Leaner Transformers Can Easily Learn to Cluster

Charlotte Park[*]
EECS, MIT
cispark@mit.edu

Kenneth Clarkson, Lior Horesh, Takuya Ito, Parikshit Ram
IBM Research
{klclarks,lhoresh}@us.ibm.com
{takuya.ito,parikshit.ram}@ibm.com

Abstract

Transformers have in-context learning capabilities, where some known learning algorithms can be executed in the forward pass through the model. Recent work shows that transformers can exactly perform Lloyd's algorithm for k-means clustering with n points in d dimensions with an embedding size $d_{\mathsf{emb}} = d + k$ (thus, requiring attention projection matrices of size $(d+k)^2$). In this work, we build upon this result in the following ways: First, we present an equally expressive but smaller transformer that executes Lloyd's algorithm with embedding size $d_{\mathsf{emb}} = (d + \lceil \log_2 k \rceil)$. Next, we train these transformers to learn the clustering algorithms given a distribution of clustering tasks, and theoretically characterize and empirically validate the factors affecting the convergence and in-distribution generalization of learning algorithms based on stochastic gradients. Finally, we probe the general clustering abilities of these learned algorithms (in the form of transformers), and try to understand situations where they succeed and fail.

1 Transformers Can Execute Clustering Algorithms In-Context

Clustering is a well-studied problem in computer science, having various applications across many fields, and possessing a nontrivial level of hardness [1, 2]. Within machine learning, clustering can help with tasks such as exploratory data analysis, unsupervised labeling, data compression, and quantization. Given a set of items, such as points in Euclidean space, the goal of k-clustering is to partition the items into k groups. Clustering is categorized through various axes such as discrete vs fuzzy or flat vs hierarchical, and have application-dependent clustering objectives such as k-means, discrete [3, 4] or soft [5, 6]; k-medoids [7, 8]; spectral [9]; kernelized [10–12]; and variants.

Here we focus on discrete (flat) Euclidean k-means clustering. Given a set of points $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ in a d-dimensional Euclidean space, with $\boldsymbol{x}_i \in \mathbb{R}^d$ for $i \in [n] \triangleq \{1, 2, \ldots, n\}$, the goal is to find k centroids/prototypes $\boldsymbol{c}_j \in \mathbb{R}^d, j \in [k]$ by solving the following problem:

$$\min_{\{\boldsymbol{c}_j \in \mathbb{R}^d, j \in [k]\}} \sum_{i=1}^{n} \min_{j \in [k]} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2 . \tag{1}$$

This has a discrete non-differentiable objective function, and so is inherently combinatorial [3, 4]. This clustering problem is known to be NP-hard even with $k = 2$, which highlights its non-trivial hardness [13], and prompts the development of various clustering algorithms [1, 2].

Beyond learning predictive models from data (like a decision tree ensemble to predict credit default or a deep convolutional neural network to label an image), an area of interest in machine learning is the ability to learn algorithms from data. In this line of work, attention-based transformers [14, 15] and recurrent neural networks [16, 17] have garnered much interest. With transformers, the ability to execute an algorithm is often studied as *in-context learning*, where a transformer is shown to be

[*]Work done as an intern at IBM Research.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

able to execute a specific learning algorithm within its forward pass [18–21]. Given this ability to express a specific (learning) algorithm in-context — thus demonstrating *algorithmic expressivity* — an important question is how and when such algorithms can be learned [20–25], and what kind of *algorithmic generalization* can be expected [21, 26]. Recently, Clarkson et al. [27] established algorithmic expressivity of a transformer with a model embedding dimensionality $d_{\mathsf{emb}} = d + k$ for Euclidean clustering, demonstrating that this architecture can exactly execute Lloyd's algorithm for k-means [4], and some variants for fuzzy [5, 6] and robust clustering [28].

Contributions. We build upon Clarkson et al. [27] to make the following contributions:

- We present a novel transformer architecture that exactly executes Lloyd's algorithm for k-means with a reduced model embedding size $d_{\mathsf{emb}} = d + \lceil \log_2 k \rceil$, an up to 75% reduction[2] in the number of transformer parameters when $k \approx d$, and much larger if $k \gg d$.
- We study a computationally cheap algorithm to train these transformers to learn to cluster in an end-to-end differentiable manner given a distribution of clustering tasks, and theoretically characterize the various factors affecting its convergence and in-distribution generalization. Specific factors of interest here are the token embedding scheme in the transformer, and the smoothing procedure to optimize the discrete objective in equation (1).

One insight from this analysis is that, for a fixed number of clusters k, and a relatively small data dimensionality d, our proposed smaller transformer with $d_{\mathsf{emb}} = d + \lceil \log_2 k \rceil$ takes longer to converge and generalize compared to the previous transformer with $d_{\mathsf{emb}} = d + k$. However, both eventually converge to the same performance, and this difference in convergence rate vanishes as d grows.

Notation. We denote the index set as $[m] \triangleq \{1, 2, \ldots, m\}$ for a positive integer $m \in \mathbb{N}$. We use lowercase letters (such as x) to denote scalars and uppercase letters (such as L) to denote functions, boldface lowercase letters ($\boldsymbol{x}$) to denote vectors, boldface uppercase ($\boldsymbol{X}$) to denote matrices, and calligraphic letters ($\mathcal{X}$) to denote sets. The ith row of a matrix $\boldsymbol{X}$ is $\boldsymbol{X}_{i,:}$, the jth column is $\boldsymbol{X}_{:,j}$, the first i rows of $\boldsymbol{X}$ is $\boldsymbol{X}_{:i,:}$. $\boldsymbol{I}_k$ is the $(k \times k)$ identity matrix. $\|\cdot\|$ denotes the ℓ_2 norm for vectors and the spectral norm for matrices. For a set of matrices $\theta = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_m\}$ (such as model parameter sets), $\|\theta\| \triangleq \max_{i \in [m]} \|\boldsymbol{W}_i\|$. For any function $F : \mathcal{X} \to \mathcal{Y}$ with learnable parameters θ, we write it as $F(\mathbf{x}; \theta)$ for any $\mathbf{x} \in \mathcal{X}$ to explicitly note the dependence on the parameters θ.

2 Expressing a Clustering Algorithm with a Leaner Transformer

The existing algorithmic expressivity of transformers exactly emulating Lloyd's algorithm [27] uses three critical components: (i) a procedure to embed the points (to be clustered) and the initial centers (to be refined), (ii) a limiting version of softmax attention with the softmax temperature tending to zero, making the softmax behave like argmax, and (iii) an update based on cross-attention and self-attention, first for the token embeddings of the points, and then for the token embeddings for the cluster centers. For a set of points $\mathcal{S} = \{\boldsymbol{x}_i, i \in [n]\} \subset \mathbb{R}^d$, represented as $\boldsymbol{X} \in \mathbb{R}^{d \times n}$, and k initial cluster centers $\{\boldsymbol{c}_j, j \in [k]\}$, represented as $\boldsymbol{C} \in \mathbb{R}^{d \times k}$, $\boldsymbol{X}$ and $\boldsymbol{C}$ are embedded in $(d + k)$ dimensions with $E_X^{\texttt{OH}} : \mathbb{R}^{d \times n} \to \mathbb{R}^{(d+k) \times n}$ and $E_C^{\texttt{OH}} : \mathbb{R}^{d \times k} \to \mathbb{R}^{(d+k) \times k}$ as follows:

$$\bar{\boldsymbol{C}}^{(0)} \leftarrow E_C^{\texttt{OH}}(\boldsymbol{C}) \triangleq \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{I}_k \end{bmatrix}, \quad \bar{\boldsymbol{X}}^{(0)} \leftarrow E_X^{\texttt{OH}}(\boldsymbol{X}) \triangleq \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(0)} \end{bmatrix}, \quad \boldsymbol{Y}^{(0)} = \mathbf{0}_{k \times n} \in \{0, 1\}^{k \times n}, \quad (\texttt{OH})$$

where the $\texttt{OH}$ denotes the one-hot encoding of the cluster indices with $\boldsymbol{I}_k$. This makes the transformer embedding dimension $d_{\mathsf{emb}} = d + k$. Beginning with $\boldsymbol{Y}^{(0)} = \mathbf{0}_{k \times n}$, the all zero $(k \times n)$ matrix, the first set of cross-attention and self-attention updates $\bar{\boldsymbol{X}}^{(t)} \to \bar{\boldsymbol{X}}^{(t+1)}$ by modifying the $\boldsymbol{Y}^{(t)}$ part of the embeddings to match the current cluster assignments via Lloyd's algorithm, that is, each point's embedding is updated to contain the one-hot encoded index of the closest center. The next set of cross and self-attention updates $\bar{\boldsymbol{C}}^{(t)} \to \bar{\boldsymbol{C}}^{(t+1)}$ by modifying the $\boldsymbol{C}^{(t)}$ part of the embeddings:

$$\bar{\boldsymbol{X}}^{(t)} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(t)} \end{bmatrix} \to \bar{\boldsymbol{X}}^{(t+1)} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(t+1)} \end{bmatrix}, \quad \bar{\boldsymbol{C}}^{(t)} = \begin{bmatrix} \boldsymbol{C}^{(t)} \\ \boldsymbol{I}_k \end{bmatrix} \to \bar{\boldsymbol{C}}^{(t+1)} = \begin{bmatrix} \boldsymbol{C}^{(t+1)} \\ \boldsymbol{I}_k \end{bmatrix}. \quad (2)$$

[2] The number of parameters in a single-headed attention-only transformer is often d_{emb}^2 for each of the query, key, value projection matrices. When $k \approx d$ and $d_{\mathsf{emb}} = (d + k)$, these matrices are of size $(2d)^2$, while $d_{\mathsf{emb}} = (d + \log_2 k)$ leads to size $(d + \log_2 d)^2$. As d grows, $(d+\log_2 d)^2 / 4d^2 \to 1/4 = 0.25$.

With the $d_{\text{emb}} = d + k$, the query/key/value projection matrices for the cross and self-attention are of $O(d_{\text{emb}}^2) \sim O((d+k)^2)$ size. To mitigate the quadratic dependence on k, we propose the following:

Definition 1 (Token embeddings)**.** *For points $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ and initial centers $\boldsymbol{C} \in \mathbb{R}^{d\times k}$, we use the embedding scheme $E_X^{\texttt{BN}} : \mathbb{R}^{d\times n} \to \mathbb{R}^{(d+\lceil \log_2 k\rceil)\times n}$ and $E_C^{\texttt{BN}} : \mathbb{R}^{d\times k} \to \mathbb{R}^{(d+\lceil \log_2 k\rceil)\times k}$:*

$$\bar{\boldsymbol{C}}^{(0)} \leftarrow E_C^{\texttt{BN}}(\boldsymbol{C}) \triangleq \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{B}_k \end{bmatrix}, \quad \bar{\boldsymbol{X}}^{(0)} \leftarrow E_X^{\texttt{BN}}(\boldsymbol{X}) \triangleq \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(0)} \end{bmatrix}, \quad \boldsymbol{Y}^{(0)} = \boldsymbol{0}_{\lceil \log_2 k\rceil \times n}, \tag{BN}$$

where $\boldsymbol{B}_k = [\boldsymbol{b}_1; \boldsymbol{b}_2; \ldots; \boldsymbol{b}_k] \in \{0,1\}^{\lceil \log_2 k\rceil \times k}$ is a binary matrix with the ith column $\boldsymbol{b}_i \in \{0,1\}^{\lceil \log_2 k\rceil}$ as the binary representation of the integer $(i-1)$ with $\lceil \log_2 k\rceil$ bits for each $i \in [k]$.

The main intuition behind this BN (binary) embedding scheme is that, while the (OH) token embedding [27] uses one-hot vectors to represent a cluster index (thus requiring k dimensions), we represent that index more compactly with a binary representation using only $\lceil \log_2 k\rceil$ dimensions. As an example, when $k = 6$, $\boldsymbol{B}_6 = \begin{bmatrix} 0,1,0,1,0,1 \\ 0,0,1,1,0,0 \\ 0,0,0,0,1,1 \end{bmatrix} \in \{0,1\}^{3\times 6}$. Then we use the following softmax attention with negative squared Euclidean distances:

Definition 2 (Attention)**.** *Given queries $\boldsymbol{Z} \in \mathbb{R}^{d_{\text{emb}}\times q}$ and keys $\boldsymbol{Z}' \in \mathbb{R}^{d_{\text{emb}}\times r}$, we define attention as:*

$$\mathcal{A}_2^{\gamma}(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})) = \boldsymbol{V}\boldsymbol{Z}' \operatorname{softmax}\left(-\frac{\gamma}{\sqrt{d_{\text{emb}}}} \operatorname{sqdist}(\boldsymbol{Q}\boldsymbol{Z}, \boldsymbol{K}\boldsymbol{Z}')\right), \tag{3}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d_{\text{emb}}\times d_{\text{emb}}}$ are the query, key and value projection matrices respectively, γ is the inverse temperature of the columnwise softmax *operation and* $\operatorname{sqdist} : \mathbb{R}^{d_{\text{emb}}\times q} \times \mathbb{R}^{d_{\text{emb}}\times r} \to \mathbb{R}^{r\times q}$ *computes the pairwise squared Euclidean distances with the (i,j)-th entry of its output given by* $\operatorname{sqdist}(\mathbf{Z}, \mathbf{Z}')_{ij} = \left\|\mathbf{Z}_{:,j} - \mathbf{Z}'_{:,i}\right\|^2$ *for some matrices $\mathbf{Z} \in \mathbb{R}^{d_{\text{emb}}\times q}, \mathbf{Z}' \in \mathbb{R}^{d_{\text{emb}}\times r}$.*

This distance-based attention [29] was used to establish algorithmic expressivity for clustering, though this can be replaced with dot-product attention if we also utilize a single-hidden-layer feedforward network in the transformer [27]. Based on the above, we consider the following transformer updates:

Definition 3 (Per-layer updates)**.** *In the $(t+1)^{th}$ transformer layer, with parameters $\Theta = \{(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 1, \ldots, 4\}$ shared across all layers, the layer input token embeddings $(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{C}}^{(t)})$ are updated with the attention operation in Definition 2 as follows:*

$$\begin{aligned} \bar{\boldsymbol{X}}^{(t+1)} &= \bar{\boldsymbol{X}}^{(t)} + \mathcal{A}_2^{\gamma}(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{C}}^{(t)}; (\boldsymbol{Q}_1, \boldsymbol{K}_1, \boldsymbol{V}_1)) + \mathcal{A}_2^{\gamma}(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{X}}^{(t)}; (\boldsymbol{Q}_2, \boldsymbol{K}_2, \boldsymbol{V}_2)), \\ \bar{\boldsymbol{C}}^{(t+1)} &= \bar{\boldsymbol{C}}^{(t)} + \mathcal{A}_2^{\gamma}(\bar{\boldsymbol{C}}^{(t)}, \bar{\boldsymbol{X}}^{(t+1)}; (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) + \mathcal{A}_2^{\gamma}(\bar{\boldsymbol{C}}^{(t)}, \bar{\boldsymbol{C}}^{(t)}; (\boldsymbol{Q}_4, \boldsymbol{K}_4, \boldsymbol{V}_4)), \end{aligned} \tag{4}$$

where the first operation on the right-hand-side after the residual connection in each of the update steps corresponds to cross-attention, while the second operation is self-attention.

Given this architecture, we can show the following algorithmic expressivity result:

Theorem 1. *Given a set of points $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ to be clustered into k disjoint sets and initial cluster centers $\boldsymbol{C} \in \mathbb{R}^{d\times k}$, both embedded with $(E_C^{\texttt{BN}}, E_X^{\texttt{BN}})$ (Definition 1) to give us a latent dimension $d_{\text{emb}} = d + \lceil \log_2 k\rceil$, there exists parameters $\Theta = \{(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 1, \ldots, 4\}$ for the transformer (Definition 3) with the inverse softmax temperature $\gamma = \infty$ such that the tth transformer layer output exactly matches the output of Lloyd's algorithm after t iterations for any $t \geq 1$.*

The precise Lloyd's algorithm is presented in Algorithm 2. The proof follows a construction similar to that of Clarkson et al. [27] with modifications to handle the binary version of the cluster indices instead of the one-hot version. See Appendix B for details and the proof. This architecture now requires $O((d + \log_2 k)^2)$ parameters reducing the dependence on k from $\Omega(k^2)$ to $O((\log_2 k)^2)$. While the number of parameters is not really a factor in the expressivity results, it can affect computational cost of the inference (which in turn affects the training time). Figure 1 compares the computational costs (in terms of forward pass time, backward pass time, GPU memory usage) of the models with $d_{\text{emb}} = d + k$ for (OH) and our proposed $d_{\text{emb}} = d + \lceil \log_2 k\rceil$ for (BN) for varying values of (d, k). While the advantage of the proposed model is clear across the board in terms of memory consumption, the gain in terms of forward/backward pass time is only significant when values of d and k are large enough. Table 1 profiles a different setup where we fix d and vary k, and see that, once k is sufficiently larger than d, the gains from the proposed (BN) can be very significant even in terms of the forward and backward pass runtimes.

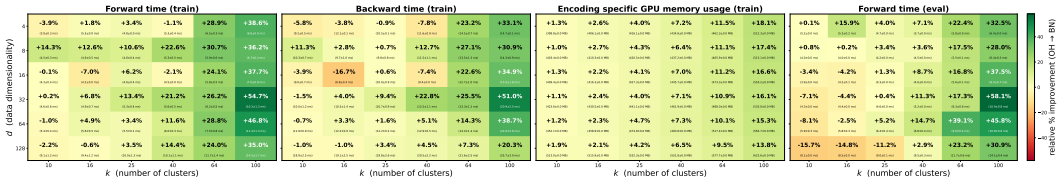


Figure 1: Comparison of the proposed (BN) embedding against the existing (OH) with varying $d \in \{4, 8, 16, 32, 64, 128\}$ and $k \in \{10, 16, 25, 40, 64, 100\}$. Each heatmap reports the relative improvement (in %) of (BN) over (OH) for each of following metrics: (i) *Forward time (train)*: forward pass time during training, (ii) *Backward time (train)*: backward pass time, (iv) *Encoding specific GPU memory usage*: peak GPU memory usage at training (less the usage from the self-attention matrices which does not depend on d_{emb}), and (iv) *Forward pass (eval)*: forward pass time with the computation graph disabled (via `model.eval()`). Each cell in the (6×6) grid in the heatmap lists the relative gain (or degradation), with actual metrics for (OH) also listed in the cell as (mean±std) for completeness; darker greens denote larger gain values. All runs use 32 clustering problems with $n = 1024$ points, and numbers are generated with 30 repetitions (10 warm-up rounds). In terms of memory usage, (BN) shows gains across the board, ranging from +1-15%. For the computation times (forward (train), backward, and forward (eval)), (BN) does not always show a positive gain for smaller values of k, especially when the baseline (OH) runtimes is already quite small (< 10ms). However, once k is large enough, the gains can be quite significant, reaching over 50% in some cases.

3 Understanding Learning Convergence and Generalization

Given the above architecture, we will train the transformers to learn a clustering algorithm given a distribution of clustering tasks. In any learning setup, we need training data, a model defining the "forward pass" or inference, and a loss that quantifies the difference between the (current) model output and the desired one. As we aim to learn a clustering *algorithm*, the training data will be a set of k-clustering tasks in d dimensions, each with a pair $(\boldsymbol{X}, \boldsymbol{C})$ of points to be clustered $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ and initial cluster centers $\boldsymbol{C} \in \mathbb{R}^{d \times k}$, generated from a distribution $\mathcal{D}$ of clustering tasks (for example, each $\boldsymbol{X}$ is randomly sampled from a different mixture of Gaussians, and $\boldsymbol{C}$ is random k-subset of $\boldsymbol{X}$). The model $F^{\gamma}_{d:k} : \mathbb{R}^{d \times n} \times \mathbb{R}^{d \times k} \to \mathbb{R}^{d \times k}$ is a single-layer version of the one presented in Definition 1, 2 and 3 with a finite γ (in all our experiments, we set $\gamma = 1$), and parameterized with $\Theta = \{(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 1, \ldots, 4\}$.

For a clustering task $(\boldsymbol{X}, \boldsymbol{C})$, the initial token embeddings $(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)})$ will be generated using embedding functions (E_C, E_X) – either with $(E^{\texttt{OH}}_C, E^{\texttt{OH}}_X)$ from (OH) or $(E^{\texttt{BN}}_C, E^{\texttt{BN}}_X)$ from (BN) – and passed through a single transformer layer to get $(\bar{\boldsymbol{X}}^{(1)}, \bar{\boldsymbol{C}}^{(1)})$, and the model output $F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)$ is the first d rows $\bar{\boldsymbol{C}}^{(1)}_{:d,:}$ of $\bar{\boldsymbol{C}}^{(1)}$ — this choice implies that we are training the model to perform **single-step clustering**. Note that we learn a separate model for each value of d and k (denoted by the $d{:}k$ subscript in $F^{\gamma}_{d:k}$); the same model can be applied to clustering tasks with different values of n.

Table 1: Comparison between proposed (BN) and existing (OH) under the setting of figure 1 but with k varied in $[6, 1000]$ (column 1) for a fixed data dimensionality $d = 32$. We note the actual forward (column 3) and backward pass (column 4) times (with the speedup of (BN) over (OH)), and the peak GPU memory usage (column 5) as described in figure 1 (and the reduction from using (BN)). We also note the additional embedding dimensions $(d_{\text{emb}} - d)$ for each value of k for the two embeddings (column 2), and the significant reduction in the additional embedding dimensions lead to the significant reductions in the runtimes once k is sufficiently larger than d with up to 12×/9× speedup in forward/backward pass, and almost 70% reduction in memory usage.

k	$(d_{\text{emb}} - d)$ OH / BN	Forward pass (ms) OH / BN [speedup]	Backward pass (ms) OH / BN [speedup]	GPU memory usage (MB) OH / BN [reduction]
6	6 / 3	4.25 / 4.83 [0.88×]	9.53 / 10.66 [0.89×]	419.93 / 416.29 [0.9%]
25	25 / 5	5.24 / 4.99 [1.05×]	11.21 / 11.01 [1.02×]	441.08 / 423.23 [4.0%]
100	100 / 7	9.76 / 4.52 [2.16×]	20.21 / 9.93 [2.03×]	532.96 / 447.04 [16.1%]
400	400 / 9	38.68 / 7.89 [4.90×]	66.13 / 16.47 [4.02×]	1095.60 / 531.02 [51.5%]
1000	1000 / 10	193.07 / 15.90 [12.14×]	308.28 / 32.33 [9.54×]	3930.34 / 1207.75 [69.3%]

Given the model output $\bar{\boldsymbol{C}}(\Theta) \triangleq F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)$, we can compute the per-task clustering objective $\mathcal{L} : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times k} \to \mathbb{R}$ using the per-point clustering objective $L : \mathbb{R}^{d} \times \mathbb{R}^{d\times k} \to \mathbb{R}$ as

$$\mathcal{L}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)) = \sum_{i=1}^{n} L(\boldsymbol{X}_{:,i}, F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)). \quad (5)$$

As a concrete example, we will use the discrete k-means objective $L(\boldsymbol{X}_{:,i}, \bar{\boldsymbol{C}}(\Theta)) = \min_{j\in[k]} \left\|\boldsymbol{X}_{:,i} - \bar{\boldsymbol{C}}(\Theta)_{:,j}\right\|^2$. With in-context supervised learning, the learning loss, such as mean-squared-error or negative cross-entropy, is continuous and thus differentiable, while the k-means objective in equation (5) is not. Hence, we will consider a smoothed surrogate objective. While $\min$ is not a differentiable function, various regularized forms of $\min$, with a convex regularizer $\Omega : \Delta^k \to \mathbb{R}$ and a penalty $\tau > 0$, are differentiable, and give us an upper-bound for the per-point k-means objective $\tilde{L}^{\tau}_{\Omega} : \mathbb{R}^{d} \times \mathbb{R}^{d\times k} \to \mathbb{R}$ of the following form (see Appendix H) for any $\boldsymbol{x} \in \mathbb{R}^d$ and cluster centers $\boldsymbol{C} \in \mathbb{R}^{d\times k}$ with simplex weights $\{\lambda_j^{\Omega_\tau} \geq 0\}_{j\in[k]}$ where $\sum_{j=1}^{k} \lambda_j^{\Omega_\tau} = 1$:

$$L(\boldsymbol{x}, \boldsymbol{C}) \triangleq \min_{j\in[k]} \|\boldsymbol{x} - \boldsymbol{C}_{:,j}\|^2 \leq \sum_{j=1}^{k} \lambda_j^{\Omega_\tau} \|\boldsymbol{x} - \boldsymbol{C}_{:,j}\|^2 \triangleq \tilde{L}^{\tau}_{\Omega}(\boldsymbol{x}, \boldsymbol{C}),$$

Here, $\tau = 0$ implies no regularization (and thus the exact minimum). While a common choice for the weights $\{\lambda_j^{\Omega_\tau}\}_{j\in[k]}$ is the softmax function (obtained by a negative Shannon entropy or NE-regularized $\min$, with τ serving as the softmax temperature), a tighter bound can be obtained with the sparsemax function which uses the L2-norm regularization leading to sparse weights [30, 31].

With this smoothed loss surrogate $\tilde{L}^{\tau}_{\Omega}$, we have $L(\boldsymbol{X}_{:,i}, \bar{\boldsymbol{C}}(\Theta)) \leq \tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \bar{\boldsymbol{C}}(\Theta))$; equality holds at $\tau = 0$. However, note that as $\tau \to 0$, $\tilde{L}^{\tau}_{\Omega}$ becomes progressively less smooth (Proposition 2 in Appendix H). Given this continuous objective, we train the transformer by solving the following:

$$\min_{\Theta} \mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim\mathcal{D}} \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)), \quad \text{where} \quad \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, \bar{\boldsymbol{C}}(\Theta)) \triangleq \sum_{i=1}^{n} \tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:i}, \bar{\boldsymbol{C}}(\Theta)). \quad (6)$$

In practice we will utilize a finite approximation of the above expectation $\mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim\mathcal{D}}$ by sampling clustering tasks from $\mathcal{D}$. The training procedure is detailed in Algorithm 1. Note that while we focus on the discrete k-means objective, this learning scheme is applicable to any discrete clustering objective L for which we can find a reasonably smoothed surrogate objective $\tilde{L}$ such that $L \leq \tilde{L}$. In each training step, we sample a batch of clustering tasks, compute the surrogate loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ of the model outputs $F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)$, and use the batch-loss-gradient to update the model.

Algorithm 1 Learning to k-cluster in d dimensions using a transformer $F^{\gamma}_{d:k}$ with learnable parameters Θ, given a distribution $\mathcal{D}$ of clustering tasks, an $\min$-regularizer $\Omega : \Delta^k \to \mathbb{R}$ with penalty $\tau > 0$, and optimization hyperparameters of learning rate η, task batch size B and M training steps.

