Learning Functions on Multiple Sets using Multi-Set Transformers (Supplementary Material)

Kira A. Selby[1,2] **Ahmad Rashid**[1,2,3] **Ivan Kobyzev**[3] **Mehdi Rezagholizadeh**[3] **Pascal Poupart**[1,2]

[1]Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
[2]Vector Institute , Toronto, Ontario, Canada
[3]Huawei Noah's Ark Lab, Montreal, Quebec, Canada

1 ATTENTION DERIVATION

A typical self-attention block with the set $X \in \mathbb{R}^{n \times d}$ as the input queries and keys/values obeys the following equation:

$$\begin{aligned} Z &= \text{MHA}(X, X) \\ &= \left[\sigma\left((XW_Q)(XW_K)^T\right)(XW_V)\right] W_O \\ &= \sigma\left(X(W_Q W_K^T)X^T\right) X W_V W_O \end{aligned}$$

If we now consider the joint set $X \bigsqcup Y \in \mathbb{R}^{n+m \times d}$ and perform self-attention on that, we find the following:

$$\begin{aligned} \begin{pmatrix} Z_X \\ Z_Y \end{pmatrix} &= \text{ATTN}\left(\begin{pmatrix} X \\ Y \end{pmatrix}, \begin{pmatrix} X \\ Y \end{pmatrix}\right) \\ &= \left[\sigma\left(\begin{pmatrix} XW_Q \\ YW_Q \end{pmatrix}\begin{pmatrix} XW_K \\ YW_K \end{pmatrix}^T\right)\begin{pmatrix} X \\ Y \end{pmatrix} W_V\right] W_O \\ &= \begin{pmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{pmatrix}\begin{pmatrix} X \\ Y \end{pmatrix} W_V W_O \end{aligned}$$

wherein

$$\begin{pmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{pmatrix} = \sigma\left(\begin{pmatrix} X(W_Q W_K^T)X^T & X(W_Q W_K^T)Y^T \\ Y(W_Q W_K^T)X^T & Y(W_Q W_K^T)Y^T \end{pmatrix}\right)$$

Then, we find that

$$\begin{aligned} Z_X &= A_{xx} X W_V W_O + A_{xy} Y W_V W_O \\ Z_Y &= A_{yx} X W_V W_O + A_{yy} Y W_V W_O \end{aligned}$$

This function remains entirely equivariant with respect to the order of the elements in the joint set $X \bigsqcup Y$ - there is no distinction between elements in X and elements in Y.

Suppose, however, that we were to let the softmax for $A_{\alpha\beta}$ be computed only over the elements of the set α, and let the parameter matrices be different for each of the 4 terms. Then, we find

$$\begin{aligned} Z_X &= \sigma(X(W_{Q,xx} W_{K,xx}^T)X^T) X W_{V,xx} W_{O,xx} \\ &\quad + \sigma(X(W_{Q,xy} W_{K,xy}^T)Y^T) Y W_{V,xy} W_{O,xy} \\ Z_Y &= \sigma(Y(W_{Q,yx} W_{K,yx}^T)X^T) X W_{V,yx} W_{O,yx} \\ &\quad + \sigma(Y(W_{Q,yx} W_{K,yy}^T)Y^T) Y W_{V,yy} W_{O,yy} \end{aligned}$$

These are just four separate attention blocks! Now we can simply write

$$Z_X = \text{MHA}_{xx}(X, X) + \text{MHA}_{xy}(X, Y)$$
$$Z_Y = \text{MHA}_{yx}(Y, X) + \text{MHA}_{yy}(Y, Y)$$

Since the attention function is equivariant with respect to the first input and invariant with respect to the second, this meets the conditions for partial equivariance. To make this slightly more general, we can now write

$$Z_X = g_x\left(\text{MHA}_{xx}(X, X), \text{MHA}_{xy}(X, Y)\right)$$
$$Z_Y = g_y\left(\text{MHA}_{yx}(Y, X), \text{MHA}_{yy}(Y, Y)\right)$$

where the function g acts on each vector in the output set independently.

