Supplement to "A Primal-Dual Framework for Transformers and Neural Networks"

A Additional details on the experiments

This section provides datasets, models, and training details for experiments in Section 3. As mentioned in Section 3, for Attention-BN models, recentering queries and keys alone is sufficient for accuracy improvement, and we weight the mean $\boldsymbol{\mu}$ in Eqn 22 with a constant β. Hence Eqn 22 is simplified to:

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \operatorname{softmax}\left((\boldsymbol{q}_i - \beta\boldsymbol{\mu})^\top(\boldsymbol{k}_j - \beta\boldsymbol{\mu})/\sqrt{D}\right)\boldsymbol{v}_j. \tag{26}$$

In our experiments, we consider the constant β in Attention-BN/BN+SH and the different downsampling scales in Attention-SH/SH+BN as hyper-parameters to finetune. All of our experiments are conducted on a server with 4 NVIDIA A100 GPUs.

A.1 UEA time series classification

Datasets and metrics The benchmark (Bagnall et al., 2018) consists of 30 datasets. Following (Wu et al., 2022), we choose 10 datasets, which vary in input sequence lengths, the number of classes, and dimensionality, to evaluate our models on temporal sequences. We report the test accuracy as evaluation for the benchmark.

Models and baselines The experiment setups and configurations for the softmax/linear baseline and our models are the same as in (Wu et al., 2022) [1] (for the PEMS-SF, SelfRegulationSCP2, UWaveGestureLibrary datasets) and (Zerveas et al., 2021) [2] (for the other tasks). In all models, the number of heads is 8, whereas the model dimension and number of transformer layers are varied. For Attention-SH/SH+BN, we downsample keys and values by the factor of 2, after every two successive heads.

A.2 Long range arena benchmark

Datasets and metrics We adopt the tasks: Listops (Nangia & Bowman, 2018), byte-level IMDb reviews text classification (Maas et al., 2011), byte-level document retrieval (Radev et al., 2013), CIFAR-10 image classification (Krizhevsky et al., 2009) and the Pathfinder challenge (Linsley et al., 2018) in the LRA benchmark for our experiments. They consist of long sequences of length $2K$, $4K$, $4K$, $1K$, and $1K$ respectively. The evaluation protocol and metric are the same as in (Tay et al., 2021).

Models and baselines All our models and softmax/linear baselines follow the same architecture and configuration as in (Zhu et al., 2021)[3]. Each model consists of two layers and 64 embedding dimensions. While one head at each layer remains intact, the keys and values of the other heads are halved in our Attention-SH/SH+BN experiments.

A.3 Image Classification on Imagenet

Datasets and metrics The ImageNet dataset (Deng et al., 2009; Russakovsky et al., 2015) consists of $1.28M$ training images and $50K$ validation images. The task is to classify 1000 categories. Top-1 and top-5 accuracies are reported.

Models and baselines Our baseline is DeiT-tiny model (Touvron et al., 2021) with 12 transformer layers, 4 attention heads per layer, and the model dimension of 192. For model setting and setting and configuration, we follow (Touvron et al., 2021)[4]. The downsampling scales in Attention-SH/BN+SH models are $[1, 1, 2, 4]$ for 4 heads at each layer, respectively.

---

[1] Implementation available at https://github.com/thuml/Flowformer.

[2] Implementation available at https://github.com/gzerveas/mvts_transformer.

[3] Implementation available at https://github.com/NVIDIA/transformer-ls.

[4] Implementation available at https://github.com/facebookresearch/deit.

Table 5: Test Accuracy (%) of the Linear Attention-BN/SH/BN+SH vs. the baseline Linear Attention (Katharopoulos et al., 2020) on the UEA Time Series Classification Archive benchmark (Bagnall et al., 2018). Our proposed attentions outperform the baseline.

Dataset/Model	*Baseline Linear*	Linear Attention-BN	Linear Attention-SH	Linear Attention-BN+SH
EthanolConcentration	33.84 ± 0.66	**34.98 ± 0.74**	34.76 ± 0.69	34.35 ± 0.70
FaceDetection	69.17 ± 0.32	69.22 ± 0.17	**69.38 ± 0.17**	69.12 ± 0.19
HandWriting	32.87 ± 0.27	32.86 ± 0.49	32.82 ± 0.12	**32.98 ± 0.36**
HeartBeat	75.61 ± 0.73	75.78 ± 0.71	74.96 ± 0.62	**75.94 ± 0.68**
JapaneseVowels	99.37 ± 0.16	**99.60 ± 0.19**	99.28 ± 0.41	99.33 ± 0.19
PEMS-SF	83.43 ± 0.88	85.74 ± 0.67	**86.51 ± 0.88**	84.97 ± 0.76
SelfRegulationSCP1	90.90 ± 0.40	91.81 ± 0.69	90.76 ± 0.59	**91.92 ± 0.60**
SelfRegulationSCP2	55.18 ± 0.89	**56.11 ± 0.94**	54.44 ± 0.88	55.74 ± 0.92
SpokenArabicDigits	**99.07 ± 0.10**	99.01 ± 0.07	99.03 ± 0.18	98.91 ± 0.17
UWaveGestureLibrary	85.63 ± 0.81	**86.04 ± 0.86**	84.89 ± 1.00	85.78 ± 0.75
Average Accuracy	72.51 ± 0.34	**73.12 ± 0.20**	72.68 ± 0.40	72.90 ± 0.23

Table 6: Top-1 and top-5 accuracy (%) of the Attention-Conv2D Deit vs. the baseline Deit with the softmax attention on the ImageNet image classification task. The Attention-Conv2D Deit significantly outperforms the baseline in both top-1 and top-5 accuracy.

Metric/Model	*Baseline Softmax Deit*	Attention-Conv2D Deit
Top-1 Acc (%)	72.23 ± 0.23	**73.18 ± 0.24**
Top-5 Acc (%)	91.13 ± 0.15	**91.52 ± 0.13**

Table 7: Test Accuracy (%) of the Attention-Conv1D vs. the baseline softmax attention on 5 tasks of the LRA benchmark (Tay et al., 2021). Our models outperform the softmax baseline.

Dataset/Model	*Baseline Softmax*	Attention-Conv1D
ListOps	36.76 ± 0.42	**37.20 ± 0.05**
Text	64.90 ± 0.07	**64.92 ± 0.44**
Retrieval	79.68 ± 0.52	**80.75 ± 0.15**
Image	**39.23 ± 1.35**	39.18 ± 0.59
Pathfinder	72.72 ± 0.75	**73.01 ± 0.24**
Average Accuracy	58.66 ± 0.26	**59.01 ± 0.20**

B Additional Experimental Results

B.1 UEA Time Series Classification using the Linear Attention-BN/SH/BN+SH

Table 5 summarizes the comparison between the Linear Attention-BN/SH/BN+SH and the baseline Linear Attention on the UEA Time Series Classification task. The Linear Attention-BN/SH/BN+SH achieve better accuracy than the Linear Attention baseline while being more efficient.

B.2 Convolution Attention

Table 6 demonstrates the advantage of Attention-Conv2D (Def. 3, Section G) over softmax Deit on the ImageNet image classification task. Furthemore, as shown in Table 7, the Attention-Conv1D (Def. 4, Section G) outperforms the baseline softmax attention on 5 tasks of the LRA benchmark (Tay et al., 2021).