Initialize $F^{\gamma}_{d:k}$ parameters Θ
for $m = 1, \ldots, M$ **do**
  `// Compute batch loss`
  Initialize batch loss $l(\Theta) = 0$
  for $b = 1, \ldots, B$ **do**
    Sample task $(\boldsymbol{X}, \boldsymbol{C}) \sim \mathcal{D}$
    $l(\Theta)$ += $\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta))$
  `// update model params`
  $\Theta \leftarrow \texttt{Update}(\Theta, \nabla_{\Theta} l(\Theta), \eta)$
return Θ

We build upon existing analysis to discuss the effect of different choices in the learning procedure (namely the regularizer Ω, the penalty τ, and the token embedding) on the training convergence. For any α-Lipschitz and β-smooth learning loss, various results establish guarantees for SGD (stochastic gradient descent) based learning [32, 33]. For a finite-sum nonconvex objective, such as the one in equation (6), M steps of SGD with a time-dependent learning rate $\eta_m, m \in [M]$ converges to ϵ-stationarity with

$$\epsilon \sim O\left(\beta\alpha^2 \frac{\sum_{m=1}^{M} \eta_m^2}{\sum_{m=1}^{M} \eta_m}\right), \quad \text{with } \eta_m = \begin{cases} \eta/m & \Rightarrow \epsilon \sim O(\beta\alpha^2/\log M) \\ \eta/\sqrt{m} & \Rightarrow \epsilon \sim O(\beta\alpha^2 \log M/\sqrt{M}) \end{cases}, \quad (7)$$

for some initial learning rate $\eta > 0$. The (α, β) properties of the loss function, along with the learning rate scheduling, drive the convergence rate guarantees. However, (α, β) also affect generalization. Given a set $\mathcal{S}$ of clustering tasks sampled from $\mathcal{D}$, model parameters Θ learned with M steps of SGD using a time dependent learning rate $\eta_m = \eta/m$, the generalization error (the gap between the true

and estimated loss) can be bounded by combining Hardt et al. [33, Theorems 2.2 and 3.12]:

$$\left|\mathbb{E}_{\mathcal{S}}\left[\mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim\mathcal{D}}\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X},\bar{\boldsymbol{C}}(\Theta)) - \frac{1}{|\mathcal{S}|}\sum_{(\boldsymbol{X},\boldsymbol{C})\in\mathcal{S}}\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X},\bar{\boldsymbol{C}}(\Theta))\right]\right| \leq \underbrace{\frac{1+\frac{1}{\beta\eta}}{|\mathcal{S}|-1}(2\eta\alpha^2)^{\frac{1}{\beta\eta+1}}M^{\frac{\beta\eta}{\beta\eta+1}}}_{\varepsilon}. \quad (8)$$

The above provides a bound on the smoothed objective $\tilde{\mathcal{L}}^{\tau}_{\Omega}$. With ε denoting the above gap bound, we have the following *in-distribution* generalization bound for the non-differentiable k-means objective:

$$\mathbb{E}_{\mathcal{S}}\left[\mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim D}\mathcal{L}(\boldsymbol{X},F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta))\right] \leq \mathbb{E}_{\mathcal{S}}\left[\frac{1}{|\mathcal{S}|}\sum_{(\boldsymbol{X},\boldsymbol{C})\in\mathcal{S}}\mathcal{L}(\boldsymbol{X},F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta))\right] + C^{\tau}_{\Omega} + \varepsilon, \quad (9)$$

where $0 \leq C^{\tau}_{\Omega} \triangleq \max_{(\boldsymbol{X},\hat{\boldsymbol{C}})}\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X},\hat{\boldsymbol{C}}) - \mathcal{L}(\boldsymbol{X},\hat{\boldsymbol{C}})$ quantifies the gap between the true non-differentiable objective $\mathcal{L}$ and smoothed $\tilde{\mathcal{L}}^{\tau}_{\Omega}$, with smaller C^{τ}_{Ω} implying tighter generalization bounds, and can be controlled with τ. Note that, under certain conditions, some regularized forms of $\min$ can ensure that this gap $C^{\tau}_{\Omega} = 0$; as an example, if there is usually a sufficiently large margin between the inter-cluster distances and intra-cluster distances, the sparsemax based upperbound is tight. If this gap C^{τ}_{Ω} is large (relative to the other terms), the guarantee is weak. Thus, it is important to ensure that this gap C^{τ}_{Ω} is sufficiently small by using a tighter smoothed upper-bound of the clustering loss. We will evaluate its effect empirically. However, the penalty τ also plays a role in the Lipschitz constant α of the smoothed $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ as we show in the following:

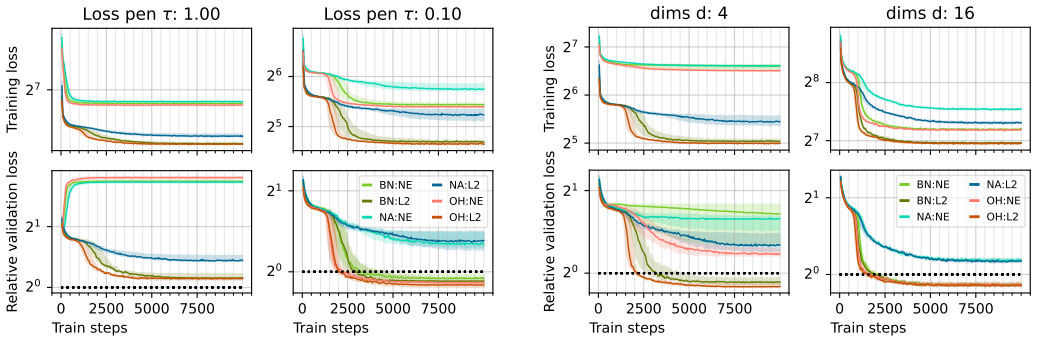
Theorem 2 (informal, see Theorem 5 in Appendix I for a detailed version)**.** *Assume bounded points $\boldsymbol{X} \in [0,1]^{d\times n}$ and initial cluster centers $\boldsymbol{C} \in [0,1]^{d\times k}$ for all clustering tasks $(\boldsymbol{X},\boldsymbol{C}) \sim \mathcal{D}$, and bounded learnable model parameters Θ, Θ'. Let d_E denote the maximum squared norm of the initial center embeddings, with $\left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\|^2 \leq d_E$ for all $j \in [k]$ for any $\boldsymbol{C}$, and let the Ω-regularized* argmax *with penalty $\tau > 0$ be (ω_τ/τ)-input-stable. Then the smoothed clustering loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ is α-Lipschitz as follows:*

$$\left|\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X},F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta)) - \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X},F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta'))\right| \leq \alpha\,\|\Theta-\Theta'\|,$$
$$\text{where } \alpha \sim O\left(\frac{kn\omega_\tau}{\tau}\cdot\max\{1,\gamma^2\}\cdot\sqrt{d_E}\cdot\max\left\{1,\left(\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)^2\right\}\cdot\exp\left(\gamma\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)\right) \quad (10)$$

This result highlights how the Lipschitz constant α of the smoothed loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ depends on (i) the regularized $\min$ penalty τ as $1/\tau$, (ii) the regularizer Ω through ω_τ, (iii) the attention inverse temperature γ, and (iv) the token embedding scheme through the d_E and d_{emb} terms, where $d_E = d+1, d_{\mathsf{emb}} = d+k$ with (OH), and $d_E = d_{\mathsf{emb}} = d + \lceil\log_2 k\rceil$ with (BN). Thus, the (OH) scheme of Clarkson et al. [27] has a better Lipschitz constant than the proposed (BN), thereby presenting a tradeoff between fewer parameters and improved convergence. However, for large d (relative to k), this difference is limited as d_{emb} and d_E scale as d for both token embeddings. This result in conjunction with those in section 2 (figure 1 and table 1) highlight a tradeoff between the architecture size and the actual training times — *the faster forward/backward passes of the leaner architecture with* (BN) *token embeddings do not necessarily translate to faster model training to convergence*.

To understand how this theoretical characterization aligns with empirical behaviour, we train this model with various configurations. The training clustering tasks contains points sampled from a mixture of isotropic normal distributions (data generation detailed in Appendix D). All experiments are repeated with 10 different random seeds (which affect model initialization, and the tasks sampled), and we aggregate the results over these 10 trials, and present the median (the curve) and the inter-quartile ranges (the translucent ribbon around the curve) of the reported metrics. We use the Adam optimizer with a learning rate of $\eta = 0.01$ for $M = 10000$ steps with a task batch size $B = 32$. Note that we *do not utilize the standard practice of gradient clipping* in our experiments to highlight the ease of training the transformer for clustering. Extended ablation of experimental configurations are presented in Appendix F.

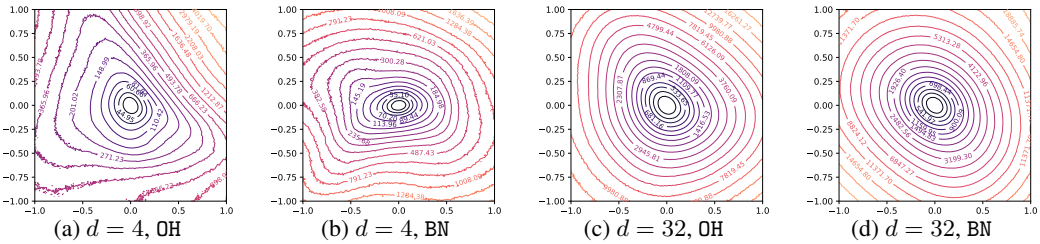
Figure 2 shows the training convergence and validation performance. In almost all cases, the training loss converges quite monotonically, and the validation performance usually tracks the training. In



(a) Varying $\tau \in \{1, 0.1\}$ with $d = 4$ dimensions. (b) Varying $d \in \{4, 16\}$ with $\tau = 0.25$.

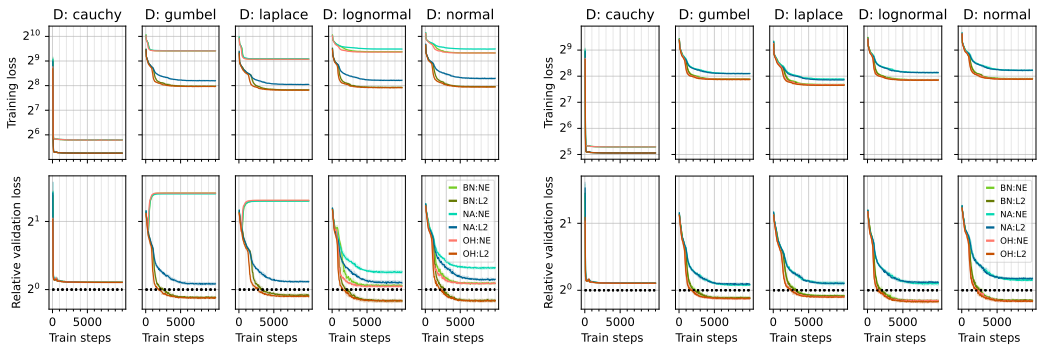
Figure 2: **Effect of data dimensionality d, $\min$-regularizer Ω with penalty τ, token embedding (E_C, E_X) on training convergence and generalization.** We show the convergence of (i) the smoothed training loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ (top row in each figure) and (ii) the *relative* non-differentiable clustering loss $\mathcal{L}$ on held-out validation clustering tasks (bottom row) for multiple configurations; the dotted black line at $y = 1$ (bottom row) corresponds to the performance of a single Lloyd's iteration on the validation tasks. See Appendix F for details and ablations. *In all cases, lower values on the vertical axes are better.* We use clustering tasks with $n = 512$ and $k = 6$. The initial learning rate for Adam is 0.01 (ablation in figure 11) with no gradient clipping. In each plot, all 6 curves correspond to the same set of clustering tasks used for training, and show the behaviour of NE-regularized $\min$ (softmax) vs L2-regularized $\min$ (sparsemax), jointly with the effect of (E_C, E_X) — (OH) vs (BN) vs no token embedding (NA). In figure 2a, we fix the data dimensionality $d = 4$ and vary the $\min$-regularization penalty τ (ablation in figure 10), and in figure 2b, we fix $\tau = 0.25$ and vary d (ablation in figure 14).

figure 2a, we vary the temperature τ of the smoothed loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$, and observe that for any value of τ, the *L2-regularized* $\min$ *always produces lower training losses, and thus tighter upper-bounds (and lower values of* C^{τ}_{Ω}*).* The worst upper-bounds with the NE-regularizer are reflected in the validation performance at $\tau = 1$, where *the learned model does not perform well on the unseen validation clustering tasks for any of the token embedding schemes* (the validation loss increases during training). In contrast, the L2-regularization is able to reduce the clustering loss on unseen tasks as training progresses, even though it is unable to match Lloyd's at $\tau = 1$. This highlights the effect of the tightness of the smoothed upper-bound discussed in equation (9). At $\tau = 0.1$, both forms of regularized $\min$ (NE and L2) produce tighter bounds, and *the learned model is able to significantly outperform Lloyd's for single-step clustering on the unseen validation tasks*, except with the NA embedding scheme (which does show reduction in validation loss though is unable to match Lloyd's). However, training converges faster for higher values of τ as implied by equation (10) though it appears to converge to a worse solution. Thus *lowering τ to tighten the upper-bound can slow down training convergence*. Finally, for all values of τ, *the training converges faster with* (OH) *compared to* (BN), though *both finally converge to a similar loss and are able to outperform single-step Lloyd's*. This effect is visible for both L2-regularized and NE-regularized $\min$ for $\tau = 0.1$. This aligns with equation (10), where (OH) has a smaller Lipschitz constant than (BN). In figure 2b, we vary the data dimensionality d, and see that the results continue to highlight the expressivity of (OH) and (BN) (and the lack thereof with NA), and the tighter upper-bounds of the L2-regularized $\min$ as discussed for figure 2a. We also see that, for a fixed $\tau = 0.25$ (and controlling for other factors), *increasing d removes the difference between the convergence of* (OH) *vs* (BN). We compare the convergence in terms of actual training times in figure 17 in appendix F to understand the interplay of the faster forward/backward pass and slower convergence of the proposed (BN).



(a) $d = 4$, OH (b) $d = 4$, BN (c) $d = 32$, OH (d) $d = 32$, BN

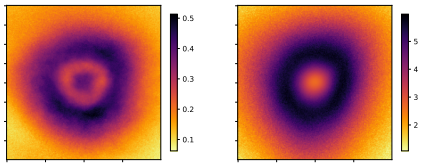
Figure 3: **Loss surfaces for different dimensionalities ($d = 4$ vs $d = 32$), token embeddings (OH) vs (BN) with the L2-regularized $\min$.** We fix $k = 6$ and vary d to visualize its effect. We use the procedure in Li et al. [34] with filterwise normalization and 2 random vectors in the parameter space.



(a) $\tau = 1$. (b) $\tau = 0.1$.

Figure 4: **Effect of data distribution.** The above figures are as described in figure 2. We train on tasks with $n = 512, d = 32, k = 10$ generated from the noted distributions. The validation loss is computed on clustering tasks generated from the training distribution (see ablation in Figure 16).

We visualize the loss surfaces [34] for the L2-regularized smoothed loss in figure 3, and see that in all cases, *the loss surfaces appear fairly unimodal without any complicated landscapes*, explaining why these models train so easily with apparent monotonic convergence. For the smaller $d = 4$, *the loss surfaces of* (`OH`) *and* (`BN`) *are markedly different* aligning with the significant difference in their d_E, d_{emb} terms in Theorem 2. For the larger $d = 32$, *the loss surfaces of* `OH` *and* `BN` *are quite similar*, as Theorem 2 suggests given the smaller relative differences between their d_E, d_{emb} terms.



(a) `normal` (b) `cauchy`

Figure 5: **Training loss gap surfaces.** We visualize the landscape of the loss gap $(\tilde{\mathcal{L}}^{\tau}_{\Omega} - \mathcal{L})$ between the smoothed $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ and the k-means loss $\mathcal{L}$ at $\tau = 0.1$.

In figure 4, we use clustering tasks with the points $\boldsymbol{X}$ generated from a mixture of isotropic distributions, and consider 5 different distribution families: `cauchy`, `gumbel`, `laplace`, `lognormal`, `normal`. We evaluate two values of τ. In all cases, the models are trained and evaluated on the same family. First, we see that our proposed training scheme is unable to generalize when trained with data from the `cauchy` distribution — the training seems to converge very quickly, but the performance on unseen clustering tasks are significantly worse than that of the single Lloyd's iteration. For the remaining families with the larger $\tau = 1$, we see that the L2-regularized $\min$ with the (`OH`) and (`BN`) token embeddings generalize well, and significantly outperform a single Lloyd's iteration; the NE-regularization is unable to match a single Lloyd's iteration, highlighting the benefit of the tighter loss upper-bound. When $\tau = 0.1$, both NE-regularized and L2-regularized $\min$ generalize equally well. Overall, we see that we learn generalizing transformers for all considered distributions except with `cauchy`; this failure remains an open question. We hypothesize that it is related to the larger gap between the smoothed upper-bound $\tilde{\mathcal{L}}$ and the discrete k-means objective $\mathcal{L}$ over the parameter space, which we visualize for `cauchy` and `normal` in figure 5 — the gap is at most 0.5 for `normal`, but has an order-of-magnitude larger range with `cauchy`, with the largest gap being greater than 5.

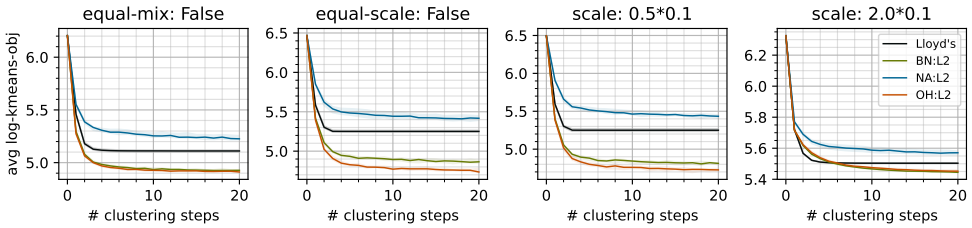


Figure 6: **Generalization to varying distributional properties.** Models are trained with $n = 512, d = 32, k = 10, \tau = 0.1$, L2-regularized $\min$ with `OH/BN/NA` token embeddings, and evaluated on $T = 20$ step clustering with tasks from mixtures of `normal` distributions. Training tasks are from equally weighted mixtures of isotropic `normal` distributions with a per-dimensional scale of 0.1. We evaluate generalization in 4 scenarios: **[Left]** *unequally weighted* mixture of isotropic distributions, **[Left center]** equally weighted mixture of *anisotropic* distributions, **[Right center]** equally weighted mixture of isotropic distributions with a *per-dimension scale smaller (half) than that during training*, and **[Right]** same as **Right center** but *with a scale larger (double) than that during training*.

4 Probing Algorithmic Generalization Beyond Training Distributions

In this section, we evaluate the trained transformers (which have implicitly learned a single-step clustering algorithm) on varied distributions of clustering tasks. First, in figure 7, we present the performance of the models (trained for single-step clustering) for multiple clustering steps. We consider two versions of multi-step clustering, and compare them to Lloyd's iterations with the same initial centers. See Appendix C for details. The results clearly show that naively applying the attention updates in equation (4) iteratively does not work beyond a single-step, as should be expected since the models are trained for single-step clustering. After the first-step, the k-means objective of the centers generated by the models diverge for all forms of token embeddings (`OH/BN/NA`) and $\min$-regularizer (NE/L2). However, treating multi-step clustering as *multiple single-steps of clustering* (thus, re-embedding the inputs at each step) does show improved clustering with multiple steps, and models with `OH/BN` embeddings significantly outperform Lloyd's iterations. Models with no token embeddings also show improvements with multiple steps but underperform Lloyd's. This highlights that *even though we train these models for single-step clustering, allowing for efficient training, they are viable for multi-step clustering as the performance improves when used iteratively in a careful manner.*

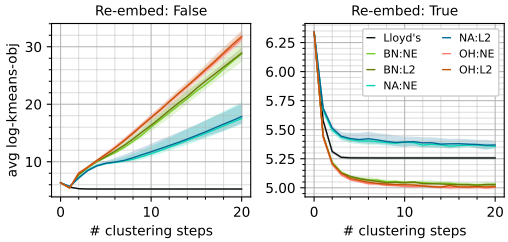


Figure 7: **Multi-step clustering performance.** Models trained with $n = 512, d = 32, k = 10, \tau = 0.1$ and used for $T = 20$ step clustering. We consider all configurations (NE/L2 regularizers and `OH/BN/NA` embeddings). **[Left]** Multi-step clustering by applying equation (4) iteratively for T steps. **[Right]** Multi-step clustering where the model input is re-embedded at each of the T steps, treating T-steps as T single-steps of clustering. See appendix C for further details.

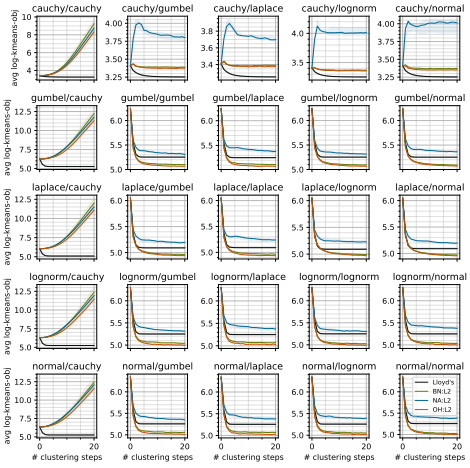


Figure 8: **Generalization to different distribution families.** The models are trained on one distribution family (each column maps to a training family), and tested for multi-step ($T = 20$) clustering from a different distribution family (each row maps to a test family).

Next, we evaluate the ability of the model (trained on clustering tasks from one distribution) to generalize to tasks from different distributions. In figure 6, we consider models trained on tasks with data sampled from an equally-weighted mixtures of isotropic `normal` distributions, and study their generalization to various changes to the mixture distributions. The learned models can perform multi-step clustering quite well, with `OH` and `BN` token embedding based models outperforming Lloyd's iterations significantly in all but one case — for data sampled from a mixture of `normal` distributions with a larger scale (than the one used during training), the models slightly underperform Lloyd's initially, but eventually outperform Lloyd's with additional clustering steps.

As a further evaluation of algorithmic generalization, we study the effect of changing the *distribution family* in figure 8 (larger version in figure 18). We see that models trained on clustering tasks with data from the `cauchy` distributions (first column in figure 8) always diverge for all distributions. This aligns with our previous results in figure 4, where models trained on `cauchy` tasks are even unable to generalize well to single-step `cauchy` clustering problems. Models trained on other distributions are also unable to perform well with `cauchy` tasks (first row in figure 8); the models with `OH` and `BN` embeddings do not diverge, but are not competitive to Lloyd's here. The remaining pairwise evaluations (the lower-right 4×4 in figure 8) all show that *the learned models can generalize well to evaluation tasks from other distributions, and continue to significantly outperform Lloyd's iterations* (for `OH/BN`). While generalization to `cauchy` is an open question, and we leave it for future work, we provide a preliminary probe in figure 9 (more tasks per family in figure 19) where we show the pairwise distributional dissimilarities between the per-task initial-center to point squared Euclidean distances. The `cauchy` tasks are clearly only similar to other `cauchy` tasks, and are quite dissimilar to the rest;

all other task families are quite similar to each other. These task (dis)similarities align quite well with the generalization (or lack thereof) we observe in figure 8.

5 Discussion

Building upon the initial findings of Garg et al. [18] and Akyürek et al. [19], the area of in-context learning with transformers is very active, usually focusing on supervised learning tasks where some labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ are provided in the context alongside a single unlabeled query $\boldsymbol{x}_q$, and the predicted output $\hat{y}_q$ for the query is generated in-context during the forward pass through the transformer. Von Oswald et al. [20] demonstrate that transformers with *linear attention* can execute gradient descent for a linear model in-context, while subsequent works study learnability with simplified linear attention [24] and the standard softmax attention [23]; note that Huang et al. [23] study a very stylized transformer with structured weight matrices. These results have been extended to demonstrate preconditioned gradient descent for a linear model [22] and functional gradient descent [25]. Usually, a differentiable supervised learning objective (such as mean squared error for regression) is considered. Our focus is on discrete clustering problems, where the learning objective is not natively differentiable, and we carefully study the effect of the necessary smoothing of the clustering objective on learnability.

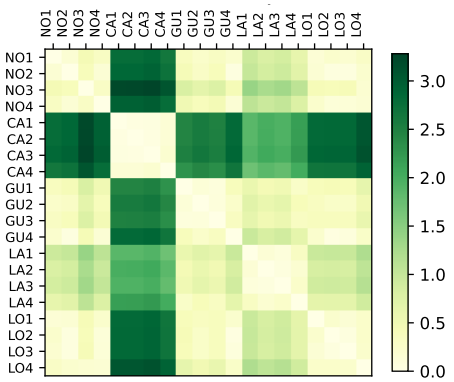


Figure 9: **Pairwise task dissimilarities.** We use 4 tasks from each of the families and compute the pairwise Earth-mover distance (EMD) between their respective distribution of the distances between the points $\boldsymbol{X}$ and initial centers $\boldsymbol{C}$, giving us a 20×20 matrix. Each row and column corresponds to a single task, and the tasks are arranged into family-based blocks in the following order: `normal`, `cauchy`, `gumbel`, `laplace`, `lognormal`. *Light shades show low EMD.*

While most existing literature focuses on linear attention transformers with (causal) self-attention, we build upon Clarkson et al. [27] and study softmax attention with an encoder-decoder model that uses both self-attention and cross-attention, though the algorithmic expressivity results rely on a *limiting softmax attention* where the attention softmax temperature goes to zero. However, this form of softmax attention is practically infeasible, and we use standard softmax attention for our learnability analysis and experiments.

Another line of research connecting transformers and clustering is the study of the token dynamics of self-attention where the tokens are iteratively updated via self-attention, and the goal is to understand the conditions under which these dynamics stabilize and converge [35, 36]. The results show that, under certain conditions, the token representations tend to self-cluster, first into a small number of clusters, and then eventually, to a single cluster. In contrast, we focus on learning clustering algorithms that are competitive to existing hand-crafted schemes, and train a single transformer layer instead of the problem setup of the aforementioned papers with multiple (possibly infinite) layers. We learn the transformer parameters from data (in this case, a collection of clustering tasks) and study their training convergence and algorithmic generalization.

Conclusion. We study the ability of transformers to express and learn clustering algorithms. First, we establish expressivity of Lloyd's algorithm for k-means clustering with a leaner transformer (Theorem 1) than existing constructions [27]. Then, we study the factors affecting the ability of transformers to learn a clustering algorithm (Theorem 2), and empirically validate their importance (Figure 2, 3, 4, 5). We show that these learned algorithms – in the form of transformers – can progressively improve clustering with iterative application (Figure 7), though naive iteration fails, and can successfully execute clustering for tasks that are different than those used for training (Figure 6, 8), though exceptions exist. This work comes with various limitations, with the main ones being (i) the restriction that transformers learn an algorithm for a specific data dimensionality d and number of clusters k, while hand-crafted algorithms are more general, and (ii) the lack of understanding of the failure to train and generalize with `cauchy` tasks. We discuss these and various others in Appendix A.

References

[1] John A Hartigan. *Clustering Algorithms*. Wiley series in probability and mathematical statistics, 1975. 1

[2] Charu C Aggarwal and Chandan K Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 2013. 1

[3] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5, pages 281–298. University of California press, 1967. 1

[4] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28 (2):129–137, 1982. 1, 2, 18

[5] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1974. 1, 2

[6] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013. 1, 2

[7] Leonard Kaufman. Partitioning around medoids (program pam). *Finding groups in data*, 344: 68–125, 1990. 1, 18

[8] Raymond T. Ng and Jiawei Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002. 1

[9] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 1

[10] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. 1

[11] Mark Girolami. Mercer kernel-based clustering in feature space. *IEEE transactions on neural networks*, 13(3):780–784, 2002.