2 EXPERIMENT DETAILS

The base architecture used in all experiments was the architecture shown in Figure 1, with the MSAB blocks replaced as appropriate for each baseline. The only exception was the PINE model, which followed the architecture described in their paper.

In all cases (except where noted otherwise), we used architectures with 4 blocks, 4 attention heads (for the transformer models), and 1-layer feedforward decoders. We used Pooling by Multiheaded Attention (PMA) (see Lee et al. [2019]) as the pooling layer for the overall network, and max pooling within each relation network block. We used layer norm around each encoder block, as well as within the transformer blocks as per usual. Each block used the same latent and hidden size, and linear projection layers were added at the beginning of the network to project the inputs to the correct dimension if needed.

For the KL and MI experiments, we trained for 100,000 batches of size 64 with a learning rate of 1e-4. The models used a latent size of 16 per input dimension and feedforward size of 32 per input dimension. The dimension-equivariant model was trained across data of multiple dimensions (1-3 for $d = 2$, 3-5 for $d = 4$, 7-9 for $d = 8$ and 14-18 for $d = 16$). Sets were generated as described in sections 5.1.1 and 5.1.2.

For the Counting experiments, we trained with a batch size of 64 using a latent size of 128 and hidden size of 256. We used a single projection layer as a decoder, with no hidden layers. For MNIST, we used a convolutional encoder with 3x3 convolutional layers of 32 and 64 filters, each followed by a max pool, with a linear projection to the latent size of 128 at the end. This encoder was pretrained for 1000 batches, then the network and encoder were trained end to end for 10,000 batches with a learning rate of 3e-4. Sets were randomly sampled with set size randomly selected in $[10, 30]$. For Omniglot, the convolutional encoder used one 7x7 conv with stride 2 and 32 filters, followed by three blocks of two 3x3 convs each with 32, 64 and 128 filters respectively. Each block was followed by a max pool, and a final linear projection to size 128 was again added at the end. This was pretrained for 300 batches, then the network itself was trained end to end for 10,000 batches with a learning rate of 1e-4. Loss was calculated by mean-squared error. Sets were randomly sampled with set size randomly selected in $[6, 10]$.

The CoCo experiments again used convolutional encoders to obtain fixed size representations of each image, and used transformer encoders to do the same for the captions. This time, the pretrained ResNet-101 model was used as the image encoder, with BERT used as the text encoder. The model was trained for 2500 batches of batch size 48 with learning rate 1e-5, using a latent size of 512 and hidden size of 1024. The set size was increased gradually according to a schedule during training, beginning at sets with size randomly selected in $[1, 5]$ for 1250 batches, $[3, 10]$ for 625 batches and $[8, 15]$ for 625 batches. Standard image preprocessing techniques were applied, with each image rescaled to 256x256, center cropped to size 224, then normalized according to the method expected by PyTorch's pretrained ResNet models (see https://pytorch.org/vision/stable/models.html). The FastText experiments used common crawl FastText vectors for English and French [1], with ground truth translations taken from MUSE [2]. The model was trained for 3125 batches of batch size 128 with learning rate 1e-5, using a latent size of 512 and hidden size of 1024. The set size was increased gradually according to a schedule during training, beginning at sets with size randomly selected in $[1, 5]$ for 1250 batches, $[3, 10]$ for 625 batches, $[8, 15]$ for 625 batches and $[10, 30]$ for 625 batches.

Meta-Dataset experiments used the same convolutional encoder architecture as the Omniglot experiments, though without pretraining. Images were preprocessed in the standard fashion performed by the Pytorch Meta-Dataset library. The model

[1] https://fasttext.cc/docs/en/crawl-vectors.html

[2] https://github.com/facebookresearch/MUSE#ground-truth-bilingual-dictionaries

was trained for 7500 batches of batch size 64 with set size randomly selected in $[10, 30]$ and learning rate 1e-5, using a latent size of 512 and hidden size of 1024. The synthetic experiments for distinguishability were trained for 7500 batches of batch size 256 with set size randomly selected in $[10, 30]$ and learning rate 1e-5, using a latent size of 8 and hidden size of 16.

