and $\mu_k = \mu(x_k)$ the estimated covariance and mean of the SGT*

model, respectively. Then the Monte-Carlo approximation of the obfuscation loss in Algorithm 1 is:

$$\mathcal{L}_O^{\text{MI}}(X_b, \tilde{X}_b; X'_b) = \frac{1}{B} \sum_{\ell', i} \log(|\Sigma_i^{-1} \Sigma_{\ell'}|) + \|\tilde{x}_i - x_{\ell'} - \mu_{\ell'}\|_{\Sigma_{\ell'}^{-1}}^2 \quad (8)$$

where x_i is an element of X_b , \tilde{x}_i is its corresponding SGT, and $x_{\ell'}$ is an element of X'_b (Terms constant in θ have been dropped).

The mutual information loss in Equation (8) can be decomposed into three pieces. The first is $\log(|\Sigma_i^{-1}|)$ which is the entropy of the components of the GMM. This term encourages higher variance in each component and is the loss that we arrive at from Bound (6). When considering the full mutual information however; the component-wise entropy is mediated by the entropy of the sampled mixture components (the $\log(|\Sigma_i^{-1} \Sigma_{\ell'}|)$ term). This encourages the covariances to be more similar to one another. This avoids having an individual component with much larger covariance than other components standing out, which would leak information. Finally, the loss includes a Mahalanobis distance term between components of the mixture, penalizing individual components whose distribution has drifted from the centers of the rest of the mixture. In this way reducing the mutual information loss, results in reducing the identifiability of any given component.

5.3. Other Obfuscation Loss Components

In this subsection we describe the two other components which contribute to the obfuscation loss function \mathcal{L}_O . In practice the mutual information loss, $\mathcal{L}_O^{\text{MI}}$, presents a difficult optimization problem and can be slow to converge or sensitive to hyperparameters. By adding additional terms to the obfuscation loss function \mathcal{L}_O we are able to both speed up convergence of the SGT and produce SGTs with higher quality obfuscations.

Absolute Cosine Similarity (AbsCos) During training of the mutual information loss it was observed that the cosine similarity between the input embeddings, X_b , and their SGT, \tilde{X}_b , slowly converged to zero. Geometrically this indicates the states of lowest mutual information which still maintain utility are orthogonal to the original input embeddings. We choose to accelerate this behavior by adding a loss component $\mathcal{L}_O^{\text{ACS}}$ which computes the absolute value of the cosine similarity between a batch of input embeddings X_b and their SGT \tilde{X}_b :

$$\mathcal{L}_O^{\text{ACS}}(X_b, \tilde{X}_b) = \left| \frac{X_b \cdot \tilde{X}_b}{\|X_b\| \|\tilde{X}_b\|} \right| \quad (9)$$

Median Norm Penalty (Norm) During training with mutual information loss, the transformed input embeddings norm tended to drift away from the original distribution of the embedding, slowing convergence. To combat this,

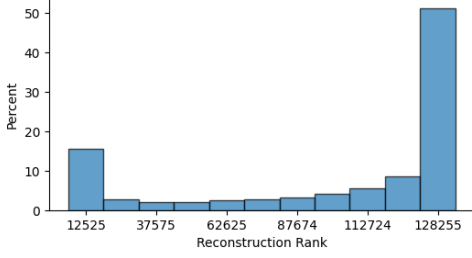


Figure 3. A histogram of reconstruction rank of obfuscations from an SGT trained with cosine-similarity obfuscation loss as opposed to considering mutual information which was discussed in Section 4. The skew towards the farthest ranks demonstrates a failure of this model not captured by the common nearest neighbor reconstructions attack.

we include a loss which penalizes the norm of the mean component the SGT:

$$\mathcal{L}_O^{\text{MNP}}(X_b) = | \|X_b + \mu_\theta(X_b)\| - T |, \quad (10)$$

where $\mu_\theta(X_b)$ is the mean of the Gaussian distribution comprising the SGT. We set T to be the median norm of the embedding vectors (i.e. $T = \text{Median}(\{\|e\| : e \in E\})$ where E is the embeddings table of \mathcal{M}). The total obfuscation loss is then the weighted sum of its constituent components:

$$\mathcal{L}_O = \alpha_1 \mathcal{L}_O^{\text{MI}} + \alpha_2 \mathcal{L}_O^{\text{ACS}} + \alpha_3 \mathcal{L}_O^{\text{MNP}}, \quad (11)$$

where each $\alpha_i \in \mathbb{R}^+$.