[12] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004. 1

[13] Sanjoy Dasgupta. The hardness of k-means clustering. *UCSD Technical Report*, 2008. URL https://cseweb.ucsd.edu/~dasgupta/papers/kmeans.pdf. 1

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[15] Mary Phuong and Marcus Hutter. Formal algorithms for transformers. *arXiv preprint arXiv:2207.09238*, 2022. URL https://arxiv.org/pdf/2207.09238.pdf. 1

[16] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. URL https://arxiv.org/pdf/1410.5401. 1

[17] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/fb87582825f9d28a8d42c5e5e5e8b23d-Paper.pdf. 1

[18] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf. 2, 10

[19] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I. 10, 16

[20] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023. URL https://proceedings.mlr.press/v202/von-oswald23a/von-oswald23a.pdf. 2, 10

[21] Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pages 19565–19594. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23l/li23l.pdf. 2, 30, 33

[22] Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=LziniAXEI9. 10

[23] Yu Huang, Yuan Cheng, and Yingbin Liang. In-context convergence of transformers. *arXiv preprint arXiv:2310.05249*, 2023. URL https://arxiv.org/pdf/2310.05249.pdf. 10

[24] Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023. URL https://arxiv.org/pdf/2306.09927.pdf. 10

[25] Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ah1BlQcLv4. 2, 10

[26] Takuya Ito, Murray Campbell, Lior Horesh, Tim Klinger, and Parikshit Ram. Quantifying artificial intelligence through algorithmic generalization. *Nature Machine Intelligence*, pages 1–11, 2025. URL https://www.nature.com/articles/s42256-025-01092-w. 2

[27] Kenneth L. Clarkson, Lior Horesh, Takuya Ito, Charlotte Park, and Parikshit Ram. Transformer circuits can realize clustering algorithms. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=2jw5U060C4. 2, 3, 6, 10, 15, 16, 18, 44

[28] Juan Antonio Cuesta-Albertos, Alfonso Gordaliza, and Carlos Matrán. Trimmed k-means: an attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553–576, 1997. URL http://dx.doi.org/10.1214/aos/1031833664. 2

[29] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. 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)*, pages 4344–4353. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1443. URL https://aclanthology.org/D19-1443/. 3

[30] André Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR, 2016. URL https://proceedings.mlr.press/v48/martins16.pdf. 5, 15, 29

[31] Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020. URL https://jmlr.csail.mit.edu/papers/volume21/19-021/19-021.pdf. 5, 30

[32] Olivier Bousquet and André Elisseeff. Algorithmic stability and generalization performance. *Advances in Neural Information Processing Systems*, 13, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf. 5

[33] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR, 2016. URL https://arxiv.org/pdf/1509.01240. 5, 6

[34] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf. 7, 8

[35] Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. The emergence of clusters in self-attention dynamics. *arXiv preprint arXiv:2305.05465*, 2023. URL https://arxiv.org/pdf/2305.05465.pdf. 10

[36] Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. A mathematical perspective on transformers. *arXiv preprint arXiv:2312.10794*, 2023. URL https://arxiv.org/pdf/2312.10794.pdf. 10

[37] Weiran Wang and Miguel A Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013. URL https://arxiv.org/pdf/1309.1541. 29

Appendix

Table of Contents

A Limitations 15

B Leaner Transformers Can Emulate Lloyd's Iteration 16

C Single-Step and Multi-Step Clustering 18

D Clustering Task Generation 19

E Computational Resources and Implementation 20

F Ablative Experiments 20

G Additional Figures 27

H Smoothed Clustering Objective via Regularized Minimum 29

I Lipschitz-Ness of the Smoothed Upperbound of the Clustering Objective 31

A Limitations

As discussed in our concluding remarks, one major limitation of our work is that we currently lack a clear understanding as to why the learning and generalization fail in certain situations. For example, it is not clear why learning and generalization is hard with `cauchy` distributions. We hypothesize that this is related to the inability of the smoothed clustering objective (via a regularized $\min$) to provide a tight upper-bound for the actual k-means clustering loss (as we show in Figure 5). However, it is not completely clear why, but we speculate that the following is a potential path to understanding: The `cauchy` distribution is fat-tailed, and hence, many points in $\boldsymbol{X}$ can be quite far from the true cluster, and act like outliers in a clustering problem. The presence of outlier values – the large pairwise squared Euclidean distances to the outliers compared to the relatively smaller pairwise squared distances to the inliers – in the regularized $\min$ can severely degrade the upper-bounds obtained via the smoothed objective. This still does not give us a clear way to mitigate the issues while training with `cauchy` tasks seen in Figure 4. Similarly, the presence of outliers in $\boldsymbol{X}$ also results in the significant pairwise distributional distances between the tasks shown in Figure 9 and Figure 19, leading to the failures of the trained transformers on `cauchy` validation tasks.

Beyond that, there are architectural limitations we inherit from Clarkson et al. [27], where the transformer learns to cluster for a specific choice of data dimensionality d and number of clusters k, though it is able to perform clustering for varying number of points n in d dimensions into k clusters (as shown in Figure 12). It is important to extend this architecture to be able to generalize to different values of k or d, and be able to train on clustering tasks with a certain k value and generalize to larger k values. One possible way to generalize in terms of d is to train the transformer with a large d'. Then any clustering task with $d < d'$ can be input to the transformer by zero-padding the necessary extra $(d' - d)$ dimensions to the original data in d dimensions. As Euclidean distances are unaffected by zero-padding, this idea should conceptually work. However, we pay the computational cost of training a model for a large d' as the model embedding dimensionality d_{emb} would scale linearly in d', while the number of parameters would scale quadratically in d'. Whether such a setup would show strong generalization performance across various values of d is unclear, and would depend on the training strategies. A similar scheme could also be considered to generalize to different values of k, where we would train a model on a large value of k'.

Additionally, there are various open questions we do not address in this paper, such as

- It is not clear if the additional dimensions – k dimensions with (`OH`) and $\lceil \log_2 k \rceil$ with (`BN`) – are necessary to learn a clustering algorithm that is competitive to Lloyd's algorithm. Empirically, we see that these additional dimensions do allow the models to surpass Lloyd's algorithm, and the models without these additional dimensions (the `NA` models) are empirically unable to match Lloyd's. However, we do not have theory codifying this need for the additional dimensions.
- Our proposed token embedding in (`BN`) can express Lloyd's algorithm for k-means clustering with the limiting softmax attention (with $\gamma \to \infty$), but can introduce an arbitrary topological order among the cluster centers with a finite γ. We already see that models with the `BN` token embedding have slower training convergence compared to the models with the `OH` token embeddings (especially for small d), but it is possible that this arbitrary order can introduce other unexpected behaviours.
- The standard transformer has an attention block and a feedforward block (which we do not consider in this work). It is not clear how the inclusion of the feedforward block would affect learning and generalization. The theoretical results in Clarkson et al. [27] utilize the feedforward block to perform Euclidean clustering with dot-product attention (instead of the negative squared Euclidean distance based attention in Definition 2), but it is not clear what role the feedforward block will play in the practical learning setup.
- While we are able to train the model efficiently for single-step clustering tasks, and utilize it for multi-step clustering by treating multi-step clustering as multiple single-step clustering tasks, it is not clear if we can learn a better model by training directly for multi-step clustering with appropriately designed objectives, and do so efficiently.
- Our discussion in Section 3 around learning convergence and generalization consider training losses that are β-smooth. We see that the L2-regularized $\min$ (using sparsemax [30]) provides a tighter upper-bound and thus generalizes better in many cases over the NE-regularized $\min$ (using softmax). However, the sparsemax operation is only differentiable almost everywhere, and is not generally β-smooth, thus leading to a mismatch between our theoretical discussion and empirical results. One possible explanation for the strong empirical performance of the L2-regularized $\min$

Algorithm 2 Lloyd's algorithm to cluster n points into k clusters

Input: Instances $\{\boldsymbol{x}_i \in \mathbb{R}^d, i \in [n]\}$
Input: Number of iterations T
Input: Initial centers $\{\boldsymbol{c}_j \in \mathbb{R}^d, j \in [k]\}$
$\forall j \in [k], \boldsymbol{c}_j^{(0)} \leftarrow \boldsymbol{c}_j$
for $t = 1, \ldots, T$ **do**
  $\forall i \in [n],\ a_i^{(t)} \leftarrow \arg\min_{j \in [k]} \left\| \boldsymbol{x}_i - \boldsymbol{c}_j^{(t-1)} \right\|^2$
  $\forall j \in [k],\ \boldsymbol{c}_j^{(t)} \leftarrow \arg\min_{\boldsymbol{c} \in \mathbb{R}^d} \sum_{i \in [n]: a_i^{(t)} = j} \|\boldsymbol{x}_i - \boldsymbol{c}\|^2$
return $\{\boldsymbol{c}_j^{(T)}, j \in [k]\}$

is that the loss surface is actually more favourable than the usual nonconvex loss landscapes of neural networks, as can be seen in Figure 3, where the loss landscapes appear convex for the larger value of d.
- We treat the softmax attention inverse temperature γ and the regularized $\min$ penalty τ as fixed hyperparameters in our analysis in Theorem 2 and in our experiments. But in practice, these parameters can be adaptive or annealed through the training and inference, and this is not something we consider here.
- Finally, while we see that the clustering algorithm learned by the transformer is able to outperform Lloyd's algorithm for k-means clustering, it is not clear what the precise clustering algorithm is that the learned model is executing. While we can consider the tools used in Akyürek et al. [19] such as the notion of "algorithmic distance", we leave this for future work.

B Leaner Transformers Can Emulate Lloyd's Iteration

We use $\boldsymbol{I}_d$ to denote the $(d \times d)$ identity matrix. Further, we specify our matrices for attention in a compact form, using the following notation. For integers $1 \le i \le j \le d$, let $\boldsymbol{I}_d^{i:j}$ be the matrix such that for any $\boldsymbol{x} \in \mathbb{R}^d$ such that

$$\boldsymbol{v} = \boldsymbol{I}_d^{i:j}\boldsymbol{x}, \quad \text{where } v_m = \begin{cases} x_m & i \le m \le j \\ 0 & o.w. \end{cases}. \tag{11}$$

The $\boldsymbol{v} = \boldsymbol{I}_d^{i:j}\boldsymbol{x}$ zeroes out the m-th entries of $\boldsymbol{x}$ with indices $m \in [1, i) \cup (j, d]$, leaving the rest unchanged. When $j = d$, we use $\boldsymbol{I}_d^{i:}$, and when $i = 1$, we use $\boldsymbol{I}_d^{:j}$.

We prove theorem 1 based on the construction in Clarkson et al. [27, Theorem 2] with minor modification. We use the selective property of attention as described in Clarkson et al. [27, Appendix D] in the form of *averaging hard attention* or AHAT as the inverse temperature of the softmax goes to infinity; this behaves as a *hardmax* when there is a unique maximum. We are restating theorem 1 as follows:

Theorem 3. *Given as input a set of points $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ and initial centers $\boldsymbol{C} \in \mathbb{R}^{d \times k}$, assuming that no iterations of Lloyd's algorithm lead to empty clusters, there exists parameters $\{(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 1, \ldots, 4\}$ for the transformer in Definition 3 with $\gamma = \infty$ and initial token embeddings $(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)})$ as in Definition 1 then, for any $t \in \mathbb{N}$, the output of the t-th transformer layer exactly matches the output of Lloyd's algorithm (Algorithm 2) with the same input after t iterations.*

Proof. With $d_{\mathsf{emb}} = d + \lceil \log_2 k \rceil$, let $\boldsymbol{K}_1 = \boldsymbol{Q}_1 = \boldsymbol{K}_2 = \boldsymbol{Q}_2 = \boldsymbol{I}_{d_{\mathsf{emb}}}^{:d}$, and $\boldsymbol{V}_1 = -\boldsymbol{V}_2 = \boldsymbol{I}_{d_{\mathsf{emb}}}^{d+1:}$. Note that $\boldsymbol{I}_{d_{\mathsf{emb}}}^{:d}$ and $\boldsymbol{I}_{d_{\mathsf{emb}}}^{d+1:}$ are $(d + \lceil \log_2 k \rceil) \times (d + \lceil \log_2 k \rceil)$ matrices.

At any step $t \ge 0$, let us assume that $\bar{\boldsymbol{X}}^{(t)} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(t)} \end{bmatrix}$, and $\bar{\boldsymbol{C}}^{(t)} = \begin{bmatrix} \boldsymbol{C}^{(t)} \\ \boldsymbol{B}_k \end{bmatrix}$. This is true at $t = 0$ by the definition of the token embeddings in definition 1. We will show that, at any $t \ge 0$ the update to $(\bar{\boldsymbol{X}}^{(t+1)}, \bar{\boldsymbol{C}}^{(t+1)})$ maintains this form where $\bar{\boldsymbol{X}}^{(t+1)} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(t+1)} \end{bmatrix}$, and $\bar{\boldsymbol{C}}^{(t+1)} = \begin{bmatrix} \boldsymbol{C}^{(t+1)} \\ \boldsymbol{B}_k \end{bmatrix}$, where

- $\boldsymbol{Y}^{(t+1)}$ is the matrix with the i-th column corresponding to the binary representation of $(j^\star(i)-1)$ in $\lceil \log_2 k \rceil$ bits, with $j^\star(i) \in [k]$ denoting the index of the cluster to which the i-th point is assigned to in the $(t+1)$-th Lloyd's iteration in line 6 of Algorithm 2.
- $\boldsymbol{C}^{(t+1)}$ corresponds to the updated cluster centers as in line 7 of Algorithm 2.

To update cluster assignments for Lloyd's algorithm, that is, to compute $\bar{\boldsymbol{X}}^{(t+1)}$ from $\bar{\boldsymbol{X}}^{(t)}$ and $\bar{\boldsymbol{C}}^{(t)}$, we compute

$$\begin{aligned}
&\mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{C}}^{(t)}; (\boldsymbol{I}_{d_{\text{emb}}}^{:d}, \boldsymbol{I}_{d_{\text{emb}}}^{:d}, \boldsymbol{I}_{d_{\text{emb}}}^{d+1:})) + \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{X}}^{(t)}; (\boldsymbol{I}_{d_{\text{emb}}}^{:d}, \boldsymbol{I}_{d_{\text{emb}}}^{:d}, -\boldsymbol{I}_{d_{\text{emb}}}^{d+1:})) \\
&\quad = \boldsymbol{I}_{d_{\text{emb}}}^{d+1:} \bar{\boldsymbol{C}}^{(t)} \mathrm{softmax}(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}(\boldsymbol{I}_{d_{\text{emb}}}^{:d} \bar{\boldsymbol{X}}^{(t)}, \boldsymbol{I}_{d_{\text{emb}}}^{:d} \bar{\boldsymbol{C}}^{(t)})) \\
&\qquad - \boldsymbol{I}_{d_{\text{emb}}}^{d+1:} \bar{\boldsymbol{X}}^{(t)} \mathrm{softmax}(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}(\boldsymbol{I}_{d_{\text{emb}}}^{:d} \bar{\boldsymbol{X}}^{(t)}, \boldsymbol{I}_{d_{\text{emb}}}^{:d} \bar{\boldsymbol{X}}^{(t)})) \qquad (12) \\
&\quad = \begin{bmatrix} \boldsymbol{0}_{d\times k} \\ \boldsymbol{B}_k \end{bmatrix} \mathrm{softmax}\left(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}\left(\begin{bmatrix} \boldsymbol{X} \\ 0_{\lceil \log_2 k \rceil \times n} \end{bmatrix}, \begin{bmatrix} \boldsymbol{C}^{(t)} \\ 0_{\lceil \log_2 k \rceil \times k} \end{bmatrix}\right)\right) \\
&\qquad - \begin{bmatrix} \boldsymbol{0}_{d\times n} \\ \boldsymbol{Y}^{(t)} \end{bmatrix} \mathrm{softmax}\left(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}\left(\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times n} \end{bmatrix}, \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times n} \end{bmatrix}\right)\right) \qquad (13) \\
&\quad = \begin{bmatrix} \boldsymbol{0}_{d\times k} \\ \boldsymbol{B}_k \end{bmatrix} \mathrm{hardmax}\left(-\mathrm{sqdist}\left(\boldsymbol{X}, \boldsymbol{C}^{(t)}\right)\right) - \begin{bmatrix} \boldsymbol{0}_{d\times n} \\ \boldsymbol{Y}^{(t)} \end{bmatrix} \mathrm{hardmax}\left(-\mathrm{sqdist}\left(\boldsymbol{X}, \boldsymbol{X}\right)\right), \qquad (14)
\end{aligned}$$

as $\mathrm{softmax}$ with inverse temperature $\gamma = \infty$ gives us $\mathrm{hardmax}$, where for any matrix $\boldsymbol{A} \in \mathbb{R}^{a\times b}$, $\mathrm{hardmax}(\boldsymbol{A}) \in \{0,1\}^{a\times b}$ is binary matrix of the same size as $\boldsymbol{A}$ with each column having exactly a single nonzero value corresponding to the unique maximum of the column, assuming each column has an unique maximum.

First, assuming $\boldsymbol{x}_i \neq \boldsymbol{x}_{i'}$ for any $i \neq i'$, $\mathrm{hardmax}(-\mathrm{sqdist}(\boldsymbol{X}, \boldsymbol{X})) = \boldsymbol{I}_n$. Next $\mathrm{hardmax}\left(-\mathrm{sqdist}\left(\boldsymbol{X}, \boldsymbol{C}^{(t)}\right)\right)$ applies a columnwise argmin to the $(k \times n)$ matrix $\mathrm{sqdist}(\boldsymbol{X}, \boldsymbol{C}^{(t)})$, thereby, each column having only the $j^\star(i)$-th entry as 1 where $j^\star(i) = \arg\min_{j\in[k]} \left\| \boldsymbol{X}_{:,i} - \boldsymbol{C}^{(t)}_{:,j} \right\|^2$. Thus $\boldsymbol{Y}^{(t+1)} = \boldsymbol{B}_k \mathrm{hardmax}\left(-\mathrm{sqdist}\left(\boldsymbol{X}, \boldsymbol{C}^{(t)}\right)\right)$ is a $(\lceil \log_2 k \rceil \times n)$ binary matrix, where the i-th column $\boldsymbol{Y}^{(t)}_{:,i}$ is the $j^\star(i)$-th column of $\boldsymbol{B}_k$, thereby denoting the updated cluster assignments as per Lloyd's algorithm (Algorithm 2, line 5). All together, we have

$$\begin{aligned}
&\mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{C}}^{(t)}; (\boldsymbol{I}_{d_{\text{emb}}}^{:d}, \boldsymbol{I}_{d_{\text{emb}}}^{:d}, \boldsymbol{I}_{d_{\text{emb}}}^{d+1:})) + \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{X}}^{(t)}; (\boldsymbol{I}_{d_{\text{emb}}}^{:d}, \boldsymbol{I}_{d_{\text{emb}}}^{:d}, -\boldsymbol{I}_{d_{\text{emb}}}^{d+1:})) \\
&\quad = \begin{bmatrix} \boldsymbol{0}_{d\times n} \\ \boldsymbol{Y}^{(t+1)} - \boldsymbol{Y}^{(t)} \end{bmatrix}. \qquad (15)
\end{aligned}$$

Adding this to $\bar{\boldsymbol{X}}^{(t)} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(t)} \end{bmatrix}$ via the residual connection in equation (4) gives us the updated $\bar{\boldsymbol{X}}^{(t+1)} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y}^{(t+1)} \end{bmatrix}$ with the updated cluster assignments $\boldsymbol{Y}^{(t+1)}$ matching Lloyd's algorithm (Algorithm 2, line 5).

Let $\boldsymbol{Q}_3 = \boldsymbol{K}_3 = \boldsymbol{Q}_4 = \boldsymbol{K}_4 = \boldsymbol{I}_{d_{\text{emb}}}^{d+1:}$, and $\boldsymbol{V}_3 = -\boldsymbol{V}_4 = \boldsymbol{I}_{d_{\text{emb}}}^{:d}$. To update cluster centers as per Lloyd's algorithm (Algorithm 2, line 6), we compute

$$\begin{aligned}
&\mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(t)}, \bar{\boldsymbol{X}}^{(t+1)}; (\boldsymbol{I}_{d_{\text{emb}}}^{d+1:}, \boldsymbol{I}_{d_{\text{emb}}}^{d+1:}, \boldsymbol{I}_{d_{\text{emb}}}^{:d})) + \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(t)}, \bar{\boldsymbol{C}}^{(t)}; (\boldsymbol{I}_{d_{\text{emb}}}^{d+1:}, \boldsymbol{I}_{d_{\text{emb}}}^{d+1:}, -\boldsymbol{I}_{d_{\text{emb}}}^{:d})) \\
&\quad = \boldsymbol{I}_{d_{\text{emb}}}^{:d} \bar{\boldsymbol{X}}^{(t+1)} \mathrm{softmax}\left(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}\left(\boldsymbol{I}_{d_{\text{emb}}}^{d+1:} \bar{\boldsymbol{C}}^{(t)}, \boldsymbol{I}_{d_{\text{emb}}}^{d+1:} \bar{\boldsymbol{X}}^{(t+1)}\right)\right) \\
&\qquad - \boldsymbol{I}_{d_{\text{emb}}}^{:d} \bar{\boldsymbol{C}}^{(t)} \mathrm{softmax}\left(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}\left(\boldsymbol{I}_{d_{\text{emb}}}^{d+1:} \bar{\boldsymbol{C}}^{(t)}, \boldsymbol{I}_{d_{\text{emb}}}^{d+1:} \bar{\boldsymbol{C}}^{(t)}\right)\right) \qquad (16) \\
&\quad = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times n} \end{bmatrix} \mathrm{softmax}\left(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}\left(\begin{bmatrix} \boldsymbol{0}_{d\times k} \\ \boldsymbol{B}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{0}_{d\times n} \\ \boldsymbol{Y}^{(t+1)} \end{bmatrix}\right)\right) \\
&\qquad - \begin{bmatrix} \boldsymbol{C}^{(t)} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times k} \end{bmatrix} \mathrm{softmax}\left(-(\gamma/\sqrt{d_{\text{emb}}})\mathrm{sqdist}\left(\begin{bmatrix} \boldsymbol{0}_{d\times k} \\ \boldsymbol{B}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{0}_{d\times k} \\ \boldsymbol{B}_k \end{bmatrix}\right)\right) \qquad (17) \\
&\quad = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times n} \end{bmatrix} \mathrm{avghardmax}\left(-\mathrm{sqdist}\left(\boldsymbol{B}_k, \boldsymbol{Y}^{(t+1)}\right)\right)
\end{aligned}$$

$$- \begin{bmatrix} \boldsymbol{C}^{(t)} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times k} \end{bmatrix} \text{hardmax}\left(-\text{sqdist}\left(\boldsymbol{B}_k, \boldsymbol{B}_k\right)\right) \tag{18}$$

where $\gamma = \infty$ gives us the columnwise hardmax and the columnwise averaging hardmax operation avghardmax. The reason for the averaging hardmax is because each column in $\text{sqdist}\left(\boldsymbol{B}_k, \boldsymbol{Y}^{(t+1)}\right)$ can have a non-singleton set of minima (or multiple maxima). As before, $\text{hardmax}\left(-\text{sqdist}\left(\boldsymbol{B}_k, \boldsymbol{B}_k\right)\right) = \boldsymbol{I}_k$.

Next, $\text{avghardmax}\left(-\text{sqdist}\left(\boldsymbol{B}_k, \boldsymbol{Y}^{(t+1)}\right)\right)$ will be a $(n \times k)$ matrix, where in the j-th column, the i-th entry will be zero if $\boldsymbol{X}_{:,i}$ is not assigned to the j-th cluster, that is the i-th column of $\boldsymbol{Y}^{(t+1)}$ does not match the j-th column of $\boldsymbol{B}_k$; otherwise, the averaging hardmax sets it to $1/m_j$ where m_j is the number of points assigned to the j-th cluster, and thus, the squared Euclidean distance between the i-th column of $\boldsymbol{Y}^{(t+1)}$ and the j-th column of $\boldsymbol{B}_k$ is zero. Note that $m_j > 0 \forall j \in [k]$ as we assume that there are no empty clusters during Lloyd's iterations. Thus, $\boldsymbol{X}\,\text{avghardmax}\left(-\text{sqdist}\left(\boldsymbol{B}_k, \boldsymbol{Y}^{(t+1)}\right)\right)$ will be a $(d \times k)$ matrix corresponding to the centers $\boldsymbol{C}^{(t+1)}$ computed with Lloyd's algorithm (Algorithm 2, line 6). Thus, we have

$$\begin{aligned} &\mathcal{A}_2^{\gamma}(\bar{\boldsymbol{C}}^{(t)}, \bar{\boldsymbol{X}}^{(t+1)}; (\boldsymbol{I}_{d_{\mathsf{emb}}}^{d+1:}, \boldsymbol{I}_{d_{\mathsf{emb}}}^{d+1:}, \boldsymbol{I}_{d_{\mathsf{emb}}}^{:d})) + \mathcal{A}_2^{\gamma}(\bar{\boldsymbol{C}}^{(t)}, \bar{\boldsymbol{C}}^{(t)}; (\boldsymbol{I}_{d_{\mathsf{emb}}}^{d+1:}, \boldsymbol{I}_{d_{\mathsf{emb}}}^{d+1:}, -\boldsymbol{I}_{d_{\mathsf{emb}}}^{:d})) \\ &\quad = \begin{bmatrix} \boldsymbol{C}^{(t+1)} - \boldsymbol{C}^{(t)} \\ \boldsymbol{0}_{\lceil \log_2 k \rceil \times k} \end{bmatrix}, \end{aligned} \tag{19}$$

and adding this to $\bar{\boldsymbol{C}}^{(t)} = \begin{bmatrix} \boldsymbol{C}^{(t)} \\ \boldsymbol{B}_k \end{bmatrix}$ gives us $\bar{\boldsymbol{C}}^{(t+1)} = \begin{bmatrix} \boldsymbol{C}^{(t+1)} \\ \boldsymbol{B}_k \end{bmatrix}$, thus tracking the same cluster centers as in Lloyd's algorithm.

Thus we have established that both the cluster assignments and the cluster center updates of Lloyd's algorithm can be maintained in $(\bar{\boldsymbol{X}}^{(t)}, \bar{\boldsymbol{C}}^{(t)})$ for any t with the leaner token embedding with $d_{\mathsf{emb}} = d + \lceil \log_2 k \rceil$. □

Note that this construction for the $\bar{\boldsymbol{X}}^{(t)} \to \bar{\boldsymbol{X}}^{(t+1)}$ is the same as in the proof of Clarkson et al. [27, Theorem 2]. The construction for the $\bar{\boldsymbol{C}}^{(t)} \to \bar{\boldsymbol{C}}^{(t+1)}$ update is different from the one in Clarkson et al. [27] as we make use of negative squared Euclidean distance based attention with $\gamma = \infty$ instead of a linear attention as in Clarkson et al. [27].

C Single-Step and Multi-Step Clustering

We set up the learning problem in equation (6) as a single-step clustering task where the transformer model generates cluster prototypes in one-shot (with a single forward pass through the model) such that the k-means objective (actually a surrogate of it) is reduced. However, almost all clustering algorithms are iterative in nature, where the cluster prototypes are iteratively refined in a way that usually progressively improves the clustering objective; Lloyd's algorithm for k-means clustering [4] and partitioning around medoids or PAM [7] are examples of such algorithms. Thus, it is natural to explore whether we can utilize this trained transformer for multi-step clustering, with the cluster prototypes being iteratively refined. Given the theoretical results in Theorem 1 and Clarkson et al. [27, Theorem 1], it would seem natural to perform T-step clustering refinement by applying the attention-based updates in definition 3 iteratively for T steps with the learned model parameters, starting from $(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}) \leftarrow (E_X(\boldsymbol{X}), E_C(\boldsymbol{C}))$, and use the first d rows $\bar{\boldsymbol{C}}_{:d,:}^{(T)}$ of the T-step output $(\bar{\boldsymbol{X}}^{(T)}, \bar{\boldsymbol{C}}^{(T)})$ as the final set of prototypes. Note that the token embedding scheme can be either the $(E_X^{\texttt{OH}}, E_C^{\texttt{OH}})$ scheme in (OH) or the proposed $(E_X^{\texttt{BN}}, E_C^{\texttt{BN}})$ scheme in (BN) in Definition 1.

However, it is important to note that this is valid only if the transformer model parameters are as in the constructive proofs of these theoretical results, and if we are using the softmax attention with the inverse temperature $\gamma \to \infty$. We have trained the transformer model with finite γ.