3 PROOF OF THEOREM 4.2

Our proof will closely follow the work of Yun et al. [2019]. In their work, they define the following theorem:

Theorem 2 [Yun et al., 2019]. *Let $1 \leq p \leq \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{PE}$ there exists a transformer network $g \in \mathcal{T}^{2,1,4}$ such that $d_p(f, g) \leq \epsilon$.*

The proof of this theorem follows several stages, enumerated here:

1. Approximate $\mathcal{F}_{\text{PE}}$ by piecewise constant functions on the grid of resolution δ, denoted $\mathbb{G}_\delta = 0, \delta, ..., 1 - \delta^{d \times n}$.
2. Approximate $\overline{\mathcal{F}}_{\text{PE}}$ by modified transformers, with hardmax replacing softmax.

   Proposition 4 [Yun et al., 2019]. *For each $\overline{f} \in \overline{\mathcal{F}}_{PE}$ and $1 \leq p < \infty$, $\exists \overline{g} \in \overline{\mathcal{T}}^{2,1,1}$ such that $d_p(\overline{f}, \overline{g}) = O(\delta^{d/p})$*

3. Approximate the class of modified transformers $\overline{\mathcal{T}}^{2,1,1}$ with $\mathcal{T}^{2,1,4}$

Each of these steps leads to a certain error under d_p, with step 1 and 3 contributing an error of order $\epsilon/3$ and step 2 contributing an error of order $\mathcal{O}(\delta^{d/p})$. This leads to a total error of less than order ϵ, so long as δ is chosen to be sufficiently small. Of these steps, we will focus our attention on step 2, as the others remain unchanged. The key is to prove our own version of proposition 4 [Yun et al., 2019]. The proof of this proposition itself follows the following steps:

1. Given $X \in \mathbb{R}^{d \times n}$, quantize X to $L^{(x)} \in G_\delta^+$, where G_δ is the $[0, 1]^{d \times n}$ grid with resolution δ and $G_\delta^+ = G_\delta \cup \{-\delta^{-nd}\}$.
2. Implement a *contextual mapping* $q(X)$ such that all elements of $q(L), q(L')$ are distinct if L, L' are not permutations of each other. This essentially maps each (x_i, X) to a unique representation.
3. Since each (x_i, X) is mapped to a unique representation, we can use feedforward networks to approximate any desired decoder to approximate any equivariant function on X.

The critical part of this proof comes in Lemma 6 from Yun et al:

Lemma 6 [Yun et al., 2019]. *Let $\widetilde{G}_\delta = \{L \in G_\delta | \; L_{:,i} \neq L_{:,j} \; \forall i \neq j\}$. Let $n \geq 2$ and $\delta^{-1} \geq 2$. Then, there exists a function $q(L)$ of the form $q(L) = u^T g_c(L)$ where $u \in \mathbb{R}^d$, and $g_c(L) : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n}$ is a function composed of $\delta^{-d} + 1$ self-attention layers using the σ_H operator, such that $q(L)$ has the following properties:*

1. *For any $L \in \widetilde{G}_\delta$, all entries of $q(L)$ are distinct*
2. *For any $L, L' \in \widetilde{G}_\delta$, if L is not a permutation of L' then all entries of $q(L)$ and $q(L')$ are distinct*
3. *For any $L \in \widetilde{G}_\delta$, all entries of q(L) are in $[t_l, t_r]$*
4. *For any $L \in G_\delta^+ \setminus \widetilde{G}_\delta$, all entries of q(L) are not in $[t_l, t_r]$*

For our purposes, we will keep the structure of this proof, and change it by substituting our own version of this lemma:

Lemma 6'. *Let $\widetilde{G}_\delta = \{L \in G_\delta | \; L_{:,i} \neq L_{:,j} \; \forall i \neq j\}$. Let $n \geq 2$ and $\delta^{-1} \geq 2$. Then, there exists a function $q(L^X, L^Y)$ on $L^X, L^Y \in G_\delta^+; L^X \neq L^Y$ consisting of $2\delta^{-d} + 4$ MSAB layers and constants $t_l^X, t_r^X, t_l^Y, t_r^Y$ such that*

1. *For any $L^X, L^Y \in G_\delta$, all entries of $q(L^X, L^Y)$ are distinct*
2. *For any $L^X, L^Y, L'^X, L'^Y \in G_\delta$, if L'^X is not a permutation of L^X and L'^Y is not a permutation of L^Y then all entries of $q(L^X, L^Y)$ and $q(L'^X, L'^Y)$ are distinct*
3. *For any $L^X, L^Y \in \widetilde{G}_\delta$, all entries of $q^X(L^X, L^Y)$ are in $[t_l^X, t_r^X]$*
4. *For any $L^X, L^Y \in \widetilde{G}_\delta$, all entries of $q^Y(L^X, L^Y)$ are in $[t_l^Y, t_r^Y]$*
5. *For any $L^X \in G_\delta^+ \setminus \widetilde{G}_\delta$, $L^Y \in G_\delta^+$, all entries of $q(L^X, L^Y)$ are not in $[t_l, t_r]$*

6. *For any $L^Y \in G_\delta^+ \setminus \widetilde{G}_\delta$, $L^X \in G_\delta^+$, all entries of $q(L^X, L^Y)$ are not in $[t_l, t_r]$*

Once our version of Lemma 6 is established, we then make a slight modification of Yun et al's Lemma 7. This lemma states:

Lemma 7 [Yun et al., 2019]. *Let $g_c : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ be the function defined in Lemma 6 [Yun et al., 2019]. Then, there exists a function $g_v : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ composed of $O(n(\frac{1}{\delta})^{dn}/n!)$ column-wise feedforward layers ($r = 1$) such that g_v is defined by a function $g_{col} : \mathbb{R}^d \to \mathbb{R}^d$,*

$$g_v(Z) = [g_{col}(Z_{:,1}), ..., g_{col}(Z_{:,n})]$$

where $\forall j = 1, ..., n$

$$g_{col}(g_c(L)_{:,j}) = \begin{cases} (A_L)_{:,j} & L \in \widetilde{G_\delta} \\ 0_d & L \in G_\delta^+ \setminus \widetilde{G_\delta} \end{cases}$$

Our modified version states

Lemma 7'. *Let $g_c : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times n}$ be the function defined in Lemma 6'. Then, there exists a function $g_v : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times n}$ composed of $O(n(\frac{1}{\delta})^{dn}/n!)$ column-wise feedforward layers ($r = 1$) such that g_v is defined by a function $g_{col} : \mathbb{R}^d \to \mathbb{R}^d$,*

$$g_v(Z) = [g_{col}(Z_{:,1}), ..., g_{col}(Z_{:,n})]$$

where $\forall i = 1, ..., n, j = 1, ...m$

$$g_{col}(g_c(L)_{:,j}) = \begin{cases} (A_L)_{:,j} & L \in \widetilde{G_\delta} \\ 0_d & L \in G_\delta^+ \setminus \widetilde{G_\delta} \end{cases}$$

The construction of $q(L^X, L^Y)$, and the proofs of Lemma 6' and 7' will follow in the subsequent sections.

3.1 CONSTRUCTION OF THE CONTEXTUAL MAPPING

The construction of this function proceeds as follows. First, note that a multi-set attention block can implement functions consisting of multiple feedforward or attention blocks successively by using the skip connections to ignore the other component when needed. It can also implement a function consisting of multiple blocks successively applied to a single one of the input sets by simply letting the action of the block on the other set be the identity. The multi-set attention block can also implement a layer which simply performs the attention computation e.g. $ATTN_{XX}(X, X)$ by letting $g_X(f_{XX}, f_{XY}) = f_{XX}$. As such, multi-set attention blocks can reproduce any function on either X or Y implemented by regular transformer blocks - such as the constructions defined in Yun et al. [2019].