B.3 Additional experiments on the UEA Timeseries Classification benchmark and the UCR Time Series Regression Archive

In this section, we further demonstrate the advantage of our Attention-BN/SH/BN+SH on additional 15 tasks in the UEA Time Series Classification benchmark and on 6 tasks in the UCR Time Series Regression benchmark. The results in Table 8 and 9 show that our Attention-BN and Attention-SH+BN outperform the baseline softmax transformers significantly on both of these benchmarks, while the attention-SH has comparable performance with the baseline but being more effiicient.

B.4 UEA Time Series Classification using the Sparse Attention-BN/SH/BN+SH

Table 10 summarizes the comparison between the Sparse Attention-BN/SH/BN+SH and the Sparse Attention baseline on a subset of the UEA Time Series Classification benchmark. Our models when combined with Sparse Attention achieve significantly better accuracy than the Sparse Attention baseline while the Sparse Attention-SH/BN+SH are more efficient (See Fig. 3 and Fig. 4 in Appendix C).

Table 8: Root mean square error (RMSE) of the Attention-BN/SH/BN+SH vs. the baseline softmax attention on 6 UCR Time Series Regression tasks (Tan et al., 2020). Smaller RMSE indicates better performance.

Dataset/Model	*Baseline Softmax*	Attention-BN	Attention-SH	Attention-BN+SH
AppliancesEnergy	3.44 ± 0.06	3.38 ± 0.34	3.39 ± 0.02	**3.37 ± 0.23**
BenzeneConcentration	0.91 ± 0.03	**0.89 ± 0.17**	1.00 ± 0.09	0.90 ± 0.08
BeijingPM10	92.31 ± 1.06	**92.00 ± 0.89**	92.82 ± 0.92	92.40 ± 0.85
BeijingPM25	59.73 ± 1.21	59.55 ± 0.92	59.66 ± 0.88	**59.24 ± 1.22**
LiveFuelMoisture	43.08 ± 0.17	**43.01 ± 0.50**	43.65 ± 0.09	43.79 ± 0.49
IEEEPPG	32.12 ± 1.25	**30.69 ± 0.64**	31.38 ± 1.02	30.73 ± 1.20
Average RMSE	38.60 ± 0.67	**38.25 ± 0.30**	38.65 ± 0.27	38.40 ± 0.51

Table 9: Accuracy (%) of the Attention-BN/SH/BN+SH vs. the baseline softmax attention on other 15 UEA Time Series classification tasks (Bagnall et al., 2018).

Dataset/Model	*Baseline Softmax*	Attention-BN	Attention-SH	Attention-BN+SH
ArticularyWordRecognition	97.44 ± 0.42	98.22 ± 0.87	97.22 ± 0.95	**98.44 ± 0.41**
BasicMotions	98.75 ± 1.25	99.38 ± 1.08	99.37 ± 1.06	**99.78 ± 0.51**
Epilepsy	**93.71 ± 1.23**	92.27 ± 0.74	89.13 ± 1.07	92.02 ± 1.02
ERing	95.18 ± 0.52	95.18 ± 0.37	94.72 ± 0.66	**95.46 ± 0.40**
FingerMovements	59.67 ± 0.47	63.00 ± 0.41	61.33 ± 0.70	**63.66 ± 0.64**
Libras	85.00 ± 0.45	**85.37 ± 0.69**	83.88 ± 0.45	85.00 ± 0.78
NATOPS	95.00 ± 0.45	95.37 ± 0.26	**96.29 ± 0.69**	95.74 ± 0.94
RacketSports	87.28 ± 0.82	87.93 ± 0.31	88.16 ± 0.54	**89.03 ± 0.64**
AtrialFibrillation	33.33 ± 2.71	41.67 ± 2.88	35.00 ± 2.89	**41.68 ± 2.80**
Cricket	94.90 ± 0.65	95.37 ± 0.65	93.98 ± 0.65	**96.29 ± 0.65**
StandWalkJump	50.00 ± 2.33	**55.55 ± 2.14**	50.01 ± 2.34	55.00 ± 2.08
HandMovementDirection	63.96 ± 2.30	64.41 ± 2.76	61.71 ± 2.64	**66.66 ± 2.54**
LSST	58.54 ± 0.54	57.05 ± 0.26	**60.34 ± 0.73**	59.91 ± 0.34
DuckDuckGeese	64.50 ± 1.96	65.00 ± 1.73	64.50 ± 1.95	**65.50 ± 1.66**
MotorImagery	58.66 ± 1.25	60.67 ± 1.69	59.00 ± 1.41	**62.00 ± 0.81**
Average Accuracy	75.73 ± 0.51	77.10 ± 0.22	75.61 ± 0.18	**77.74 ± 0.24**

Table 10: Test Accuracy (%) of the Sparse Attention-BN/SH/BN+SH vs. the baseline Sparse Attention (Child et al., 2019) on a subset of the UEA Time Series Classification Archive benchmark (Bagnall et al., 2018). Our proposed attentions outperform the baseline.

Dataset/Model	*Baseline Sparse*	Sparse Attention-BN	Sparse Attention-SH	Sparse Attention-BN+SH
EthanolConcentration	33.33 ± 1.23	33.33 ± 0.78	32.50 ± 0.57	**33.46 ± 0.71**
FaceDetection	68.58 ± 0.95	68.65 ± 0.44	**68.67 ± 0.78**	68.44 ± 0.51
HandWriting	31.08 ± 0.38	31.79 ± 0.44	32.75 ± 0.39	**33.37 ± 0.61**
HeartBeat	74.95 ± 0.81	75.98 ± 0.72	74.96 ± 0.80	**76.09 ± 0.75**
JapaneseVowels	99.45 ± 0.10	**99.54 ± 0.12**	99.18 ± 0.14	99.36 ± 0.34
PEMS-SF	82.08 ± 0.63	83.81 ± 0.47	82.66 ± 0.63	**84.01 ± 0.89**
SelfRegulationSCP1	91.24 ± 0.85	91.69 ± 0.42	91.47 ± 0.84	**91.70 ± 0.16**
SelfRegulationSCP2	55.18 ± 0.69	**58.52 ± 0.71**	55.92 ± 0.94	56.67 ± 0.68
SpokenArabicDigits	99.04 ± 0.06	99.10 ± 0.15	99.06 ± 0.13	**99.15 ± 0.09**
UWaveGestureLibrary	84.90 ± 0.39	85.73 ± 0.38	85.31 ± 0.88	**86.56 ± 0.25**
Average Accuracy	71.98 ± 0.38	72.81 ± 0.15	72.25 ± 0.24	**72.88 ± 0.35**

Table 11: Test Accuracy (%) of the Attention-BN/BN+SH with β is learnable or set as a hyperparameter on the retrieval task (Tay et al., 2021).

Model	Retrieval
Attention-BN (learn β)	80.77 ± 0.23
Attention-BN+SH (learn β)	**81.31 ± 0.25**
Attention-BN (β as a hyperparameter)	81.05 ± 0.08
Attention-BN+SH (β as a hyperparameter)	81.20 ± 0.11

B.5 ATTENTION-BN/BN+SH WITH LEARNABLE β

We experiment with our Attention-BN/BN+SH with learnable β on the retrieval task. Table 11 shows that learning β does not improve much over setting β to be a hyperparameter.