Thus, in the following, we precisely explain what we mean by *single-step clustering* (used during training) vs *multi-step clustering* (used for evaluating the model as a clustering algorithm). First, we will demonstrate how standard algorithms such as Lloyd's perform multi-step clustering by applying the same operation iteratively on evolving inputs (data and current centers). Then we will use the same intuition to motivate how we can similarly use the model iteratively on evolving inputs.

For a clustering task with data $\boldsymbol{X}$ and initial centers $\boldsymbol{C}^{(0)}$, T iterations of Lloyd's algorithm (algorithm 2) does the following:

- [Step 1] $\boldsymbol{X}, \boldsymbol{C}^{(0)} \rightarrow$ assign/center-update $\rightarrow \boldsymbol{C}^{(1)}$,
- [Step 2] $\boldsymbol{X}, \boldsymbol{C}^{(1)} \rightarrow$ assign/center-update $\rightarrow \boldsymbol{C}^{(2)}$,
- $\cdots$
- [Step T] $\boldsymbol{X}, \boldsymbol{C}^{(T-1)} \rightarrow$ assign/center-update $\rightarrow \boldsymbol{C}^{(T)}$.

With a trained model $F^{\gamma}_{d:k}(\cdot, \cdot; \Theta)$ which uses token embeddings (`OH`/`BN`) and attention based updates in (4), we can perform T iterations of multi-step clustering in the same spirit as the above iterative Lloyd's algorithm by applying the same model $F^{\gamma}_{d:k}$ on evolving inputs (data and current centers):

- [Step 1] $\boldsymbol{X}, \boldsymbol{C}^{(0)} \rightarrow F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}^{(0)}; \Theta) \rightarrow \boldsymbol{C}^{(1)}$,
- [Step 2] $\boldsymbol{X}, \boldsymbol{C}^{(1)} \rightarrow F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}^{(1)}; \Theta) \rightarrow \boldsymbol{C}^{(2)}$,
- $\cdots$
- [Step T] $\boldsymbol{X}, \boldsymbol{C}^{(T-1)} \rightarrow F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}^{(T-1)}; \Theta) \rightarrow \boldsymbol{C}^{(T)}$.

This follows the iterative nature of Lloyd's algorithm, and this is what is evaluated in figure 7 [right], figure 6, figure 8, and figure 12. With single-step clustering, we refer to the fact that, during training, this model is executed on each training clustering task for only 1 step, and the training loss is computed with $\boldsymbol{C}^{(1)}$, the output of the [Step 1] update. This makes the training extremely efficient as we are training a single transformer layer without any recursion. However, we see that this model learns some clustering mechanisms as we can reapply it for T and progressively improve clustering quality, much like applying the "assign/center-update" step in Lloyd's algorithm iteratively.

Figure 7 [left] is ablating the role of the token embedding in $F^{\gamma}_{d:k}$ by evaluating an alternate form of multi-step clustering that only applies the attention updates in (4) iteratively as follows:

- [Step 0] $\boldsymbol{X}, \boldsymbol{C}^{(0)} \rightarrow$ (`OH`)/(`BN`) $\rightarrow \bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}$,
- [Step 1] $\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)} \rightarrow$ eq. (4) $\rightarrow \bar{\boldsymbol{X}}^{(1)}, \bar{\boldsymbol{C}}^{(1)}$,
- [Step 2] $\bar{\boldsymbol{X}}^{(1)}, \bar{\boldsymbol{C}}^{(1)} \rightarrow$ eq. (4) $\rightarrow \bar{\boldsymbol{X}}^{(2)}, \bar{\boldsymbol{C}}^{(2)}$,
- $\cdots$,
- [Step T] $\bar{\boldsymbol{X}}^{(T-1)}, \bar{\boldsymbol{C}}^{(T-1)} \rightarrow$ eq. (4) $\rightarrow \bar{\boldsymbol{X}}^{(T)}, \bar{\mathbf{C}}^{(T)}$,
- [final step] $\bar{\mathbf{C}}^{(T)} \rightarrow \mathbf{C}^{(T)}$.

The results in figure 7 [left] shows that just applying the attention updates recursively does not improve the clustering quality, and both the token embeddings (`OH`)/(`BN`) and the attention updates (4) in the trained model $F^{\gamma}_{d:k}(\cdot, \cdot; \Theta)$ are essential for the trained transformer to successfully cluster iteratively.

D Clustering Task Generation

To train the transformer, we generate clustering tasks $(\boldsymbol{X}, \boldsymbol{C}) \sim \mathcal{D}$ from a distribution $\mathcal{D}$ where $\boldsymbol{X}$ denotes the points to be clustered, and $\boldsymbol{C}$ denotes the initial cluster centers. We consider mixture distributions (such as mixture of `normal` distributions or mixture of `cauchy` distributions, etc) to generate $\boldsymbol{X} \in [0, 1]^{d \times n}$ in the following manner:

- Randomly sample k centers $\{\boldsymbol{l}_j \sim \mathcal{U}(0, 1)^d, j \in [k]\}$ in d dimensions uniformly from the unit cube $[0, 1]^d$
- Randomly sample k d-dimensional scales $\{\boldsymbol{s}_j \sim \mathcal{U}(0, 1)^d, j \in [k]\}$ if using anisotropic distributions else $\boldsymbol{s}_j = \mathbf{1}_d \forall j \in [k]$.
- Randomly sample k weights $\{w_j, j \in [k]\}$ from the k-dimensional simplex Δ^k if using unequally weighted mixtures else $w_j = 1/k \forall i \in [k]$
- Create a mixture distribution $\mathcal{T} \triangleq \{(\boldsymbol{l}_j, \boldsymbol{s}_j, w_j)_{j \in [k]}\}$ and sample n samples $\boldsymbol{x}_i \sim \mathcal{T}$ in d dimensions to get $\hat{\boldsymbol{X}} \in \mathbb{R}^{d \times n}$.
- $\boldsymbol{X} \leftarrow$ `MinMaxScaler`$(\hat{\boldsymbol{X}})$

Note that, as the data dimensionality d increases, the diameter of the set of samples increases. Given the set of samples, we randomly sample k points from $\boldsymbol{X}$ to define the set of initial centers $\boldsymbol{C} \in [0, 1]^{d \times k}$.

The motivation behind generating clustering tasks where $\boldsymbol{X} \in [0,1]^{d\times n}, \boldsymbol{C} \in [0,1]^{d\times k}$ is that this aligns with our theoretical analyses, and any general k-means clustering problem with $\hat{\boldsymbol{X}} \in \mathbb{R}^{d\times n}, \hat{\boldsymbol{C}} \in \mathbb{R}^{d\times k}$ can be mapped accordingly given the translation and scaling invariance of the k-means clustering problem. Let $\boldsymbol{v}_{\mathsf{lb}} \in \mathbb{R}^d$ be the vector of the per-dimension minimum values for all the points in $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{C}}$ combined. Then setting $\bar{\boldsymbol{X}} \leftarrow \hat{\boldsymbol{X}} - \boldsymbol{v}_{\mathsf{lb}}\mathbf{1}_n^\top, \bar{\boldsymbol{C}} \leftarrow \hat{\boldsymbol{C}} - \boldsymbol{v}_{\mathsf{lb}}\mathbf{1}_k^\top$ moves all the points and centers to the nonnegative quadrant $\mathbb{R}^d_{\geq 0}$. Let $v_{\mathsf{ub}} \in \mathbb{R}_+$ be the single largest scalar value in $\bar{\boldsymbol{X}}, \bar{\boldsymbol{C}}$. We assume that this $v_{\mathsf{ub}} > 0$, otherwise the problem is trivial with $\hat{\boldsymbol{X}} = \mathbf{0}_{d\times n}, \boldsymbol{C} = \mathbf{0}_{d\times k}$. Then setting $\boldsymbol{X} \leftarrow \bar{\boldsymbol{X}}/v_{\mathsf{ub}}, \boldsymbol{C} \leftarrow \bar{\boldsymbol{C}}/v_{\mathsf{ub}}$ ensures that $\boldsymbol{X} \in [0,1]^{d\times n}, \boldsymbol{C} \in [0,1]^{d\times k}$ while the k-means problem remains the same due to the translation and scaling invariance.

E Computational Resources and Implementation

All experiments were executed on a Intel i7 Core CPU (16 threads, 64GB memory), and a NVIDIA V100 GPU (32GB memory). Each experiment was executed with 10 random seeds and all results are aggregated across these 10 trials. The implementation is in Pytorch and available at this GitHub repository (`https://github.com/rithram/kmeans-trf`). We train models with 300 different configurations — 50 for each of the 6 different combinations of regularized min, softmax vs sparsemax, and initial token embedding, one-hot (`OH`) vs binary (`BN`) vs none (`NA`). With 10 repetitions for each, we train a total of 3000 models, each taking around 15-20 minutes to train, for a total training compute time of 1000 hours. The loss surface generation for figure 3 took about 15 minutes per model for a total of 1 hour of compute. Each of the inference evaluations in section 4 took around 5 minutes each for total of $(25 + 2 + 4) \times 5 \approx 150$ minutes. The timed runs in figure 1 and table 1 in section 2 and figure 17 in appendix F are performed on a NVIDIA RTX PRO 3000 Blackwell Generation Laptop GPU.

F Ablative Experiments

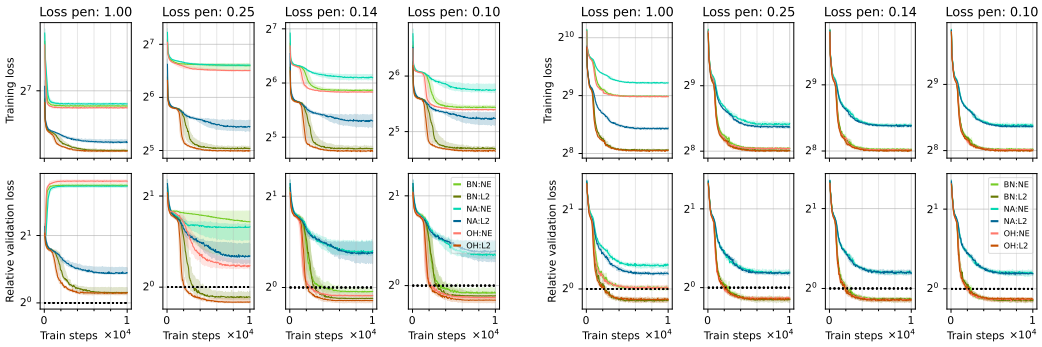
In this section, we present ablations across various axes, presenting the training loss trajectory as the training progresses, and the relative validation loss. Given a transformer parameterized with Θ, for an input $(\boldsymbol{X}, \boldsymbol{C}) \sim \mathcal{D}$ sampled from the distribution $\mathcal{D}$ of clustering tasks, we will denote the updated centers after a single transformer layer as $\bar{\boldsymbol{C}}(\Theta)$. Then, the training loss $\mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim\mathcal{D}}\tilde{\mathcal{L}}^\tau_\Omega(\boldsymbol{X}, \bar{\boldsymbol{C}}(\Theta))$ (defined in equation (6)) is the smoothed upper-bound of the desired k-means loss $\mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim\mathcal{D}}\mathcal{L}(\boldsymbol{X}, \bar{\boldsymbol{C}}(\Theta))$ (defined in equation (1)) with $\mathcal{L}(\boldsymbol{X}, \bar{\boldsymbol{C}}(\Theta)) \leq \tilde{\mathcal{L}}^\tau_\Omega(\boldsymbol{X}, \bar{\boldsymbol{C}}(\Theta))$, and the inequality approaching equality as the temperature $\tau \to 0$. We define the *relative validation loss* as the ratio of the k-means objective with the centers $\bar{\boldsymbol{C}}(\Theta)$ obtained with a single transformer layer to the k-means objective with the centers $\boldsymbol{C}^\star$ generated from a single iteration of Lloyd's algorithm (Algorithm 2) with the same set of points $\boldsymbol{X}$ and initial centers $\boldsymbol{C}$. Thus the relative validation error is given by

$$\mathbb{E}_{(\boldsymbol{X},\boldsymbol{C})\sim\mathcal{D}} \frac{\mathcal{L}(\boldsymbol{X}, \bar{\boldsymbol{C}}(\Theta))}{\mathcal{L}(\boldsymbol{X}, \boldsymbol{C}^\star)}. \tag{20}$$

Note that, for estimating the training loss and the relative validation loss, we use different sets of clustering tasks, and the clustering tasks in the validation set are not used for training. A relative validation error of less than 1 denotes that the single transformer layer outperforms Lloyd's algorithm (on average) when considered for a single iteration.

Each figure in these ablations has 6 curves corresponding to (i) the min-regularizer Ω — the negative Shannon entropy or NE-regularizer (denoted by **NE**) vs the L2-regularizer (denoted by **L2**), and (ii) the use of the one-hot initial token embeddings $(E^{\mathtt{OH}}_X, E^{\mathtt{OH}}_C)$ defined in (`OH`) with $d_{\mathsf{emb}} = d+k$ and denoted by **OH** vs the binary initial token embeddings $(E^{\mathtt{BN}}_X, E^{\mathtt{BN}}_C)$ defined in (`BN`) with $d_{\mathsf{emb}} = d + \lceil \log_2 k \rceil$ and denoted by **BN** vs no initial token embedding with $d_{\mathsf{emb}} = d$ denoted by **NA**. Thus we have 6 lines corresponding to labels BN:NE, BN:L2, NA:NE, NA:L2, OH:NE, OH:L2. Each experiment is executed for 10 trials with different random seeds, and each curve reports the median performance metrics, with a ribbon around it denoting the inter-quartile range of the performance metrics.

Varying the min-regularization penalty τ in the smoothed loss $\tilde{\mathcal{L}}^\tau_\Omega$. In Figure 10, we present the effect of varying the min-regularization penalty τ in the smoothed upper-bound $\tilde{\mathcal{L}}^\tau_\Omega$, highlighting that,

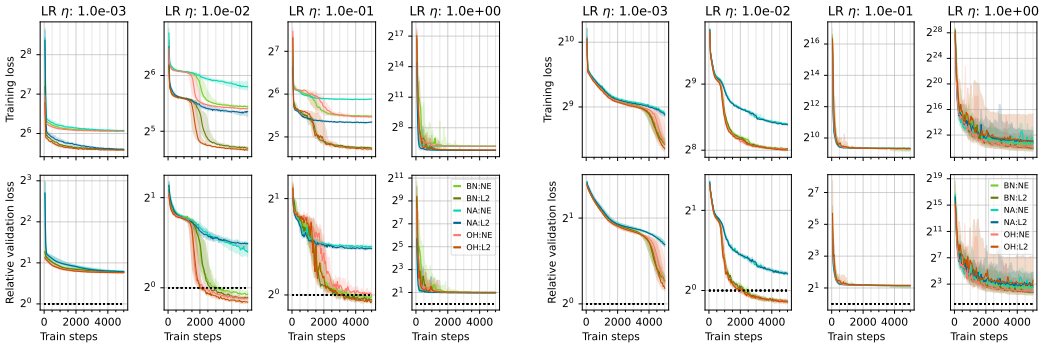


(a) Clustering tasks in $d = 4$ dimensions.

(b) Clustering tasks in $d = 32$ dimensions.

Figure 10: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) for varying values of the min-regularization penalty τ in the training loss $\tilde{\mathcal{L}}_\Omega^\tau$ (the smoothed upper-bound of the discrete non-differentiable k-means objective $\mathcal{L}$). This is an extended version of Figure 2a. We consider $\tau \in \{1, 0.25, 0.14, 0.1\}$. These set of results correspond to a training with clustering tasks involving $n = 512$ points and $k = 6$ clusters with $d = 4$ (figure 10a) and $d = 32$ (figure 10b) dimensions. Here the initial learning rate $\eta = 0.01$ for the Adam optimizer with $M = 10000$ training steps with a task batch-size $B = 32$.

as the penalty τ decreases, the training loss (the upper-bound) becomes tighter (and thus has lower values), and the performance on the validation tasks actually starts matching or even outperforming 1-step Lloyd's algorithm. Note that, for the larger value of d, the number of dimensions, the min-regularization penalty τ does not need to be too low for strong performance. This result also highlights that the L2-regularized min provides tighter upper-bounds, which translate to better validation performance, with the difference most apparent at large values of τ and the smaller data dimensionality of $d = 4$, when the penalization is stronger in general, and the upper-bounds are looser but smoother. The training loss converges faster at the higher values of τ, though the validation performance highlights that the faster training loss convergence does not necessary guarantee strong validation performance because of the gap between the smoothed upper-bound and the actual discrete clustering objective we care about.



(a) Clustering tasks in $d = 4$ dimensions.

(b) Clustering tasks in $d = 32$ dimensions.

Figure 11: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) for varying values of the initial learning rate η for the Adam optimizer selected from $\{0.001, 0.01, 0.1, 1.0\}$ with $M = 5000$ training steps with a task batch-size $B = 32$. These set of results correspond to a training with clustering tasks involving $n = 512$ points and $k = 6$ clusters with $d = 4$ (figure 11a) and $d = 32$ (figure 11b) dimensions with a min-regularization penalty $\tau = 0.1$.

Varying the learning rate η in Algorithm 1. In Figure 11, we present the effect of varying the initial learning rate in the Adam optimizer. The results indicate that, for low and moderately high

dimensions d, a low learning rate of 0.001 shows monotonic convergence but is slower than a learning rate of 0.01. On the other hand, larger values of learning rates such as 0.1 and 1.0 show that both the training and the validation losses do not decrease monotonically, especially at the higher $d = 32$ dimensions. The larger learning rates ($\eta = 0.1, 1.0$ for $d = 32$ and $\eta = 1.0$ for $d = 4$) converge to a worse local minima, and thus do not generalize well. In all cases where the training converges favourably, the L2-regularized $\min$ continues to converge to a better minima (both in terms of training loss and validation) than the NE-regularized $\min$ (controlling for all other factors).

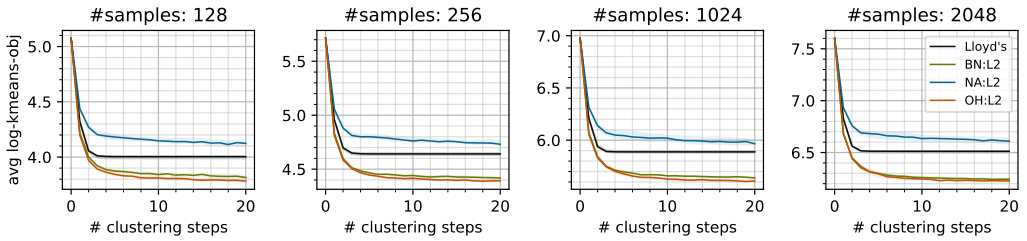
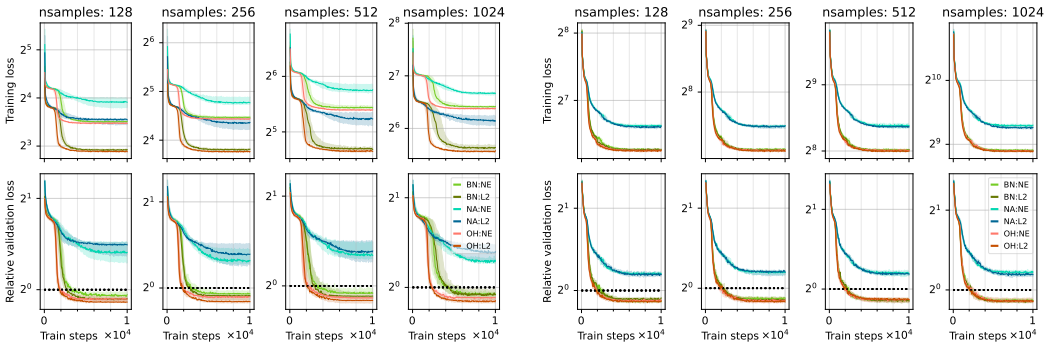


Figure 12: **"Length-generalization" of learned models.** We consider models trained with $n = 512, d = 32, k = 10, \tau = 0.1$ utilized for $T = 20$ step clustering on tasks with $\{128, 256, 1024, 2048\}$ points. We only consider L2-regularized $\min$ and the three forms of token embeddings, `OH`, `BN` and `NA` as described in Figure 2. The performance of Lloyd's algorithm is plotted in black.

Evaluating the "length generalization" of models, learning on one n, and generalizing to other values of n Here we evaluate the ability of these trained models to length generalize — the ability of models to be trained on problems of a certain size, but generalize to problems of different (often larger) sizes. In the context of clustering, we train the models on (single-step) clustering tasks with $n = 512$ samples (for a mixture of `normal` distributions), and then evaluate their (multi-step) clustering performance on problem with varying number of samples (again from the same distribution as it was trained on) in Figure 12. The results show that *the learned model can generalize to larger clustering problems*. The models using the (`OH`) and (`BN`) token embeddings outperform Lloyd's iterations, both for smaller problems (with 128 and 256 samples) and larger problems (with 1024 and 2048 samples).



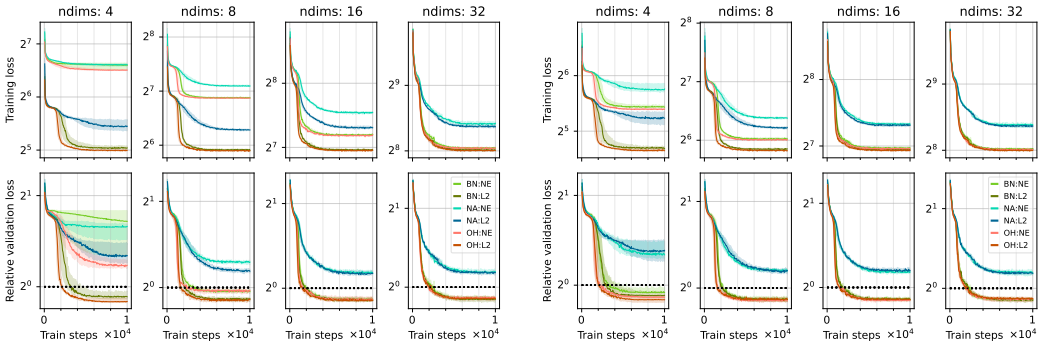
(a) Clustering tasks in $d = 4$ dimensions.

(b) Clustering tasks in $d = 32$ dimensions.

Figure 13: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) for varying values of the number of points n in the clustering tasks used for training the transformers, with $n \in \{128, 256, 512, 1024\}$. These set of results correspond to a training with clustering tasks involving $k = 6$ clusters with $d = 4$ (figure 13a) and $d = 32$ (figure 13b) dimensions. Here we use the initial learning rate $\eta = 0.01$ for the Adam optimizer with $M = 10000$ training steps, a task batch-size $B = 32$, and the $\min$-regularization penalty $\tau = 0.1$.

Varying the number of points n to be clustered in the training tasks. In Figure 13, we study effect of varying the number of points n in the clustering tasks $(\boldsymbol{X}, \boldsymbol{C})$ with $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ used

during training, selecting $n \in \{128, 256, 512, 1024\}$ for clustering tasks in $d \in \{4, 32\}$ dimensional Euclidean space. The results indicate that the number of points in the clustering tasks do not seem to have a significant effect on the training convergence or in-distribution generalization for any data dimensionality d, $\min$-regularizer Ω or initial token embedding scheme (`OH` vs `BN` vs `NA`). This combined with our length-generalization results in Figure 12 indicate that a good strategy would be to train the transformer with clustering tasks with a small n as that would be computationally efficient, and still be able to use this transformer for larger clustering tasks with larger number of points. Note that, for a fixed d and k and initial token embedding choice (`OH` vs `BN` vs `NA`), the number of learnable parameters remains the same as we vary n in the training clustering tasks.



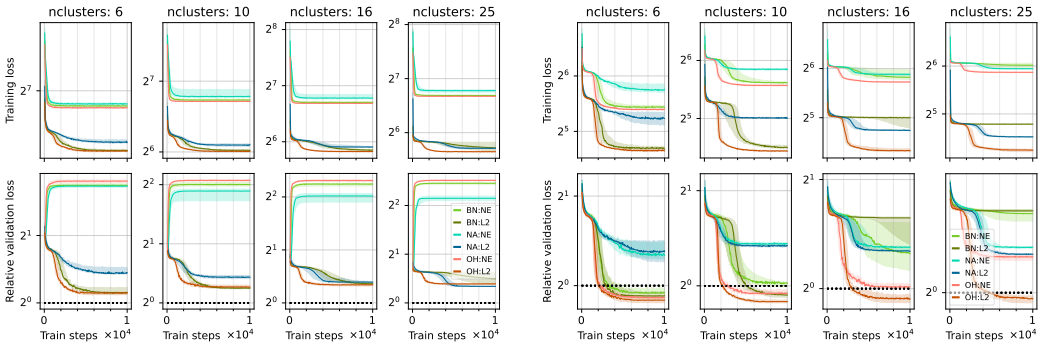
(a) $\min$-regularization penalty $\tau = 0.25$.

(b) $\min$-regularization penalty $\tau = 0.1$.

Figure 14: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) for varying values of the dimensionality d of the clustering tasks used for training the transformers, with $d \in \{4, 8, 16, 32\}$. This is an extended version of Figure 2b. These set of results correspond to a training with clustering tasks involving $n = 512$ points, $k = 6$ clusters, trained with the smoothed loss with a loss $\min$-regularizer penalty $\tau = 0.25$ (figure 14a) and $\tau = 0.1$ (figure 14b). Here the initial learning rate $\eta = 0.01$ for the Adam optimizer with $M = 10000$ training steps, each with a task batch-size $B = 32$.

Varying the data dimensionality d of the clustering tasks. In Figure 14, we study the effect of the data dimensionality d of the clustering tasks used to train the transformer. We consider two different values for the $\min$-regularization penalty τ, with smaller value of τ implying tighter upper-bounds to the discrete clustering loss. For the larger penalty value of $\tau = 0.25$ in Figure 14a, we can see that the L2-regularized $\min$ based smoothed loss converges and generalizes better than the NE-regularized $\min$ especially at a low $d = 4$; also note that the `OH` token embedding converges faster than the `BN` token embeddings as previously discussed (though both generalize comparably). However, as the data dimensionality d increases, this difference between L2-regularized and NE-regularized $\min$ vanishes. This is because, as we consider all $\boldsymbol{X} \in [0, 1]^{d \times n}$ and $\boldsymbol{C} \in [0, 1]^{d \times k}$, the pairwise squared Euclidean distances generally increases with d, and thus the regularized $\min$ produces tighter upper-bounds with increasing d for a fixed $\min$-regularization penalty τ. We will discuss this further in Appendix H. Note that the number of parameters in the transformer model grows quadratically with d, but we do not see any clear adverse effect of this increase in the number of learnable parameters.

Varying the number of clusters k in the training tasks. In Figure 15, we carefully study the effect of varying the number of clusters k in the training tasks to learning convergence and generalization. To understand the interplay of the number of clusters k, data dimensionality d and the $\min$-regularization penalty τ, we consider a pair of values of d and for τ, and for each of the 4 combinations, we vary $k \in \{6, 10, 16, 25\}$. First, in Figure 15a, we consider a low $d = 4$ and high penalty $\tau = 1$, implying a looser smoother upper-bound. In this case, the L2-regularized $\min$ is able to learn and somewhat generalize, but is unable to match the results of a single Lloyd's iteration (the black dotted line) for any value of k; the NE-regularized $\min$ is unable to learn at all as is evidenced by the high relative loss on the single-step validation clustering tasks. One observation is that the convergence (and generalization) of `BN` token embedding relatively deteriorates with increasing k as one would expect from our results in Theorem 2, where the Lipschitz constant depends on $\sqrt{d + \lceil \log_2 k \rceil}$, and thus increases with k (thereby hurting convergence). In contrast, the `OH` token embedding does not seem



(a) $d = 4, \tau = 1$.