5.4. Demonstrative Obfuscation Losses

In this section we describe two obfuscation loss functions that we experimented with to demonstrate useful properties of the final obfuscation loss, \mathcal{L}_O . The first is the component wise Gaussian entropy (**Comp. Entr.**) from Bound (6):

$$\mathcal{L}_O^{\text{CGE}}(\tilde{X}_b) = \log(-|\Sigma_\theta(X_b)|^{-1}). \quad (12)$$

The second is a cosine-similarity penalty (**CosSim**) which does not encourage orthogonality, but rather antipodality:

$$\mathcal{L}_O^{\text{BCS}}(X_b, \tilde{X}_b) = \frac{X_b \cdot \tilde{X}_b}{\|X_b\| \|\tilde{X}_b\|}. \quad (13)$$

Both losses encourage differences between inputs and obfuscations but as we will show in the results they provide in weak or superficial protection of data.

6. Privacy Metrics

A primary measure of an obfuscations strength is its resilience to reconstruction attacks. Many reconstruction attacks are based on the nearest clean neighbors in the embedding table to the obfuscation [25], [36], [37]. In this section we define the nearest neighbor attacks, language aware attacks, and then provide empirical evidence that the distance of an obfuscation to the other embeddings in the LLM’s embedding table provides an attack vector missed by nearest neighbors.

6.1. Nearest/ N^{th} Neighbor Reconstruction

Define the reconstruction rank of an obfuscation \tilde{X} of clean embedding X_i to be

$$R(\tilde{X}) = \text{rank}_i \left(\{ \|\tilde{X} - X_k\|_2 : k \in \{1, \dots, |V|\} \} \right).$$

That is, $R(\tilde{X}) = r$ means that the clean token embedding, X , which generated \tilde{X} is the r^{th} closest token embedding to \tilde{X} .

The relative distance of an obfuscation to each clean token carries valuable information. In Figure 3 we compute the reconstruction rank of obfuscations across the Alpaca dataset and plot its histogram. This specific SGT was trained with the CosSim loss (see Equation (13)) rather than obfuscation loss \mathcal{L}_O in Equation (11). With this basic cosine-similarity loss, we see a bimodal behavior in the rank histogram. There is a small peak at the nearest neighbors followed by a larger peak at the *farthest* neighbors, implying that:

Reconstructions based on the nearest neighbors would perform worse than those which considered the farthest ones¹.

This motivates two reconstruction attacks for a more robust evaluation obfuscations:

- 1) **NN**: The nearest neighbor reconstruction of an obfuscation \tilde{X} is X_{i^*} where i^* satisfies $R(\tilde{X}, i^*) = 1$.
- 2) **MRP**: Given a maximum rank probability r over a dataset D , the max-rank-probability reconstruction (MRP) of an obfuscation \tilde{X} is X_{i^*} where i^* satisfies $R(\tilde{X}, i^*) = r$. That is, it is the r^{th} farthest clean neighbor to the obfuscation.

6.2. Language Aware Reconstruction

The state-of-the-art for reconstruction of input text from transformed embeddings is a language aware attack known as BeamClean [38]. This technique jointly estimates the noise parameters of the transformed embeddings and decodes token sequences by integrating a language-model prior.

6.3. Transformation Metrics

In this section we describe metrics based on how well an obfuscation changes a token embedding. We begin with metrics based on reconstruction attacks. The general robustness of obfuscations against these attacks is measured by the failure rate of the attacks:

- **NN-FR**: The failure rate of a nearest neighbor reconstruction attack.
- **MRP-FR**: The failure rate of a max rank probability reconstruction attack.

1. Note this assumes additional greybox information about the SGT, e.g. some knowledge of the rank histogram or ability to run an SGT on a small dataset.

- **BeamClean-FR:** The failure rate of a BeamClean attack.

Based on our empirical observations we see the peak of the rank histograms are at the nearest and farthest ranks (Figure 3) across models. For this reason we report the following metrics.