C ADDITIONAL RESULTS ON EFFICIENCY ANALYSIS

This section provides more efficiency analysis on our models.

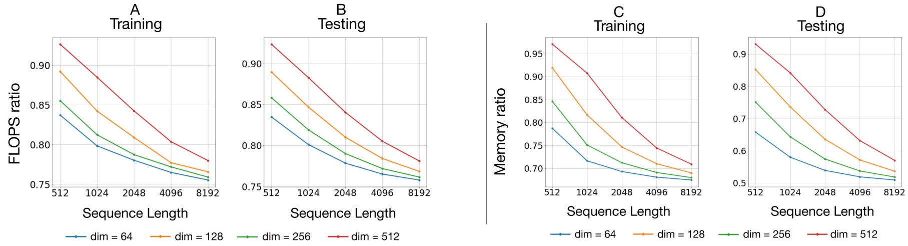

Figure 2: (Left) FLOPS ratios and (Right) memory usage ratios between the Attention-SH and the softmax attention baseline trained on the LRA retrieval task for different model dimensions and sequence lengths.

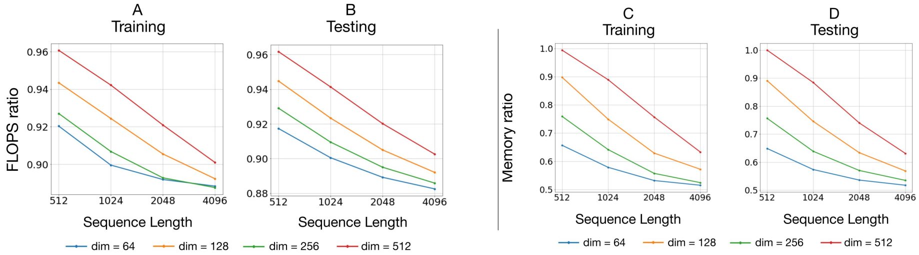

Figure 3: (Left) FLOPS ratios and (Right) memory usage ratios between the Sparse Attention-BN+SH and the Sparse Attention baseline trained on the LRA retrieval task for different model dimensions and sequence lengths. When using our models, the reduction in computation and memory improves with sequence length. When scaling up the model with greater model dimension, our methods remain significantly more efficient than the baseline.

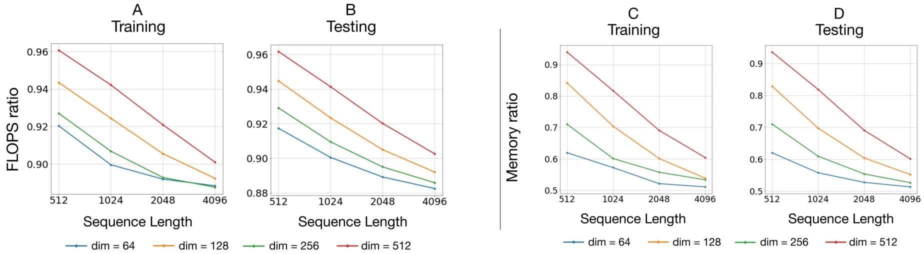

Figure 4: (Left) FLOPS ratios and (Right) memory usage ratios between the Sparse Attention-SH and the Sparse Attention baseline trained on the LRA retrieval task for different model dimensions and sequence lengths. When using our models, the reduction in computation and memory improves with sequence length. When scaling up the model with greater model dimension, our methods remain significantly more efficient than the baseline.

Attention-SH. Fig.2 shows the efficiency benefits of our Attention-SH when trained on the retrieval task. Same as in the case of Attention-SH+BN, the efficiency benefits of our Attention-SH over the baseline Softmax attention grows when N and D increase.

Sparse Attention-SH/BN+SH. Fig.3 and Fig.4 show that the efficiency advantages of our Sparse Attention-BN+SH and Sparse Attention-SH, respectively, increase as the model dimension D and sequence length N grow. All models are trained on the LRA retrieval task. In addition to the efficiency advantage, the Sparse Attention-BN+SH also significantly outperforms the Sparse Attention baseline in terms of accuracy in this task (79.86% vs. 78.20%) while the Sparse Attention-SH achieves a comparable result to the baseline. More accuracy advantages of the Sparse Attention-BN/SH/BN+SH over the Sparse Attention baseline are given in Table 10.

D DERIVING SOFTMAX ATTENTION.

Choosing the appropriate $h(\boldsymbol{x})$ and $\Phi(\boldsymbol{x})$ allows us to derive the popular softmax attention given in Eqn. 1 and 2. In particular, if we choose $h(\boldsymbol{x}) := \sum_j^N \Phi(\boldsymbol{x})^T\Phi(\boldsymbol{k}_j)$, Eqn. 10 becomes

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \frac{\Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{k}_j)}{\sum_{j'}^{N} \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{k}_{j'})} \boldsymbol{v}_j + \boldsymbol{b} = \frac{\sum_{j=1}^{N} \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j'}^{N} \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{k}_{j'})} + \boldsymbol{b}. \tag{27}$$

We then select $\Phi(\boldsymbol{x}) = \left(a_{l_0}^{(0)}, a_1^{(1)}, \ldots, a_{l_1}^{(1)}, \ldots, a_1^{(t)}, \ldots, a_{l_t}^{(t)}, \ldots\right)$ where $l_t = \binom{D+t-1}{t}$ and

$$a_l^{(t)} = \frac{(x_1/\sqrt[4]{D})^{n_1} \ldots (x_D/\sqrt[4]{D})^{n_D}}{\sqrt{n_1! \ldots n_D!}} \mid n_1 + \cdots + n_D = t, \ 1 \leq l \leq l_t. \tag{28}$$

Since

$$\exp\left(\boldsymbol{x}^T \boldsymbol{y}\right) = \sum_{t=0}^{\infty} \frac{(\boldsymbol{x}^T \boldsymbol{y})^t}{t!} = \sum_{t=0}^{\infty} \sum_{n_1+\cdots+n_D=t} \left(\frac{x_1^{n_1} \ldots x_D^{n_D}}{\sqrt{n_1! \ldots n_D!}}\right)\left(\frac{y_1^{n_1} \ldots y_D^{n_D}}{\sqrt{n_1! \ldots n_D!}}\right), \tag{29}$$

then Eqn. 27 becomes

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \frac{\sum_{t=0}^{\infty} \sum_{n_1+\cdots+n_D=t} \left(\frac{\left(\frac{x_1}{\sqrt[4]{D}}\right)^{n_1} \cdots \left(\frac{x_D}{\sqrt[4]{D}}\right)^{n_D}}{\sqrt{n_1! \ldots n_D!}}\right)\left(\frac{\left(\frac{k_{j1}}{\sqrt[4]{D}}\right)^{n_1} \cdots \left(\frac{k_{jD}}{\sqrt[4]{D}}\right)^{n_D}}{\sqrt{n_1! \ldots n_D!}}\right)}{\sum_{j'=1}^{N} \sum_{t=0}^{\infty} \sum_{n_1+\cdots+n_D=t} \left(\frac{\left(\frac{x_1}{\sqrt[4]{D}}\right)^{n_1} \cdots \left(\frac{x_D}{\sqrt[4]{D}}\right)^{n_D}}{\sqrt{n_1! \ldots n_D!}}\right)\left(\frac{\left(\frac{k_{j'1}}{\sqrt[4]{D}}\right)^{n_1} \cdots \left(\frac{k_{j'D}}{\sqrt[4]{D}}\right)^{n_D}}{\sqrt{n_1! \ldots n_D!}}\right)} \boldsymbol{v}_j + \boldsymbol{b}$$