Let $g_c(L)$ be the iterated selective shift network defined in Yun et al. [2019], consisting of n selective shift operations followed by a final global shift layer. This results in the mapping

$$u^T g_c(L) = \widetilde{\ell}_j + \delta^{-(n+1)d} \widetilde{\ell}_n$$

where $\widetilde{\ell}_j$ is the j-th output of the selective shift layers, sorted in ascending order. We will now construct our own analogous network, $g_c(L^X, L^Y)$. Let $g_s(L)$ be the selective shift portion of $g_c(L)$. Then, we let the first δ^{-d} blocks implement $g_s(L^X)$ on L^X alone while performing the identity operation on L^Y ($T_{XY}, T_{YX} = 0$ for this component). The next δ^{-d} blocks do the same thing on L^Y, while performing the identity on L^X. The next block then applies a modified global shift to each set with attention component $\delta^{(n+1)d}\psi(\cdot; 0)$ - the same global shift as in Yun et al. [2019]. This shift is applied with attention over X, and a scaled version is also applied with attention over Y - i.e. shifting L^X by $\delta^{-(n+1)d} \max_k u^T L^X_{:,k}$ *and* $\delta^d \max_k u^T L^Y_{:,k}$ (and the same in reverse for L^Y). This can be implemented by a single MSAB block with $T_{XX} = T_{YY} = \delta^{-(n+1)d}\psi(\cdot; 0)$ and $T_{XY} = T_{YX} = \delta^d \psi(\cdot; 0)$. This comprises our $g_c(L^X, L^Y)$ block, and results in an output of

$$q^X(L^X, L^Y)_j = u^T g_c^X(L^X, L^Y)_{:,j} = \widetilde{\ell}_j^X + \delta^{-(n+1)d}\, \widetilde{\ell}_n^X + \delta^{(m+1)d}\, \widetilde{\ell}_m^Y$$
$$q^Y(L^X, L^Y)_j = u^T g_c^Y(L^X, L^Y)_{:,j} = \widetilde{\ell}_j^Y + \delta^{-(n+1)d}\, \widetilde{\ell}_m^Y + \delta^{(n+1)d}\, \widetilde{\ell}_n^X$$

3.2 PROOF OF LEMMA 6'

The proof of Lemma 6' proceeds much as the proof of Lemma 6 [Yun et al., 2019]. We must now check that all conditions are satisfied.

3.2.1 Property 2

For the second property, let us begin by considering the case where L^X, L'^X are not permutations of each other. Then, analogous to Yun et al., we have that

$$u^T g_c^X(L^X, L^Y)_{:,j} \in [\delta^{-(n+1)d}\widetilde{\ell}_n^X + \delta^{(m+1)d}\widetilde{\ell}_m^Y, \delta^{-(n+1)d}\widetilde{\ell}_n^X + \delta^{(m+1)d}\widetilde{\ell}_m^Y + \delta^{-(n+1)d+1} - \delta^{-nd+1})$$

As in Yun et al., $L^X, L'^X \in \widetilde{G_\delta}$ which are not permutations of each other must result in $\widetilde{\ell}_n^X, \widetilde{\ell}_n'^X$ differing by at least δ. By Lemma 10, distinct L^Y, L'^Y can lead to $\widetilde{\ell}_m^Y, \widetilde{\ell}_m'^Y$ differing by a value strictly less than $\delta^{-(m+1)d+1}$. The smallest net change this can result in is $\delta^{-(n+1)d} \cdot \delta - \delta^{(m+1)d} \cdot \delta^{-(m+1)d+1} = \delta^{-(n+1)d+1} - \delta$. Since this is larger than the width of the original interval and the intervals are open on at least one end, the intervals must thus be disjoint, and thus if L^X and L'^X are distinct, Q^X and Q'^X must be distinct. Now, consider the case where $L^X, L'^X \in \widetilde{G_\delta}$ are permutations of each other, but $L^Y, L'^Y \in \widetilde{G_\delta}$ are not. In this case, since $\widetilde{\ell}_m^Y, \widetilde{\ell}_m'^Y$ must differ by at least δ, Q^X and Q'^X must again be distinct. Since $|\widetilde{\ell}_m^Y - \widetilde{\ell}_m'^Y| < \delta^{-(m+1)d+1}$, the resulting change in Q^X must be strictly less than δ. Since $\widetilde{\ell}_j^X, \widetilde{\ell}_k^X$ must be separated by at least δ for $j \neq k$, $Q_j^X \neq Q_k'^X$ for any $j \neq k$. Thus, all entries of Q^X and Q'^X must be distinct in this case as well. These results apply symmetrically for Q^Y and Q'^Y, and thus this proves Property 2.