(TTR-k) The Token Transformation Rate at k over a dataset of obfuscated tokens D is

$$\text{TTR-k} = 100 \times \frac{|\{\tilde{X} \in D : k < R(\tilde{X})\}|}{|D|}$$

i.e. the percent of obfuscated tokens whose clean embedding is not in its set of k -nearest-neighbors. Concretely, the TTR-1 metric is exactly the familiar metric of nearest clean token transformation rate [25], [36], [37].

(SymTTR-k) The symmetric TTR at k over a dataset of obfuscated tokens D is

$$\text{SymTTR-k} = 100 \times \frac{|\{\tilde{X} \in D : k < R(\tilde{X}) < |V| - k\}|}{|D|}$$

i.e. the percent of obfuscated tokens whose clean embedding is not in the set of k -nearest or k -farthest neighbors.

The TTR- k and SymTTR- k metrics extend upon this nearest neighbors based metric to help understand not only whether the transformed embedding is mapping to a transformed token, but how the transformed embeddings are being distributed with respect to the embeddings which appear in the LLM’s embedding table geometrically.

Another approach for measuring the privacy leakage of reconstructions of transformed embeddings are the PII accuracy (**PII**) metrics developed in PAPILLON [4]. This technique evaluates the privacy of reconstructed text, rather than token level reconstruction, embeddings by using a LLM-as-a-judge to determine whether the reconstructions contain the private attributes of the original plaintext and ensures that the reconstructions are not transforming the text in a way in which the sensitive data can be easily recovered. We report the percent of PII-Ratio in the input text which was recovered by a reconstruction. Given an input text T and reconstructed text T^*

$$\text{PII-Ratio}(T, T^*) = \frac{\text{Percent of } T^* \text{ which is PII}}{\text{Percent of } T \text{ which is PII}}$$

6.4. Information Theoretic Metrics

In this section we describe metrics for determining the amount of information leaked by an obfuscation. These metrics do not have direct attacks associated with them, but minimizing them increases an obfuscation’s resilience across attack types.

The TTR and SymTTR metrics were *a specific* way of measuring the exploitability of the rank histogram but are not exhaustive. Fully quantifying the additional information in the rank histogram can be measured by its entropy, as it can be interpreted as measuring the vulnerability to any attack on the distribution of the rank histogram. To this end,

we report the entropy of each SGT’s rank histogram normalized by a uniform (i.e. maximally entropic) distribution:

$$\text{Hist. Entropy} = H[R(\tilde{X})]/\log(N)$$

where N is the number of clean embedding vectors.

Additionally, we report the mutual information (**MI**) of our models approximated via a Monte-Carlo approximation on the testing dataset. The mutual information is computed at the feature level rather than the whole embedding to account for the high-dimensional embeddings of the target model.

By choosing an a-priori reconstruction success rate, $1 - \delta_0$, one can invert Equation (4) to obtain an explicit bound on the reconstruction probability δ . We report these bounds (**PAC-Adv**) in Table 1 and approximate the a-priori success as $1/N$, where N is the number of token embeddings. Importantly, any mutual information value above $\log(N)$ provides an upper bound on reconstruction success of more than 100% and so falls out of the scope of Equation 4. Since our mutual information is computed at the feature level these bounds represent the robustness of reconstruction of individual features of token embeddings.

7. Experiments

The loss ablation studies use the Llama 3.2 1B Instruct as the target model [39] with the SGT models trained on the OpenOrca dataset [40]. Due to the number of losses tested and the need for hyperparameter tuning we train these models on 15% of the dataset and select the best performer based on SymTTR. Obfuscation quality is evaluated on the Alpaca dataset [41], [42]. To measure the utility of SGTs we compare a model using SGT’s obfuscated embeddings with the target model on several LLM benchmarks (Arc Challenge, Arc Easy, Openbook QA, PIQA, TruthfulQA, IFEval, MMLU 0-shot, and Hellaswag 10-shot [43], [44], [45], [46], [47], [48]). We then report the average performance difference from the target model across all benchmarks (Utility). Models were each trained on a single Nvidia A100s 80GB for roughly six hours.

Additionally, we perform evaluation on the PUPA dataset [49]. On this dataset we compute the NN and BeamClean failure rates. Additionally, we use the PAPILLON framework [49] for computing PII contained in the original text as well as the reconstructions and report the ratio (PII-Ratio). Due to the compute required for BeamClean only a subset of the SGTs trained for Llama-1B are evaluated in this way.