$$= \sum_{j=1}^{N} \frac{\exp\left(\left(\frac{\boldsymbol{x}}{\sqrt[4]{D}}\right)^{\top} \frac{\boldsymbol{k}_j}{\sqrt[4]{D}}\right)}{\sum_{j'=1}^{N} \exp\left(\left(\frac{\boldsymbol{x}}{\sqrt[4]{D}}\right)^{\top} \frac{\boldsymbol{k}_{j'}}{\sqrt[4]{D}}\right)} \boldsymbol{v}_j + \boldsymbol{b} = \sum_{j=1}^{N} \frac{\exp\left(\boldsymbol{x}^{\top} \boldsymbol{k}_j / \sqrt{D}\right)}{\sum_{j'=1}^{N} \exp\left(\boldsymbol{x}^{\top} \boldsymbol{k}_{j'} / \sqrt{D}\right)} \boldsymbol{v}_j + \boldsymbol{b}. \tag{30}$$

Let $\boldsymbol{x} = \boldsymbol{q}_i$, $\boldsymbol{b} = 0$ and relax the boundness constraint of $\boldsymbol{v}_j$ in Remark 1. Eqn. 30 becomes Eqn. 2 of the softmax attention (Vaswani et al., 2017).

E BATCH NORMALIZED ATTENTION: DERIVATION OF EQN. 24

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \text{softmax}\left(\sum_{d=1}^{D} \frac{(\boldsymbol{q}_i(d) - \boldsymbol{\mu}(d))(\boldsymbol{k}_j(d) - \boldsymbol{\mu}(d))}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right) \boldsymbol{v}_j$$

$$= \sum_{j=1}^{N} \text{softmax}\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_j(d) - \boldsymbol{q}_i(d)\boldsymbol{\mu}(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_j(d) + \boldsymbol{\mu}(d)\boldsymbol{\mu}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right) \boldsymbol{v}_j$$

$$= \sum_{j=1}^{N} \frac{\exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_j(d) - \boldsymbol{q}_i(d)\boldsymbol{\mu}(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_j(d) + \boldsymbol{\mu}(d)\boldsymbol{\mu}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right)}{\sum_{j'=1}^{N} \exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_{j'}(d) - \boldsymbol{q}_i(d)\boldsymbol{\mu}(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_{j'}(d) + \boldsymbol{\mu}(d)\boldsymbol{\mu}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right)} \boldsymbol{v}_j$$

$$= \sum_{j=1}^{N} \frac{\exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_j(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_j(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right) \exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{\mu}(d)\boldsymbol{\mu}(d) - \boldsymbol{q}_i(d)\boldsymbol{\mu}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right)}{\sum_{j'=1}^{N} \exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_{j'}(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_{j'}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right) \exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{\mu}(d)\boldsymbol{\mu}(d) - \boldsymbol{q}_i(d)\boldsymbol{\mu}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right)} \boldsymbol{v}_j$$

$$= \sum_{j=1}^{N} \frac{\exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_j(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_j(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right)}{\sum_{j'=1}^{N} \exp\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_{j'}(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_{j'}(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right)} \boldsymbol{v}_j$$

$$= \sum_{j=1}^{N} \text{softmax}\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_j(d) - \boldsymbol{\mu}(d)\boldsymbol{k}_j(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right) \boldsymbol{v}_j$$

$$= \sum_{j=1}^{N} \text{softmax}\left(\sum_{d=1}^{D} \frac{\boldsymbol{q}_i(d)\boldsymbol{k}_j(d) - \frac{1}{N}\sum_{j'=1}^{N} \boldsymbol{k}_{j'}(d)\boldsymbol{k}_j(d)}{\sqrt{D}(\sigma_d^2 + \epsilon)}\right) \boldsymbol{v}_j. \tag{31}$$

F ATTENTION WITH THE RESIDUAL CONNECTION AND MATRIX PROJECTIONS

In this supplement, we first discuss attention with the residual connection and matrix projections in Appendix F.

Suppose we are given a training data $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\} \subset \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^{D_x}$ and $\mathcal{Y} = \mathbb{R}^{D_v}$. Here, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are the training inputsd, and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ are the training targets. In order to derive the attention with the residual connection and query, key, and value matrix projections, we define the function f as follows

$$\boldsymbol{y} = f(\boldsymbol{x}) := \mathbf{W}\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x})}{h(\boldsymbol{x})} + \boldsymbol{x} + \boldsymbol{b}, \tag{32}$$

where $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^{D_x}$, $\mathbf{W}^{\text{proj}} = [\boldsymbol{w}_1^{\text{proj}}, \dots, \boldsymbol{w}_D^{\text{proj}}]^\top \in \mathbb{R}^{D\times D_x}$, $\Phi(\cdot) = [\phi_1(\cdot), \dots, \phi_{D_\phi}(\cdot)] : \mathbb{R}^D \to \mathbb{R}^{D_\phi}$, $\mathbf{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_{D_v}]^\top \in \mathbb{R}^{D_v \times D_\phi}$, $\boldsymbol{b} \in \mathbb{R}^{D_v}$, and $h(\boldsymbol{x})$ is a vector-scalar function. We fit the function f to the training data $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ with an $\mathbb{L}_2$ regularization on $\mathbf{W}$ and $\mathbf{W}^{proj}$ by solving the following convex optimization problem:

$$\begin{aligned}
&\underset{\substack{\mathbf{W}, \mathbf{W}^{\text{proj}} \\ \boldsymbol{\xi}_j, \tilde{\boldsymbol{\xi}}_j, j=1,\dots,N}}{\text{minimize}} \quad \frac{1}{2}\sum_{d=1}^{D_v}\|\boldsymbol{w}_d\|^2 + \frac{1}{2}\sum_{d=1}^{D_v}\|\boldsymbol{w}_d^{\text{proj}}\|^2 + C\sum_{j=1}^{N}\sum_{d=1}^{D_v}\left(\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\xi}}_j(d)\right) \\
&\text{subject to} \quad \begin{cases} \boldsymbol{y}_j(d) - \boldsymbol{w}_d^\top\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)/h(\boldsymbol{x}_j) - \boldsymbol{x}_j - \boldsymbol{b}(d) \le \epsilon + \boldsymbol{\xi}_j(d) \\ \boldsymbol{w}_d^\top\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)/h(\boldsymbol{x}_j) + \boldsymbol{x}_j + \boldsymbol{b}(d) - \boldsymbol{y}_j(d) \le \epsilon + \tilde{\boldsymbol{\xi}}_j(d) \\ \boldsymbol{\xi}_j(d), \tilde{\boldsymbol{\xi}}_j(d) \ge 0 \end{cases}, \; j = 1, \dots, N, \; d = 1, \dots, D_v.
\end{aligned} \tag{33}$$

