A. How Hard is it to Find a Surrogate Dataset?

To motivate the need for data-free approaches to model extraction, we show here that an adversary relying on a surrogate dataset must ensure that its distribution is close to the one of the victim's training set. Otherwise, model extraction will return a poor approximation of the victim.

Consider a *victim* machine learning model, $\mathcal{V}$, trained on a proprietary dataset, $\mathcal{D}_{\mathcal{V}}$. The victim model reveals its predictions through either a prediction API (as is common in MLaaS) or through the deployment of the model on devices accessible to adversaries. The adversary, $\mathcal{A}$, attempts to steal $\mathcal{V}$ by querying it with a surrogate dataset, $\mathcal{D}_{\mathcal{S}}$. This surrogate dataset is assumed to be publicly available or easier to access because it does not need to be labeled.

We now perform a systematic study of the features that characterize the *closeness* of $\mathcal{D}_{\mathcal{S}}$ when compared to $\mathcal{D}_{\mathcal{V}}$. Let the private and surrogate datasets $\mathcal{D}_{\mathcal{V}}$ and $\mathcal{D}_{\mathcal{S}}$ be characterized by $\mathcal{D} = \{\mathcal{X}, P(X), \mathcal{Y}, P(Y|X)\}$ [35]. The private and surrogate datasets can vary in three ways, which we will illustrate in the following with object recognition tasks:

1. **A:** $(\mathcal{X}_{\mathcal{V}} \neq \mathcal{X}_{\mathcal{S}})$. When the inputs of $\mathcal{D}_{\mathcal{V}}$ and $\mathcal{D}_{\mathcal{S}}$ belong to different feature spaces (i.e. domain). In computer vision, for example, this can be a scenario wherein the input data (e.g., images, videos) for the datasets contain a different number of channels or pixels.
2. **B:** $(P(X_{\mathcal{V}}) \neq P(X_{\mathcal{S}}))$. While the input domain of both the surrogate and private datasets is the same, their marginal probability distribution is different. For example, when the semantic nature is different for the two datasets (images of animals, digits, etc).
3. **C:** $(P(Y_{\mathcal{V}}|X_{\mathcal{V}}) \neq P(Y_{\mathcal{S}}|X_{\mathcal{S}}))$. We consider a setting where the semantic distribution $(P(X))$ is the same, but the class-conditional probability distributions of the victim and surrogate training sets are different, e.g. when the two datasets have class imbalance.

A.1. Optimization Problem

The logit distribution of the victim network can often have a strong affinity toward the true label. To address this issue, Hinton et al. suggested scaling the logits to make the probability distributions more informative [18].

$$\mathcal{V}_i(x) = \frac{\exp\left(v_i(x)/\tau\right)}{\sum_j \exp\left(v_j(x)/\tau\right)}; \quad \mathcal{L} = -\tau^2 \mathrm{KL}\left(\mathcal{V}(x), \mathcal{S}(x)\right)$$

where $\tau > 1$ is referred to as the temperature scaling parameter. In the knowledge distillation literature, a combination of both the cross entropy and knowledge distillation loss are used to query the teacher [9]. However, model extraction we rely only on the KL divergence loss because the queries are made from a surrogate dataset which may not have any semantic binding to the true class.

A.2. Experimental Setting

Hyperparameters To search over a meaningful hyperparameter space for the temperature co-efficient τ, we refer to prior work in knowledge distillation such as [9, 18, 22] to confine the search over $\tau \in \{1, 3, 5, 10\}$. We used the SGD optimizer, and train the CIFAR10 students for 100 epochs, while the SVHN students were trained for 50 epochs. We experimented with two different learning schedules: (1) cyclic learning rate [38]; and (2) step-decay learning rate. For step-decay, we reduced the learning rate by a factor of 0.2 at 30%, 60%, and 80% of the training process. In both cases, the maximum learning rate was set to 0.1.

Experimental validation. To illustrate our argument, we next detail the relation between a task-specific surrogate dataset and the accuracy of state-of-the-art model extraction techniques. The victim models under attack are ResNet-18-8x models, their accuracy is reported in Table 4. Further details on the victim models training are provided in Section 5). We find that querying from the original dataset yields the most query-efficient and accurate extraction results. This is not surprising given that this setup corresponds to the original knowledge distillation setting. Our observations are made on both CIFAR10 and SVHN:

- **CIFAR10.** We benchmarked model extraction on 4 surrogate datasets, each reflecting a different property detailed above: CIFAR10 [21], CIFAR100 [21] (BC), SVHN [30] (AB) and MNIST [23] (AB). To ensure a fair comparison, we bound the maximum number of distinct samples queried by 50,000 while performing model extraction.
- **SVHN.** We evaluated model extraction by querying from SVHN, SVHN$_{skew}$ (C), MNIST (A) and CIFAR10 (B) as surrogate datasets. Similar to the case for CIFAR10, we cap the maximum number of distinct samples queried to 50,000.

