Supplement to “Transformer with a Mixture of Gaussian Keys”

In this supplementary material, we provide experimental details and additional experiments of Transformer-MGK and Transformer-MLK.

A ADDITIONAL EXPERIMENTS

A.1 EXPERIMENT DETAILS

In this section, we provide model and training details for experiments in Section 3. All our experiments are conducted on a server with 4 NVIDIA A100 GPUs.

A.1.1 LONG RANGE ARENA BENCHMARK

Datasets and metrics We consider the following tasks in the LRA benchmark: Listops (Nangia & Bowman, 2018), byte-level IMDb reviews text classification (Maas et al., 2011), and byte-level document retrieval (Radev et al., 2013). These tasks involve long sequences of length $2K$, $4K$, and $4K$, respectively. We follow the setup/evaluation protocol in (Tay et al., 2021) and report the test accuracy for individual task and the average result across all tasks.

Models and baselines We use the softmax transformer (Vaswani et al., 2017) and linear transformer (Katharopoulos et al., 2020) as our baselines. All models have 2 layers, 64 embedding dimension, and 128 hidden dimension. The number of heads in each layer are set to 1, 2, 4, and 8. For Transformer-MGK/MLKs and their shifted versions, we share π_{jr} for all position j and learn it for each head. The initial value for each π_{jr} is set to 0.5. For Transformer-sMGK, we learn $\mathbf{b_r}$ and initialize its elements from a standard normal distribution. Each σ_{jr} is a constant with value $\sqrt{D}$ where D is the dimension of each head, which is the same as in the baselines models

Details about the Long Range Arena (LRA) benchmarks can be found in the original paper (Tay et al., 2021). Our implementation is based on the public code by (Xiong et al., 2021), and we follow their training procedures. The training setting and additional baseline model details are provided in the configuration file used in (Xiong et al., 2021) and available at https://github.com/mlpen/Nystromformer/blob/main/LRA/code.

A.1.2 LANGUAGE MODELING ON WIKITEXT-103

Datasets and metrics WikiText-103 consists of articles from Wikipedia and is a dataset with long contextual dependencies. The training set is made up of about $28K$ articles containing $103M$ running words; this corresponds to text blocks of about 3600 words. The validation and test sets are composed of $218K$ and $246K$ running words, respectively. Each of them contains 60 articles and about $268K$ words. Our experiment follows the standard setting (Merity et al., 2017; Schlag et al., 2021) and splits the training data into L-word independent long segments. For evaluation, we use a batch size of 1, and go through the text sequence with a sliding window of size L. We consider only the last position for computing perplexity (PPL) except in the first segment, where all positions are evaluated as in (Al-Rfou et al., 2019; Schlag et al., 2021).

Models and baselines Our language modeling implementation is based on the public code https://github.com/IDSIA/lmtool-fwp by (Schlag et al., 2021). We use their small and medium model configurations for models in our experiments. In particular, for small models, we set the key, value, and query dimension to 128, and the training and evaluation context length to 256. For medium models, we set the key, value, and query dimension to 256, and the training and evaluation context length to 384. In both configurations, we set the number of heads to 8, the feed-forward layer dimension to 2048, and the number of layers to 16. For Transformer-MGK/MLK, we use our 4 and 8-head versions to compare with the 8-head baselines. π_{ir}, $\mathbf{b_r}$ and σ_{jr} for this task follow our setings for LRA experiments. Other than those, our language modeling share the same configurations as the baselines.

We train our models for language modeling on 2 A100, 40GB each with a batch size of 96, and each model is trained for 120 epochs. We apply 10% dropout (Hanson, 1990; Srivastava et al., 2014) and use the Adam optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.00025 and 2000 steps for learning rate warm-up.

A.1.3 NEURAL MACHINE TRANSLATION ON IWSLT’14 GERMAN TO ENGLISH

Datasets and metrics The IWSLT14 German-English dataset (Cettolo et al., 2014) contains $153K$ training, $7K$ validation, and $7K$ test TED-talks scripts German-English translated sentences. We follow the same preprocessing steps as in (Ott et al., 2019). We use the BiLingual Evaluation Understudy (BLEU) score as our evaluation metric for this task. All trained models are evaluated using the evaluation protocol in (Ott et al., 2019).

Models and baselines The baseline model we use in this machine translation task is an encoder-decoder transformer with six encoder/decoder layers, four attention heads per layer. The embedding dimension and the hidden size are 512 and 1024, respectively, for both encoder and decoder. These architecture configurations are the same for Transformer-MGK/sMGK except for the number of attention heads per layer, which is reduced by half. We share π_{jr} across all position j and learn it for each head. The initial value for each π_{jr} is set to 0.5. For Transformer-sMGK, we learn $\boldsymbol{b}_r$ and initialize its elements from a standard normal distribution. Each σ_{jr} is a constant with the value $\sqrt{D}$ where D is the dimension of each head, which is the same as in the baselines models. Our experiments follow the settings from (Ott et al., 2019), and our implementation is based on the public code https://github.com/pytorch/fairseq/tree/main/examples/translation.

A.2 MORE EMPIRICAL ANALYSIS OF TRANSFORMER-MGKS/MLKS TRAINED FOR LANGUAGE MODELING

In Table 3 in the main text, we show the improvements in valid and test perplexity of our Transformer-MGKs/MLKs compared with the baseline softmax and linear transformers. In particular, Transformer-MGK/MLKs with the same number of heads as the baselines, e.g. 8 heads, significantly improve the baselines during training while Transformer-MGK/MLKs with 4 head achieve comparable or better performance than the 8-head baselines. In this section, we provide more empirical analysis to shed light on those results. Figure 4 shows the validation perplexity curve of our models versus the softmax and linear transformers.