(b) $d = 4, \tau = 0.1$.

(c) $d = 32, \tau = 1$.

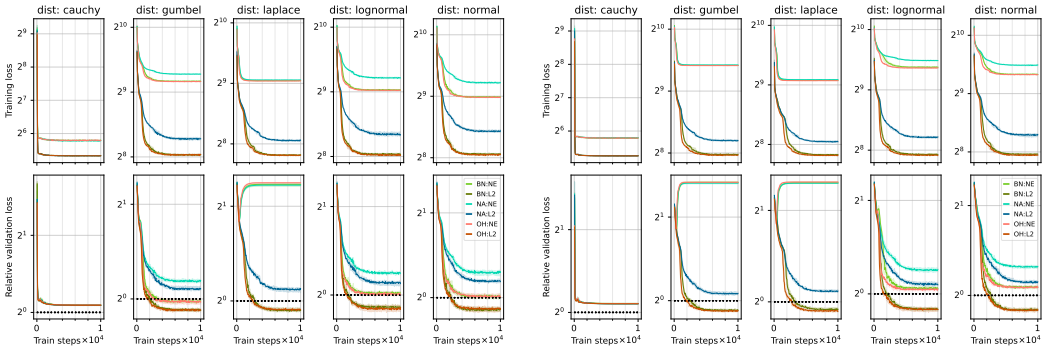
(d) $d = 32, \tau = 0.1$.

Figure 15: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) for varying values of the number of clusters k in the clustering tasks used for training the transformers, with $k \in \{6, 10, 16, 25\}$. These set of results correspond to a training with clustering tasks with $n = 512$ points. To highlight the interplay of the dimensionality d and the $\min$-regularization penalty τ, we consider four cases: $d = 4, \tau = 1$ (figure 15a), $d = 4, \tau = 0.1$ (figure 15b), $d = 32, \tau = 1$ (figure 15c), and $d = 32, \tau = 0.1$ (figure 15d). In all cases, the initial learning rate $\eta = 0.01$ for the Adam optimizer with $M = 10000$ training steps each with a task batch-size $B = 32$.

to be affected by the increase in k, which corroborates our theoretical results in Theorem 2 where the Lipschitz constant would depend on $(d+1)/\sqrt{d+k}$, and thus is not adversely affected by increasing k. Fixing the dimension $d = 4$, and lowering the $\min$-regularization $\tau = 0.1$ in Figure 15b, thereby using a tighter (less smooth) upper-bound for training, we see that the NE-regularized $\min$ is able to somewhat learn and generalize for smaller values of k, with its performance deteriorating with increasing k; L2-regularized $\min$ with `OH` token embedding does significantly better and outperforms Lloyd's for all values of k considered. For smaller values of k, the `BN` token embedding is able to outperform Lloyd's but converges and generalizes slower than `OH` as our theory suggests; for larger values of k, the convergence and generalization of `BN` token embedding slows down considerably, and even underperforms the model with no token embedding (`NA`). As we increase the data dimensionality to $d = 32$ and utilize a larger $\min$-regularization penalty $\tau = 1$ in Figure 15c, we see that L2-regularized $\min$ with `OH` and `BN` token embeddings outperforms Lloyd's for all values of k. However, with NE-regularized $\min$, the smoothed upper-bound $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ for a fixed τ gets progressively looser with larger values of k, and this is evident in the worsening validation losses. Finally, fixing the data dimensionality to $d = 32$ and lowering the $\min$-regularization penalty to $\tau = 0.1$ for a tighter clustering loss upper-bound in Figure 15d, we see that both L2-regularized and NE-regularized $\min$ for both `OH` and `BN` token embeddings outperform Lloyd's on the validation tasks. The only thing of note here is that, at the largest value of $k = 25$, the model with the `OH` token embeddings experiences intermediate numerical issues for 3/10 random trials in figure 15d which throws the model off, and the training does not recover in the allotted 10000 steps, thereby broadening the inter-quartile range

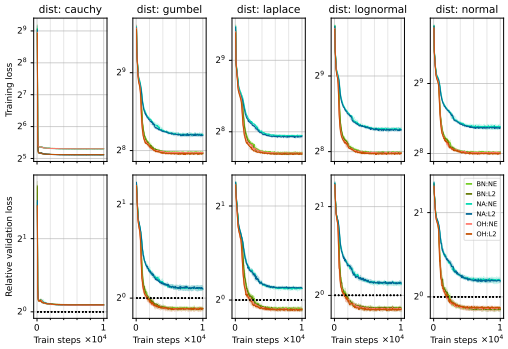
unfavourably; note that the median performance continues to outperform Lloyd's. We would like to note a couple of points regarding this:

- First, we did not use the standard practice of gradient clipping in any of our results, and our results indicate that the learning problem is well-structured enough for the training to easily converge in almost all cases (also highlighted by the loss landscapes in figure 3).
- Second, for the 3/10 random trials where the training gets stuck, adding gradient clipping mitigates the issue of the numerical instability, and these previously failing training runs converge as expected consistent with the other random trials. We presented the results in figure 15d without gradient clipping to be consistent across all evaluations.
- Finally, this only occurs with the `OH` token embedding for both NE (softmax) and L2 (sparsemax) regularization, where $d_{\text{emb}} = d + k = 57$ with $d = 32, k = 25$, and the training is not able to find the right basin of attraction in this relatively high $O(d_{\text{emb}}^2) \sim 3600$ dimensional loss landscape once the training is destabilized. In contrast, for the same setup (same random seed and clustering tasks for training), the training converges fine with the `BN` token embedding, where $d_{\text{emb}} = d + \lceil \log_2 k \rceil = 37$ with $d = 32, k = 25, \lceil \log_2 k \rceil = 5$, implying that the loss landscapes are easier to navigate in a relatively lower dimensionality $O(d_{\text{emb}}^2) \sim 1600$.

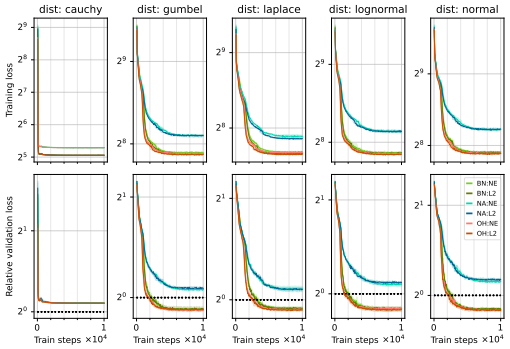


(a) $k = 6, \tau = 1.$

(b) $k = 10, \tau = 1.$



(c) $k = 6, \tau = 0.1.$

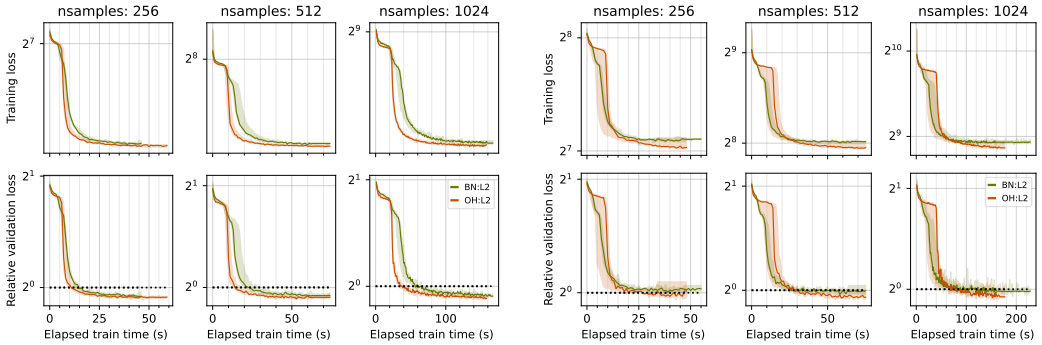


(d) $k = 10, \tau = 0.1.$

Figure 16: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) for varying the distribution of the points to the clustered. These set of results correspond to a training with clustering tasks with $n = 512$ points in $d = 32$ dimensions, and is an extended version of Figure 4. To highlight the interplay of the number of clusters k and the loss temperature τ, we consider four cases: $k = 6, \tau = 1$ (figure 16a), $k = 10, \tau = 1$ (figure 16b), $k = 6, \tau = 0.1$ (figure 16c), and $k = 10, \tau = 0.1$ (figure 16d). In all cases, the initial learning rate $\eta = 0.01$ for the Adam optimizer with $M = 10000$ training steps and a per-step task batch-size $B = 32$.

Varying the distribution of the data from which the points are generated for the training tasks. In Figure 16, we build upon Figure 4, and provide an extended evaluation of the effect of the data distribution from which the set of points $\boldsymbol{X} \in [0, 1]^{d \times n}$ are generated for any clustering tasks $(\boldsymbol{X}, \boldsymbol{C})$. As with Figure 4, we see that our proposed learning scheme is unable to learn from and generalize to clustering tasks generated from a mixture of `cauchy` distributions under any of the configuarations

considered. Focusing on the remaining distributions, namely `gumbel`, `laplace`, `lognormal`, and `normal`, we see that, at a larger $\min$-regularization of $\tau = 1$ in Figure 16a and Figure 16b, the L2-regularized $\min$ provides better generalization than the NE-regularized $\min$, with NE-regularized $\min$ failing to generalize at all with the `gumbel` and `laplace` distributions; this behaviour is more evident with the larger $k = 10$ value in Figure 16b. With a smaller $\min$-regularization penalty $\tau = 0.1$ providing tighter clustering loss upper-bounds for training in Figure 16c and Figure 16d, we see that both NE-regularized and L2-regularized $\min$ with `OH` and `BN` token embeddings can outperform Lloyd's across all data distributions.



(a) $k = 16, d = 16, \tau = 1$.

(b) $k = 25, d = 32, \tau = 1$.

Figure 17: Training loss convergence (top row) and relative validation loss from equation (20) (bottom row) in terms of actual training time for varying values of $n \in \{256, 512, 1024\}$. In one case, we use $k = 16, d = 16$ with $\tau = 1$ (figure 17a), while in another, we use $k = 25, d = 32$, again with $\tau = 1$ (figure 17b). Here we limit ourselves to the L2-regularized $\min$, and compare the `OH` and `BN` token embeddings, giving us only the BN:L2 and the OH:L2 curves.

Interplay of the lower computational overhead and slower convergence of the leaner (`BN`). The results in figure 1 and table 1 indicate that the proposed `BN` token embeddings can provide significant computational gains over the existing `OH` token embeddings primarily on account of having smaller embeddings sizes d_{emb} (`OH` with $(d+k)$ vs `BN` with $(d+\lceil \log_2 k \rceil)$). However, the results in theorem 2 indicate that the model with the `BN` token embeddigns will have slower convergence guarantees (in terms of the number of optimization steps). To understand their interplay, we consider two settings of (d, k), and compare the convergence (of training and validation loss) in terms of actual elapsed time in figure 17. These training runs use gradient clipping to ensure that all the model trainings avoided any gradient explosions and numerical issues. With moderately small $d = k = 16$ in figure 17a, we see that the model with the `BN` token embeddings converges signficantly slower than the model with the `OH` token embedding, though both appear to eventually converge close to each other, and better than single step Lloyd's update for all values of n. However, as the data dimensionality d and the number of clusters k grows to $d = 32, k = 25$ in figure 17b, we see that, initially, the model with `BN` token embeddings converges faster in terms of the runtime, with both methods eventually matching or outperforming single-step Lloyd's updates by the end of their 10000 training steps (which take different amounts of time for both).

G Additional Figures

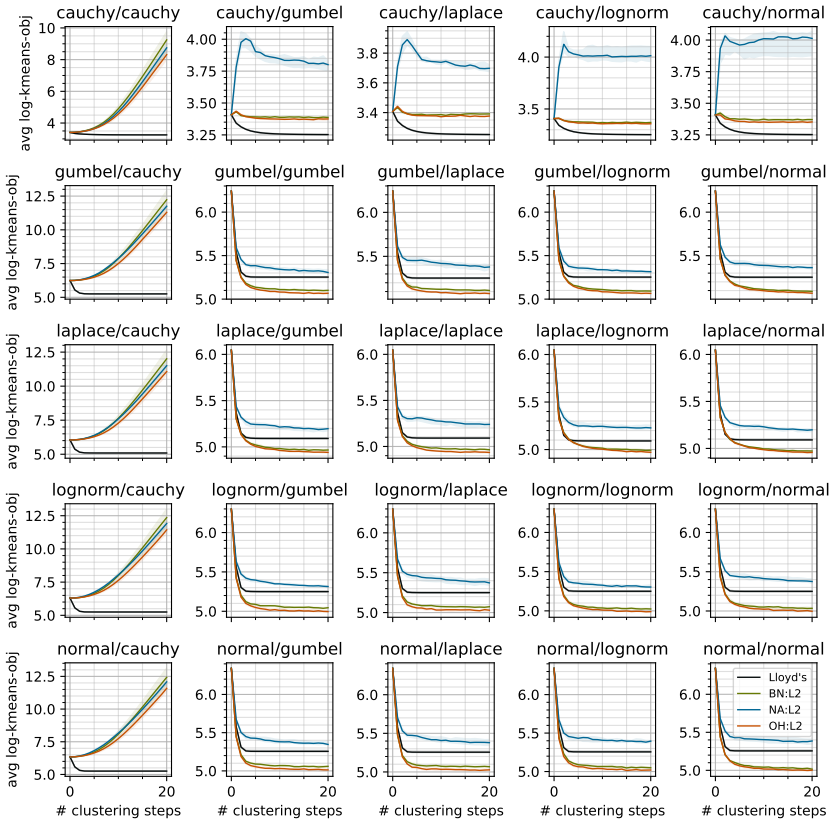


Figure 18: **Generalization to different distribution families.** Larger version of Figure 8. The models are trained on one distribution family (each column maps to a training family), and tested for multi-step ($T = 20$) clustering from a different distribution family (each row maps to a test family). We consider 5 distributions — `cauchy`, `gumbel`, `laplace`, `lognormal` and `normal` — giving us a 5×5 pairwise evaluations. Note that the pairwise evaluations on the "diagonal" correspond to training and evaluations tasks being sampled from the same distribution (though the tasks themselves are different, and the training only considers single-step clustering while we are evaluating multi-step clustering). We only consider models trained with the L2-regularized $\min$ and the three forms of token embeddings, `OH`, `BN` and `NA` as described in Figure 2. Each curve corresponds to the inter-quartile range aggregated over 10 trials. The performance of Lloyd's is in black.

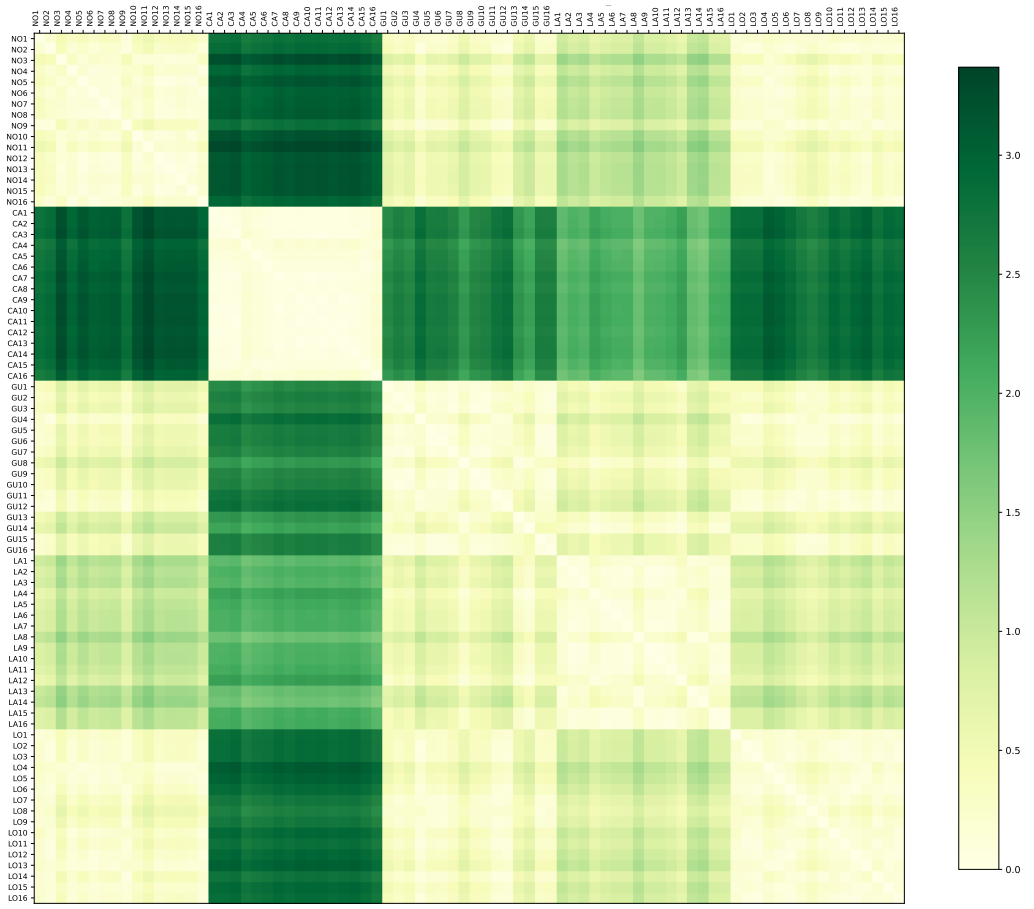


Figure 19: **Pairwise task dissimilarities.** A version of Figure 9 with 16 tasks per distribution family. We use the 16 tasks from each of the families and compute the pairwise Earth-mover distance between their respective distribution of the squared Euclidean distances $\{\|\boldsymbol{X}_{:,i} - \boldsymbol{C}_{:,j}\|^2, i \in [n], j \in [k]\}$ between the points $\boldsymbol{X}$ and the initial centers $\boldsymbol{C}$, giving us a 80×80 matrix. Each row and column corresponds to a single task, and the tasks are arranged into family-based blocks in the following order: `normal`, `cauchy`, `gumbel`, `laplace`, `lognormal`. *Light shades show low distances.* The "diagonal" corresponds to tasks distances to themselves, and thus are zero (the lightest shade). `cauchy` tasks are only similar to other `cauchy` tasks, and significantly dissimilar to tasks from other distribution families, with the `laplace` distribution appearing the least dissimilar (but still quite dissimilar).

H Smoothed Clustering Objective via Regularized Minimum

For the purposes of clustering, our main target is the non-differentiable per-point k-means loss $L : \mathbb{R}^d \times \mathbb{R}^{d\times k} \to \mathbb{R}$, given a sample $\boldsymbol{x} \in \mathbb{R}^d$ and cluster centers $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k] \in \mathbb{R}^{d\times k}$:

$$L(\boldsymbol{x}, \boldsymbol{C}) = \min_{j\in[k]} \|\boldsymbol{x} - \boldsymbol{c}_j\|^2, \tag{21}$$

with the complete k-means objective in equation (1) given by $\sum_{i\in[n]} L(\boldsymbol{x}_i, \boldsymbol{C})$. We can obtain a differentiable, trainable surrogate for $L(\boldsymbol{x}, \boldsymbol{C})$ in a few steps. First, by defining function $R : \mathbb{R}^d \times \mathbb{R}^{d\times k} \to \mathbb{R}^k$ where $\boldsymbol{r} = R(\boldsymbol{x}, \boldsymbol{C})$ has $r_j \triangleq \|\boldsymbol{x} - \boldsymbol{C}_{:,j}\|^2$ for each $j \in [k]$, we have equivalently

$$L(\boldsymbol{x}, \boldsymbol{C}) = \min_{j\in[k]} r_j \quad \text{where} \quad \boldsymbol{r} = R(\boldsymbol{x}, \boldsymbol{C}). \tag{22}$$

Let Δ^k be the $(k-1)$-dimensional probability simplex with $\Delta^k = \{\boldsymbol{p} \in \mathbb{R}^k_{\geq 0} : \|\boldsymbol{p}\|_1 = 1\}$. Since $\min_{\boldsymbol{\lambda}\in\Delta^k} \langle \boldsymbol{\lambda}, \boldsymbol{r}\rangle = \min_{j\in[k]} r_j$, we could *also* equivalently say, again with $\boldsymbol{r} = R(\boldsymbol{x}, \boldsymbol{C})$,

$$L(\boldsymbol{x}, \boldsymbol{C}) \triangleq \mathtt{L}(\boldsymbol{r}) = \langle P(\boldsymbol{r}), \boldsymbol{r}\rangle, \quad \text{where } P(\boldsymbol{r}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{\lambda}\in\Delta^k} \langle \boldsymbol{\lambda}, \boldsymbol{r}\rangle \tag{23}$$

We haven't changed $L(\boldsymbol{x}, \boldsymbol{C})$, which remains non-differentiable. However, we can regularize the minimization of $\boldsymbol{\lambda}$, obtaining a surrogate $\tilde{L}^\tau_\Omega(\boldsymbol{x}, \boldsymbol{C})$, defined as

$$\tilde{L}^\tau_\Omega(\boldsymbol{x}, \boldsymbol{C}) \triangleq \tilde{\mathtt{L}}^\tau_\Omega(\boldsymbol{r}) = \langle P^\tau_\Omega(\boldsymbol{r}), \boldsymbol{r}\rangle, \quad \text{where } P^\tau_\Omega(\boldsymbol{r}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{\lambda}\in\Delta^k} \langle \boldsymbol{\lambda}, \boldsymbol{r}\rangle + \tau\Omega(\boldsymbol{\lambda}), \tag{24}$$

where $\tau > 0$ is the penalty on the regularization $\Omega : \Delta^k \to \mathbb{R}$, with $\tau = 0$ yields the unregularized $\arg\min$. If Ω is strongly convex, $\tilde{\mathtt{L}}^\tau_\Omega(\boldsymbol{r})$ is differentiable with respect to $\boldsymbol{r}$. Note that, for any $\tau > 0$, $P^\tau_\Omega(\boldsymbol{r}) = P^1_\Omega(\boldsymbol{r}/\tau)$, and thus we can just operate with $P^1_\Omega(\boldsymbol{r})$. We will write P^1_Ω as P_Ω for the ease of exposition with:

$$P_\Omega(\boldsymbol{r}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{\lambda}\in\Delta^k} \langle \boldsymbol{\lambda}, \boldsymbol{r}\rangle + \Omega(\boldsymbol{\lambda}) \tag{25}$$

For $\boldsymbol{\lambda} \in \Delta^k$, if $\Omega(\boldsymbol{\lambda}) = \sum_{j\in[k]} \lambda_j \log \lambda_j$, the negative Shannon entropy, then the corresponding $P_{\text{NE}}(\boldsymbol{r}/\tau)$ are the softmax weights (actually "softmin" weights) with the temperature τ, giving us the following "soft-min" k-means objective $\tilde{L}^\tau_{\text{NE}}$:

$$\begin{aligned} P_{\text{NE}}(\boldsymbol{r}/\tau)_j &= \frac{\exp(-r_j/\tau)}{\sum_{j'\in[k]} \exp(-r_{j'}/\tau)}, \\ \tilde{L}^\tau_{\text{NE}}(\boldsymbol{x}, \boldsymbol{C}) \triangleq \tilde{\mathtt{L}}^\tau_{\text{NE}}(\boldsymbol{r}) &= \langle P_{\text{NE}}(\boldsymbol{r}/\tau), \boldsymbol{r}\rangle = \sum_{j\in[k]} \frac{\exp(-r_j/\tau) r_j}{\sum_{j'\in[k]} \exp(-r_{j'}/\tau)}. \end{aligned} \tag{26}$$

Note that $L(\boldsymbol{x}, \boldsymbol{C}) \leq \tilde{L}^\tau_{\text{NE}}(\boldsymbol{x}, \boldsymbol{C})$ with the equality achieved when $\tau \to 0$. In the learning procedure, we use equation (26) instead of equation (21) (or equation (23)) to have a differentiable objective.

An alternative choice for the regularization for $\boldsymbol{\lambda} \in \Delta^k$ is $\Omega(\boldsymbol{\lambda}) = {}^1\!/_2 \|\boldsymbol{\lambda}\|_2^2$, yielding

$$P_{\text{L2}}(\boldsymbol{r}) = \operatorname*{arg\,min}_{\boldsymbol{\lambda}\in\Delta^k} \langle \boldsymbol{\lambda}, \boldsymbol{r}\rangle + {}^1\!/_2 \|\boldsymbol{\lambda}\|_2^2 = \operatorname*{arg\,min}_{\boldsymbol{\lambda}\in\Delta^k} {}^1\!/_2 \|\boldsymbol{r}\|^2 + \langle \boldsymbol{\lambda}, \boldsymbol{r}\rangle + {}^1\!/_2 \|\boldsymbol{\lambda}\|_2^2 = \operatorname*{arg\,min}_{\boldsymbol{\lambda}\in\Delta^k} {}^1\!/_2 \|\boldsymbol{r} + \boldsymbol{\lambda}\|^2,$$

the closest point to $(-\boldsymbol{r})$ in Δ^k. For $\tau > 0$ this yields the sparsemax weights $P_{\text{L2}}(\boldsymbol{r}/\tau)$, which can be computed as:

$$P_{\text{L2}}(\boldsymbol{r}/\tau)_j = [-(r_j/\tau) - \kappa(-\boldsymbol{r}/\tau)]_+, \quad \text{with } [a]_+ = \max\{a, 0\}, a \in \mathbb{R}, \tag{27}$$

where $\kappa : \mathbb{R}^k \to \mathbb{R}$ is the appropriately defined threshold function such that, for any $\boldsymbol{z} \in \mathbb{R}^k$, $\sum_{j\in[k]} [z_j - \kappa(\boldsymbol{z})]_+ = 1$; see Wang and Carreira-Perpinán [37] and Martins and Astudillo [30, Proposition 1]. For any $\boldsymbol{z} \in \mathbb{R}^k$, let its entries be ordered as $z_{(1)} \geq z_{(2)} \geq \cdots \geq z_{(k)}$. Then $\kappa(\boldsymbol{z})$ is computed as follows:

$$N(\boldsymbol{z}) = \max\left\{ m \in [k] : 1 + m z_{(m)} > \sum_{j\leq m} z_{(j)} \right\}, \quad \kappa(\boldsymbol{z}) = \frac{\left(\sum_{j\leq N(\boldsymbol{z})} z_{(j)}\right) - 1}{N(\boldsymbol{z})}. \tag{28}$$

This results in sparser weights, where some of the $P_{\text{L2}}(\boldsymbol{r}/\tau)_j$ values can be exactly zero. This gives us the following differentiable upper-bound of the k-means objective:

$$\tilde{L}^{\tau}_{\text{L2}}(\boldsymbol{x}, \boldsymbol{C}) \triangleq \tilde{\text{L}}^{\tau}_{\text{L2}}(\boldsymbol{r}) = \langle P_{\text{L2}}(\boldsymbol{r}/\tau), \boldsymbol{r} \rangle = \sum_{j \in [k]} \left[-(r_j/\tau) - \kappa(-\boldsymbol{r}/\tau) \right]_+ r_j. \tag{29}$$

Again $L(\boldsymbol{x}, \boldsymbol{C}) \leq \tilde{L}^{\tau}_{\text{L2}}(\boldsymbol{x}, \boldsymbol{C})$ with equality now achieveable at $\tau > 0$ (as opposed to at $\tau \to 0$ for NE-regularization) and depends on this notion of "margin" [31]. We will also consider this as our learning loss. Using Blondel et al. [31, Proposition 1, Approximation Error], we can show that, for the k-means objective, for the same regularization penalty $\tau > 0$, the sparsemax-based upper-bound in equation (29) provides a tighter upper than the softmax-based upper-bound in equation (26).