Finally, as a control for our experiments, we also studied the extraction accuracy of the models when trained using totally random queries.

Dataset adaptation. The SVHN, CIFAR10, and CIFAR100 datasets contain 32×32 color images. To query networks trained on CIFAR10 and SVHN with images from the MNIST dataset, which contains grayscale images of size 28×28, we re-scaled the image and repeated the same input across all three RGB channels. In case of random input generation, we sample input tensors from a normal distribution with mean 0 and variance 1. Note that the teacher networks were trained on normalized datasets in the first place. Finally, in case of SVHN$_{skew}$, we supplied images from only

	Victim	CIFAR10	CIFAR100	SVHN	MNIST	SVHN$_{skew}$	Random
CIFAR10	95.5%	95.2%	93.5%	66.6%	37.2%	-	10.0%
SVHN	96.2%	96.0%	-	96.3%	89.5%	96.1%	84.1%

Table 4. Model Extraction accuracy across various surrogate datasets. Victim models were trained on the CIFAR10 and SVHN datasets, and the source accuracies are reported under the heading 'Victim'

the first 5 classes of the dataset to skew the distribution of the modified dataset.

Results. We present the results for accuracy of extracted models across various surrogate datasets for CIFAR10 and SVHN in Table 4. Recall that both the CIFAR10 and CIFAR100 datasets are subsets from the same TinyImages [40] dataset. We find that the identical source distribution was extremely useful in making relevant queries to the CIFAR10 teacher. The accuracy of the extracted model reached 93.5%, just below the 95.5% accuracy of the teacher model. However, when we used the SVHN surrogate dataset to query the CIFAR10 teacher, with a different source distribution, the model extraction performance dropped remarkably, attaining a maximum of 66.6% across all of the hyperparameters tried. In the most extreme scenario when querying the CIFAR10 teacher with MNIST–a dataset with disjoint feature space both in terms of number of pixels, and number of channels)—model accuracy did not improve beyond 37.2%.

On the contrary, we notice that the victim trained on the SVHN dataset is much easier for the adversary to extract. Surprisingly, even when the victim is queried with completely random inputs, the extracted model attains an accuracy of over 84% on the original SVHN test set. Further, nearly all surrogate datasets are able to achieve greater than 90% accuracy on the test set. We hypothesize that this observation is linked to how the digit classification task, at the root of SVHN, is a simpler task for neural networks to solve, and the underlying representations (hence, not being as complex as for CIFAR10) can be learnt even when queried over random inputs.

Given the current understanding of model extraction, we make two conclusions: (1) the success of model extraction largely depends on the complexity of the task that the victim model aims to solve; and (2) similarity to source domain is critical for extracting machine learning models that solve complex tasks. We posit that it may be nearly as expensive for the adversary to extract a CIFAR10 machine learning model with a good surrogate dataset, as is training from scratch. A weaker or non-task specific dataset may have lesser costs, but has high accuracy trade-offs.

B. Recovering logits from probabilities

The main difficulty with computing $\mathcal{L}_{\ell_1}$ is that it requires access to $\mathcal{V}$'s logits v_i, but we only have access to the probabilities of each class (i.e., after the softmax is applied to the logits). In a first approximation, the logits can be recovered by computing the log-probabilities but the resulting approximate logits are computed up to an additive constant $C(x)$ to which we don't have access in a black-box setting. This additive constant is the same for all logits but is different from one image to another. Related works on adversarial examples [4, 8] use losses that are the difference of two logits, effectively canceling out the additive constant. In our case, the logits need to be used individually which makes the ℓ_1 loss more difficult to compute in our setting.