Figure 5 visualizes the attention matrices from a randomly selected sample for the trained softmax transformer with 8 heads and Transformer-MGKs with 8 heads and 4 heads. These visualizations show that Transformer-MGKs attend to more diverse positions in all heads and layers than the softmax attention baseline. We also compare the rank distributions of these attention matrices computed from 1000 samples at each layer in the model. Figure 6 presents the rank histograms of the 8-head softmax attention and 4-head and 8-head MGK attentions for the 1^{st} and 5^{th} layer. It is clear that attention matrices from the Transformer-MGKs have higher ranks than those in the softmax transformer, which implies that Transformer-MGK can attend to more diverse regions than the softmax transformer without the need of using more attention heads.

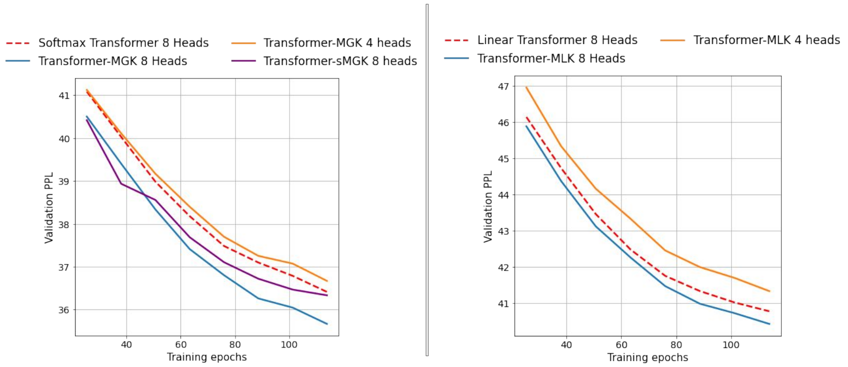

Figure 4: Validation perplexity of Transformer-MGK vs. the softmax transformer (Left) and Transformer-MLK vs. the linear transformer (Right) for language modeling on WikiText-103.

.

A.3 ADDITIONAL TRAINING RESULTS FOR LRA

In this section, we provide additional experimental results on the LRA benchmark. Figure 7 compares the computational cost measured by FLOPs and model complexity in term of the number of parameters of different inference and learning methods for Transformer-MGK. The computational costs of Transformer-MGK/sMGK and Transformer-MGK Hard-E/Soft-E are on par with each other, while Transformer-sMGK uses fewer parameters than the other without trade-off in performance 4 for all tasks. The naming is as explained in Section 3.4 in the main text. In addition, Figure 8 visualizes the attention matrices in the 4-head linear transformer baseline, 4-head Transformer-MLK, and 2-head Transformer-MLK trained on the document retrieval task.

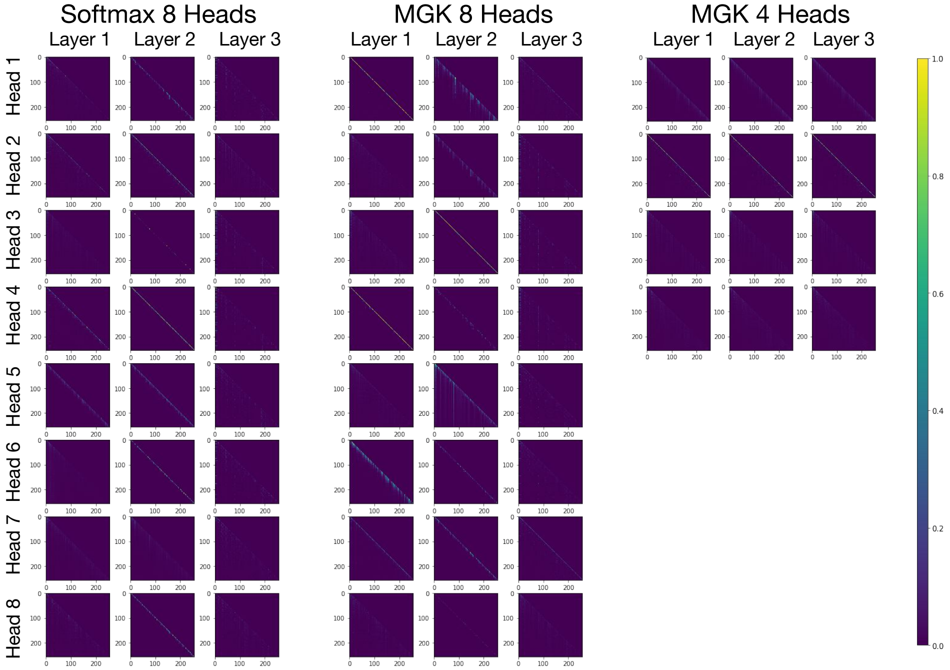

Figure 5: Visualization of attention matrices from 8-head softmax transformer (Left), 8-head Transformer-MGK (Middle), and 4-head Transformer-MGK (Right) trained on WikiText-103 language modeling. Here, we plot the attention matrices for all heads and layers in the models. The sequence length is 256 and the size of each matrix is 256×256.

.

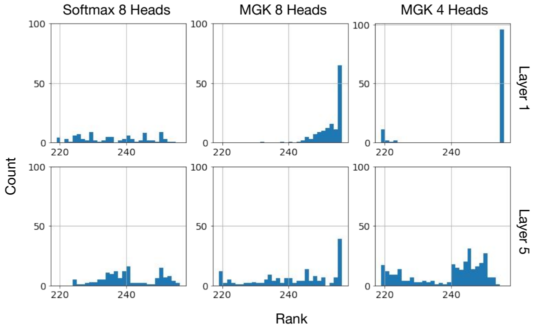

Figure 6: Rank distributions of attention matrices from 8-head softmax transformer (Left), 8-head Transformer-MGK (Middle), and 4-head Transformer-MGK (Right) trained on WikiText-103 language modeling. The rank histograms are computed from 1000 attention matrices at each layer. The attention matrices from Transformer-MGKs have higher ranks than those from softmax transformers. This implies that Transformer-MGKs have more diverse attention patterns, which allows us to reduce the number of heads in Transformer-MGKs.