7.1. Model Comparisons

Obfuscation Losses. To understand the impact of the different constituent losses in (11) we train SGTs using only the mutual information loss (MI), only the absolute cosine loss (AbsCos), and the combination of the two (MI+AbsCos). Additionally, we compare with the demonstrative losses in Section 5.4 (i.e. CosSim and Comp. Entr.).

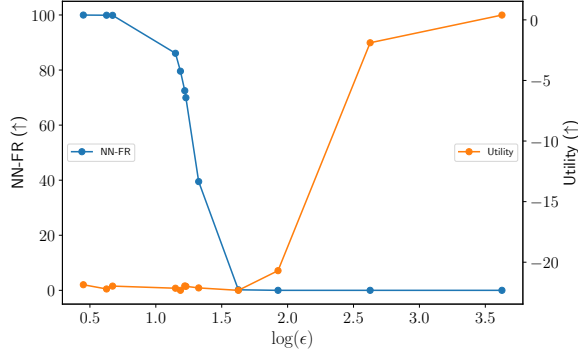


Figure 4. Constant noise (Gaussian) Utility (left, blue) and nearest-neighbor-reconstruction failure rate (right, orange) for increasing standard deviation. Constant noise degrades utility by 20 percentage points compared to the baseline model before beginning to provide any obfuscation.

Constant Noise. We experiment with constant noise obfuscations (Gaussian). For each embedding in a sequence we add a constant Gaussian and vary the strength of the noise via its variance. These are analogous to local Differential Privacy (DP) [50] and so we report the ϵ -DP value for the standard deviation of the Gaussian mechanism [51].

Large Models Fully Trained. To verify that the SGT generalize to true Large Language Models, we apply the MI+ AbsCos + Normloss to train SGT models for: Llama-3.3-70B, Qwen3-32B [52], and the Llama-70B distilled DeepSeek-R1-70B [53]. These models were trained for two epochs on the full OpenOrca. Results are reported a validation subset of OpenOrca (up to half a percent of the dataset depending on number of nodes available at the time). All large models were trained with the full MI+ AbsCos + Norm loss.

These larger models were trained up to 64 nodes of 8 Nvidia A100 80GB GPUs (hardware issues led to use of 32 nodes for some training restarts). Models were trained for up to 2 days depending on hardware configurations and model context lengths. Models were trained using FSDP2 x Tensor-Parallelism [54].

7.2. Results

Constant noise obfuscations are unable to usefully obfuscate. Figure 4 plots the Utility and nearest-neighbor reconstruction failure rate (NN-FR) for constant noise obfuscations. This unlearned noise catastrophically degrades model Utility, reducing performance by 20 percentage points compared to baseline for all ϵ values smaller than 42. At which point utility begins to recover but no privacy is provided as the NN-FR is effectively 0%.

SGTs preserve target model performance. Table 1 shows that in general the SGT does not degrade model performance. The AbsCos maintained near identical performance with the baseline, differing only by 0.01 percentage points (pp). This was followed closely by the MI (-0.72 pp) and MI+ AbsCos + Norm (-1.97 pp). The exception was Comp. Entr. which had an over 13 pp drop in performance.

Nearest Neighbor Reconstructions are an insufficient measure of obfuscation protection. The top performer in terms of NN-FR was CosSim, achieving an impressive 99.98%. However, the MRP-FR was only 3.21%. As alluded to in Section 6, this indicates the model is successfully moving the input away from its original location but is laundering information about the embeddings in other ways. This is validated by the histogram entropy (Hist. Entr.). The entropy of the rank histogram of was only 2.47% of a fully entropic histogram (i.e. a uniform distribution). This together with the less than 2% SymTTR values implies that the model learned to consistently move directly away from the input, resulting in tokens that were "different" but not truly secure. These results underscore the importance of considering information theoretic measures of privacy beyond just nearest neighbor based reconstructions.

Controlling mutual information requires it to be used in the loss. The losses which did not include the mutual information (i.e., CosSim, CosSim+ Comp. Entr., Comp. Entr.+Cat, AbsCos) provided no control over the mutual information. Table 1 shows that the geometric based losses involving cosine-similarity had dramatically large mutual information values with AbsCos having MI of over 32 quintillion. Additionally, encouraging large standard deviations of components (Comp. Entr.) provided some control over mutual information but not enough to provide PAC Privacy guarantees. Only the mutual information losses MI, MI+ AbsCos, and MI+ AbsCos + Norm were able to effectively control the mutual information with the latter providing the best protection against attacks.