Proposition 1 (Blondel et al. [31])**.** *Suppose* $\mathsf{domain}(\Omega)$ *contains all vectors in* $\mathcal{Y} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k\}$*, where* $\boldsymbol{e}_j$ *is the* j^{th} *characteristic vector. If such an* Ω *is* γ*-strongly convex and bounded, with* $L \leq \Omega(\boldsymbol{\lambda}) \leq U$ *for all* $\boldsymbol{\lambda} \in \mathsf{domain}(\Omega)$*, then* $1/2 \left\| P(\boldsymbol{z}) - P_\Omega(\boldsymbol{z}) \right\|^2 \leq \frac{U-L}{\gamma}$ *for any* $\boldsymbol{z} \in \mathbb{R}^k$*.*

For our purposes $\mathsf{domain}(\Omega) = \Delta^k$ and thus, $\mathcal{Y} \subset \Delta^k$. Quantifying the tightness of the upper-bound, we have

$$\begin{aligned} 0 < \tilde{L}^{\tau}_{\Omega}(\boldsymbol{x}, \boldsymbol{C}) - L(\boldsymbol{x}, \boldsymbol{C}) &= \tilde{\text{L}}^{\tau}_{\Omega}(\boldsymbol{r}) - \text{L}(\boldsymbol{r}) \\ &= \langle P_\Omega(\boldsymbol{r}/\tau) - P(\boldsymbol{r}), \boldsymbol{r} \rangle \leq \left\| P(\boldsymbol{r}) - P_\Omega(\boldsymbol{r}/\tau) \right\| \left\| \boldsymbol{r} \right\|. \end{aligned} \tag{30}$$

Thus, a smaller value of $\left\| P(\boldsymbol{r}) - P_\Omega(\boldsymbol{r}/\tau) \right\|$ implies a tighter upper-bound. Note that for $\tau > 0$, $P(\boldsymbol{r}) = P(\boldsymbol{r}/\tau)$, and thus $\left\| P(\boldsymbol{r}) - P_\Omega(\boldsymbol{r}/\tau) \right\| = \left\| P(\boldsymbol{r}/\tau) - P_\Omega(\boldsymbol{r}/\tau) \right\|$, so it suffices to bound $\left\| P(\boldsymbol{z}) - P_\Omega(\boldsymbol{z}) \right\|$ for any $\boldsymbol{z} \in \mathbb{R}^k$

For sparsemax with $\Omega(\boldsymbol{\lambda}) = 1/2 \left\| \boldsymbol{\lambda} \right\|_2^2$, Ω is γ-strongly convex with $\gamma = 1$, and $\Omega(\boldsymbol{\lambda}) = 1/2 \left\| \boldsymbol{\lambda} \right\|_2^2 \in [1/2k, 1/2]$ for any $\boldsymbol{\lambda} \in \Delta^k$. Thus $1/2 \left\| P(\boldsymbol{z}) - P_\Omega(\boldsymbol{z}) \right\|^2 \leq 1/2(1 - 1/k) < 1/2$ for any $\boldsymbol{z} \in \mathbb{R}^k$ using proposition 1. Thus $\left\| P(\boldsymbol{r}/\tau) - P_\Omega(\boldsymbol{r}/\tau) \right\| \leq \sqrt{1 - 1/k} < 1$.

For softmax with $\Omega(\boldsymbol{\lambda}) = \sum_{j \in [k]} \lambda_j \log \lambda_j$, with $0 \leq \lambda_j \leq M \leq 1$, Ω is γ-strongly convex with $\gamma = 1/M$. Naively $M = 1$ and thus, Ω would be 1-strongly convex. When $\boldsymbol{\lambda}$ is one-hot, $\Omega(\boldsymbol{\lambda}) = 0 = U$, while $\lambda_j = 1/k \forall j \in [k]$ gives us $\Omega(\boldsymbol{\lambda}) = -\log k = L$. Thus, we have

$$1/2 \left\| P(\boldsymbol{z}) - P_\Omega(\boldsymbol{z}) \right\|^2 \leq \frac{U - L}{\gamma} = \log k. \tag{31}$$

Thus, for softmax, $\left\| P(\boldsymbol{r}/\tau) - P_\Omega(\boldsymbol{r}/\tau) \right\| \leq \sqrt{2 \log k}$, giving a looser bound than with sparsemax of $\sqrt{1 - 1/k}$.

Next, we will establish the Lipschitz continuity of the smoothed objective $\tilde{\text{L}}^{\tau}_{\Omega}$. First, let us assume that the regularized $\min$ operation is ω-input-stable. That is:

Assumption 1. *The regularized* $\min$ *operation* $P_\Omega : \mathbb{R}^k \to \Delta^k$ *in equation* (25) *is* ω_τ*-input-stable for a* τ*-dependent* $\omega_\tau > 0$*, such that, for any* $\boldsymbol{r}, \boldsymbol{r}' \in \mathbb{R}^k$

$$\left\| P_\Omega(\boldsymbol{r}/\tau) - P_\Omega(\boldsymbol{r}'/\tau) \right\|_1 \leq (\omega_\tau/\tau) \left\| \boldsymbol{r} - \boldsymbol{r}' \right\|_1. \tag{32}$$

This assumption is already satisfied for softmax with bounded inputs $\boldsymbol{r}, \boldsymbol{r}'$ [21, Lemma B.1]:

$$\left\| P_{\text{NE}}(\boldsymbol{r}/\tau) - P_{\text{NE}}(\boldsymbol{r}'/\tau) \right\|_1 \leq \frac{2 \exp(\delta/\tau)}{k\tau} \left\| \boldsymbol{r} - \boldsymbol{r}' \right\|_1, \tag{33}$$

$$\text{where } \delta \geq (\max_{j \in [k]} r_j) - (\min_{j \in [k]} r_j) \, \forall \boldsymbol{r} \qquad \text{and } \delta \geq (\max_{j \in [k]} r'_j) - (\min_{j \in [k]} r'_j) \, \forall \boldsymbol{r}', \tag{34}$$

with the corresponding $\omega_\tau = \exp(\delta/\tau)/k$. Thus we can show the following result:

Proposition 2. *The smoothed* k*-means objective* $\tilde{L}^{\tau}_{\Omega} : \mathbb{R}^k \to \mathbb{R}$ *in equation* (24) *is Lipschitz continuous, such that, with any* $\boldsymbol{r}, \boldsymbol{r}' \in [0, r^\star]^k$*, we have*

$$\left| \tilde{L}^{\tau}_{\Omega}(\boldsymbol{r}) - \tilde{L}^{\tau}_{\Omega}(\boldsymbol{r}') \right| = \left| \langle P_\Omega(\boldsymbol{r}/\tau), \boldsymbol{r} \rangle - \langle P_\Omega(\boldsymbol{r}'/\tau), \boldsymbol{r}' \rangle \right| \leq \left(1 + \frac{r^\star \omega_\tau}{\tau} \right) \left\| \boldsymbol{r} - \boldsymbol{r}' \right\|_1. \tag{35}$$

Proof. First, we have the following inequalities by simply applying the triangle inequalities:

$$\begin{aligned}
&|\langle P_\Omega(\boldsymbol{r}/\tau), \boldsymbol{r}\rangle - \langle P_\Omega(\boldsymbol{r}'/\tau), \boldsymbol{r}'\rangle| \\
&\quad \le |\langle P_\Omega(\boldsymbol{r}/\tau), (\boldsymbol{r}-\boldsymbol{r}')\rangle| + |\langle (P_\Omega(\boldsymbol{r}/\tau) - P_\Omega(\boldsymbol{r}'/\tau)), \boldsymbol{r}'\rangle| \\
&\quad \le \max_{j\in[k]} |P_\Omega(\boldsymbol{r}/\tau)_j| \, \|\boldsymbol{r}-\boldsymbol{r}'\|_1 + \max_{j\in[k]} |r'_j| \, \|P_\Omega(\boldsymbol{r}/\tau) - P_\Omega(\boldsymbol{r}'/\tau))\|_1 .
\end{aligned}$$

Noting that $\max_{j\in[k]} P_\Omega(\boldsymbol{r}/\tau)_j \le 1$ as $P_\Omega : \mathbb{R}^k \to \Delta^k$, and $r_j, r'_j \in [0, r^\star]$, and applying equation (32) gives us the result in equation (35). □

This result indicates that the Lipschitz constant of the smoothed loss usually increases as the min-regularization penalty τ decreases. In general, the Lipschitz constant depends on $(1/\tau)$. Furthermore, if ω_τ increases as τ decreases, then the Lipschitz constant also increases. With softmax, $\omega_\tau = \exp(\delta/\tau)/k$ also increases as τ decreases.

I Lipschitz-Ness of the Smoothed Upperbound of the Clustering Objective

In this section, we will establish the Lipschitz constant for the smoothed upper-bound $\tilde{\mathcal{L}}^\tau_\Omega$ of the original discrete non-differentiable clustering objective $\mathcal{L}$. We begin with some standard assumptions:

Assumption 2 (bounded inputs)**.** *For any clustering task $(\boldsymbol{X}, \boldsymbol{C}) \sim \mathcal{D}$ with $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ and $\boldsymbol{C} \in \mathbb{R}^{d\times k}$, we assume that any point and initial cluster center is bounded to the unit hypercube, that is, $\boldsymbol{X}_{:,i} \in [0,1]^d$ for all $i \in [n]$ and $\boldsymbol{C}_{:,j} \in [0,1]^d$ for all $j \in [k]$.*

We explicitly enforce this boundedness in all our experiments.

Assumption 3 (bounded model parameters)**.** *We assume that the query, key and value projection matrices $(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 1, \dots, 4$ have a bounded spectral norm throughout the learning process, that is,*

$$\|\boldsymbol{Q}_\mu\| \le \Gamma_1, \quad \|\boldsymbol{K}_\mu\| \le \Gamma_1, \quad \|\boldsymbol{V}_\mu\| \le \Gamma_2, \quad \mu = 1, \dots, 4. \tag{36}$$

For the ease of exposition, we assume that the query and key projection matrices share the norm bound.

Lemma 1 (bounded model outputs)**.** *Under Assumptions 2 and 3, and the attention based updates in Definition 3, the initial token embeddings, the intermediate token embeddings, and the cluster centers $\boldsymbol{C}(\Theta) \triangleq F^\gamma_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)$ output by the model $F^\gamma_{d:k}$ with initial token embedding scheme (E_C, E_X) for any clustering task $(\boldsymbol{X}, \boldsymbol{C})$ have bounded norms as follows:*

$$\begin{aligned}
&\forall i \in [n], \left\|\bar{\boldsymbol{X}}^{(0)}_{:,i}\right\| \le \sqrt{d}, && \left\|\bar{\boldsymbol{X}}^{(1)}_{:,i}\right\| \le (1+\Gamma_2)\sqrt{d} + \Gamma_2\sqrt{d_E}, \\
&\forall j \in [k], \left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\| \le \sqrt{d_E}, && \|\boldsymbol{C}(\Theta)_{:,j}\| \le (1 + 2\Gamma_2 + 2\Gamma_2^2)\sqrt{d_E}, \\
&\multicolumn{2}{c}{\textit{where } d_E = \begin{cases} d+1 & \textit{if using } (\texttt{OH}) \\ d + \lceil \log_2 k \rceil & \textit{if using } (\texttt{BN}) \end{cases}.}
\end{aligned} \tag{37}$$

Proof. Denoting $\bar{\boldsymbol{X}}^{(0)} \leftarrow E_X(\boldsymbol{X})$ and $\bar{\boldsymbol{C}}^{(0)} \leftarrow E_C(\boldsymbol{C})$, we can see that $\left\|\bar{\boldsymbol{X}}^{(0)}_{:,i}\right\| \le \sqrt{d}$ for any $i \in [n]$ given Assumption 2 and the fact that $\boldsymbol{Y}^{(0)} = \boldsymbol{0}_{(d_{\text{emb}}-d)\times n}$ for any initial token embedding E_X for the points. For the embedded initial cluster centers, for any $j \in [k]$, their norms $\left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\| \le \sqrt{d+1}$ with the OH token embedding scheme $E^{\texttt{OH}}_C$ and $\left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\| \le \sqrt{d + \lceil \log_2 k \rceil}$ with the binary token embedding scheme $E^{\texttt{BN}}_C$. Thus we can see that $\left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\| \le \sqrt{d_E}$ with d_E defined in equation (37).

Since the output of $\boldsymbol{C}(\Theta) \triangleq F^\gamma_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)$ is the first d rows $\bar{\boldsymbol{C}}^{(1)}_{:d,:}$ of the token embeddings for the cluster centers $\bar{\boldsymbol{C}}^{(1)}$ after a single transformer layer with updates defined as in Definition 3, then we have

$$\|\boldsymbol{C}(\Theta)_{:,j}\| = \left\|\bar{\boldsymbol{C}}^{(1)}_{:d,j}\right\| \le \left\|\bar{\boldsymbol{C}}^{(1)}_{:,j}\right\| = \left\|\bar{\boldsymbol{C}}^{(0)}_{:,j} + \left(\boldsymbol{V}_3 \sum_{i=1}^n a_{ji} \bar{\boldsymbol{X}}^{(1)}_{:,i}\right) + \left(\boldsymbol{V}_4 \sum_{j'=1}^k b_{jj'} \bar{\boldsymbol{C}}^{(0)}_{:,j'}\right)\right\| \tag{38}$$

$$\leq \left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\| + \max_{i\in[n]} \left\|\boldsymbol{V}_3 \bar{\boldsymbol{X}}^{(1)}_{:,i}\right\| + \max_{j'\in[k]} \left\|\boldsymbol{V}_4 \bar{\boldsymbol{C}}^{(0)}_{:,j'}\right\|, \tag{39}$$

where a_{ji} are the cross-attention weights and $b_{jj'}$ are the self-attention weights in the cluster center updates in equation (4). The first inequality holds above since the norm of the first d dimensions is smaller than the norm of all the d_{emb} dimensions. The last inequality applies triangle inequality, and the fact that the attention weights lie on a simplex (that is, $a_{ji} \in [0,1], \sum_i a_{ji} = 1$ and $b_{jj'} \in [0,1], \sum_{j'} b_{jj'} = 1$) thus corresponding to a convex sum of the value vectors, and a convex sum is upper-bounded by the maximum. Now, we need to bound the norm of the updated token embeddings $\bar{\boldsymbol{X}}^{(1)}$ of the points:

$$\forall i \in [n], \left\|\bar{\boldsymbol{X}}^{(1)}_{:,i}\right\| = \left\|\bar{\boldsymbol{X}}^{(0)}_{:,i} + \left(\boldsymbol{V}_1 \sum_{j'=1}^{k} a'_{ij'} \bar{\boldsymbol{C}}^{(0)}_{:,j'}\right) + \left(\boldsymbol{V}_2 \sum_{i'=1}^{n} b'_{ii'} \bar{\boldsymbol{X}}^{(0)}_{:,i'}\right)\right\| \tag{40}$$

$$\leq \left\|\bar{\boldsymbol{X}}^{(0)}_{:,i}\right\| + \max_{j'\in[k]} \left\|\boldsymbol{V}_1 \bar{\boldsymbol{C}}^{(0)}_{:,j'}\right\| + \max_{i'\in[n]} \left\|\boldsymbol{V}_2 \bar{\boldsymbol{X}}^{(0)}_{:,i'}\right\| \tag{41}$$

$$\leq \sqrt{d} + \Gamma_2\sqrt{d_E} + \Gamma_2\sqrt{d}, \tag{42}$$

where $a'_{ij'}$ and $b'_{ii'}$ are the cross-attention and self-attention weights for the point embedding updates in equation (4). This gives us the bound on $\left\|\bar{\boldsymbol{X}}^{(1)}_{:,i}\right\|$ in equation (37) in the statement of the result. Plugging this in equation (39), we see

$$\|\boldsymbol{C}(\Theta)_{:,j}\| \leq \left\|\bar{\boldsymbol{C}}^{(0)}_{:,j}\right\| + \max_{i\in[n]} \left\|\boldsymbol{V}_3 \bar{\boldsymbol{X}}^{(1)}_{:,i}\right\| + \max_{j'\in[k]} \left\|\boldsymbol{V}_4 \bar{\boldsymbol{C}}^{(0)}_{:,j'}\right\|, \tag{43}$$

$$\leq \sqrt{d_E} + \Gamma_2(\sqrt{d} + \Gamma_2\sqrt{d_E} + \Gamma_2\sqrt{d}) + \Gamma_2\sqrt{d_E} \leq (1 + 2\Gamma_2 + 2\Gamma_2^2)\sqrt{d_E}, \tag{44}$$

thus giving us equation (37) in the statement of lemma. □

Lemma 2. *Under Assumptions 2, 3, 1, and the conditions of Lemma 1, for any clustering task $(\boldsymbol{X}, \boldsymbol{C}) \sim \mathcal{D}$ with n points in d dimensions and k clusters, the smoothed loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ is input-stable with respect to the output of the model $F^{\gamma}_{d:k}$ upon changes to the model parameters Θ as follows for any pair of parameters Θ, Θ':*

$$\begin{aligned}
&\left|\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)) - \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta'))\right| \\
&\quad \leq \kappa_1(n, \Gamma_2, d_E, \omega_\tau, \tau) \left\|F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta) - F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta')\right\|_{2,1} \\
&\quad \textit{where } \kappa_1(n, \Gamma_2, d_E, \omega_\tau, \tau) = 4n(1+\Gamma_2)^2\sqrt{d_E}\left(1 + \frac{4(1+\Gamma_2)^4 d_E \omega_\tau}{\tau}\right) \sim O\left(\frac{n\omega_\tau d_E^{3/2}}{\tau}\right).
\end{aligned} \tag{45}$$

Proof. Denoting the output cluster centers as $\boldsymbol{C}(\Theta) \triangleq F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta)$ and $\boldsymbol{C}(\Theta') \triangleq F^{\gamma}_{d:k}(\boldsymbol{X}, \boldsymbol{C}; \Theta')$, first, we can see the expand the clustering loss $\tilde{\mathcal{L}}^{\tau}_{\Omega}$ to a per-point loss $\tilde{L}^{\tau}_{\Omega}$ as follows

$$\left|\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, \boldsymbol{C}(\Theta)) - \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, \boldsymbol{C}(\Theta'))\right| = \left|\sum_{i=1}^{n} \tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \boldsymbol{C}(\Theta)) - \sum_{i=1}^{n} \tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \boldsymbol{C}(\Theta'))\right| \tag{46}$$

$$\leq \sum_{i=1}^{n} \left|\tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \boldsymbol{C}(\Theta)) - \tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \boldsymbol{C}(\Theta'))\right| \tag{47}$$

$$\leq n \max_{i\in[n]} \left|\tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \boldsymbol{C}(\Theta)) - \tilde{L}^{\tau}_{\Omega}(\boldsymbol{X}_{:,i}, \boldsymbol{C}(\Theta'))\right|. \tag{48}$$

We will bound the above for a generic $\boldsymbol{x} \in [0,1]^d$. Using the notation that $\boldsymbol{r} = R(\boldsymbol{x}, \boldsymbol{C}(\Theta))$ and $\boldsymbol{r}' = R(\boldsymbol{x}, \boldsymbol{C}(\Theta'))$, where $R : \mathbb{R}^d \times \mathbb{R}^{d\times k} \to \mathbb{R}^k$ is as previously defined where the j-th entry of its output $r_j = \|\boldsymbol{x} - \boldsymbol{C}(\Theta)_{:,j}\|^2$ (and correspondingly $r'_j = \|\boldsymbol{x} - \boldsymbol{C}(\Theta')_{:,j}\|^2$), we have $\boldsymbol{r}, \boldsymbol{r}' \in [0, r^\star]^k$ with

$$r^\star \triangleq \max_{\boldsymbol{x}\in[0,1]^d, j\in[k], \Theta} \|\boldsymbol{x} - \boldsymbol{C}(\Theta)_{:,j}\|^2 \leq \left(\max_{\boldsymbol{x}\in[0,1]^d} \|\boldsymbol{x}\| + \max_{j\in[k],\Theta} \|\boldsymbol{C}(\Theta)_{:,j}\|\right)^2 \tag{49}$$

$$\leq \left(\sqrt{d} + (1 + 2\Gamma_2 + 2\Gamma_2^2)\sqrt{d_E}\right)^2 \tag{50}$$

$$\leq 4d_E(1 + \Gamma_2 + \Gamma_2^2)^2 \leq 4(1 + \Gamma_2)^4 d_E. \tag{51}$$

where we bounded $\max_{j,\Theta} \|\boldsymbol{C}(\Theta)\|$ using Lemma 1. Now we can utilize Proposition 2 as follows:

$$\left|\tilde{L}_\Omega^\tau(\boldsymbol{x}, \boldsymbol{C}(\Theta)) - \tilde{L}_\Omega^\tau(\boldsymbol{x}, \boldsymbol{C}(\Theta'))\right| = \left|\tilde{\mathrm{L}}_\Omega^\tau(\boldsymbol{r}) - \tilde{\mathrm{L}}_\Omega^\tau(\boldsymbol{r}')\right| \tag{52}$$

$$\leq \left(1 + \frac{r^\star \omega_\tau}{\tau}\right) \|\boldsymbol{r} - \boldsymbol{r}'\|_1 \tag{53}$$

$$= \left(1 + \frac{4(1 + \Gamma_2)^4 d_E \omega_\tau}{\tau}\right) \|\boldsymbol{r} - \boldsymbol{r}'\|_1 . \tag{54}$$

Finally, we can bound $\|\boldsymbol{r} - \boldsymbol{r}'\|_1$ as follows:

$$\|\boldsymbol{r} - \boldsymbol{r}'\|_1 = \sum_{j=1}^{k} |r_j - r'_j| \tag{55}$$

$$= \sum_{j=1}^{k} \left|\left\langle \left(\boldsymbol{C}(\Theta)_{:,j} - \boldsymbol{C}(\Theta')_{:,j}\right), \left(\boldsymbol{C}(\Theta)_{:,j} + \boldsymbol{C}(\Theta')_{:,j} - 2\boldsymbol{x}\right)\right\rangle\right| \tag{56}$$

$$\leq \sum_{j=1}^{k} \|\boldsymbol{C}(\Theta)_{:,j} - \boldsymbol{C}(\Theta')_{:,j}\| \, \|\boldsymbol{C}(\Theta)_{:,j} + \boldsymbol{C}(\Theta')_{:,j} - 2\boldsymbol{x}\| \tag{57}$$

$$\leq \sum_{j=1}^{k} \|\boldsymbol{C}(\Theta)_{:,j} - \boldsymbol{C}(\Theta')_{:,j}\| \left(2\sqrt{d} + 2(1 + 2\Gamma_2 + 2\Gamma_2^2)\sqrt{d_E}\right) \tag{58}$$

$$\leq \left(4(1 + \Gamma_2)^2 \sqrt{d_E}\right) \sum_{j=1}^{k} \|\boldsymbol{C}(\Theta)_{:,j} - \boldsymbol{C}(\Theta')_{:,j}\| \tag{59}$$

$$= 4(1 + \Gamma_2)^2 \sqrt{d_E} \, \|\boldsymbol{C}(\Theta) - \boldsymbol{C}(\Theta')\|_{2,1} . \tag{60}$$

Finally, combining the above with equation (54) and equation (48) gives us the inequality in equation (45) in the statement of the result. □

The above result ties the sensitivity of the loss to the model parameters $\left|\tilde{\mathcal{L}}_\Omega^\tau(\boldsymbol{X}, \boldsymbol{C}(\Theta)) - \tilde{\mathcal{L}}_\Omega^\tau(\boldsymbol{X}, \boldsymbol{C}(\Theta'))\right|$ to the sensitivity of the model output to the model parameters $\|\boldsymbol{C}(\Theta) - \boldsymbol{C}(\Theta')\|_{2,1}$. To bound this $\|\boldsymbol{C}(\Theta) - \boldsymbol{C}(\Theta')\|_{2,1}$ with perturbations to the model parameters, we have to consider the attention based updates in Definition 2. Since there are four different attention operations in equation (3) in Definition 2, we will first present bounds on the sensitivity of the output of a single attention module $\mathcal{A}_2^\gamma(\cdots)$ to changes in the specific model parameters (the query/key/value projection matrices) for generic query and key tokens $\boldsymbol{Z}, \boldsymbol{Z}'$, and then apply them to each of the four attention modules with the appropriate keys and queries. These results are presented in the following, and then summarized in Table 2.

Definition 4. *For any $\boldsymbol{v} \in \mathbb{R}^m$, let its dispersion be defined as $D(\boldsymbol{v})$ with a function $D : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ such that*

$$D(\boldsymbol{v}) = \left(\max_{i \in [m]} v_i\right) - \left(\min_{i \in [m]} v_i\right). \tag{61}$$

Lemma 3 (adapted from Li et al. [21], Lemma B.3)**.** *For any $\boldsymbol{v}, \boldsymbol{v}' \in R^m$ such that $D(\boldsymbol{v}), D(\boldsymbol{v}') \leq \delta$ for some $\delta > 0$, the softmax operation is bounded and input-stable as follows:*

$$\|\mathrm{softmax}(\boldsymbol{v})\|_\infty \leq \frac{\exp(\delta)}{m}, \quad \|\mathrm{softmax}(\boldsymbol{v}) - \mathrm{softmax}(\boldsymbol{v}')\|_1 \leq \frac{2\exp(\delta)}{m} \|\boldsymbol{v} - \boldsymbol{v}'\|_1 . \tag{62}$$

Lemma 4 (attention score dispersion)**.** *For queries $\boldsymbol{Z} \in \mathbb{R}^{d_{\mathsf{emb}} \times m}$ and keys $\boldsymbol{Z}' \in \mathbb{R}^{d_{\mathsf{emb}} \times m'}$ such that $\|\boldsymbol{Z}_{:,i}\| \leq \rho$ for all $i \in [m]$ and $\left\|\boldsymbol{Z}'_{:,i}\right\| \leq \rho'$ for all $i \in [m']$, under the assumption 3, for $\boldsymbol{v}_i = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}, \boldsymbol{K}_\mu \boldsymbol{Z}')$, we have the dispersion bounded as*

$$\forall i \in [m], D(\boldsymbol{v}_i) \leq \delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}}) = (\gamma/\sqrt{d_{\mathsf{emb}}})\Gamma_1^2(\rho + \rho')^2. \tag{63}$$

Proof. We have the following for any $i \in [m]$:

$$D(\boldsymbol{v}_i) = \max_{i' \in [m']} \left\{ -(\gamma/\sqrt{d_{\mathsf{emb}}}) \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,i'} \right\|^2 \right\} - \min_{i' \in [m']} \left\{ -(\gamma/\sqrt{d_{\mathsf{emb}}}) \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,i'} \right\|^2 \right\} \tag{64}$$

$$= \gamma/\sqrt{d_{\mathsf{emb}}} \left(\max_{i' \in [m']} \left\{ \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,i'} \right\|^2 \right\} - \min_{i' \in [m']} \left\{ \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,i'} \right\|^2 \right\} \right) \tag{65}$$

$$\leq \gamma/\sqrt{d_{\mathsf{emb}}} \max_{i' \in [m']} \left\{ \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,i'} \right\|^2 \right\} \tag{66}$$

$$\leq \gamma/\sqrt{d_{\mathsf{emb}}} \left(\|\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}\| + \max_{i' \in [m']} \left\| \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,i'} \right\| \right)^2 \leq \gamma/\sqrt{d_{\mathsf{emb}}} \left(\Gamma_1(\rho + \rho') \right)^2, \tag{67}$$

giving us the definition of $\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})$ in the statement of the result. □

Lemma 5. *Under conditions of Lemma 4, the attention operation is stable with respect to the value projection matrix as follows:*

$$\left\| \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}'_\mu)) \right\|_{2,1} \leq \kappa_2(m, \rho') \left\| \boldsymbol{V}_\mu - \boldsymbol{V}'_\mu \right\|, \tag{68}$$

with $\kappa_2(m, \rho') = m\rho'$.