A.4 ABLATION STUDY ON THE IMPACT OF THE MIXTURE OF KEYS, THE GAUSSIAN DISTANCE, AND THE KEY SHIFTING

In this section, we conduct an ablation study of the Transformer-MGK on the LRA retrieval task to investigate where the performance improvement is from. In particular, we would like to understand the impact of the following factors on the performance of Transformer-MGK: 1) the mixture of keys, 2) the Gaussian distance, and 3) the key shifting. We summarize our empirical results in Table 5 and discuss the impact of 1, 2 and 3 below.

Impact of the Mixture of Keys We apply our mixture of keys (MGK) approach to the softmax transformer using the dot product between queries and keys instead of the Gaussian distance as in our paper. We name this model Softmax MGK. We compare the Softmax MGK that has 1 head (Sofmax MGK 1 head in Table 5) with the baseline softmax transformers that use 1 and 2 heads (Softmax 1 head and Softmax 2 heads in Table 5). Results in Table 5 show that the Softmax MGK 1 head outperforms both the baseline softmax transformers of 1 and 2 heads. Note that our Softmax MGK

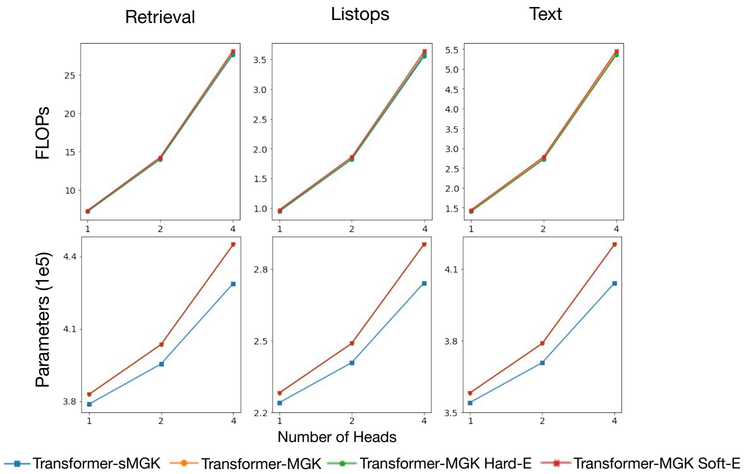

Figure 7: Model complexity (Top) and computational cost (Bottom) of different inference and learning methods for Transformer-MGK trained on the document retrieval task. While computational costs are almost the same, Transformer-sMGK has more advantage in model size, comparing to Transformer-MGK, Transformer-MGK Hard-E, and Soft-E. The naming is as explained in Section 3.4 in the main text.

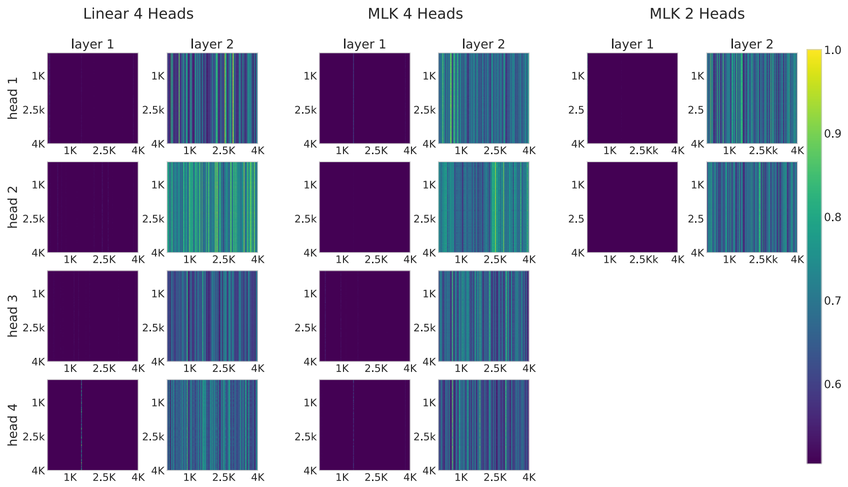

Figure 8: Visualization of attention matrices in the 4-head linear transformer baseline (Left), 4-head Transformer-MLK (Middle), and 2-head Transformer-MLK (Right) trained on the document retrieval task. Here, the sequence length is 4000, and the size of each matrix is 4000×4000.

1 head is more efficient than the baseline of 2 heads in terms of the number of parameters and the number of FLOPs. These results confirm the benefit of using MGK.

Impact of using the Gaussian distance Next, we compare the softmax MGK with the Gaussian MGK. Here the Gaussian MGK is the Transformer-MGK proposed and discussed in our paper, which computes the attention scores using the MGK approach and the Gaussian distance between the queries and keys. Results in Table 5 suggest that the Gaussian MGK 1 head improves over the Softmax MGK 1 head (80.63% vs. 79.23%). This result justifies the advantage of using Gaussian distance over dot product to compute the attention scores.

Impact of key shifting Finally, we apply key shifting to both Softmax MGK and Gaussian MGK (Softmax sMGK and Gaussian sMGK in Table 5). From Table 5, we observe that the Softmax sMGK 1 head and Gaussian sMGK 1 head outperform the Softmax MGK 1 head and Gaussian MGK 1 head, respectively. These results, again, corroborate the benefit of using key shifting.

We also include the result for the Gaussian 1 head model in Table 5. This Gaussian 1 head model is similar to the Softmax 1 head model but uses the Gaussian distance to compute the attention scores. Comparing the results of the Gaussian 1 head model, the Softmax 1 head model, and the Gaussian

Table 5: Ablation study on the impact of the mixture of keys, the Gaussian distance, and the key shifting on the LRA retrieval task. We denote the softmax Transformer by Softmax. All Softmax models (i.e., Softmax 2 heads, Softmax 1 head, Softmax MGK 1 head, and Softmax sMGK 1 head) use dot product to compute the attention scores. All Gaussian models (i.e., Gaussian MGK 1 head, Gaussian sMGK 1 head, and Gaussian 1 head) use Gaussian distance to compute the attention scores. We denote MGK with key shifting by sMGK. Here MGK is used to denote our approach of using a mixture of keys at each timestep.