The full obfuscation loss provide best protection against attacks. The Comp. Entr.+Cat showed weakest protection overall which demonstrates how the approximation of Bound (6) is not sufficient for privacy protection. The pure MI also performed poorly, though this may have been a result of under-convergence due to hyperparameter sensitivity or being a more difficult optimization problem. The absolute cosine loss (AbsCos) provided the best protection in terms of Utility and reconstructions which we tested against. However, as shown with the discrepancy between protection against NN-FR versus MRP-FR seen in the CosSim loss, protection against a specific reconstruction does not imply protection against all. Having high mutual information is a signal that there is information an adversary may be able to exploit. The obfuscation losses that controlled mutual information as these are the only that achieved bounds on feature level reconstruction probability (PAC-Adv), with the full loss achieving 12.69% bound on reconstruction.

The SGT can protect PII at the text level. Table 7.2 shows the evaluation of the SGT models on the PUPA dataset. All SGTs were able to resist BeamClean with roughly the same strength as against NN. Aside from the MI, all losses were able to maintain strong privacy on the dataset and reduce the recovered PII substantially. The MI+ AbsCos + Norm decreased recoverable PII by over 50% (46.98% vs PII-Ratio NN and 47.04% vs BeamClean). AbsCos achieved an impressive reduction of PII recovered by over 90% (roughly 1.6 PII-Ratio against both reconstructions).

TABLE 1. SGT LOSS ABLATION. MEASURING THE REDUCING IN PERFORMANCE (UTILITY), RESILIENCE TO OBFUSCATIONS (NN-FR, MRP-FR, SYMTTR-10, AND SYMTTR-100), AND INFORMATION THEORETIC PROTECTION (HIST. ENTROPY, MI, AND PAC-ADV). THE MI+ ABSCOS + NORM LOSS PROVIDES STRONG PERFORMANCE ACROSS ALL THREE CATEGORIES.

SGT Loss	Utility (\uparrow)	NN-FR (\uparrow)	MRP-FR (\uparrow)	SymTTR-10 (\uparrow)	SymTTR-100 (\uparrow)	Hist. Entr. (\uparrow)	MI (\downarrow)	PAC-Adv (\downarrow)
Comp. Entr.	-13.84	88.41	88.41	75.99	49.75	58.60	22.19	100.0
CosSim	-5.49	99.98	3.21	1.43	1.07	2.47	3.2×10^{19}	100.0
AbsCos	-0.01	94.90	94.90	93.36	92.86	87.62	6×10^9	100.0
MI	-0.72	34.43	34.43	18.96	17.39	23.96	1.48	16.39
MI+ AbsCos	-2.44	81.23	81.23	71.36	66.88	75.72	1.91	20.56
MI+ AbsCos + Norm	-1.97	93.00	93.00	90.42	88.15	91.90	1.11	12.69

TABLE 2. PERFORMANCE METRICS FOR LARGE MODELS TRAINED FOR TWO EPOCHS ON OPENORCA. THE SGT MANAGES TO MAINTAIN HIGH UTILITY WHILE PROVIDING STRONG PROTECTION.

Target Model	Utility (\uparrow)	NN-FR (\uparrow)	MRP-FR (\uparrow)	TTR-50 (\uparrow)	SymTTR-100 (\uparrow)	Hist. Entropy (\uparrow)	MI (\downarrow)	PAC-Adv (\downarrow)
Llama-3.3-70B	-0.38	98.22	98.22	98.11	98.11	80.14	0.26	3.54
DeepSeek-R1-70B	-0.46	97.39	97.39	97.23	97.23	78.05	0.28	3.74
Qwen3-32B	-0.29	94.95	94.95	94.54	66.39	67.67	0.23	3.03

TABLE 3. PUPA DATASET RESULTS. THE FAILURE RATE OF NN AND BEAMCLEAN RECONSTRUCTIONS AS WELL AS THE PII RECOVERY RATIO (PII-RATIO) ARE REPORTED.

SGT Loss	FR (\uparrow)		PII-Ratio (\downarrow)	
	NN	BeamClean	NN	BeamClean
MI	23.38	22.17	97.57	97.66
AbsCos	97.67	97.64	1.60	1.61
MI+ AbsCos	75.17	75.16	65.04	65.03
MI+ AbsCos + Norm	86.77	86.66	46.98	47.04

tions). The results show that even against more sophisticated reconstructions, the SGT is able to protect private data at the text level and is not simply changing unimportant tokens.