To overcome this issue, we propose to approximate the true logits of each image x in two steps. First, compute the logarithm of the probability vector $V(x)$.

$$\tilde{v}_i(x) = \log \mathcal{V}_i(x) = v_i + C(x) \tag{8}$$

Then, compute the approximate true logits $v_i^*(x)$ by subtracting the log-probability vector with its own mean:

$$\begin{aligned} v_i^*(x) &= \tilde{v}_i(x) - \frac{1}{K}\sum_{j=1}^{K} \tilde{v}_j(x) \\ &= v_i(x) - \frac{1}{K}\sum_{j=1}^{K} v_j(x) \approx v_i(x) \end{aligned} \tag{9}$$

The second equality holds because the mean of the log-probability vector $\tilde{v}_i(x)$ is equal to the mean of the true victim logits $v_i(x)$ plus the mean of the additive constant (i.e. the $C(x)$ itself). By analyzing the mean values of the true logits from various pre-trained models—which proves to be negligible in comparison to the logit values themselves, we provide empirical evidence in Section 6.3 that this recovers a highly accurate approximation of the true logits $v_i^*(x)$.

C. Examples of Synthetic Images

Figure 7 shows 4 images from the generator towards the end of the attack on CIFAR-10. We do not observe any similarities with the images from the original training dataset.

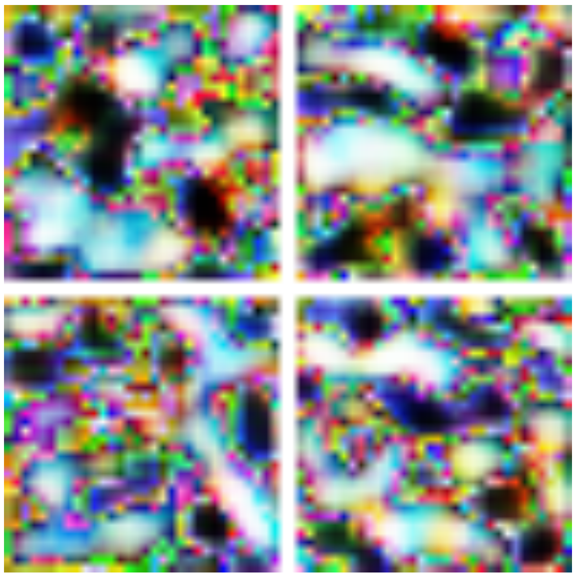
Figure 7. Four synthetic images from the generator.

D. Hypothesis 1: Justification

D.1. Preliminary results

Lemma 1. *If $\mathcal{S}(x) \in (0,1)^K$ is the softmax output of a differentiable function (e.g. a neural network) on an input x and s is the corresponding logits vector, then the Jacobian matrix $J = \frac{\partial \mathcal{S}}{\partial s}$ has an eigenvalue decomposition and all its eigenvalues are in the interval [0, 1].*

Proof. By definition:

$$\forall i \in \{1 \dots K\},\ \mathcal{S}_i = \frac{\exp(s_i)}{\sum_{k=1}^{K} \exp(s_k)}$$

For some $i, j \in \{1 \dots K\}$, if $i \neq j$:

$$\begin{aligned}\frac{\partial \mathcal{S}_i}{\partial s_j} &= -\exp(s_j) \frac{\exp(s_i)}{\left(\sum_{k=1}^{K} \exp(s_k)\right)^2} \\ &= -\mathcal{S}_i \mathcal{S}_j\end{aligned}$$

if $i = j$:

$$\begin{aligned}\frac{\partial \mathcal{S}_i}{\partial s_j} &= \frac{\exp(s_i)(\sum_{k=1}^{K} \exp(s_k)) - \exp(s_j)\exp(s_i)}{\left(\sum_{k=1}^{K} \exp(s_k)\right)^2} \\ &= \frac{\exp(s_i)}{\sum_{k=1}^{K} \exp(s_k)} - \frac{\exp(s_i)^2}{(\sum_{k=1}^{K} \exp(s_k))^2} \\ &= \mathcal{S}_i(1 - \mathcal{S}_i)\end{aligned}$$

Therefore, $\forall x$,

$$J = \frac{\partial \mathcal{S}}{\partial s} = \begin{bmatrix} \mathcal{S}_1(1-\mathcal{S}_1) & -\mathcal{S}_1\mathcal{S}_2 & \dots & -\mathcal{S}_1\mathcal{S}_K \\ -\mathcal{S}_1\mathcal{S}_2 & \mathcal{S}_2(1-\mathcal{S}_2) & \dots & -\mathcal{S}_2\mathcal{S}_K \\ \vdots & \vdots & & \vdots \\ -\mathcal{S}_1\mathcal{S}_K & -\mathcal{S}_2\mathcal{S}_K & \dots & \mathcal{S}_K(1-\mathcal{S}_K) \end{bmatrix}$$