Method	Accuracy (%)
Softmax 2 heads	79.10
Softmax sMGK 1 head	**79.81**
Softmax MGK 1 head	79.23
Softmax 1 head	77.90
Gaussian sMGK 1 head	**81.23**
Gaussian MGK 1 head	80.63
Gaussian 1 head	80.38

Table 6: Comparing the GPU memory footprint and computational time overhead (seconds/iteration) between our 4-head Transformer-MGKs/MLKs and the 8-head softmax/linear transformer baselines trained on the LRA retrieval task at test time. Our Transformer-MGKs/MLKs save much more memory and have significantly smaller wall-clock time compared to the baselines. Here we use a batch size of 32 for each iteration.

Method	Memory (Gb)	Time overhead (seconds/iteration)
Softmax 8 heads	58.00	0.357
Transformer-MGK 4 heads	53.58	0.278
Transformer-sMGK 4 heads	**45.50**	**0.227**
Linear 8 heads	3.38	0.055
Transformer-MLK 4 heads	2.90	0.043
Transformer-sMLK 4 heads	**2.83**	**0.042**

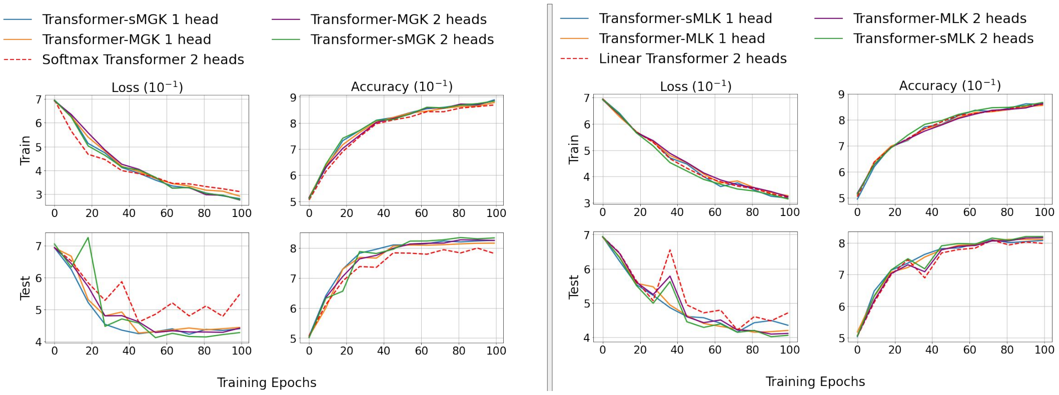

Figure 9: Training and test loss/accuracy of Transformer-MGK vs. softmax transformer (Left) and Transformer-MLK vs. linear Transformer (Right) on the retrieval task, which has the longest average sequence-length and attention span among the LRA tasks (Tay et al., 2021). In training, we apply early stopping to avoid overfitting. That explains why the training loss and accuracy curves stop early. The test loss/accuracy curves already converge. The impressive performance of Transformer-MGK/MLK on this challenging task validates the capability of our models to capture long-range dependencies via learning a diversity of attention patterns.

MGK 1 head model reported in Table 5 further confirms the advantage of using MGK and Gaussian distance.

A.5 TIME OVERHEAD AND MEMORY FOOTPRINT ANALYSIS

Table 6 compares the GPU memory footprint and computational time overhead (seconds/iteration) of our 4-head Transformer-MGKs/MLKs with those of the 8-head softmax/linear transformer baselines at test time. All models are trained on the LRA retrieval task, and we use a batch size of 32 for each iteration. Our MGK/MLK models save memory and reduce wall-clock time significantly compared to the baselines. Using key shifting in our models, i.e. Transformer-sMGKs/sMLKs, helps improve the efficiency further.

A.6 LEARNING CURVES SHOWING CONVERGENCE

In this section, we replot Figure 1 and 4 to show the convergence of our trainings. In Figure 1, the results do not seem to converge since we plot the loss and the accuracy in log-scale. In Figure 4, the

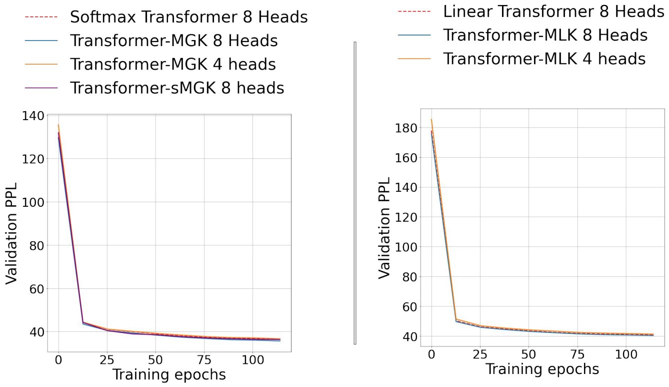

Figure 10: Validation perplexity of the Transformer-MGK vs. the softmax transformer (Left) and the Transformer-MLK vs. the linear transformer (Right) for language modeling on WikiText-103. Training converges on this task after 500000 iterations, equivalent to 115 epochs .

.