The SGT performs well for large models. Table 2 shows the performance of The SGT for large models. Across all three models the SGT maintains utility to within 0.5pp of their respective baseline models. The SGT for Qwen3 had the highest utility (-0.29pp); however, its SymTTR-100 was lowest (66.39%) as was its rank histogram entropy (67.67%). This suggests that further hyperparameter tuning (e.g. increasing the strength of the MI loss) could further improve performance. The remaining metrics are all higher than what was observed in the Llama-1B loss ablation likely due to the substantially longer training time allocated to these models. Overall, the protection offered by the SGT for the large models is high in both transformation metrics and information theoretic measures.

MI losses need more time to converge. We emphasize that, due to computational constraints, the loss ablation in Table 1 was done on a small subset of OpenOrca (15%) and so some of the performance advantages of AbsCos come from it being a much simpler optimization problem than $\mathcal{L}_O^{\text{MI}}$. As seen in the larger model experiments, Table 2, where the MI+ AbsCos+ Norm was trained for a full 2 epochs, the MI+ AbsCos+ Norm loss is able to achieve impressive protection even on transformation metrics. That is, when allowed to converge MI+ AbsCos+ Norm is able

to obtain strong transformation metrics with the additional benefit of having theoretical guarantees on reconstruction.

8. Conclusion

In this paper we develop an obfuscation mechanism, the Stained Glass Transform, that information theoretically protects the input embeddings of an LLM at inference time with as little as a 0.29 percentage point drop compared to the LLM’s benchmark performance on untransformed embeddings. We provided a theoretical characterization of the distribution of the SGT across a dataset as well as details of how to perform inference and train an SGT model for a target LLM. Using the characterization of the SGT’s distribution across the dataset we derived an obfuscation loss that reduces the mutual information (MI) between input embeddings and their obfuscations. We then provided experimental evidence that, while losses based on common MI bounds provide some control over MI, having the MI in the loss was the only way to obtain information theoretic guarantees of protection. Finally, we showed that the SGT generalizes across model architectures and sizes, maintaining their target model’s utility while providing strong protection.

References

- [1] R. A. Husein, H. Aburajouh, and C. Catal, “Large language models for code completion: A systematic literature review,” *Computer Standards & Interfaces*, vol. 92, p. 103917, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0920548924000862>
- [2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS ’20. Red Hook, NY, USA: Curran Associates Inc., 2020.
- [3] F. Mireshghallah, H. A. Inan, M. Hasegawa, V. Rühle, T. Berg-Kirkpatrick, and R. Sim, “Privacy regularization: Joint privacy-utility optimization in language models,” 2021. [Online]. Available: <https://arxiv.org/abs/2103.07567>