The Lagrangian of the optimization problem 33 is given by

$$\begin{aligned}
\mathcal{L}_1 :=& \frac{1}{2}\sum_{d=1}^{D_v}\|\boldsymbol{w}_d\|^2 + \frac{1}{2}\sum_{d=1}^{D_v}\|\boldsymbol{w}_d^{\text{proj}}\|^2 + C\sum_{j=1}^{N}\sum_{d=1}^{D_v}\left(\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\xi}}_j(d)\right) - \sum_{j=1}^{N}\sum_{d=1}^{D_v}\left(\boldsymbol{\eta}_j(d)\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\eta}}_j(d)\tilde{\boldsymbol{\xi}}_j(d)\right) \\
&- \sum_{j=1}^{N}\sum_{d=1}^{D_v}\boldsymbol{\alpha}_j(d)\left(\epsilon + \boldsymbol{\xi}_j(d) - \boldsymbol{y}_j(d) + \boldsymbol{w}_d^\top\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x}_j)} + \boldsymbol{x}_j + \boldsymbol{b}(d)\right) \\
&- \sum_{j=1}^{N}\sum_{d=1}^{D_v}\tilde{\boldsymbol{\alpha}}_j(d)\left(\epsilon + \tilde{\boldsymbol{\xi}}_j(d) + \boldsymbol{y}_j(d) - \boldsymbol{w}_d^\top\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x}_j)} - \boldsymbol{x}_j - \boldsymbol{b}(d)\right),
\end{aligned} \tag{34}$$

Similar to the derivation in Section 2.1, the partial derivatives of $\mathcal{L}_1$ with respect to the primal variable $\boldsymbol{w}_d$, $d = 1, \dots, D_v$, have to vanish for optimality, which leads to

$$\partial_{\boldsymbol{w}_d}\mathcal{L}_1 = \boldsymbol{w}_d - \sum_{j=1}^{N}(\boldsymbol{\alpha}_j(d) - \tilde{\boldsymbol{\alpha}}_j(d))\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x}_j)} = 0 \Rightarrow \boldsymbol{w}_d = \sum_{j=1}^{N}(\boldsymbol{\alpha}_j(d) - \tilde{\boldsymbol{\alpha}}_j(d))\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x}_j)}. \tag{35}$$

Note that here we only find the form of the optimal solution for $\mathbf{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_{D_v}]^\top$. The optimal value of $\mathbf{W}^{\text{proj}}$ can then be found by optimization algorithm such as the (stochastic) gradient descent when training the transformer.

Let $\boldsymbol{v}_j = [\frac{\boldsymbol{\alpha}_j(1) - \tilde{\boldsymbol{\alpha}}_j(1)}{h(\boldsymbol{x}_j)}, \dots, \frac{\boldsymbol{\alpha}_j(D_v) - \tilde{\boldsymbol{\alpha}}_j(D_v)}{h(\boldsymbol{x}_j)}]^\top$, $j = 1, \dots, N$, we obtain the following support vector expansion of the function f:

$$\begin{aligned}
f(\boldsymbol{x}) &= \left[\sum_{j=1}^{N}(\boldsymbol{\alpha}_j(1) - \tilde{\boldsymbol{\alpha}}_j(1))\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x}_j)}, \dots, \sum_{j=1}^{N}(\boldsymbol{\alpha}_j(D_v) - \tilde{\boldsymbol{\alpha}}_j(D_v))\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x}_j)}\right]^\top \frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x})}{h(\boldsymbol{x})} + \boldsymbol{x} + \boldsymbol{b}, \\
&= \left[\sum_{j=1}^{N}\frac{\boldsymbol{\alpha}_j(1) - \tilde{\boldsymbol{\alpha}}_j(1)}{h(\boldsymbol{x}_j)}\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x})^\top\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x})}, \dots, \sum_{j=1}^{N}\frac{\boldsymbol{\alpha}_j(D_v) - \tilde{\boldsymbol{\alpha}}_j(D_v)}{h(\boldsymbol{x}_j)}\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x})^\top\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x})}\right]^\top + \boldsymbol{x} + \boldsymbol{b}, \\
&= \underbrace{\sum_{j=1}^{N}\frac{\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x})^\top\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)}{h(\boldsymbol{x})}\boldsymbol{v}_j + \boldsymbol{x}}_{\text{Residual connection}} + \boldsymbol{b}.
\end{aligned} \tag{36}$$

Here, the support vector expansion of f already includes a residual connection. The softmax attention can then be derived by selecting $h(\boldsymbol{x}) := \sum_j^N \Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x})^T\Phi(\mathbf{W}^{\text{proj}}\boldsymbol{x}_j)$ and choosing Φ as in Eqn. 28 in Section 2.1. Note that in Eqn. 36, $\{\boldsymbol{x}_j\}_{j=1}^N$ and $\boldsymbol{x}$ are the training samples and test sample, respectively. In order to derive the key, query, and value matrix projections in attention, we can then relax Eqn. 36 by letting $\mathbf{W}^{\text{proj}}\boldsymbol{x}_j = \mathbf{W}_K\boldsymbol{x}_j$, $\mathbf{W}^{\text{proj}}\boldsymbol{x} = \mathbf{W}_Q\boldsymbol{x}$, $\boldsymbol{v}_j = \mathbf{W}_V\boldsymbol{x}_j$ and choosing the test sample $\boldsymbol{x}$ among the training samples $\{\boldsymbol{x}_j\}_{j=1}^N$.

Remark 6 *Here, for self-attention, we choose the test sample* $\boldsymbol{x}$ *among the training samples* $\{\boldsymbol{x}_j\}_{j=1}^N$ *to compute the attention score of a token to other tokens in the same sequence. For cross-attention where a token in a sequence attends to tokens in another sequence, this constraint can be removed.*

G 2D-CONVOLUTION ATTENTION

In this section, we discuss attention with 2D-convolution. Suppose we are given a training data $\{(\boldsymbol{x}_1^{train}, \boldsymbol{y}_1^{train}), \dots, (\boldsymbol{x}_{N_H \times N_W}^{train}, \boldsymbol{y}_{N_H \times N_W}^{train})\} \subset \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^{D_x}$ and $\mathcal{Y} = \mathbb{R}^{D_v}$. Here, $\boldsymbol{x}_1^{train}, \dots, \boldsymbol{x}_{N_H \times N_W}^{train}$ are the training inputs, and $\boldsymbol{y}_1^{train}, \dots, \boldsymbol{y}_{N_H \times N_W}^{train}$ are the training targets. Let $\mathbf{X}^{train} \in \mathbb{R}^{N_H \times N_W \times D_x}$ be the 3D-tensor of training inputs, where $\mathbf{X}^{train}(h, w, d) = \boldsymbol{x}_{N_W \times (h-1)+w}^{train}(d)$. Given a new set of inputs $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_H \times N_W}\} \subset \mathcal{X}$ and the corresponding 3D-tensor $\mathbf{X} \in \mathbb{R}^{N_H \times N_W \times D_x}$ of these inputs, where $\mathbf{X}(h, w, d) = \boldsymbol{x}_{N_W \times (h-1)+w}(d)$. We consider the function f applying on the 3D-tensor $\mathbf{X}$ and taking the following form