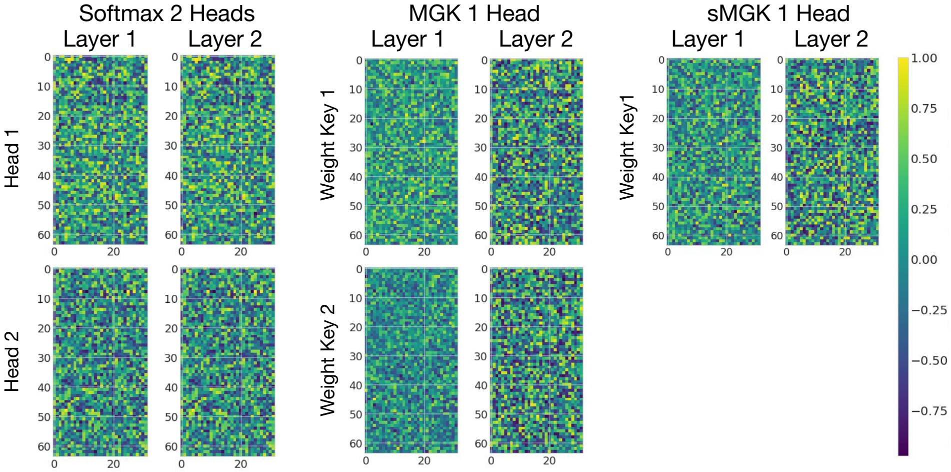

Figure 11: Weight matrices $\mathbf{W}_K$ for computing the keys, for all heads and layers, in the 2-head softmax transformer baseline (Left), the 1-head Transformer-MGK with 2 keys (Middle), and the 1-head Transformer-sMGK with 2 keys (Right) trained on the LRA retrieval task. Here, the dimension of each head $D = 32$ and that of input x_i is $D_x = 64$. Hence, each weight matrix has the shape of $(64, 32)$.

Table 7: The learned mixing coefficient π_{jr} of all heads and layers in the 1-head Transformer-MGKs trained on the LRA retrieval task. Here we use the same π_{j1}, π_{j2} for all time step $j = 1, \ldots, N$.

Method	Layer 1		Layer 2	
	π_{j1}	π_{j2}	π_{j1}	π_{j2}
Transformer-sMGK 1 heads	0.488	0.512	0.500	0.500
Transformer-MGK 1 heads	0.502	0.498	0.497	0.503

results do not seem to converge since we zoom into the specific range on the y and x-axes. Figure 9 and 10 are the replotted versions of Figure 1 and 4, respectively. In Figure 9, the training loss/accuracy curves stop early because we use early stopping to avoid overfitting. The test loss/accuracy curves in this figure already converge.

A.7 WEIGHT MATRICES OF THE KEYS, KEYS AND MIXING COEFFICIENT

In this section, we analyze the learned π_{jr}, $\boldsymbol{k}_{jr}$, and $\mathbf{W}_{K_r}$, $j = 1, \ldots, N$ and $r = 1, \ldots, M$ in the Transformer-MGK trained on the LRA retrieval task. In all of our experiments, we set $M = 2$. In Figure 11 and 12, we visualize the weight matrices $\mathbf{W}_K$ that computes the keys and the keys $\mathbf{K}$, respectively, for all heads and layers in the 2-head softmax transformer baseline, the 1-head Transformer-MGK with 2 keys, and the 1-head Transformer-sMGK with 2 keys trained on the LRA retrieval task. Note that for the keys, we only plot the first 100 tokens. Also, Table 7 summarizes the

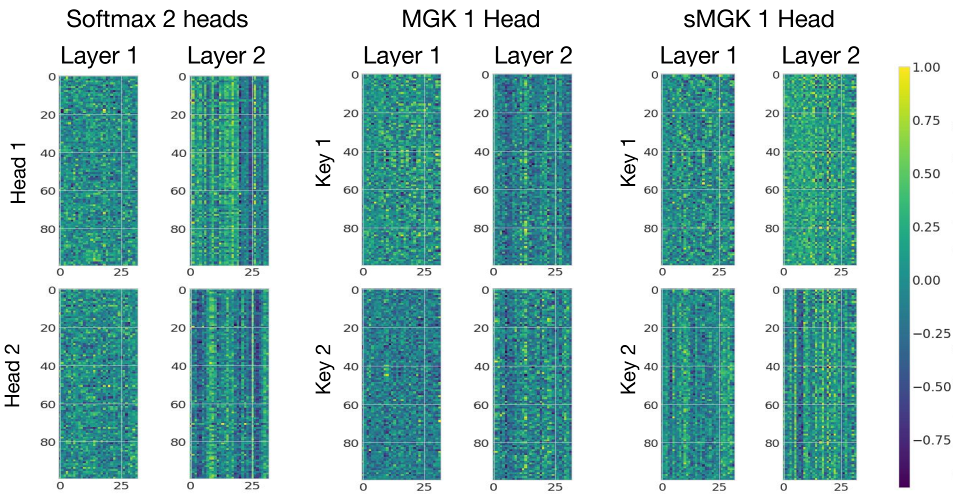

Figure 12: Key embeddings $\mathbf{K}$ for all heads and layers of the 2-head softmax transformer baseline (Left), the 1-head Transformer-MGK with 2 keys (Middle), and the 1-head Transformer-sMGK with 2 keys (Right) trained on the LRA retrieval task. Here the dimension D of each head is 32, and we plot the key embeddings of the first 100 tokens in a randomly chosen sequence. Hence, each key matrix has the shape of $(100, 32)$.

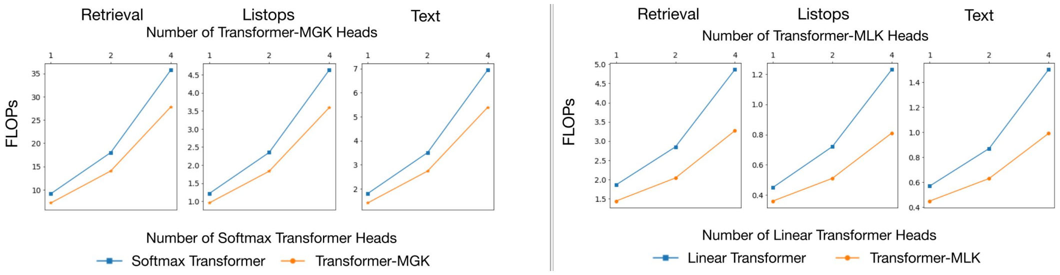

Figure 13: Computational cost (FLOPs) for each training iteration of Transformer-MGK vs. the baseline softmax transformer (Left) and Transformer-MLK vs. the baseline linear transformer (Right) on the LRA retrieval task. The efficiency advantage of Transformer-MGK/MLK over the baselines grows with the number of heads. Here, the batch size of each training iteration is 32.

learned mixing coefficient π_{jr} of all heads and layers in the 1-head Transformer-MGK trained on the same retrieval task. Here we use the same π_{j1}, π_{j2} for all time step $j = 1, \dots, N$.