- [4] L. Siyan, V. C. Raghuram, O. Khattab, J. Hirschberg, and Z. Yu, "Papillon: Privacy preservation from internet-based and local language model ensembles," 2025. [Online]. Available: <https://arxiv.org/abs/2410.17127>
- [5] S. Martínez, D. Sánchez, A. Valls, and M. Batet, "Privacy protection of textual attributes through a semantic-based masking method," *Information Fusion*, vol. 13, no. 4, pp. 304–314, 2012, information Fusion in the Context of Data Privacy. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1566253511000157>
- [6] A. Vats, Z. Liu, P. Su, D. Paul, Y. Ma, Y. Pang, Z. Ahmed, and O. Kalinli, "Recovering from privacy-preserving masking with large language models," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 10 771–10 775.
- [7] S. Sousa and R. Kern, "How to keep text private? a systematic review of deep learning methods for privacy-preserving natural language processing," *Artif. Intell. Rev.*, vol. 56, no. 2, p. 1427–1492, May 2022. [Online]. Available: <https://doi.org/10.1007/s10462-022-10204-6>
- [8] J. X. Morris, V. Kuleshov, V. Shmatikov, and A. M. Rush, "Text embeddings reveal (almost) as much as text," 2023. [Online]. Available: <https://arxiv.org/abs/2310.06816>
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.
- [10] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:49313245>
- [11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:160025533>
- [12] R. Grabler, M. Hirschmanner, H. A. Frijns, and S. T. Koeszegi, "Privacy agents: Utilizing large language models to safeguard contextual integrity in elderly care," *parameters*, vol. 4, no. 28, p. 37, 2024.
- [13] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of large language models (llms) and llm agents: A literature review," *High-Confidence Computing*, p. 100300, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2667295225000042>
- [14] Y. Dou, I. Krsek, T. Naous, A. Kabra, S. Das, A. Ritter, and W. Xu, "Reducing privacy risks in online self-disclosures with language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 13 732–13 754. [Online]. Available: <https://aclanthology.org/2024.acl-long.741/>
- [15] P. West, A. Holtzman, J. Buys, and Y. Choi, "BottleSum: Unsupervised and self-supervised sentence summarization using the information bottleneck principle," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3752–3761. [Online]. Available: <https://aclanthology.org/D19-1389/>
- [16] P. Mai, R. Yan, Z. Huang, Y. Yang, and Y. Pang, "Split-and-denoise: protect large language model inference with local differential privacy," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.
- [17] V. Prodomo, R. Gonzalez, and M. Gramaglia, "SCIPER: Secure Collaborative Inference via Privacy-Enhancing Regularization," *IEEE Transactions on Privacy*, vol. 1, no. 01, pp. 57–68, Jan. 2024. [Online]. Available: <https://doi.ieeeecomputersociety.org/10.1109/TP.2024.3513254>
- [18] X. Shen, Y. Liu, H. Liu, J. Hong, B. Duan, Z. Huang, Y. Mao, Y. Wu, and D. Wu, "A split-and-privatize framework for large language model fine-tuning," 2023. [Online]. Available: <https://arxiv.org/abs/2312.15603>
- [19] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018. [Online]. Available: <https://arxiv.org/abs/1812.00564>
- [20] N. G. Evgenidis, N. A. Mitsiou, S. A. Tegos, P. D. Diamantoulakis, and G. K. Karagiannidis, "Split learning in computer vision for semantic segmentation delay minimization," 2024. [Online]. Available: <https://arxiv.org/abs/2412.14272>
- [21] Z. Wang, G. Yang, H. Dai, and C. Rong, "Privacy-preserving split learning for large-scaled vision pre-training," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1539–1553, 2023.
- [22] F. Mireshghallah, M. Taram, A. Jalali, A. T. T. Elthakeb, D. Tullsen, and H. Esmailzadeh, "Not all features are equal: Discovering essential features for preserving prediction privacy," in *Proceedings of the Web Conference 2021*, ser. WWW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 669–680. [Online]. Available: <https://doi.org/10.1145/3442381.3449965>
- [23] P. Vepakomma, A. Singh, O. Gupta, and R. Raskar, "NoPeek: Information leakage reduction to share activations in distributed deep learning," in *2020 International Conference on Data Mining Workshops (ICDMW)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2020, pp. 933–942. [Online]. Available: <https://doi.ieeeecomputersociety.org/10.1109/ICDMW51313.2020.00134>
- [24] L. Duan, J. Sun, J. Jia, Y. Chen, and M. Gorlatova, "Reimagining mutual information for enhanced defense against data leakage in collaborative inference," *Advances in Neural Information Processing Systems*, vol. 37, pp. 44 479–44 500, 2024.
- [25] P. Mai, R. Yan, Z. Huang, Y. Yang, and Y. Pang, "Split-and-denoise: Protect large language model inference with local differential privacy," *arXiv preprint arXiv:2310.09130*, 2023.
- [26] A. Makhdoumi, S. Salamatian, N. Fawaz, and M. Médard, "From the information bottleneck to the privacy funnel," in *2014 IEEE Information Theory Workshop (ITW 2014)*, 2014, pp. 501–505.
- [27] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *CoRR*, vol. abs/1612.00410, 2016. [Online]. Available: <http://arxiv.org/abs/1612.00410>
- [28] H. Xiao, G. E. Suh, and S. Devadas, "Formal privacy proof of data encoding: The possibility and impossibility of learnable encryption," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1834–1848.
- [29] M. Sridhar, H. Xiao, and S. Devadas, "PAC-private algorithms," 2024. [Online]. Available: <https://eprint.iacr.org/2024/718>
- [30] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for gaussian mixture random vectors," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2008, pp. 181–188.
- [31] A. Eskenazis, P. Nayar, and T. Tkocz, "Gaussian mixtures: entropy and geometric inequalities," *The Annals of Probability*, vol. 46, no. 5, pp. 2908–2945, 2018.
- [32] A. Kolchinsky and B. D. Tracey, "Estimating mixture entropy with pairwise distances," *Entropy*, vol. 19, no. 7, p. 361, 2017.
- [33] N. Schraudolph, "Gradient-based manipulation of nonparametric entropy estimates," pp. 828–837, 2004.
- [34] S. Shwartz, M. Zibulevsky, and Y. Y. Schechner, "Fast kernel entropy estimation and optimization," *Signal Processing*, vol. 85, no. 5, pp. 1045–1058, 2005, information Theoretic Signal Processing. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0165168405000216>
- [35] P. Viola, N. Schraudolph, and T. Sejnowski, "Empirical entropy manipulation for real-world problems," vol. 8, 03 1998.