$$f(\boldsymbol{x}_i) = \mathbf{W}\frac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}, s))(i))}{h(\boldsymbol{x}_i)}, \; i = 1, \dots, N_H \times N_W \tag{37}$$

where Conv2D is the depth-wise 2D-convolution (Howard et al., 2017), with the kernel size $s \times s$ and identical kernel channels, applied on the input tensor $\mathbf{X}$. Here, the last dimension of $\mathbf{X}$, i.e., D_x, is the depth. Also, $\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \dots, \phi_{D_\phi}(\boldsymbol{x})] \in \mathbb{R}^{D_\phi}$, $\mathbf{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_{D_v}]^\top \in \mathbb{R}^{D_v \times D_\phi}$, $\boldsymbol{b} \in \mathbb{R}^{D_v}$, and h is a vector-scalar function. We fit the function f to the training data $\{(\boldsymbol{x}_1^{train}, \boldsymbol{y}_1^{train}), \dots, (\boldsymbol{x}_{N_H \times N_W}^{train}, \boldsymbol{y}_{N_H \times N_W}^{train})\}$ with an $\mathbb{L}_2$ regularization on $\mathbf{W}$, i.e., a ridge regression, by solving the following convex optimization problem:

$$\begin{aligned}
&\underset{\substack{\mathbf{W} \\ \boldsymbol{\xi}_j, \tilde{\boldsymbol{\xi}}_j, j=1,\dots,N_H \times N_W}}{\text{minimize}} && \frac{1}{2}\|\mathbf{W}\|_F^2 + C\sum_{j=1}^{N_H \times N_W}\sum_{d=1}^{D_v}\left(\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\xi}}_j(d)\right) = \frac{1}{2}\sum_{d=1}^{D_v}\|\boldsymbol{w}_d\|^2 + C\sum_{j=1}^{N_H \times N_W}\sum_{d=1}^{D_v}\left(\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\xi}}_j(d)\right) \\
&\text{subject to} && \begin{cases} \boldsymbol{y}_j^{train}(d) - \boldsymbol{w}_d^\top \dfrac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))}{h(\boldsymbol{x}_j^{train})} - \boldsymbol{b}(d) \le \epsilon + \boldsymbol{\xi}_j(d) \\ \boldsymbol{w}_d^\top \dfrac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))}{h(\boldsymbol{x}_j^{train})} + \boldsymbol{b}(d) - \boldsymbol{y}_j^{train}(d) \le \epsilon + \tilde{\boldsymbol{\xi}}_j(d) \\ \boldsymbol{\xi}_j(d), \tilde{\boldsymbol{\xi}}_j(d) \ge 0, \; j = 1, \dots, N_H \times N_W, \; d = 1, \dots, D_v. \end{cases}
\end{aligned} \tag{38}$$

The Lagrangian of the optimization problem 38 is given by

$$\begin{aligned}
\mathcal{L} := &\frac{1}{2}\sum_{d=1}^{D_v}\|\boldsymbol{w}_d\|^2 + C\sum_{j=1}^{N_H \times N_W}\sum_{d=1}^{D_v}\left(\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\xi}}_j(d)\right) - \sum_{j=1}^{N_H \times N_W}\sum_{d=1}^{D_v}\left(\boldsymbol{\eta}_j(d)\boldsymbol{\xi}_j(d) + \tilde{\boldsymbol{\eta}}_j(d)\tilde{\boldsymbol{\xi}}_j(d)\right) \\
&- \sum_{j=1}^{N_H \times N_W}\sum_{d=1}^{D_v}\boldsymbol{\alpha}_j(d)\left(\epsilon + \boldsymbol{\xi}_j(d) - \boldsymbol{y}_j^{train}(d) + \boldsymbol{w}_d^\top \frac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))}{h(\boldsymbol{x}_j^{train})} + \boldsymbol{b}(d)\right) \\
&- \sum_{j=1}^{N_H \times N_W}\sum_{d=1}^{D_v}\tilde{\boldsymbol{\alpha}}_j(d)\left(\epsilon + \tilde{\boldsymbol{\xi}}_j(d) + \boldsymbol{y}_j^{train}(d) - \boldsymbol{w}_d^\top \frac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))}{h(\boldsymbol{x}_j^{train})} - \boldsymbol{b}(d)\right),
\end{aligned} \tag{39}$$

Similar to the derivation in Section 2.1 in the main text, the partial derivatives of $\mathcal{L}$ with respect to the primal variable $\boldsymbol{w}_d$, $d = 1, \dots, D_v$, have to vanish for optimality, which leads to

$$\partial_{\boldsymbol{w}_d}\mathcal{L} = \boldsymbol{w}_d - \sum_{j=1}^{N_H \times N_W}(\boldsymbol{\alpha}_j(d) - \tilde{\boldsymbol{\alpha}}_j(d))\frac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))}{h(\boldsymbol{x}_j^{train})} = 0 \tag{40}$$

$$\Rightarrow \boldsymbol{w}_d = \sum_{j=1}^{N_H \times N_W}(\boldsymbol{\alpha}_j(d) - \tilde{\boldsymbol{\alpha}}_j(d))\frac{\Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))}{h(\boldsymbol{x}_j^{train})}. \tag{41}$$

Let $\boldsymbol{v}_j = [\frac{\boldsymbol{\alpha}_j(1)-\tilde{\boldsymbol{\alpha}}_j(1)}{h(\boldsymbol{x}_j^{train})}, \dots, \frac{\boldsymbol{\alpha}_j(D_v)-\tilde{\boldsymbol{\alpha}}_j(D_v)}{h(\boldsymbol{x}_j^{train})}]^\top$, $j = 1, \dots, N_H \times N_W$, and substitute Eqn. 41 into Eqn. 38, we obtain the following support vector expansion of the linear basis function f:

$$
\begin{aligned}
f(\boldsymbol{x}_i) &= \left[\sum_{j=1}^{N_H \times N_W} \frac{\boldsymbol{\alpha}_j(1)-\tilde{\boldsymbol{\alpha}}_j(1)}{h(\boldsymbol{x}_j^{train})} \frac{\mathbf{A}_{ij}}{h(\boldsymbol{x}_i)}, \dots, \sum_{j=1}^{N_H \times N_W} \frac{\boldsymbol{\alpha}_j(D_v)-\tilde{\boldsymbol{\alpha}}_j(D_v)}{h(\boldsymbol{x}_j^{train})} \frac{\mathbf{A}_{ij}}{h(\boldsymbol{x}_i)} \right]^\top + \boldsymbol{b}, \\
&= \sum_{j=1}^{N_H \times N_W} \frac{\mathbf{A}_{ij}}{h(\boldsymbol{x}_i)} \boldsymbol{v}_j + \boldsymbol{b}, \qquad (42)
\end{aligned}
$$

where $\mathbf{A}_{ij} := \Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}, s))(i))^\top \Phi(\text{Flatten}(\text{Conv2D}(\mathbf{X}^{train}, s))(j))$.