A.8 Additional Computational Complexity (FLOPs) Analysis at Training Time

Figure 13 demonstrates the computational cost (FLOPs) for each training iteration of the Transformer-MGK vs. the baseline softmax transformer (Left) and the Transformer-MLK vs. the baseline linear transformer (Right) on the LRA retrieval task. The efficiency advantage of Transformer-MGK/MLK over the baselines grows with the number of heads.

Figure 14 shows the computational cost per training iteration (measured in FLOPs) of different inference and learning methods for Transformer-MGK trained on the document retrieval task. Transformer-sMGK, Transformer-MGK, Transformer-MGK Hard-E, and Soft-E have similar computational costs.

A.9 Scaling to 12-Head Baseline Models for the Retrieval task.

To further study the scalability of our model, in this section, we investigate the performance of our 6-head Transformer-MGKs/MLKs in comparison with the 12-head baseline softmax/linear transformers on the retrieval task. Table 8 indicates that our 6-head Transform-MGKs/MLKs significantly outperform 12-head softmax/linear transformers, respectively. Moreover, comparing these results to those in Table 1 and Table 2, although the 12-head softmax/linear transformers improve over the 8-head ones, their accuracies are still worse than or only equivalent to those of our 4-head and even 2-head Transformer-MGK/MLK models.

A.10 Neural Machine Translation on IWSLT'14 German to English

Table 9 shows that the 2-head Transformer-MGK/sMGK models have comparable or better BLEU scores than the 4-head softmax transformer baseline.

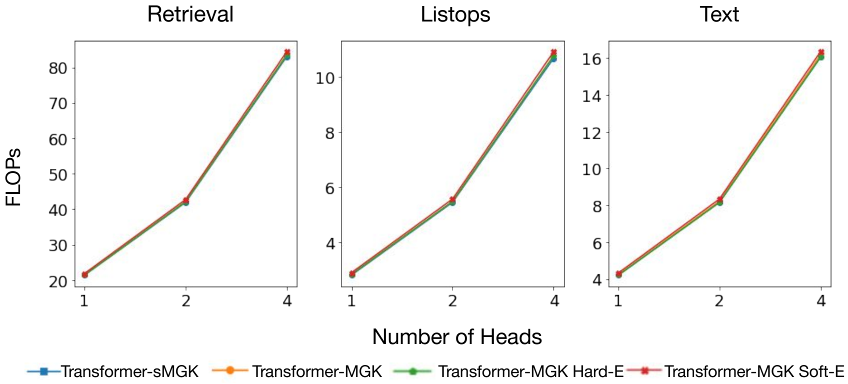

Figure 14: Computational cost (measured in FLOPs) per training iteration of different inference and learning methods for Transformer-MGK trained on the LRA retrieval task. Computational costs are almost the same for Transformer-sMGK, Transformer-MGK, Transformer-MGK Hard-E, and Soft-E. The naming is as explained in Section 3.4 in the main text.

Table 8: Test Accuracy (%) of 6-head Transformer-MGKs/MLKs compared with the baseline 12-head softmax and linear transformers on the retrieval task. Our 6-head Transformer-MGKs/MLKs significantly outperform softmax and linear transformers, respectively, while being more efficient in terms of computational cost, model size, and memory usage.

Method	Accuracy (%)
Softmax 12 heads	82.18
Transformer sMGK 6 head	**83.31**
Transformer MGK 6 head	83.05
Linear sMGK 12 head	81.97
Transformer sMLK 6 head	**82.80**
Transformer MLK 6 head	82.11

Table 9: Machine translation BLEU scores of 2-head Transformer-MGKs on the IWSLT14 De-En dataset is better than or equivalent to that of the 4-head baseline.

Method	BLEU score
Softmax 4 heads	34.42
Transformer sMGK 2 head	**34.69**
Transformer MGK 2 head	34.34

A.11 COMPARISON TO MULTI-QUERY ATTENTION

In this section, we compare our MGK approach to the multi-query attention (Shazeer, 2019). The multi-query attention shares the same set of keys and values at different heads to reduce the memory-bandwidth cost during incremental inference, which allows faster inference since the size of the reloaded ”keys” and ”values” tensors are significantly reduced. On another hand, our Transformer-MGK models the key at each head as a Gaussian mixture model, which leads to the use of multiple keys at each head and allows us to decrease the number of attention heads. This helps reduce the computations and parameters needed to calculate additional queries and values. If using key shifting (see option (B) in paragraph Design Options for Keys in Section 2.3), the MGK approach also helps reduce the computations and parameters needed to calculate additional keys. The advantages of Transformer-MGK hold in both training and inference, including incremental inference. We have provided a detailed analysis on the computational complexity and the number of parameters of the Transformer-MGK in comparison with the corresponding softmax transformer in Appendix B. Combining the multi-query attention and our MGK approach is interesting since each method has its own advantage and they are complementary to each other. In particular, we can let the transformer model share the same set of values and mixtures of keys at different heads. This approach can potentially have the advantages of both multi-query attention and MGK.

B AN ANALYSIS ON THE COMPUTATIONAL COMPLEXITY AND THE NUMBER OF PARAMETERS IN TRANSFORMER-MGK AND THE SOFTMAX TRANSFORMER

In this section, we compare the computational complexity and the number of parameters in transformer-MGK with M keys at each timestep and H/M heads to the baseline softmax transformer that has 1 key at each timestep and H heads. Here we choose M such that H is a multiple of M and use keys design option (A) that make the key $\boldsymbol{k}_{jr}$ a linear projection of the input $\boldsymbol{x}_j$, i.e. $\boldsymbol{k}_{jr} = \boldsymbol{x}_j \mathbf{W}_{K_r}^{\top}$, where $\boldsymbol{x}_j \in \mathbb{R}^{1\times D_x}$, $\mathbf{W}_{K_r} \in \mathbb{R}^{D\times D_x}$ and $r = 1, 2, \dots, M$ (see Design Options for Keys in Section 2.3 for more details). To simplify notation and without loss of generalization, we let $M = 2$ as in our experiments and assume that $D_v = D$, i.e., the values have the same feature dimension as the queries and the keys. To simplify the computation, we also do not take the softmax operator into account since this operator yields similar costs when applied in Transformer-MGK and softmax transformer.