- [36] X. Zhang, Y. Pang, Y. Kang, W. Chen, L. Fan, H. Jin, and Q. Yang, “No free lunch theorem for privacy-preserving llm inference,” *Artificial Intelligence*, vol. 341, p. 104293, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370225000128>
- [37] Y. Li, Z. Tan, and Y. Liu, “Privacy-preserving prompt tuning for large language model services,” *arXiv preprint arXiv:2305.06212*, 2023.
- [38] K. Kale, K. Mylonakis, J. Roberts, and S. Roy, “Beamclean: Language aware embedding reconstruction,” 2025. [Online]. Available: <https://arxiv.org/abs/2505.13758>
- [39] Z. M. Aaron Grattafiori, “The llama 3 herd of models,” 2024. [Online]. Available: <https://arxiv.org/abs/2407.21783>
- [40] W. Lian, B. Goodson, E. Pentland, A. Cook, C. Vong, and “Teknum”, “Openorca: An open dataset of gpt augmented flan reasoning traces,” <https://huggingface.co/datasets/Open-Orca/OpenOrca>, 2023.
- [41] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah, “Orca: Progressive learning from complex explanation traces of gpt-4,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.02707>
- [42] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, “Stanford alpaca: An instruction-following llama model,” https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [43] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, “Think you have solved question answering? try arc, the AI2 reasoning challenge,” *CoRR*, vol. abs/1803.05457, 2018. [Online]. Available: <http://arxiv.org/abs/1803.05457>
- [44] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, “Can a suit of armor conduct electricity? a new dataset for open book question answering,” in *EMNLP*, 2018.
- [45] M. Seo, T. Kwiatkowski, A. P. Parikh, A. Farhadi, and H. Hajishirzi, “Phrase-indexed question answering: A new challenge for scalable document comprehension,” in *EMNLP*, 2018.
- [46] S. Lin, J. Hilton, and O. Evans, “Truthfulqa: Measuring how models mimic human falsehoods,” *CoRR*, vol. abs/2109.07958, 2021. [Online]. Available: <https://arxiv.org/abs/2109.07958>
- [47] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, “Measuring massive multitask language understanding,” *CoRR*, vol. abs/2009.03300, 2020. [Online]. Available: <https://arxiv.org/abs/2009.03300>
- [48] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, “Hellaswag: Can a machine really finish your sentence?” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- [49] L. Siyan, V. C. Raghuram, O. Khattab, J. Hirschberg, and Z. Yu, “Papillon: Privacy preservation from internet-based and local language model ensembles,” *arXiv preprint arXiv:2410.17127*, 2024.
- [50] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of Cryptography*, S. Halevi and T. Rabin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.
- [51] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, p. 211–407, Aug. 2014. [Online]. Available: <https://doi.org/10.1561/04000000042>
- [52] Q. Team, “Qwen3 technical report,” 2025. [Online]. Available: <https://arxiv.org/abs/2505.09388>
- [53] DeepSeek-AI, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” 2025. [Online]. Available: <https://arxiv.org/abs/2501.12948>
- [54] W. Liang, T. Liu, L. Wright, W. Constable, A. Gu, C.-C. Huang, I. Zhang, W. Feng, H. Huang, J. Wang, S. Purandare, G. Nadathur, and S. Idreos, “Torchitan: One-stop pytorch native solution for production ready LLM pretraining,” in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=SFN6Wm7YBI>