Same as in Section 2.1, we set $\boldsymbol{b}_s = 0$. To derive the softmax normalization in attention, we choose $h(\boldsymbol{x}_i) := \sum_{j=1}^{N} \mathbf{A}_{ij}$ and select Φ as in Eqn. 28. Let the training inputs $\{\boldsymbol{x}_1^{train}, \dots, \boldsymbol{x}_{N_H \times N_W}^{train}\} \subset \mathcal{X}$ be the attention keys $\{\boldsymbol{k}_1, \dots, \boldsymbol{k}_{N_H \times N_W}\} \subset \mathcal{K}$, where $\mathcal{K} = \mathbb{R}^D$, in self-attention. Also, let the new inputs $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_H \times N_W}\} \subset \mathcal{X}$ be the attention queries $\{\boldsymbol{q}_1, \dots, \boldsymbol{q}_{N_H \times N_W}\} \subset \mathcal{K}$ in self-attention. We define the 2D-Convolution Attention (Attention-Conv2D) as follows:

Definition 3 (2D-Convolution Attention) *Given a set of the key and value vectors* $\{\boldsymbol{k}_j, \boldsymbol{v}_j\}_{j=1}^{N_H \times N_W}$, *and a set of the query vectors* $\{\boldsymbol{q}_i\}_{i=1}^{N_H \times N_W}$. *Denote the key tensor and the query tensor by* $\mathbf{K} \in \mathbb{R}^{N_H \times N_W \times D}$ *and* $\mathbf{Q} \in \mathbb{R}^{N_H \times N_W \times D}$, *respectively, where* $\mathbf{K}(h, w, d) = \boldsymbol{k}_{N_W \times (h-1)+w}(d)$ *and* $\mathbf{Q}(h, w, d) = \boldsymbol{q}_{N_W \times (h-1)+w}(d)$. *The 2D-Convolution Attention (Attention-Conv2D) computes the corresponding output vector* $\boldsymbol{h}_i$ *of the query* $\boldsymbol{q}_i$ *by the following attention formula:*

$$
\boldsymbol{h}_i = \sum_{j=1}^{N} \mathit{softmax}\left(\mathit{Flatten}(\mathit{Conv2D}(\mathbf{Q}, s))(i)^\top \mathit{Flatten}(\mathit{Conv2D}(\mathbf{K}, s))(j) / \sqrt{D} \right) \boldsymbol{v}_j, \qquad (43)
$$

where the $Conv2D(\cdot, s)$ *is the depth-wise 2D-convolution (Howard et al., 2017) with the kernel size* $s \times s$ *and identical kernel channels.*

Remark 7 (Convolutional Projection for Attention in the Convolutional vision Transformer) *The convolutional projections used in the Convolutional vision Transformer (CvT) (Wu et al., 2021) can be derived from Eqn. 42 by letting the training input tensor* $\mathbf{X}^{train}$ *to be the 2D input matrix of size* $N \times D_x$ *of the self-attention layer (see Section 1.1 in the main text) reshaped into a 3D tensor of size* $N_H \times N_W \times D_x$ *where* $N = N_H \times N_W$. *Here, to avoid confusion, we denote the input of the self-attention layer by* $\mathbf{X}^{input}$ *and its reshaped version by* $\mathit{Reshape2D}(\mathbf{X}^{input})$. *We then replace the depth-wise 2D-convolution by the depth-wise separable 2D-convolution in (Wu et al., 2021) and remove the constraint that the kernels have identical channels. In order to derive the convolutional projections for the keys, queries, and values in CvT, for* $i, j = 1, \dots, N$, *we let*

$$
\begin{aligned}
\boldsymbol{k}_j &= \mathit{Flatten}(\mathit{Conv2D}(\mathbf{X}^{train}, s))(j)) = \Phi(\mathit{Flatten}(\mathit{Conv2D}(\mathit{Reshape2D}(\mathbf{X}^{input}), s, \mathbf{W}_K))(j), \\
\boldsymbol{q}_i &= \mathit{Flatten}(\mathit{Conv2D}(\mathbf{X}, s))(i)) = \Phi(\mathit{Flatten}(\mathit{Conv2D}(\mathit{Reshape2D}(\mathbf{X}^{input}), s, \mathbf{W}_Q))(i), \\
\boldsymbol{v}_j &= \mathit{Flatten}(\mathit{Conv2D}(\mathbf{X}^{train}, s))(j)) = \Phi(\mathit{Flatten}(\mathit{Conv2D}(\mathit{Reshape2D}(\mathbf{X}^{input}), s, \mathbf{W}_V))(j). \qquad (44)
\end{aligned}
$$

Here, we specify the kernel/filter $\mathbf{W}_K$, $\mathbf{W}_Q$, *and* $\mathbf{W}_V$ *to emphasize that the convolutional projections in CvT uses different kernels to compute keys, queries, and values in self-attention. Eqn. 44 matches the convolutional projects in CvT. By choosing* h *and* Φ *similar to above, we can derive the convolutional attention in CvT.*

H 1D-CONVOLUTION ATTENTION

Following the derivation for the Attention-Conv2D in Appendix G above, we can derive the 1D-Convolution Attention (Attention-Conv1D) in a similar way by letting $\mathbf{X}^{train} \in \mathbb{R}^{N \times D_x}$ and $\mathbf{X} \in \mathbb{R}^{N \times D_x}$ be 2D-matrices of training inputs and new inputs, respectively, and by replacing Conv2D by Conv1D, which is the depth-wise 1D-convolution, with the kernel size $s \times 1$ and identical kernel channels, applied on the input tensor $\mathbf{X}$. Here, the last dimension of $\mathbf{X}$, i.e., D_x, is the depth. We define the 1D-Convolution Attention (Attention-Conv1D) as follows:

Definition 4 (1D-Convolution Attention) *Given a set of the key and value vectors* $\{\boldsymbol{k}_j, \boldsymbol{v}_j\}_{j=1}^{N}$, *and a set of the query vectors* $\{\boldsymbol{q}_i\}_{i=1}^{N}$. *Denote the key matrix and the query matrix by* $\mathbf{K} :=$

$[\boldsymbol{k}_1, \dots, \boldsymbol{k}_N]^\top \in \mathbb{R}^{N\times D}$ and $\mathbf{Q} := [\boldsymbol{q}_1, \dots, \boldsymbol{q}_N]^\top \in \mathbb{R}^{N\times D}$, respectively. The 1D-Convolution Attention (Attention-Conv1D) computes the corresponding output vector $\boldsymbol{h}_i$ of the query $\boldsymbol{q}_i$ by the following attention formula:

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \mathit{softmax}\left(\mathit{Conv1D}(\mathbf{Q}, s)(i)^\top \mathit{Conv1D}(\mathbf{K}, s)(j)/\sqrt{D}\right) \boldsymbol{v}_j, \tag{45}$$

where the $Conv1D(\cdot, s)$ is the depth-wise 1D-convolution with the kernel size $s \times 1$ and identical kernel channels.