B.1 COMPUTATIONAL COMPLEXITY

(i) Softmax H-head attention: The number of computations in a softmax H-head attention is $N^2H(4D-1) + NHD(6D_x + 2HD - 5)$.

Explanation: To calculate the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$ in Step 1 in Section 1.1 at each head, we need $3NDD_x$ multiplications and $3ND(D_x-1)$ additions. In total, these need $3ND(2D_x-1)$ computations. Next, to compute the product $\mathbf{Q}\mathbf{K}^{\top}$ in Eqn. (1), we need N^2D multiplications and $N^2(D-1)$ additions. Similarly, the product $\mathbf{AV}$ requires N^2D multiplications and $N(N-1)D$ additions. In total, computing the output sequence $\mathbf{H}$ in Eqn. (1) at each head requires $3ND(2D_x-1)+N^2D+N^2(D-1)+N^2D+N(N-1)D = N^2(4D-1)+ND(6D_x-4)$ computations. The total computation for all H heads is then

$$\begin{aligned} &H(N^2(4D-1) + ND(6D_x-4)) + NHD(2HD-1) \\ &= N^2H(4D-1) + NHD(6D_x + 2HD - 5), \end{aligned}$$

where the extra $NHD(2HD-1)$ is from the linear projection by $\mathbf{W}^O$.

(ii) Mixture of 2 Gaussian keys attention with $H/2$-head: The number of computations in a mixture of 2 Gaussian keys attention with $H/2$-head is $N^2H(3D-0.5) + NHD(4D_x + HD - 4)$.

Explanation: Similar to the above derivation, in a Mixture of M Gaussian keys attention, to compute the output sequence $\mathbf{H}$ we need $N^2((2M+2)D-1) + ND((M+2)(2D_x-1)-1)$ computations. Note that, to compute the Gaussian distances between the queries q_i and the keys k_j as in the mixture of M Gaussian keys attention and to compute their dot product as in the softmax attention, we need the similar number of computations. Therefore, the total computation for all H/M heads is then

$$\begin{aligned} &(H/M)(N^2((2M+2)D-1) + ND((M+2)(2D_x-1)-1)) + NHD(2(H/M)D-1) \\ &= N^2H\left(\frac{2(M+1)D-1}{M}\right) + NHD\left(\frac{2(M+2)}{M}D_x + \frac{2}{M}HD - \frac{3M+2}{M}\right). \end{aligned}$$

Let $M = 2$, then the total computation of the mixture of 2 Gaussian keys attention is given by $N^2H(3D-0.5) + NHD(4D_x + HD - 4)$.

Soft-max H-head attention versus mixture of 2 Gaussian keys attention with H/2-head: Given the results in (i) and (ii), when compared to the baseline softmax H-head attention, our mixture of 2 Gaussian keys attention with $H/2$-head saves

$$N^2H(D-0.5) + NHD(2D_x + HD - 1)$$

computations. When N is large, this difference is significant. In conclusion, the mixture of 2 Gaussian keys attention with H/2-heads has cheaper computational complexity than that of soft-max H-head attention.

B.2 THE NUMBER OF PARAMETERS

(iii) Softmax H-head attention: The number of parameters in a softmax H-head attention is $3HDD_x + (HD)^2$.

Explanation: $3HDD_x$ is from the linear projects to calculate the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$ in Step 1 in Section 1.1. $(HD)^2$ is from the linear project to compute the final output as in Eqn. (3).

(iv) Mixture of 2 Gaussian Keys attention with $H/2$-head: The number of parameters in a Mixture of 2 Gaussian Keys attention with $H/2$-head is $2HDD_x + 0.5(HD)^2 + H$.

Explanation: The linear projections to calculate $\mathbf{Q}$ and $\mathbf{V}$ contribute $HDD_x/2$ parameters each. The linear projection to calculate $\mathbf{K}$ contributes HDD_x. The linear project to compute the final output has dimension $(H/2)D \times HD$, so it contributes $0.5(HD)^2$ parameters. H more parameters is from the prior π_{jr}. These priors contribute 2 parameters at each head since we make $\{\pi_{j1}, \pi_{j2}\}$ for all $j = 1, \ldots, N$ the same.

Soft-max H-head attention versus mixture of 2 Gaussian keys attention with H/2-head: Given the results in (iii) and (iv), when compared to the baseline softmax H-head attention, our mixture of 2 Gaussian keys attention with $H/2$-head saves $HDD_x + 0.5(HD)^2 - H$ parameters. When H and D are large, this saving is significant.

C PROOFS OF MAIN RESULTS

In this appendix, we provide proofs for the main results in the paper.

C.1 PROOF OF THEOREM 1

To ease the presentation of the proof, for any probability distribution G, we denote

$$p_G(x) := \int f(x-\theta)dG(\theta) = \int \phi(x|\theta, \sigma^2\mathbf{I})dG(\theta),$$

for all $x \in \mathbb{R}^d$ where $f(x) = \frac{1}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$ for given $\sigma > 0$. It means that p_G is the convolution of f and the probability distribution G. Since the space of Gaussian mixtures is dense in the space of continuous probability measures (Bacharoglou, 2010), it indicates that there exists probability distribution G_1 such that

$$\sup_{x \in \mathbb{R}^d} |p(x) - p_{G_1}(x)| \leq \frac{\epsilon}{2}. \tag{14}$$

Our next step is to prove that there exists a probability measure G_2 with at most K supports where $K \leq (C\log(1/\epsilon))^d$ for some universal constant C such that

$$\sup_{x \in \mathbb{R}^d} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}. \tag{15}$$

Indeed, from Lemma A.1 in (Ghosal & van der Vaart, 2001), for any $k \geq 1$ there exists a probability distribution G_2 with at most $(2k-2)^d$ supports such that

$$\int \theta^{\alpha} d(G_1 - G_2)(\theta) = 0, \tag{16}$$

for any $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d) \in \mathbb{N}^d$ such that $0 \leq |\alpha| = \sum_{j=1}^d \alpha_j \leq 2k-2$, Here, $\theta^{\alpha} = \prod_{j=1}^d \theta_j^{\alpha_j}$.