Proof. Expanding out the attention operation, we can see that

$$\left\| \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}'_\mu)) \right\|_{2,1}$$

$$= \sum_{i=1}^{m} \left\| \sum_{j=1}^{m'} a_{ij} \boldsymbol{V}_\mu \boldsymbol{Z}'_{:,j} - \sum_{j=1}^{m'} a_{ij} \boldsymbol{V}'_\mu \boldsymbol{Z}'_{:,j} \right\| = \sum_{i=1}^{m} \left\| (\boldsymbol{V}_\mu - \boldsymbol{V}'_\mu) \sum_{j=1}^{m'} a_{ij} \boldsymbol{Z}'_{:,j} \right\| \tag{69}$$

$$\leq \sum_{i=1}^{m} \left\| (\boldsymbol{V}_\mu - \boldsymbol{V}'_\mu) \right\| \left\| \sum_{j=1}^{m'} a_{ij} \boldsymbol{Z}'_{:,j} \right\| \leq \sum_{i=1}^{m} \left\| \boldsymbol{V}_\mu - \boldsymbol{V}'_\mu \right\| \max_{j \in [m']} \left\| \boldsymbol{Z}'_{:,j} \right\| \leq m\rho' \left\| \boldsymbol{V}_\mu - \boldsymbol{V}'_\mu \right\|, \tag{70}$$

where a_{ij} are the softmax attention weights for the i-th query computed by $\mathrm{softmax}(-\gamma/\sqrt{d_{\mathsf{emb}}}\mathrm{sqdist}(\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}, \boldsymbol{K}_\mu \boldsymbol{Z}'))$, thus giving us $\kappa_2(m, \rho')$ in the statement of the result. □

Lemma 6. *Under conditions of Lemma 4, the attention operation is stable with respect to the query projection matrix as follows:*

$$\begin{aligned} &\left\| \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}'_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu)) \right\|_{2,1} \\ &\quad \leq \kappa_3(m, \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \boldsymbol{Q}_\mu - \boldsymbol{Q}'_\mu \right\|, \\ &\quad \textit{where} \quad \kappa_3(m, \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) = \frac{4m\gamma\Gamma_1\Gamma_2\rho\rho'(\rho + \rho')}{\sqrt{d_{\mathsf{emb}}}} \exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})). \end{aligned} \tag{71}$$

Proof. First, focusing on just the i-th query corresponding to $\boldsymbol{Z}_{:,i}$ for some $i \in [m]$, let $\boldsymbol{v} = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}, \boldsymbol{K}_\mu \boldsymbol{Z}')$ and $\boldsymbol{v}' = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}'_\mu \boldsymbol{Z}_{:,i}, \boldsymbol{K}_\mu \boldsymbol{Z}')$, Lemma 4 tells us that their dispersions are bounded as $D(\boldsymbol{v}), D(\boldsymbol{v}') \leq \delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})$. Now the $\|\boldsymbol{v} - \boldsymbol{v}'\|_1$ can be bounded as

$$\|\boldsymbol{v} - \boldsymbol{v}'\|_1 = \sum_{j=1}^{m'} |v_j - v'_j| = \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \sum_{j=1}^{m'} \left| \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,j} \right\|^2 - \left\| \boldsymbol{Q}'_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,j} \right\|^2 \right| \tag{72}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \sum_{j=1}^{m'} \left\| (\boldsymbol{Q}_\mu - \boldsymbol{Q}'_\mu) \boldsymbol{Z}_{:,i} \right\| \left\| (\boldsymbol{Q}_\mu + \boldsymbol{Q}'_\mu) \boldsymbol{Z}_{:,i} - 2\boldsymbol{K}_\mu \boldsymbol{Z}'_{:,j} \right\| \tag{73}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} m' \left\| (\boldsymbol{Q}_\mu - \boldsymbol{Q}'_\mu) \right\| \|\boldsymbol{Z}_{:,i}\| \left(\|\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}\| + \left\| \boldsymbol{Q}'_\mu \boldsymbol{Z}_{:,i} \right\| + 2 \left\| \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,j} \right\| \right) \tag{74}$$

$$\leq \frac{m'\gamma}{\sqrt{d_{\mathsf{emb}}}} \left\| (\boldsymbol{Q}_\mu - \boldsymbol{Q}'_\mu) \right\| \rho \left(2\Gamma_1(\rho + \rho') \right). \tag{75}$$

Combining this with Lemma 3, we can see that

$$\|\mathrm{softmax}(\boldsymbol{v}) - \mathrm{softmax}(\boldsymbol{v}')\|_1 \leq \frac{2\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{m'}\|\boldsymbol{v}-\boldsymbol{v}'\|_1 \tag{76}$$

$$\leq \frac{2\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{m'}\frac{m'\gamma}{\sqrt{d_{\mathsf{emb}}}}\rho\left(2\Gamma_1(\rho+\rho')\right)\left\|\boldsymbol{Q}_\mu - \boldsymbol{Q}'_\mu\right\| \tag{77}$$

$$= \frac{4\gamma\Gamma_1\rho(\rho+\rho')\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{\sqrt{d_{\mathsf{emb}}}}\left\|\boldsymbol{Q}_\mu - \boldsymbol{Q}'_\mu\right\|. \tag{78}$$

Denoting $\boldsymbol{A} = \mathrm{softmax}(-(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z},\boldsymbol{K}_\mu\boldsymbol{Z}'))$ and $\boldsymbol{A}' = \mathrm{softmax}(-(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}'_\mu\boldsymbol{Z},\boldsymbol{K}_\mu\boldsymbol{Z}'))$ with the softmax applied columnwise, we have $\boldsymbol{A},\boldsymbol{A}' \in \mathbb{R}^{m'\times m}$. Now expanding the left-hand-side of equation (71), and we see that

$$\left\|\mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}'_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu))\right\|_{2,1}$$

$$= \sum_{i=1}^{m}\left\|\boldsymbol{V}_\mu\sum_{j=1}^{m'}(A_{ji}-A'_{ji})\boldsymbol{Z}'_{:,j}\right\| \tag{79}$$

$$\leq \|\boldsymbol{V}_\mu\|\sum_{i=1}^{m}\left\|\sum_{j=1}^{m'}(A_{ji}-A'_{ji})\boldsymbol{Z}'_{:,j}\right\| \leq \Gamma_2\sum_{i=1}^{m}\sum_{j=1}^{m'}|A_{ji}-A'_{ji}|\left\|\boldsymbol{Z}'_{:,j}\right\| \tag{80}$$

$$\leq \Gamma_2\rho'\sum_{i=1}^{m}\left\|\boldsymbol{A}_{:,i}-\boldsymbol{A}'_{:,i}\right\|_1 \tag{81}$$

$$\leq \Gamma_2\rho' m\frac{4\gamma\Gamma_1\rho(\rho+\rho')\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{\sqrt{d_{\mathsf{emb}}}}\left\|\boldsymbol{Q}_\mu-\boldsymbol{Q}'_\mu\right\|, \tag{82}$$

where we invoked equation (78), noting that $\boldsymbol{A}_{:,i} = \mathrm{softmax}(\boldsymbol{v})$ and $\boldsymbol{A}'_{:,i} = \mathrm{softmax}(\boldsymbol{v}')$ and summed over $i \in [m]$. This gives us the coefficient $\kappa_3(m,\rho,\rho',\Gamma_1,\Gamma_2,\gamma,d_{\mathsf{emb}})$ in equation (71) in the statement of the result. □

Lemma 7. *Under conditions of Lemma 4, the attention operation is stable with respect to the key projection matrix as follows:*

$$\begin{aligned}&\left\|\mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}'_\mu,\boldsymbol{V}_\mu))\right\|_{2,1}\\ &\quad\leq \kappa_4(m,\rho,\rho',\Gamma_1,\Gamma_2,\gamma,d_{\mathsf{emb}})\left\|\boldsymbol{K}_\mu-\boldsymbol{K}'_\mu\right\|,\\ &\quad\textit{where}\quad \kappa_4(m,\rho,\rho',\Gamma_1,\Gamma_2,\gamma,d_{\mathsf{emb}}) = \frac{4m\gamma\Gamma_1\Gamma_2\rho'^2(\rho+\rho')}{\sqrt{d_{\mathsf{emb}}}}\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}})).\end{aligned} \tag{83}$$

Proof. First, focusing on just the i-th query corresponding to $\boldsymbol{Z}_{:,i}$ for some $i \in [m]$, let $\boldsymbol{v} = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z}_{:,i},\boldsymbol{K}_\mu\boldsymbol{Z}')$ and $\boldsymbol{v}' = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z}_{:,i},\boldsymbol{K}'_\mu\boldsymbol{Z}')$, Lemma 4 tells us that their dispersions are bounded as $D(\boldsymbol{v}), D(\boldsymbol{v}') \leq \delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}})$. Now the $\|\boldsymbol{v}-\boldsymbol{v}'\|_1$ can be bounded as

$$\|\boldsymbol{v}-\boldsymbol{v}'\|_1 = \sum_{j=1}^{m'}|v_j - v'_j| = \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}}\sum_{j=1}^{m'}\left|\left\|\boldsymbol{Q}_\mu\boldsymbol{Z}_{:,i}-\boldsymbol{K}_\mu\boldsymbol{Z}'_{:,j}\right\|^2 - \left\|\boldsymbol{Q}_\mu\boldsymbol{Z}_{:,i}-\boldsymbol{K}'_\mu\boldsymbol{Z}'_{:,j}\right\|^2\right| \tag{84}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}}\sum_{j=1}^{m'}\left\|(\boldsymbol{K}_\mu-\boldsymbol{K}'_\mu)\boldsymbol{Z}'_{:,j}\right\|\left\|(\boldsymbol{K}_\mu+\boldsymbol{K}'_\mu)\boldsymbol{Z}'_{:,j}-2\boldsymbol{Q}_\mu\boldsymbol{Z}_{:,i}\right\| \tag{85}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}}m'\left\|(\boldsymbol{K}_\mu-\boldsymbol{K}'_\mu)\right\|\left\|\boldsymbol{Z}'_{:,j}\right\|\left(\left\|\boldsymbol{K}_\mu\boldsymbol{Z}'_{:,j}\right\|+\left\|\boldsymbol{K}'_\mu\boldsymbol{Z}'_{:,j}\right\|+2\left\|\boldsymbol{Q}_\mu\boldsymbol{Z}_{:,i}\right\|\right) \tag{86}$$

$$\leq \frac{m'\gamma}{\sqrt{d_{\mathsf{emb}}}}\left\|(\boldsymbol{K}_\mu-\boldsymbol{K}'_\mu)\right\|\rho'\left(2\Gamma_1(\rho'+\rho)\right). \tag{87}$$

Combining this with Lemma 3, we can see that

$$\|\mathrm{softmax}(\boldsymbol{v}) - \mathrm{softmax}(\boldsymbol{v}')\|_1 \leq \frac{2\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{m'}\|\boldsymbol{v}-\boldsymbol{v}'\|_1 \tag{88}$$

$$\leq \frac{2\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{m'}\frac{m'\gamma}{\sqrt{d_{\mathsf{emb}}}}\rho'\left(2\Gamma_1(\rho+\rho')\right)\left\|\boldsymbol{K}_\mu - \boldsymbol{K}'_\mu\right\| \tag{89}$$

$$= \frac{4\gamma\Gamma_1\rho'(\rho+\rho')\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{\sqrt{d_{\mathsf{emb}}}}\left\|\boldsymbol{K}_\mu - \boldsymbol{K}'_\mu\right\|. \tag{90}$$

Denoting $\boldsymbol{A} = \mathrm{softmax}(-(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z}, \boldsymbol{K}_\mu\boldsymbol{Z}'))$ and $\boldsymbol{A}' = \mathrm{softmax}(-(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z}, \boldsymbol{K}'_\mu\boldsymbol{Z}'))$ with the softmax applied columnwise, we have $\boldsymbol{A}, \boldsymbol{A}' \in \mathbb{R}^{m'\times m}$. Now expanding the left-hand-side of equation (83), and we see that

$$\left\|\mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}'_\mu,\boldsymbol{V}_\mu))\right\|_{2,1}$$

$$= \sum_{i=1}^m \left\|\boldsymbol{V}_\mu \sum_{j=1}^{m'}(A_{ji}-A'_{ji})\boldsymbol{Z}'_{:,j}\right\| \tag{91}$$

$$\leq \|\boldsymbol{V}_\mu\|\sum_{i=1}^m\left\|\sum_{j=1}^{m'}(A_{ji}-A'_{ji})\boldsymbol{Z}'_{:,j}\right\| \leq \Gamma_2\sum_{i=1}^m\sum_{j=1}^{m'}|A_{ji}-A'_{ji}|\left\|\boldsymbol{Z}'_{:,j}\right\| \tag{92}$$

$$\leq \Gamma_2\rho'\sum_{i=1}^m\left\|\boldsymbol{A}_{:,i}-\boldsymbol{A}'_{:,i}\right\|_1 \tag{93}$$

$$\leq \Gamma_2\rho' m\frac{4\gamma\Gamma_1\rho'(\rho+\rho')\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))}{\sqrt{d_{\mathsf{emb}}}}\left\|\boldsymbol{K}_\mu-\boldsymbol{K}'_\mu\right\|, \tag{94}$$

where we invoked equation (90), noting that $\boldsymbol{A}_{:,i} = \mathrm{softmax}(\boldsymbol{v})$ and $\boldsymbol{A}'_{:,i} = \mathrm{softmax}(\boldsymbol{v}')$ and summed over $i \in [m]$. This gives us the coefficient $\kappa_4(m,\rho,\rho',\Gamma_1,\Gamma_2,\gamma,d_{\mathsf{emb}})$ in equation (83) in the statement of the result. □

Lemma 8. *Under conditions of Lemma 4, the attention operation is stable with respect to the keys as follows:*

$$\begin{aligned}
&\forall \boldsymbol{Z}', \boldsymbol{Z}'' \in \mathbb{R}^{d_{\mathsf{emb}}\times m'}, \forall j \in [m'], \left\|\boldsymbol{Z}'_{:,j}\right\| \leq \rho', \left\|\boldsymbol{Z}''_{:,j}\right\| \leq \rho',\\
&\|\mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}'';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu))\|_{2,1}\\
&\quad\leq \kappa_5(m,m',\rho,\rho',\Gamma_1,\Gamma_2,\gamma,d_{\mathsf{emb}})\|\boldsymbol{Z}'-\boldsymbol{Z}''\|_{2,1},\\
&\textit{where}\quad \kappa_5(m,m',\rho,\rho',\Gamma_1,\Gamma_2,\gamma,d_{\mathsf{emb}})\\
&\quad= \left(1+4\left(\frac{\gamma}{\sqrt{d_{\mathsf{emb}}}}\right)\Gamma_1^2\rho'(\rho+\rho')\right)\left(\frac{m\Gamma_2}{m'}\right)\exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}})).
\end{aligned} \tag{95}$$

Proof. We can denote with $\boldsymbol{A} = \mathrm{softmax}(-(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z}, \boldsymbol{K}_\mu\boldsymbol{Z}'))$ using the keys $\boldsymbol{Z}'$, and the corresponding $\boldsymbol{A}' = \mathrm{softmax}(-(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu\boldsymbol{Z}, \boldsymbol{K}_\mu\boldsymbol{Z}''))$ with the keys $\boldsymbol{Z}''$ (the softmax applied columnwise) we have $\boldsymbol{A}, \boldsymbol{A}' \in \mathbb{R}^{m'\times m}$. Now expanding the left-hand-side of equation (95), and we see that

$$\begin{aligned}
&\|\mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z},\boldsymbol{Z}'';(\boldsymbol{Q}_\mu,\boldsymbol{K}_\mu,\boldsymbol{V}_\mu))\|_{2,1}\\
&\quad= \|\boldsymbol{V}_\mu\boldsymbol{Z}'\boldsymbol{A} - \boldsymbol{V}_\mu\boldsymbol{Z}''\boldsymbol{A}'\|_{2,1} \leq \|\boldsymbol{V}_\mu(\boldsymbol{Z}'-\boldsymbol{Z}'')\boldsymbol{A}\|_{2,1} + \|\boldsymbol{V}_\mu\boldsymbol{Z}''(\boldsymbol{A}-\boldsymbol{A}')\|_{2,1}.
\end{aligned} \tag{96}$$

We can bound each of these terms separately. First, noting that $A_{ji} \leq \exp(\delta(\rho,\rho',\Gamma_1,\gamma,d_{\mathsf{emb}}))/m'$ for any (i,j) from Lemma 3, we have

$$\|\boldsymbol{V}_\mu(\boldsymbol{Z}'-\boldsymbol{Z}'')\boldsymbol{A}\|_{2,1} = \sum_{i=1}^m\|\boldsymbol{V}_\mu(\boldsymbol{Z}'-\boldsymbol{Z}'')\boldsymbol{A}_{:,i}\| \leq \|\boldsymbol{V}_\mu\|\sum_{i=1}^m\|(\boldsymbol{Z}'-\boldsymbol{Z}'')\boldsymbol{A}_{:,i}\| \tag{97}$$

$$= \|\boldsymbol{V}_\mu\| \sum_{i=1}^{m} \left\| \sum_{j=1}^{m'} (\boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j}) A_{ji} \right\| \leq \|\boldsymbol{V}_\mu\| \sum_{i=1}^{m} \sum_{j=1}^{m'} \left\| \boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j} \right\| |A_{ji}| \tag{98}$$

$$\leq \Gamma_2 \frac{\exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}}))}{m'} \sum_{i=1}^{m} \sum_{j=1}^{m'} \left\| \boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j} \right\| \tag{99}$$

$$= \Gamma_2 \frac{\exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}}))}{m'} m \left\| \boldsymbol{Z}' - \boldsymbol{Z}'' \right\|_{2,1}. \tag{100}$$

Next, focusing on just the i-th query corresponding to $\boldsymbol{Z}_{:,i}$ for some $i \in [m]$, let $\boldsymbol{v} = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}, \boldsymbol{K}_\mu \boldsymbol{Z}')$ and $\boldsymbol{v}' = -(\gamma/\sqrt{d_{\mathsf{emb}}})\mathrm{sqdist}(\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i}, \boldsymbol{K}_\mu \boldsymbol{Z}'')$, Lemma 4 tells us that their dispersions are bounded as $D(\boldsymbol{v}), D(\boldsymbol{v}') \leq \delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})$. Note that $\boldsymbol{A}_{:,i} = \mathrm{softmax}(\boldsymbol{v})$, $\boldsymbol{A}'_{:,i} = \mathrm{softmax}(\boldsymbol{v}')$. Now the $\|\boldsymbol{v} - \boldsymbol{v}'\|_1$ can be bounded as

$$\|\boldsymbol{v} - \boldsymbol{v}'\|_1 = \sum_{j=1}^{m'} |v_j - v'_j| = \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \sum_{j=1}^{m'} \left| \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,j} \right\|^2 - \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} - \boldsymbol{K}_\mu \boldsymbol{Z}''_{:,j} \right\|^2 \right| \tag{101}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \sum_{j=1}^{m'} \left\| \boldsymbol{K}_\mu (\boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j}) \right\| \left\| \boldsymbol{K}_\mu (\boldsymbol{Z}'_{:,j} + \boldsymbol{Z}''_{:,j}) - 2\boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} \right\| \tag{102}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \sum_{j=1}^{m'} \|\boldsymbol{K}_\mu\| \left\| \boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j} \right\| \left(\left\| \boldsymbol{K}_\mu \boldsymbol{Z}'_{:,j} \right\| + \left\| \boldsymbol{K}_\mu \boldsymbol{Z}''_{:,j} \right\| + 2 \left\| \boldsymbol{Q}_\mu \boldsymbol{Z}_{:,i} \right\| \right) \tag{103}$$

$$\leq \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \sum_{j=1}^{m'} \Gamma_1 \left\| \boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j} \right\| (2\Gamma_1(\rho' + \rho)) \tag{104}$$

$$= \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \Gamma_1 (2\Gamma_1(\rho' + \rho)) \sum_{j=1}^{m'} \left\| \boldsymbol{Z}'_{:,j} - \boldsymbol{Z}''_{:,j} \right\| \tag{105}$$

$$= \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} \Gamma_1 (2\Gamma_1(\rho' + \rho)) \left\| \boldsymbol{Z}' - \boldsymbol{Z}'' \right\|_{2,1}. \tag{106}$$

Combining this with Lemma 3, we can see that

$$\|\mathrm{softmax}(\boldsymbol{v}) - \mathrm{softmax}(\boldsymbol{v}')\|_1 \leq \frac{2\exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}}))}{m'} \|\boldsymbol{v} - \boldsymbol{v}'\|_1 \tag{107}$$

$$\leq \frac{2\exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}}))}{m'} \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} 2\Gamma_1^2(\rho' + \rho) \left\| \boldsymbol{Z}' - \boldsymbol{Z}'' \right\|_{2,1} \tag{108}$$

Considering the second term on the right-hand-side of equation (96), we see that

$$\|\boldsymbol{V}_\mu \boldsymbol{Z}''(\boldsymbol{A} - \boldsymbol{A}')\|_{2,1} = \sum_{i=1}^{m} \left\| \boldsymbol{V}_\mu \boldsymbol{Z}''(\boldsymbol{A}_{:,i} - \boldsymbol{A}'_{:,i}) \right\| = \sum_{i=1}^{m} \left\| \sum_{j=1}^{m'} \boldsymbol{V}_\mu \boldsymbol{Z}''_{:,j} (A_{ji} - A'_{ji}) \right\| \tag{109}$$

$$\leq \sum_{i=1}^{m} \sum_{j=1}^{m'} \left\| \boldsymbol{V}_\mu \boldsymbol{Z}''_{:,j} \right\| |A_{ji} - A'_{ji}| \leq \sum_{i=1}^{m} \Gamma_2 \rho' \sum_{j=1}^{m'} |A_{ji} - A'_{ji}| \tag{110}$$

$$= \Gamma_2 \rho' \sum_{i=1}^{m} \left\| \boldsymbol{A}_{:,i} - \boldsymbol{A}'_{:,i} \right\|_1 \tag{111}$$

$$\leq \Gamma_2 \rho' \sum_{i=1}^{m} \frac{2\exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}}))}{m'} \frac{\gamma}{\sqrt{d_{\mathsf{emb}}}} 2\Gamma_1^2(\rho' + \rho) \left\| \boldsymbol{Z}' - \boldsymbol{Z}'' \right\|_{2,1}, \tag{112}$$

where we applied equation (108) noting that $\boldsymbol{A}_{:,i} = \mathrm{softmax}(\boldsymbol{v})$ and $\boldsymbol{A}'_{:,i} = \mathrm{softmax}(\boldsymbol{v}')$. Now combining equation (100) and equation (112) in equation (96), we have

$$\begin{aligned}
&\|\mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}'; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu)) - \mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}''; (\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu))\|_{2,1} \\
&\leq \|\boldsymbol{Z}' - \boldsymbol{Z}''\|_{2,1} \frac{m\Gamma_2}{m'} \exp(\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})) \left(1 + 4\frac{\gamma}{\sqrt{d_{\mathsf{emb}}}}\Gamma_1^2\rho'(\rho' + \rho)\right) \qquad (113) \\
&= \|\boldsymbol{Z}' - \boldsymbol{Z}''\|_{2,1} \kappa_5(m, m', \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}), \qquad (114)
\end{aligned}$$

as in equation (95) in the statement of the result. □

We summarize the above results — the sensitivity of the attention module output to perturbations to its parameters (value projection matrix in Lemma 5, query projection matrix in Lemma 6, key projection matrix in Lemma 7) and its input (keys in Lemma 8) — in the following Table 2 for convenience, noting the factors that affect the bounds in the first sub-table (such as the number of queries or key, their respective norm upper-bounds, parameter matrix upper-bounds and such), and how the factors affect the bounds in the second and third sub-tables (in terms of whether they worsen or improve with an increase in the factor).

Table 2: Different quantities affecting the various bounds in Lemma 4, Lemma 3, Lemma 5, Lemma 6, Lemma 7 and Lemma 8. First, we provide a description of the terms, and then we present the effect on the bound when the terms of interest are increased. In almost all cases, the bounds worsen with increase in the term. The only quantities that lead to a decrease in the terms are the embedding dimension d_{emb} and the number of key tokens m'. We

Term	Description
m	Number of query tokens
m'	Number of key tokens
ρ	Upper-bound on the query token norms
ρ'	Upper-bound on the key token norms
Γ_1	Upper-bound on the spectral norm of the query/key projection matrix
Γ_2	Upper-bound on the spectral norm of the value projection matrix
γ	Inverse temperature for the attention operation in equation (3)
d_{emb}	Token embedding dimension for the transformer architecture
$\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})$	Maximum dispersion in the attention softmax (defined in Lemma 4)
$\kappa_2(m, \rho')$	Stability constant w.r.t. the value projection matrix in Lemma 5
$\kappa_3(m, \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$	Stability constant w.r.t. the query projection matrix in Lemma 6
$\kappa_4(m, \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$	Stability constant w.r.t. the key projection matrix in Lemma 7
$\kappa_5(m, m', \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$	Stability constant w.r.t. the key tokens in Lemma 8

	$\delta(\rho, \rho', \Gamma_1, \gamma, d_{\mathsf{emb}})$ (lemma 4)	$\kappa_2(m, \rho')$ (lemma 5)	$\kappa_3(m, \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$ (lemma 6)
	$(\gamma/\sqrt{d_{\mathsf{emb}}})\Gamma_1^2(\rho + \rho')^2$	$m\rho'$	$4(\gamma/\sqrt{d_{\mathsf{emb}}})m\rho\rho'(\rho + \rho')\Gamma_1\Gamma_2 \exp(\delta(\cdots))$
m		▲	▲
m'			
ρ	▲		▲
ρ'	▲	▲	▲
Γ_1	▲		▲
Γ_2			▲
γ	▲		▲
d_{emb}	▼		▼

	$\kappa_4(m, \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$ (lemma 7)	$\kappa_5(m, m', \rho, \rho', \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$ (lemma 8)
	$4(\gamma/\sqrt{d_{\mathsf{emb}}})m\rho'^2(\rho + \rho')\Gamma_1\Gamma_2 \exp(\delta(\cdots))$	$(m\Gamma_2/m')(1 + 4(\gamma/\sqrt{d_{\mathsf{emb}}})\Gamma_1^2\rho'(\rho + \rho'))\exp(\delta(\cdots))$
m	▲	▲
m'		▼
ρ	▲	▲
ρ'	▲	▲
Γ_1	▲	▲
Γ_2	▲	▲
γ	▲	▲
d_{emb}	▼	▼

Table 3: The value of the relevant terms from Table 2 in the different attention operations in Definition 2. We write the attention operations compactly as $\mathcal{A}_2^\gamma(\boldsymbol{Z}, \boldsymbol{Z}', (\cdot, \mu))$ for query tokens $\boldsymbol{Z} \in \mathbb{R}^{d_{\text{emb}} \times m}$, key tokens $\boldsymbol{Z}' \in \mathbb{R}^{d_{\text{emb}} \times m'}$, where (i) the query token norms are bounded as $\|\boldsymbol{Z}_{:,i}\| \leq \rho$ for all $i \in [m]$, (ii) the key token norms are $\left\|\boldsymbol{Z}'_{:,j}\right\| \leq \rho'$ for all $j \in [m']$, and (iii) the $(\cdots, \mu)$ corresponds to the appropriate $(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu)$ query/key/value projection matrices. The bounds on the norms of the key and query tokens are obtained from Lemma 1.