Note that if L^X, L'^X are not permutations of each other then $u^T g_c^X(L^X, L^Y), u^T g_c^X(L'^X, L'^Y)$ must be separated from each other by at least δ, whereas if L^X, L'^X are permutations of each other, but L^Y, L'^Y are not, $u^T g_c^X(L^X, L^Y), u^T g_c^X(L'^X, L'^Y)$ must be separated by at least δ^{m+2}. In general then, all entries of $u^T g_c^X(L^X, L^Y), u^T g_c^X(L'^X, L'^Y)$ must be separated from each other by at least δ^{m+2} (and conversely with δ^{n+2} for Y).

3.2.2 Properties 3-4

By the same procedure as Yun et al (in B.5.1), we can see that $q^X(L^X, L^Y)$ obeys

$$(\delta^{-2nd+1} + \delta^{2d+1})(\delta^{-d} - 1) \leq u^T g_c^X(L^X, L^Y) < (\delta^{-(2n+1)d+1} + \delta^{d+1})(\delta^{-d} - 1)$$

if $L^X, L^Y \in \widetilde{G}_\delta$. Thus, $\forall L^X, L^Y \in \widetilde{G}_\delta$ $q^X(L^X, L^Y) \in [t_l, t_r]$ where

$$t_l^X = (\delta^{-2nd+1} + \delta^{2d+1})(\delta^{-d} - 1)$$
$$t_r^X = (\delta^{-(2n+1)d+1} + \delta^{d+1})(\delta^{-d} - 1)$$

The same holds for $q^Y(L^X, L^Y)$ with

$$t_l^Y = (\delta^{-2md+1} + \delta^{2d+1})(\delta^{-d} - 1)$$
$$t_r^Y = (\delta^{-(2m+1)d+1} + \delta^{d+1})(\delta^{-d} - 1)$$

3.2.3 Property 1

Within Q^X and Q^Y, all entries must be distinct since $\widetilde{\ell}_1 < ... < \widetilde{\ell}_n$. Suppose $n \neq m$. Consider the bounds t_l^X, t_r^X from the previous section. Without loss of generality, suppose $m < n$. If $m = n - k$ then we have $t_r^Y = (\delta^{-(2n+1-2k)d+1} + \delta^{d+1})(\delta^{-d} - 1) < t_l^X$ and thus Q^X and Q^Y belong to disjoint intervals.

If $n = m$, then we can apply a similar argument as we did in proving Property 2, and argue that $\widetilde{\ell}_n^X, \widetilde{\ell}_n^Y$ which are not permutations must differ by at least δ. This results in intervals shifted from each other by at least $\delta^{-(n+1)d+1} - \delta^{d+1}$, which will always be larger than $\delta^{-(n+1)d+1} - \delta^{-nd+1}$, which is the width of the intervals. Thus, in this case too Q^X and Q^Y must be distinct, and Property 1 is proven.