Now, for any $M \geq 2a\sqrt{d}$, we have $\|x - \theta\| \geq \|x\| - \|\theta\| > M - a\sqrt{d} > M/2$ as long as $\|x\| > M$ and $\theta \in [-a, a]^d$. It indicates that

$$\begin{aligned}
\sup_{\|x\|>M} |p_{G_1}(x) - p_{G_2}(x)| &= \sup_{\|x\|>M} \left|\int f(x-\theta)d(G_1 - G_2)(\theta)\right| \\
&\leq \sup_{\|x\|>M} \int \frac{1}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{\|x-\theta\|^2}{2\sigma^2}\right)d(G_1 + G_2)(\theta) \\
&\leq \frac{2}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{M^2}{8\sigma^2}\right).
\end{aligned} \tag{17}$$

On the other hand, for any $k \geq 1$ we also have that

$$
\begin{aligned}
\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| &= \sup_{\|x\| \leq M} \left| \int f(x-\theta) d(G_1 - G_2)(\theta) \right| \\
&\leq \sup_{\|x\| \leq M} \left| \int \left(f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right) d(G_1 - G_2)(\theta) \right| \\
&+ \sup_{\|x\| \leq M} \left| \int \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} d(G_1 - G_2)(\theta) \right| \\
&= \sup_{\|x\| \leq M} \left| \int \left(f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right) d(G_1 - G_2)(\theta) \right|,
\end{aligned} \tag{18}
$$

where the final equality is stems from

$$
\int \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} d(G_1 - G_2)(\theta) = 0,
$$

which is due to Eqn. (16).

To further bound the right-hand-side (RHS) of Eqn. (18), we use the following inequality:

$$
\left| \exp(y) - \sum_{j=0}^{k-1} (y)^j / j! \right| \leq |y|^k / k!
$$

for any $y \in \mathbb{R}$. Since $k! \geq (k/e)^k$ for any $k \geq 1$, the above bound can be rewritten as

$$
\left| \exp(y) - \sum_{j=0}^{k-1} (y)^j / j! \right| \leq \frac{|ye|^k}{k^k}. \tag{19}
$$

Further simplification of Eqn. (18) leads to

$$
\begin{aligned}
\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| &\leq \sup_{\|x\| \leq M} \int \left| f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right| d(G_1 + G_2)(\theta) \\
&\leq 2 \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \left| f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right| \\
&= \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \frac{2}{(\sqrt{2\pi}\sigma)^d} \left| \exp\left(-\frac{\|x-\theta\|^2}{2\sigma^2} \right) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{\sigma^{2j} j!} \right| \\
&\leq \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \frac{e^k \|x-\theta\|^{2k}}{\sigma^{2k} (2k)^k},
\end{aligned}
$$

where the final inequality is based on an application of inequality (19) with $y = -\|x-\theta\|^2/(2\sigma^2)$. For $\|x\| \leq M$ and $\theta \in [-a,a]^d$, we have $\|x-\theta\| \leq \|x\| + \|\theta\| \leq M + a\sqrt{d}$. Therefore, we further have

$$
\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \frac{e^k \|x-\theta\|^{2k}}{\sigma^{2k} (2k)^k} \leq \frac{e^k (M + a\sqrt{d})^{2k}}{\sigma^{2k} (2k)^k}.
$$

When $M \geq 2a\sqrt{d}$, we have $M + a\sqrt{d} \leq \frac{3M}{2}$ and the above bound leads to

$$
\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{(9e)^k M^{2k}}{(8\sigma^2 k)^k}. \tag{20}
$$

By choosing $M^2 = 8\sigma^2 \log(1/\epsilon')$ for some $\epsilon' > 0$, the bounds in Eqns. (17) and (20) become

$$\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{2}{(\sqrt{2\pi}\sigma)^d}\epsilon',$$
$$\sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{(9e)^k (\log(1/\epsilon'))^k}{k^k}. \tag{21}$$

As long as we choose $k = 9e^2 \log(1/\epsilon')$ and $\epsilon' \leq 1$, we have

$$\sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| \leq e^{-k} = e^{-9e^2 \log(1/\epsilon')} = (\epsilon')^{9e^2} \leq \epsilon'. \tag{22}$$

By choosing $\epsilon' = \frac{\epsilon}{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}$, the results from Eqns. (21) and (22) indicate that

$$\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}, \quad \text{and} \quad \sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}.$$

Therefore, if we choose $M = 8\sigma^2 \log\left(\frac{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}{\epsilon}\right)$ and $k = 9e^2 \log\left(\frac{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}{\epsilon}\right)$, we have

$$\sup_{x \in \mathbb{R}^d} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}.$$

It indicates that we obtain the conclusion of claim (15) by choosing $K = (2k-2)^d \leq \left(18e^2 \log\left(\frac{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}{\epsilon}\right)\right)^d$. Combining the results from Eqns. (14) and (15), we have

$$\sup_{x \in \mathbb{R}^d} |p(x) - p_{G_2}(x)| \leq \sup_{x \in \mathbb{R}^d} |p(x) - p_{G_1}(x)| + \sup_{x \in \mathbb{R}^d} |p_{G_1}(x) - p_{G_2}(x)| \leq \epsilon.$$

As a consequence, we obtain the conclusion of the theorem.