The matrix J is real-valued symmetric, therefore it has an eigen-decomposition with real eigenvalues. $\exists \lambda_1, \lambda_2, \dots, \lambda_K \in \mathbb{R}, X_1, X_2, \dots, X_K \neq 0$ such that:

$$\forall i \in \{1 \dots K\}, JX_i = \lambda_i X_i$$

Let us prove that all eigenvalues are in the interval [0, 1]. Suppose for a contradiction that one eigenvalue λ is strictly negative. Let the associated eigenvector be:

$$X = [x_1, x_2, \dots, x_K]^T$$

The i-th component of the vector JX is:

$$\begin{aligned}[JX]_i &= \mathcal{S}_i x_i - \mathcal{S}_i \sum_{k=1}^{K} x_k \mathcal{S}_k \\ &= \mathcal{S}_i x_i - \mathcal{S}_i \langle X, \mathcal{S} \rangle\end{aligned}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product.
Since X is an eigenvector we have,

$$JX = \lambda X$$

So $\forall\ i$,

$$\begin{aligned}\mathcal{S}_i x_i - \mathcal{S}_i \langle X, \mathcal{S} \rangle &= \lambda x_i \\ x_i(\mathcal{S}_i - \lambda) &= \mathcal{S}_i \langle X, \mathcal{S} \rangle\end{aligned}$$

Since $X \neq 0$, $\exists\ i_0$ such that $x_{i_0} \neq 0$. Furthermore, λ is strictly negative so:

$$x_{i_0}(\mathcal{S}_{i_0} - \lambda) = \mathcal{S}_{i_0} \langle X, \mathcal{S} \rangle \neq 0$$

Therefore, the inner product on the right hand side is non-zero.

In addition, $\lambda < 0$ implies that $\mathcal{S}_i - \lambda > \mathcal{S}_i > 0$ so $\forall\ i, x_i$ and $\langle X, \mathcal{S} \rangle$ have the same sign. There are two cases left.

<u>If $\langle X, \mathcal{S} \rangle > 0$</u>, then $\forall\ i, x_i > 0$ and:

$$\begin{aligned}x_i(\mathcal{S}_i - \lambda) &> x_i \mathcal{S}_i \\ \mathcal{S}_i \langle X, \mathcal{S} \rangle &> x_i \mathcal{S}_i\end{aligned}$$

By summing on all i we obtain:

$$\begin{aligned}\sum_{i=1}^{K} \mathcal{S}_i \langle X, \mathcal{S} \rangle &> \sum_{i=1}^{K} x_i \mathcal{S}_i \\ \langle X, \mathcal{S} \rangle \sum_{i=1}^{K} \mathcal{S}_i &> \langle X, \mathcal{S} \rangle \\ \langle X, \mathcal{S} \rangle &> \langle X, \mathcal{S} \rangle\end{aligned}$$

Which is an **absurdity**.

If $\langle X, \mathcal{S} \rangle < 0$, then $\forall\, i, x_i < 0$ and:

$$x_i(\mathcal{S}_i - \lambda) < x_i \mathcal{S}_i$$
$$\mathcal{S}_i \langle X, \mathcal{S} \rangle < x_i \mathcal{S}_i$$

The same summation and reasoning yields an **absurdity**. We just proved that all the eigenvalues of J are non-negative.

Lastly, the trace of the Jacobian matrix $tr(J)$ equals the sum of all eigenvalues. Computing the trace yields:

$$\begin{aligned} tr(J) = \sum_{i=1}^{K} \lambda_i &= \sum_{i=1}^{K} \mathcal{S}_i(1 - \mathcal{S}_i) \\ &= \sum_{i=1}^{K} \mathcal{S}_i - \sum_{i=1}^{K} \mathcal{S}_i^2 \\ &= 1 - \sum_{i=1}^{K} \mathcal{S}_i^2 < 1 \end{aligned}$$

Since $\lambda_i \geq 0$ and $\sum_{i=1}^{K} \lambda_i < 1$, all eigenvalues must be in the interval [0, 1], which concludes the proof.