3.2.4 Properties 5-6 - Case 1

Take $L^Y \in G_\delta^+$ and $L^X \in G_\delta \setminus \widetilde{G}_\delta$. This corresponds to an L^X with duplicate columns but within the region of compact support. In this case,

$$\widetilde{\ell}_n^X \leq \delta^{-(n-1)d+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2$$

Then,

$$\begin{aligned}
u^T g_c^X(L^X, L^Y)_{:,j} &= \widetilde{\ell}_j^X + \delta^{-(n+1)d}\, \widetilde{\ell}_n^X + \delta^{(m+1)d}\, \widetilde{\ell}_m^Y \\
&\leq (\delta^{-(n+1)d} + 1)(\delta^{-(n-1)d+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2) + \delta^{(m+1)d}(\delta^{-md+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2)) \\
&< \delta^{-2nd+1}(\delta^{-d} - 1) + \delta^{-(n-1)d+1}(\delta^{-d} - 1) - \delta^{-(n+1)d+1}(\delta^{-d} - 1)^2 + \delta^{d+1}(\delta^{-d} - 1) \\
&< \delta^{-2nd+1}(\delta^{-d} - 1) + \delta^{-(n-1)d+1}(\delta^{-d} - 1)(1 + \delta^{nd} - \delta^{-2d}(\delta^{-d} - 1)) \\
&< \delta^{-2nd+1}(\delta^{-d} - 1) < t_l^X
\end{aligned}$$

since $\delta^{-d} \geq 2$. Thus, Q^X falls strictly outside $[t_l^X, t_r^X]$, proving Property 6 for the case when $L^Y \in G_\delta^+$ and $L^X \in G_\delta \setminus \widetilde{G}_\delta$. The same applies in reverse for Property 7.

3.2.5 Properties 5-6 - Case 2

Take $L^Y \in G_\delta^+$ and $L^X \in G_\delta^+ \setminus G_\delta$. This corresponds to an L^X that contains at least one element outside the region of compact support. This leads to columns of L^X containing negative values. Note first that for a column $L_{:,j}^X$ containing $-\delta^{-nd}$, $\ell_j^X = \widetilde{\ell}_j^X$ as the selective shift operation does not alter it. From Yun et al, we have that $\widetilde{\ell}_j^X \leq -\delta^{-nd} + \delta^{-d+1} - 1 < 0$, and that the last layer shifts negative values by $\delta^{-(n+1)d} \min_k \widetilde{\ell}_k^X$. After this shift is applied and our additional attention-based shift is applied,

$$\begin{aligned}
u^T g_c^X(L^X, L^Y)_{:,j} &\leq (-\delta^{-nd} + \delta^{-d+1} - 1)(1 + \delta^{-(n+1)d}) + \delta^{(m+1)d}\widetilde{\ell}_m^Y \\
&\leq (-\delta^{-nd} + \delta^{-d+1} - 1)(1 + \delta^{-(n+1)d}) + \delta^{(m+1)d}\delta^{-(m+1)d+1} \\
&\leq -\delta^{-(2n+1)d} + \delta^{-(n+2)d+1} + \delta < 0 < t_l^X
\end{aligned}$$

where we use that $\widetilde{\ell}_m^Y \leq \delta^{-(m+1)d+1}$ and $\delta^{-1} \geq 2$. Thus, any negative column is mapped to a value outside of $[t_l^X, t_r^X]$. Note that this holds for any L^Y, including an $L^Y \in G_\delta^+ \setminus G_\delta$.

In the case where *all* columns are negative, the argument proceeds exactly as in Yun et al, and all elements are straightforwardly less than zero as shown above. In the case where only some columns are negative, the negative columns are mapped to negative values as before, and the positive columns satisfy $u^T g_c^X(L^X, L^Y)_{:,i} \geq \delta^{-(2n+2)d+}1$ before we apply our attention shift layer. If $\max_k \widetilde{\ell}_k^Y > 0$, then $u^T g_c^X(L^X, L^Y)_{:,i} \geq \delta^(- (2n+2)d + 1$ still holds. If not, all entries of $\widetilde{\ell}_k^Y$ must be negative, which means $\max_k \widetilde{\ell}_k^Y = -\delta^{-md}(\delta^{-d} - 1)$. We then have

$$u^T g_c^X(L^X, L^Y)_{:,i} \geq \delta^{-(2n+2)d+1} - \delta^d(\delta^{-d} - 1) > t_r^X \tag{1}$$

and thus Property 5 is proved for all cases (and symmetrically for Property 6).