	$\mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\cdot, 1))$	$\mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\cdot, 2))$	$\mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\cdot, 3))$	$\mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\cdot, 4))$
m	n	n	k	k
m'	k	n	n	k
ρ	$\sqrt{d}$	$\sqrt{d}$	$\sqrt{d_E}$	$\sqrt{d_E}$
ρ'	$\sqrt{d_E}$	$\sqrt{d}$	$(1+\Gamma_2)\sqrt{d} + \Gamma_2\sqrt{d_E}$	$\sqrt{d_E}$

We can now apply these results to the attention modules in Definition 2 to bound the stability of the model output $\|\boldsymbol{C}(\Theta) - \boldsymbol{C}(\Theta')\|_{2,1}$. In Table 3, we note the relevant quantities necessary to apply our general results in Table 2 to the specific attention operations in Definition 2. This presentation is for the ease of exposition, and we will formally present the bounds in the following.

Theorem 4. *Assuming bounded inputs (assumption 2) and bounded model parameters (assumption 3), we can bound the stability of the model output $\boldsymbol{C}(\Theta)$ with respect to perturbations to the model parameters Θ as follows for any Θ, Θ' given $(\boldsymbol{X}, \boldsymbol{C})$ as input:*

$$\|\boldsymbol{C}(\Theta) - \boldsymbol{C}(\Theta')\|_{2,1} \leq \phi \cdot \|\Theta - \Theta'\|,$$
$$\textit{where } \phi \sim O\left(\max\{1, \gamma^2\} \cdot \left(k\sqrt{d_E}\right) \cdot \max\left\{1, \left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)^2\right\} \cdot \textit{poly}\left(\exp\left(\gamma \frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\right)\right). \tag{115}$$

Proof. Given that $\Theta = \{(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 1, \ldots, 4\}$ and $\Theta' = \{(\boldsymbol{Q}'_\mu, \boldsymbol{K}'_\mu, \boldsymbol{V}'_\mu), \mu = 1, \ldots, 4\}$, we will study the effect of modifying a single weight matrix at a time for ease of exposition (for example, effect of modifying $\boldsymbol{Q}_1$ to $\boldsymbol{Q}'_1$). In terms of notation, when we modify a single weight matrix (say $\boldsymbol{Q}_1$ to $\boldsymbol{Q}'_1$), and compare the outputs, we denote the outputs of the respective models as $\boldsymbol{C}(\boldsymbol{Q}_1)$ and $\boldsymbol{C}(\boldsymbol{Q}'_1)$ to make it explicit that we are studying the effect of just modifying $\boldsymbol{Q}_1$ to $\boldsymbol{Q}'_1$. To make explicit the effect of the model parameters, we will also write the final embeddings of points and clusters centers as $\bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_\mu)$ vs $\bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}'_\mu)$ and $\bar{\boldsymbol{C}}^{(1)}(\boldsymbol{Q}_\mu)$ vs $\bar{\boldsymbol{C}}^{(1)}(\boldsymbol{Q}'_\mu)$ only if the change in the considered weight matrix affects the token embeddings. Note that modifying any of the $(\boldsymbol{Q}_\mu, \boldsymbol{K}_\mu, \boldsymbol{V}_\mu), \mu = 3, 4$ does not affect the final point embeddings $\bar{\boldsymbol{X}}^{(1)}$, and in those situations, we will just denote the point embeddings as $\bar{\boldsymbol{X}}^{(1)}$.

With that in mind, we first study the effect of modifying $(\boldsymbol{Q}_4, \boldsymbol{K}_4, \boldsymbol{V}_4)$ one at a time, applying Lemma 6, Lemma 7, Lemma 5, respectively. We see that

$$\|\boldsymbol{C}(\boldsymbol{Q}_4) - \boldsymbol{C}(\boldsymbol{Q}'_4)\|_{2,1} \leq \left\|\bar{\boldsymbol{C}}^{(1)}(\boldsymbol{Q}_4) - \bar{\boldsymbol{C}}^{(1)}(\boldsymbol{Q}'_4)\right\|_{2,1} \tag{116}$$
$$= \left\|\mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_4, \boldsymbol{K}_4, \boldsymbol{V}_4)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}'_4, \boldsymbol{K}_4, \boldsymbol{V}_4))\right\|_{2,1} \tag{117}$$
$$\leq \kappa_3(k, \sqrt{d_E}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \|\boldsymbol{Q}_4 - \boldsymbol{Q}'_4\| \tag{118}$$
$$\|\boldsymbol{C}(\boldsymbol{K}_4) - \boldsymbol{C}(\boldsymbol{K}'_4)\|_{2,1} \leq \left\|\bar{\boldsymbol{C}}^{(1)}(\boldsymbol{K}_4) - \bar{\boldsymbol{C}}^{(1)}(\boldsymbol{K}'_4)\right\|_{2,1} \tag{119}$$
$$= \left\|\mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_4, \boldsymbol{K}_4, \boldsymbol{V}_4)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_4, \boldsymbol{K}'_4, \boldsymbol{V}_4))\right\|_{2,1} \tag{120}$$
$$\leq \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \|\boldsymbol{K}_4 - \boldsymbol{K}'_4\| \tag{121}$$
$$\|\boldsymbol{C}(\boldsymbol{V}_4) - \boldsymbol{C}(\boldsymbol{V}'_4)\|_{2,1} \leq \left\|\bar{\boldsymbol{C}}^{(1)}(\boldsymbol{V}_4) - \bar{\boldsymbol{C}}^{(1)}(\boldsymbol{V}'_4)\right\|_{2,1} \tag{122}$$

$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_4, \boldsymbol{K}_4, \boldsymbol{V}_4)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_4, \boldsymbol{K}_4, \boldsymbol{V}_4')) \right\|_{2,1} \tag{123}$$

$$\leq \kappa_2(k, \sqrt{d_E}) \, \|\boldsymbol{V}_4 - \boldsymbol{V}_4'\| \, . \tag{124}$$

Next, we consider the effect of modifying $(\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)$ one at a time, applying Lemma 6, Lemma 7, Lemma 5, respectively:

$$\|\boldsymbol{C}(\boldsymbol{Q}_3) - \boldsymbol{C}(\boldsymbol{Q}_3')\|_{2,1} = \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\boldsymbol{Q}_3', \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{125}$$

$$\leq \kappa_3(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \, \|\boldsymbol{Q}_3 - \boldsymbol{Q}_3'\| \tag{126}$$

$$\|\boldsymbol{C}(\boldsymbol{K}_3) - \boldsymbol{C}(\boldsymbol{K}_3')\|_{2,1} = \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\boldsymbol{Q}_3, \boldsymbol{K}_3', \boldsymbol{V}_3)) \right\|_{2,1} \tag{127}$$

$$\leq \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \, \|\boldsymbol{K}_3 - \boldsymbol{K}_3'\| \tag{128}$$

$$\|\boldsymbol{C}(\boldsymbol{V}_3) - \boldsymbol{C}(\boldsymbol{V}_3')\|_{2,1} = \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}; (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3')) \right\|_{2,1} \tag{129}$$

$$\leq \kappa_2(k, \sqrt{d_E}(1+2\Gamma_2)) \, \|\boldsymbol{V}_3 - \boldsymbol{V}_3'\| \tag{130}$$

Studying the effect of modifying $(\boldsymbol{Q}_1, \boldsymbol{K}_1, \boldsymbol{V}_1)$ one at a time, we need to incorporate the effect of the modified $\bar{\boldsymbol{X}}^{(1)}$ using Lemma 8 along with the use of Lemma 6, Lemma 7, Lemma 5:

$$\|\boldsymbol{C}(\boldsymbol{Q}_1) - \boldsymbol{C}(\boldsymbol{Q}_1')\|_{2,1} \tag{131}$$

$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_1); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_1'); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{132}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_1) - \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_1') \right\|_{2,1} \tag{133}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_1, \boldsymbol{K}_1, \boldsymbol{V}_1)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_1', \boldsymbol{K}_1, \boldsymbol{V}_1)) \right\|_{2,1} \tag{134}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \cdot \kappa_3(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \, \|\boldsymbol{Q}_1 - \boldsymbol{Q}_1'\| \tag{135}$$

$$\|\boldsymbol{C}(\boldsymbol{K}_1) - \boldsymbol{C}(\boldsymbol{K}_1')\|_{2,1} \tag{136}$$

$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_1); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_1'); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{137}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_1) - \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_1') \right\|_{2,1} \tag{138}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_1, \boldsymbol{K}_1, \boldsymbol{V}_1)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_1, \boldsymbol{K}_1', \boldsymbol{V}_1)) \right\|_{2,1} \tag{139}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \cdot \kappa_4(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \, \|\boldsymbol{K}_1 - \boldsymbol{K}_1'\| \tag{140}$$

$$\|\boldsymbol{C}(\boldsymbol{V}_1) - \boldsymbol{C}(\boldsymbol{V}_1')\|_{2,1} \tag{141}$$

$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_1); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_1'); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{142}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_1) - \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_1') \right\|_{2,1} \tag{143}$$

$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$$
$$\left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_1, \boldsymbol{K}_1, \boldsymbol{V}_1)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{C}}^{(0)}; (\boldsymbol{Q}_1, \boldsymbol{K}_1, \boldsymbol{V}_1')) \right\|_{2,1} \tag{144}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \cdot \kappa_2(n, \sqrt{d_E}) \left\| \boldsymbol{V}_1 - \boldsymbol{V}_1' \right\| \tag{145}$$

Finally, we consider $(\boldsymbol{Q}_2, \boldsymbol{K}_2, \boldsymbol{V}_2)$ employing the previously used results:

$$\left\| \boldsymbol{C}(\boldsymbol{Q}_2) - \boldsymbol{C}(\boldsymbol{Q}_2') \right\|_{2,1} \tag{146}$$
$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_2); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_2'); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{147}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_2) - \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{Q}_2') \right\|_{2,1} \tag{148}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$$
$$\left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\boldsymbol{Q}_2, \boldsymbol{K}_2, \boldsymbol{V}_2)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\boldsymbol{Q}_2', \boldsymbol{K}_2, \boldsymbol{V}_2)) \right\|_{2,1} \tag{149}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \kappa_3(n, \sqrt{d}, \sqrt{d}, \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \boldsymbol{Q}_2 - \boldsymbol{Q}_2' \right\| \tag{150}$$

$$\left\| \boldsymbol{C}(\boldsymbol{K}_2) - \boldsymbol{C}(\boldsymbol{K}_2') \right\|_{2,1} \tag{151}$$
$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_2); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_2'); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{152}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_2) - \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{K}_2') \right\|_{2,1} \tag{153}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$$
$$\left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\boldsymbol{Q}_2, \boldsymbol{K}_2, \boldsymbol{V}_2)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\boldsymbol{Q}_2, \boldsymbol{K}_2', \boldsymbol{V}_2)) \right\|_{2,1} \tag{154}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \kappa_4(n, \sqrt{d}, \sqrt{d}, \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \boldsymbol{K}_2 - \boldsymbol{K}_2' \right\| \tag{155}$$

$$\left\| \boldsymbol{C}(\boldsymbol{V}_2) - \boldsymbol{C}(\boldsymbol{V}_2') \right\|_{2,1} \tag{156}$$
$$= \left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_2); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{C}}^{(0)}, \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_2'); (\boldsymbol{Q}_3, \boldsymbol{K}_3, \boldsymbol{V}_3)) \right\|_{2,1} \tag{157}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \left\| \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_2) - \bar{\boldsymbol{X}}^{(1)}(\boldsymbol{V}_2') \right\|_{2,1} \tag{158}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}})$$
$$\left\| \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\boldsymbol{Q}_2, \boldsymbol{K}_2, \boldsymbol{V}_2)) - \mathcal{A}_2^\gamma(\bar{\boldsymbol{X}}^{(0)}, \bar{\boldsymbol{X}}^{(0)}; (\boldsymbol{Q}_2, \boldsymbol{K}_2, \boldsymbol{V}_2')) \right\|_{2,1} \tag{159}$$
$$\leq \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\mathsf{emb}}) \kappa_2(n, \sqrt{d}) \left\| \boldsymbol{V}_2 - \boldsymbol{V}_2' \right\| \tag{160}$$

Putting all these together, and noting the definition of $\|\Theta - \Theta'\|$ as the maximum of the norms of the differences of the individual weight matrices, we can see that:

$$\left\| \boldsymbol{C}(\Theta) - \boldsymbol{C}(\Theta') \right\|_{2,1} \leq \phi \cdot \left\| \Theta - \Theta' \right\|, \tag{161}$$

where the quantity ϕ is defined as (also noting the equations from which the quantity is derived):

$$\phi = \max \left\{ \begin{array}{ll} \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_3(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (135) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_4(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (140) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_2(n, \sqrt{d_E}) & (145) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_3(n, \sqrt{d}, \sqrt{d}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (150) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_4(n, \sqrt{d}, \sqrt{d}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (155) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_2(n, \sqrt{d}) & (160) \\ \kappa_3(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (126) \\ \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (128) \\ \kappa_2(k, \sqrt{d_E}(1+2\Gamma_2)) & (130) \\ \kappa_3(k, \sqrt{d_E}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (118) \\ \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) & (121) \\ \kappa_2(k, \sqrt{d_E}) & (124) \end{array} \right\} \tag{162}$$

To simplify the above definition of ϕ, we note that, for a fixed $(m, \rho, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}})$, a larger ρ' implies larger constant. Thus, we can see that

$$\kappa_2(k, \sqrt{d_E}) < \kappa_2(k, \sqrt{d_E}(1+2\Gamma_2)) \tag{163}$$

$$\kappa_2(k, \sqrt{d}) < \kappa_2(k, \sqrt{d_E}) \tag{164}$$

$$\kappa_3(k, \sqrt{d_E}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) < \kappa_3(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \tag{165}$$

$$\kappa_3(n, \sqrt{d}, \sqrt{d}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) < \kappa_3(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \tag{166}$$

$$\kappa_4(k, \sqrt{d_E}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) < \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \tag{167}$$

$$\kappa_4(n, \sqrt{d}, \sqrt{d}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) < \kappa_4(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}), \tag{168}$$

thus simplifying further to

$$\phi = \max \left\{ \begin{array}{l} \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_3(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_4(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_2(n, \sqrt{d_E}) \\ \kappa_3(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \\ \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \\ \kappa_2(k, \sqrt{d_E}(1+2\Gamma_2)) \end{array} \right\} \tag{169}$$

Next given the definitions of κ_3 and κ_4 in Lemma 6 and Lemma 7 respectively, we can see that if $\rho < \rho'$, $\kappa_4(\cdot)$ provides a larger constant, giving us:

$$\phi = \max \left\{ \begin{array}{l} \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_4(n, \sqrt{d}, \sqrt{d_E}, \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \\ \kappa_5(k, n, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \cdot \kappa_2(n, \sqrt{d_E}) \\ \kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \\ \kappa_2(k, \sqrt{d_E}(1+2\Gamma_2)) \end{array} \right\} \tag{170}$$

Writing things out explicitly while highlighting the dependencies on problem and architectural choices, $n, k, d_{\text{emb}}, d_E$, we first apply the definition of $\kappa_2(\cdots)$ in Lemma 5 to $\kappa_2(k, \sqrt{d_E}(1+2\Gamma_2))$ to get:

$$\kappa_2(k, \sqrt{d_E}(1+2\Gamma_2)) = [1+2\Gamma_2] \; k d_E^{1/2} \tag{171}$$

Next, we apply the definition of $\kappa_4(\cdots)$ in Lemma 7 to $\kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}})$ to get:

$$\kappa_4(k, \sqrt{d_E}, \sqrt{d_E}(1+2\Gamma_2), \Gamma_1, \Gamma_2, \gamma, d_{\text{emb}}) \tag{172}$$

$$= 4(\gamma/\sqrt{d_{\text{emb}}}) k d_E (1+2\Gamma_2)^2 \sqrt{d_E}(2+2\Gamma_2)\Gamma_1\Gamma_2 \exp\left((\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2 d_E (2+2\Gamma_2)^2\right) \tag{173}$$

$$\leq \frac{kd_E^{3/2}}{\sqrt{d_{\text{emb}}}}\left[8\gamma(1+2\Gamma_2)^2(1+\Gamma_2)\Gamma_1\Gamma_2\right]\exp\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\left[\gamma\Gamma_1^2(2+2\Gamma_2)^2\right]\right) \tag{174}$$

$$\leq \left[8\gamma(1+2\Gamma_2)^3\Gamma_1\Gamma_2\right]\cdot\frac{kd_E^{3/2}}{\sqrt{d_{\text{emb}}}}\cdot\exp\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\left[\gamma\Gamma_1^2(2+2\Gamma_2)^2\right]\right) \tag{175}$$

Applying definitions of $\kappa_2(\cdots)$ and $\kappa_5(\cdots)$ from Lemma 5 and Lemma 8 respectively to $\kappa_5(k,n,\sqrt{d_E},\sqrt{d_E}(1+2\Gamma_2),\Gamma_1,\Gamma_2,\gamma,d_{\text{emb}})\,\kappa_2(n,\sqrt{d_E})$, we get:

$$\kappa_5(k,n,\sqrt{d_E},\sqrt{d_E}(1+2\Gamma_2),\Gamma_1,\Gamma_2,\gamma,d_{\text{emb}})\,\kappa_2(n,\sqrt{d_E}) \tag{176}$$

$$= \frac{k\Gamma_2}{n}\left[1+4(\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2 d_E(1+2\Gamma_2)(2+2\Gamma_2)\right]\exp\left((\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2 d_E(2+2\Gamma_2)^2\right)n\sqrt{d_E} \tag{177}$$

$$\leq \Gamma_2\left[kd_E^{1/2}+\frac{kd_E^{3/2}}{\sqrt{d_{\text{emb}}}}\left[8\gamma\Gamma_1^2(1+2\Gamma_2)^2\right]\right]\exp\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\left[\gamma\Gamma_1^2(2+2\Gamma_2)^2\right]\right) \tag{178}$$

Applying definitions of $\kappa_4(\cdots)$ and $\kappa_5(\cdots)$ from Lemma 7 and Lemma 8 respectively to $\kappa_5(k,n,\sqrt{d_E},\sqrt{d_E}(1+2\Gamma_2),\Gamma_1,\Gamma_2,\gamma,d_{\text{emb}})\,\kappa_4(n,\sqrt{d},\sqrt{d_E},\Gamma_1,\Gamma_2,\gamma,d_{\text{emb}})$, we get:

$$\kappa_5(k,n,\sqrt{d_E},\sqrt{d_E}(1+2\Gamma_2),\Gamma_1,\Gamma_2,\gamma,d_{\text{emb}})\,\kappa_4(n,\sqrt{d},\sqrt{d_E},\Gamma_1,\Gamma_2,\gamma,d_{\text{emb}}) \tag{179}$$

$$= \frac{k\Gamma_2}{n}\left[1+4(\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2 d_E(1+2\Gamma_2)(2+2\Gamma_2)\right]\exp\left((\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2 d_E(2+2\Gamma_2)^2\right)$$
$$\cdot\, 4(\gamma/\sqrt{d_{\text{emb}}})nd_E(\sqrt{d}+\sqrt{d_E})\Gamma_1\Gamma_2\exp\left((\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2(\sqrt{d}+\sqrt{d_E})^2\right) \tag{180}$$

$$\leq k\Gamma_2\left[1+\frac{d_E}{\sqrt{d_{\text{emb}}}}\left[8\gamma\Gamma_1^2(1+2\Gamma_2)^2\right]\right]\cdot\frac{d_E^{3/2}}{\sqrt{d_{\text{emb}}}}\left[8\gamma\Gamma_1\Gamma_2\right]\cdot\exp\left((\gamma/\sqrt{d_{\text{emb}}})\Gamma_1^2\left[d_E(2+2\Gamma_2)^2+4d_E\right]\right) \tag{181}$$

$$= \left[8\gamma\Gamma_1\Gamma_2^2\right]\cdot\left[\frac{kd_E^{3/2}}{\sqrt{d_{\text{emb}}}}+\frac{kd_E^{5/2}}{d_{\text{emb}}}\left[8\gamma\Gamma_1^2(1+2\Gamma_2)^2\right]\right]\cdot\exp\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\left[4\gamma\Gamma_1^2\left[(1+\Gamma_2)^2+1\right]\right]\right) \tag{182}$$

Note that in the above terms, there are two quantities that seem to repeatedly occur: (i) $k\sqrt{d_E}$, and (ii) $\frac{d_E}{\sqrt{d_{\text{emb}}}}$. Considering Γ_1,Γ_2 to be constants that do not depend on the architectural choices or problem setting (such as the number of clusters, points, data dimensionality or token embedding scheme), we can write ϕ as following:

$$\phi = \max\left\{\begin{array}{l} c_1\left(k\sqrt{d_E}\right) \\ c_2\cdot\gamma\cdot\left(k\sqrt{d_E}\right)\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\cdot\exp\left(c_3\cdot\gamma\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\right) \\ \Gamma_2\left[\left(k\sqrt{d_E}\right)+c_4\cdot\gamma\cdot\left(k\sqrt{d_E}\right)\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\right]\cdot\exp\left(c_3\cdot\gamma\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\right) \\ \gamma\cdot\left(k\sqrt{d_E}\right)\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\left[c_5+c_6\cdot\gamma\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\right]\cdot\exp\left(c_7\cdot\gamma\cdot\left(\frac{d_E}{\sqrt{d_{\text{emb}}}}\right)\right) \end{array}\right\} \tag{183}$$

where

$$c_1 = (1+2\Gamma_2) \tag{184}$$
$$c_2 = 8\Gamma_1\Gamma_2 c_1^3 = 4\Gamma_1\Gamma_2(1+2\Gamma_2)^3 \tag{185}$$
$$c_3 = 4\Gamma_1^2(1+\Gamma_2)^2 \tag{186}$$
$$c_4 = 8\Gamma_1^2 c_1^2 = 4\Gamma_1^2(1+2\Gamma_2)^2 \tag{187}$$
$$c_5 = 8\Gamma_1\Gamma_2^2 \tag{188}$$
$$c_6 = c_4\cdot c_5 = 64\Gamma_1^3\Gamma_2^2(1+2\Gamma_2)^2 \tag{189}$$
$$c_7 = c_3 + 4\Gamma_1^2 = 4\Gamma_1^2((1+\Gamma_2)^2+1). \tag{190}$$

Thus, simplifying all, we can see that

$$\phi \sim O\left(\max\{1,\gamma^2\}\cdot\left(k\sqrt{d_E}\right)\cdot\max\left\{1,\left(\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)^2\right\}\cdot\mathsf{poly}\left(\exp\left(\gamma\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)\right)\right), \quad (191)$$

which gives us the result in equation (115) in the statement of the theorem. □

If $d_E > \sqrt{d_{\mathsf{emb}}}$, which is the case with the proposed binary initial token embeddings, with $d_E = d + \lceil \log_2 k \rceil$ and $d_{\mathsf{emb}} = d + \lceil \log_2 k \rceil$, thus, $\frac{d_E}{\sqrt{d_{\mathsf{emb}}}} = \sqrt{d + \lceil \log_2 k \rceil} > 1$, we can see that

$$\phi \sim O\left(\max\{1,\gamma^2\}\cdot\left(k\sqrt{d_E}\right)\cdot\left(\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)^2\cdot\mathsf{poly}\left(\exp\left(\gamma\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)\right)\right). \quad (192)$$

However $d_E < \sqrt{d_{\mathsf{emb}}}$ is possible with the `OH` initial token embedding of Clarkson et al. [27] as $d_E = d + 1$ and $d_{\mathsf{emb}} = d + k$, and $k \sim O((d+1)^2)$ implies that $d + 1 < \sqrt{d + k}$. In this case, we have

$$\phi \sim O\left(\max\{1,\gamma^2\}\cdot\left(k\sqrt{d_E}\right)\cdot\mathsf{poly}\left(\exp\left(\gamma\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)\right)\right). \quad (193)$$

In all cases, for a fixed k, an architecture with a larger d_{emb} and smaller d_E results in a smaller ϕ.

Combining the above Theorem 4 with Lemma 2, we can obtain the following main result regarding the Lipschitz-ness of the smoothed clustering loss function $\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta))$ with respect to the learnable model parameters Θ (this is an extended version of Theorem 2):

Theorem 5. *Assuming bounded points and initial cluster centers $(\boldsymbol{X},\boldsymbol{C}) \sim \mathcal{D}$ (assumption 2), and bounded learnable model parameters Θ, Θ' (assumption 3), we can show that*

$$\left|\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta)) - \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta'))\right| \leq \varphi\,\|\Theta - \Theta'\|,$$
$$\textit{where } \varphi \sim O\left(\frac{kn\omega_{\tau}}{\tau}\cdot\max\{1,\gamma^2\}\cdot\sqrt{d_E}\cdot\max\left\{1,\left(\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)^2\right\}\cdot\mathit{poly}\left(\exp\left(\gamma\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)\right)\right) \quad (194)$$

Proof. We can get the above result by combining equation (115) in Theorem 4 with equation (45) in Lemma 2:

$$\left|\tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta)) - \tilde{\mathcal{L}}^{\tau}_{\Omega}(\boldsymbol{X}, F^{\gamma}_{d:k}(\boldsymbol{X},\boldsymbol{C};\Theta'))\right| \leq \varphi\cdot\|\Theta - \Theta'\|$$
$$\text{where } \varphi \sim O\left(\frac{n\omega_{\tau}}{\tau}\cdot\max\{1,\gamma^2\}\cdot\left(k\sqrt{d_E}\right)\cdot\max\left\{1,\left(\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)^2\right\}\cdot\mathsf{poly}\left(\exp\left(\gamma\frac{d_E}{\sqrt{d_{\mathsf{emb}}}}\right)\right)\right), \quad (195)$$

thus giving us the main result. □

This highlights the role of the different choices we make such as the choice of the $\min$-regularizer Ω, the corresponding regularization penalty τ, the attention inverse temperature γ, the number of points n and the number of clusters k, the embedding dimension d_{emb} and the maximum initial cluster center embedding norm d_E.

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 claims regarding novel theoretical constructions and understanding are presented in Section 2 and Section 3, while the claims regarding the empirical validation of the theoretical insights is presented in Section 3 and Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Explicit discussions of limitations are presented in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: The main theoretical results in this paper, Theorem 1 and Theorem 2, are presented in complete details with necessary assumptions and elaborate proofs in Appendix B and Appendix I respectively.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: The experimental settings are clearly described in Section 3 and Section 4, and further details regarding the data generation, compute resources, and hyperparameter ablations are presented in Appendix D, Appendix E and Appendix F respectively.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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: The learning algorithm and experimental hyperparameters and ablations are detailed in the Appendix, and the code and instructions are part of the supplementary material, where the `README.md` provides instructions for setting up the environment, while the `expts.md` details the exact command to train and evaluate the models, and to generate the plots in the paper. The code was submitted with the original submission, and now is available at https://github.com/rithram/kmeans-trf.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: The experimental settings are clearly described in Section 3 and Section 4, and further details regarding the data generation, compute resources, and hyperparameter ablations are presented in Appendix D, Appendix E and Appendix F respectively.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

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: As discussed in Appendix E, and in the experiments for Figure 2, all configurations are repeated 10 times with different seeds, and the results are aggregated over these 10 trials. In all the results, we report the median values and the inter-quartile ranges are shown as translucent ribbons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: The compute resources for the experiments are detailed in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: This paper focuses on technical research, not involving any human subjects, and solely utilized synthetic data generated from standard probability distributions. This work also does not pose any societal impact or harm.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This paper focuses on ability of an architecture to express and learn an algorithm, and this work has no societal impact.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper focuses on ability of an architecture to express and learn an algorithm, and poses no risks that would necessitate safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: [N/A]

    Justification: This paper does not make use of existing assets except standard python libraries that are appropriately attributed.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: This paper does not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: [N/A]

    Justification: This paper does not involve any crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: [N/A]

    Justification: No human subjects were involved in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were solely used as part of the Paper Assistant Tool provided through OpenReview and the NeurIPS program committee to provide a review of an initial draft.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.