□

Lemma 2. *In the same setting as for Lemma 1, if J is the Jacobian matrix $\frac{\partial \mathcal{S}}{\partial s}$ then for any vector Z we have:*

$$\|JZ\| \leq \|Z\|$$

Proof. Let $\lambda_1, \lambda_2, \ldots, \lambda_K$ be the eigenvalues of J and $X_1, X_2, \ldots, X_K$ be the associated eigenvectors. We can decompose Z with the orthonormal eigenvector basis:

$$Z = \sum_{i=1}^{K} \alpha_i X_i$$

Computing the product JZ yields:

$$JZ = \sum_{i=1}^{K} \lambda_i \alpha_i X_i$$

The norm of the product is:

$$\begin{aligned} \|JZ\| = (JZ)^T(JZ) &= \sum_{i=1}^{K} \lambda_i^2 \alpha_i^2 \\ &\leq \sum_{i=1}^{K} \alpha_i^2 \end{aligned}$$

because $\forall\, i, |\lambda_i| \leq 1$ (see Lemma 1). Since the eigenvector basis is orthonormal we have

$$\|JZ\| \leq \sum_{i=1}^{K} \alpha_i^2 = \|Z\|$$

□

Lemma 3. *Let $\mathcal{S}(x)$ and $\mathcal{V}(x)$ be the softmax output of two differentiable functions (e.g. neural networks) on an input x, with respective logits $s(x)$ and $v(x)$. When $\mathcal{S}$ converges to $\mathcal{V}$, then $\frac{\partial \mathcal{S}}{\partial s}$ converges to $\frac{\partial \mathcal{V}}{\partial v}$.*

Proof. Recall that

$$\frac{\partial \mathcal{S}}{\partial s} = \begin{bmatrix} \mathcal{S}_1(1-\mathcal{S}_1) & -\mathcal{S}_1\mathcal{S}_2 & \ldots & -\mathcal{S}_1\mathcal{S}_K \\ -\mathcal{S}_1\mathcal{S}_2 & \mathcal{S}_2(1-\mathcal{S}_2) & \ldots & -\mathcal{S}_2\mathcal{S}_K \\ \vdots & \vdots & & \vdots \\ -\mathcal{S}_1\mathcal{S}_K & -\mathcal{S}_2\mathcal{S}_K & \ldots & \mathcal{S}_K(1-\mathcal{S}_K) \end{bmatrix}$$

If $\mathcal{V}_i(x) - \mathcal{S}_i(x) = \epsilon_i(x)$, then:

$$\begin{aligned} \mathcal{S}_i(1-\mathcal{S}_i) &= (\mathcal{V}_i + \epsilon_i)(1 - \mathcal{V}_i - \epsilon_i) \\ &= \mathcal{V}_i(1-\mathcal{V}_i) + \epsilon_i(1-\mathcal{V}_i) - \epsilon_i^2 \\ &= \mathcal{V}_i(1-\mathcal{V}_i) + o(1) \end{aligned}$$

and

$$\begin{aligned} -\mathcal{S}_i\mathcal{S}_j &= -(\mathcal{V}_i + \epsilon_i)(\mathcal{V}_j + \epsilon_j) \\ &= -\mathcal{V}_i\mathcal{V}_j - \mathcal{V}_i\epsilon_j - \mathcal{V}_j\epsilon_i - \epsilon_i\epsilon_j \\ &= -\mathcal{V}_i\mathcal{V}_j + o(1) \end{aligned}$$

Therefore, we can write:

$$\frac{\partial \mathcal{S}}{\partial s} = \frac{\partial \mathcal{V}}{\partial v} + \bar{\epsilon}(x)$$

where $\bar{\epsilon}(x)$ converges to the null matrix as $\mathcal{S}$ converges to $\mathcal{V}$. In other words we can write:

$$\frac{\partial \mathcal{S}}{\partial s} \underset{\mathcal{S} \to \mathcal{V}}{\approx} \frac{\partial \mathcal{V}}{\partial v}$$

□

D.2. Justification of the hypothesis.

Hypothesis 1 states that for two differentiable functions with softmax output $\mathcal{S}$ and $\mathcal{V}$, and respective logits s and v, the gradients of the KL divergence loss $\mathcal{L}_{KL}$ with respect to the input should be small compared to the gradients of the ℓ_1 norm loss $\mathcal{L}_{\ell_1}$ as $\mathcal{S}$ converges to $\mathcal{V}$. $\forall x \in [-1, 1]^d$:

$$\|\nabla_x \mathcal{L}_{\text{KL}}(x)\| \underset{\mathcal{S} \to \mathcal{V}}{\ll} \|\nabla_x \mathcal{L}_{\ell_1}(x)\|$$

Proof. First, we note that:

$$\sum_{i=1}^{K} \mathcal{S}_i = 1$$

implies

$$\sum_{i=1}^{K} \frac{\partial \mathcal{S}_i}{\partial x} = \mathbf{0}$$

And the same holds for $\mathcal{V}$ because both are probability distributions.