3.3 PROOF OF LEMMA 7'

This proof very closely follows the proof shown in Yun et al, with a few small changes. The first layer used to map all invalid entries to strictly negative numbers becomes

$$Z \mapsto Z - (M - 1)\mathbf{1}_{n+m}(\phi^X(u^T Z^X) + \phi^Y(u^T Z^Y)) \tag{2}$$

where M is the maximum value of the image of $g_c(G_\delta^+, G_\delta^+)$ and

$$\phi^X(t) = \begin{cases} 0 & t \in [t_l^X, t_r^X] \\ 1 & t \notin [t_l^X, t_r^X] \end{cases}$$

$$\phi^Y(t) = \begin{cases} 0 & t \in [t_l^Y, t_r^Y] \\ 1 & t \notin [t_l^Y, t_r^Y] \end{cases}$$

This layer is applied to both X and Y, and ensures that if either X or Y contain invalid elements, the entirety of both sets are mapped to negative values. The next d layers, which map all negative entries to the zero matrix, remain unchanged and are applied again to both X and Y.

The remaining layers must now map $g_c^X(L^X, L^Y)$ to A_L^X and $g_c^Y(L^X, L^Y)$ to A_L^Y. In a similar fashion to Yun et al, we add $\mathcal{O}\left((n+m)(1/\delta)^{d(n+m)}/(n!m!)\right)$ feedforward layers, each of which maps one value of $u^T g_c^X(L^X, L^Y)$ or $u^T g_c^Y(L^X, L^Y)$ to the correct output while leaving the others unaffected. For a given value of $u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j}$ these layers take the form:

$$Z^X \mapsto Z^X + \left((A_{\bar{L}}^X)_{:,j} - g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j}\right) \phi^X \left(u^T Z^X - u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_n^T\right)$$
$$Z^Y \mapsto Z^Y + \left((A_{\bar{L}}^Y)_{:,j} - g_c^Y(\bar{L}^X, \bar{L}^Y)_{:,j}\right) \phi^Y \left(u^T Z^Y - u^T g_c^Y(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_m^T\right)$$

wherein

$$\phi^X(t) = \begin{cases} 0 & t < -\delta^{m+2}/2 \text{ or } t \geq \delta^{m+2}/2 \\ 1 & -\delta^{m+2}/2 \leq t < \delta^{m+2}/2 \end{cases} \qquad \phi^Y(t) = \begin{cases} 0 & t < -\delta^{n+2}/2 \text{ or } t \geq \delta^{n+2}/2 \\ 1 & -\delta^{n+2}/2 \leq t < \delta^{n+2}/2 \end{cases}$$

If $Z = g_c(L^X, L^Y)$ where L^X is not a permutation of $\bar{L}^X$ and L^Y is not a permutation of $\bar{L}^Y$, then $\phi^X\left(u^T Z - u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_n^T\right) = 0$ and Z is unchanged. If on the other hand L^X *is* a permutation of $\bar{L}^X$ and L^Y *is* a permutation of $\bar{L}^Y$, with $\bar{L}_{:,j}^X = L_{:,i}^X$, then $\phi^X\left(u^T Z^X - u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_n^T\right) = (e^{(i)})^T$ and $g_c^X(L^X, L^Y)$ is mapped to $(A_L^X)_{:,i}$. Thus, this layer maps $g_c^X(L^X, L^Y)_{:,j}$ to the correct output for the specific inputs $\bar{L}^X, \bar{L}^Y$ (or permutations thereof), and does not affect any other inputs. By stacking $\mathcal{O}\left((n+m)(1/\delta)^{d(n+m)}/(n!m!)\right)$ of these layers together, we achieve the correct output for any possible inputs L^X, L^Y.