Then we compute the gradients for both loss functions:

<u>For the ℓ_1 norm loss:</u>

$$\nabla_x \mathcal{L}_{\ell_1}(x) = \sum_{i=1}^{K} sign(v_i - s_i) \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right)$$

<u>For the KL divergence loss:</u>

$$\begin{aligned}
\nabla_x \mathcal{L}_{\text{KL}}(x) &= \sum_{i=1}^{K} \frac{\partial \mathcal{V}_i}{\partial x} \log \mathcal{V}_i + 1 \frac{\partial \mathcal{V}_i}{\partial x} - \frac{\partial \mathcal{V}_i}{\partial x} \log \mathcal{S}_i - \frac{\partial \mathcal{S}_i}{\partial x} \frac{\mathcal{V}_i}{\mathcal{S}_i} \\
&= \sum_{i=1}^{K} \frac{\partial \mathcal{V}_i}{\partial x} + \sum_{i=1}^{K} \frac{\partial \mathcal{V}_i}{\partial x} \log \frac{\mathcal{V}_i}{\mathcal{S}_i} - \frac{\partial \mathcal{S}_i}{\partial x} \frac{\mathcal{V}_i}{\mathcal{S}_i} \\
&= \sum_{i=1}^{K} \frac{\partial \mathcal{V}_i}{\partial x} \log \frac{\mathcal{V}_i}{\mathcal{S}_i} - \frac{\partial \mathcal{S}_i}{\partial x} \frac{\mathcal{V}_i}{\mathcal{S}_i}
\end{aligned}$$

When $\mathcal{S}$ converges to $\mathcal{V}$, we can write

$$\mathcal{V}_i(x) = \mathcal{S}_i(x)(1 + \delta_i(x))$$

where $\delta_i(x) \underset{\mathcal{S} \to \mathcal{V}}{\to} 0$.

Since $\log 1 + x \approx x$ when x is close to 0 we can write:

$$\begin{aligned}
\nabla_x \mathcal{L}_{\text{KL}}(x) &= \sum_{i=1}^{K} \frac{\partial \mathcal{V}_i}{\partial x} \log \frac{\mathcal{V}_i}{\mathcal{S}_i} - \frac{\partial \mathcal{S}_i}{\partial x} \frac{\mathcal{V}_i}{\mathcal{S}_i} \\
&\approx \sum_{i=1}^{K} \frac{\partial \mathcal{V}_i}{\partial x} \delta_i - \frac{\partial \mathcal{S}_i}{\partial x} (1 + \delta_i) \\
&\approx \sum_{i=1}^{K} \delta_i \left(\frac{\partial \mathcal{V}_i}{\partial x} - \frac{\partial \mathcal{S}_i}{\partial x} \right) + \sum_{i=1}^{K} \frac{\partial \mathcal{S}_i}{\partial x} \\
&\approx \sum_{i=1}^{K} \delta_i \left(\frac{\partial \mathcal{V}_i}{\partial x} - \frac{\partial \mathcal{S}_i}{\partial x} \right) \\
&\approx \sum_{i=1}^{K} \delta_i \frac{\partial \mathcal{V}}{\partial v} \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \text{(Lemma 3)} \\
&\approx \frac{\partial \mathcal{V}}{\partial v} \sum_{i=1}^{K} \delta_i \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right)
\end{aligned}$$

Using Lemma 2, the norm is upper bounded by:

$$\|\nabla_x \mathcal{L}_{\text{KL}}(x)\| \leq \left\| \sum_{i=1}^{K} \delta_i \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \right\| \tag{10}$$

For the ℓ_1 norm, however, the norm is:

$$\|\nabla_x \mathcal{L}_{\ell_1}(x)\| = \left\| \sum_{i=1}^{K} sign(v_i - s_i) \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \right\| \tag{11}$$

From equation 10, we can observe that each term is negligible compared to its counterpart in equation 11: for all i we have:

$$\left\| \delta_i \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \right\| \leq \epsilon \left\| \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \right\|$$

And also $\forall i$:

$$\left\| sign(v_i - s_i) \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \right\| = \left\| \left(\frac{\partial v_i}{\partial x} - \frac{\partial s_i}{\partial x} \right) \right\|$$

Therefore, by summing these terms on the index i we can expect the KL divergence gradient to be small in magnitude compared to those of the ℓ_1 norm. However, it does not seem possible to prove this result rigorously without further assumptions on the data distribution or the mode of convergence of $\mathcal{S}$.

$\square$