I ATTENTION WITH BATCH NORMALIZATION AND SCALED HEADS

The Attention-BN+SH combines both the Attention-BN and Attention-SH. The Attention-BN+SH fits the function f^s, $s = 1, \dots, H$, in Eqn. 17 with training sets $\{(\boldsymbol{k}_1^1, \boldsymbol{y}_1^1), \dots, (\boldsymbol{k}_{N_1}^1, \boldsymbol{y}_{N_1}^1)\}, \dots, \{(\boldsymbol{k}_1^H, \boldsymbol{y}_1^H), \dots, (\boldsymbol{k}_{N_H}^H, \boldsymbol{y}_{N_H}^H)\} \subset \mathcal{K} \times \mathcal{Y}$ of different sizes $N_1, \dots, N_H$, where $\mathcal{K} = \mathbb{R}^D$ and $\mathcal{Y} = \mathbb{R}^{D_v}$. The function f^s is defined as:

$$f^s(\boldsymbol{x}) := \mathbf{W}^s \frac{\Phi((\boldsymbol{x} - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})}{h^s((\boldsymbol{x} - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})} + \boldsymbol{b}^s, \tag{46}$$

where

$$\boldsymbol{\mu}^s = \frac{1}{N_s}\sum_{j=1}^{N_s} \boldsymbol{k}_j^s,\ \mathbf{s}^{s^{-1}} = \left[\frac{1}{\sqrt{\sigma_1^{s^2} + \epsilon}}, \dots, \frac{1}{\sqrt{\sigma_D^{s^2} + \epsilon}}\right]^\top,\ \sigma_d^{s^2} = \frac{1}{N_s}\sum_{j=1}^{N_s} (\boldsymbol{k}_j^s(d) - \boldsymbol{\mu}^s(d))^2. \tag{47}$$

Following the same derivation as in Section 2.1, we derive the following support vector expansion of f^s

$$f^s(\boldsymbol{x}) = \sum_{j=1}^{N_s} \frac{\Phi((\boldsymbol{x} - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})^\top \Phi((\boldsymbol{k}_j^s - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})}{h^s((\boldsymbol{x} - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})} \boldsymbol{v}_j^s + \boldsymbol{b}^s. \tag{48}$$

Here, $\boldsymbol{v}_j^s = \left[\frac{\boldsymbol{\alpha}_j^s(1) - \tilde{\boldsymbol{\alpha}}_j^s(1)}{h^s((\boldsymbol{k}_j^s - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})}, \dots, \frac{\boldsymbol{\alpha}_j^s(D_v) - \tilde{\boldsymbol{\alpha}}_j^s(D_v)}{h^s((\boldsymbol{k}_j^s - \boldsymbol{\mu}^s) \odot \mathbf{s}^{s^{-1}})}\right]^\top$, where $\boldsymbol{\alpha}_j^s$ and $\tilde{\boldsymbol{\alpha}}_j^s$ are the dual variables, $j = 1, \dots, N$. Same as in Section 2.1, in Eqn. 48, we choose Φ as in Eqn. 28, $h^s(\boldsymbol{x}) := \sum_j^{N_s} \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{k}_j^s)$, and $\boldsymbol{b}^s = 0$ to obtain the Batch Normalized Attention with Scaled Heads (Attention-BN+SH), which is defined as follows:

Definition 5 (Batch Normalized Attention with Scaled Heads) *Given H sets of the key and value vectors $\{\boldsymbol{k}_j^1, \boldsymbol{v}_j^1\}_{j=1}^{N_1}, \dots, \{\boldsymbol{k}_j^H, \boldsymbol{v}_j^H\}_{j=1}^{N_H}$, for each set of H query vectors $\mathbf{q}_i^1, \dots, \mathbf{q}_i^H$, $i = 1, \dots, N$, the Batch Normalized Attention with Scaled Heads (Attention-BN+SH) computes the corresponding output vector $\mathbf{h}_i$ of the queries $\mathbf{q}_i^1, \dots, \mathbf{q}_i^H$ by the following attention formula:*

$$\boldsymbol{h}_i = \sum_{s=1}^{H} \mathbf{W}_O^s \left(\sum_{j=1}^{N_s} \mathit{softmax}\left(((\boldsymbol{q}_i^s - \boldsymbol{\mu}^s) \odot \boldsymbol{s}^{s^{-1}})^\top ((\boldsymbol{k}_j^s - \boldsymbol{\mu}^s) \odot \boldsymbol{s}^{s^{-1}})/\sqrt{D}\right) \boldsymbol{v}_j^s\right), \tag{49}$$

where

$$\boldsymbol{\mu}^s = \frac{1}{N_s}\sum_{j=1}^{N_s} \boldsymbol{k}_j^s,\ \boldsymbol{s}^{s^{-1}} = \left[\frac{1}{\sqrt{\sigma_1^{s^2} + \epsilon}}, \dots, \frac{1}{\sqrt{\sigma_D^{s^2} + \epsilon}}\right]^\top,\ \sigma_d^{s^2} = \frac{1}{N_s}\sum_{j=1}^{N_s} (\boldsymbol{k}_j^s(d) - \boldsymbol{\mu}^s(d))^2. \tag{50}$$

Following the same Remark 5 in Section 2.3.2, given input sequence $\mathbf{X} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times D_x}$ of N feature vectors in self-attention, in order to generate the sets of $\{\boldsymbol{k}_j^s, \boldsymbol{v}_j^s\}_{j=1}^{N_s}$ at the scale s^{th}, we can downsample the input $\mathbf{X}$ before projecting into the key matrix $\mathbf{K}$ and the value matrix $\mathbf{V}$. In this paper, we use the average-pooling to downsample $\mathbf{X}$.

As in the same case of Attention-BN, for Attention-BN+SH, recentering queries and keys alone are sufficient for accuracy improvement, and we weight the mean $\boldsymbol{\mu}$ in Eqn 49 with a constant β. Hence Eqn. 49 is simplified to:

$$\boldsymbol{h}_i = \sum_{s=1}^{H} \mathbf{W}_O^s \left(\sum_{j=1}^{N_s} \mathrm{softmax}\left((\boldsymbol{q}_i^s - \beta\boldsymbol{\mu}^s)^\top (\boldsymbol{k}_j^s - \beta\boldsymbol{\mu}^s)/\sqrt{D}\right) \boldsymbol{v}_j^s\right). \tag{51}$$

Table 12: The values of β for Linear Attention-BN/BN+SH and Sparse Attention-BN/BN+SH trained on the selected 10 UEA tasks.

Dataset/Model	Linear Attention-BN	Linear Attention-BN+SH	Sparse Attention-BN	Sparse Attention-BN+SH
EthanolConcentration	0.15	0.95	0.8	0.2
FaceDetection	0.6	0.6	0.6	0.6
HandWriting	0.25	0.3	0.3	0.3
HeartBeat	0.6	0.15	0.4	0.5
JapaneseVowels	0.6	0.6	0.6	0.6
PEMS-SF	0.35	0.65	0.5	0.6
SelfRegulationSCP1	0.35	0.25	0.1	0.9
SelfRegulationSCP2	0.75	0.15	0.5	0.3
SpokenArabicDigits	0.6	0.6	0.6	0.6
UWaveGestureLibrary	0.65	0.55	0.9	0.3

Table 13: The values of β for Attention-BN/BN+SH trained on 25 UEA Time Series classification tasks (Bagnall et al., 2018) and 6 UEA Time Series Regression tasks.

Dataset/Model	Attention-BN	Attention-BN+SH
EthanolConcentration	0.25	0.15
FaceDetection	0.6	0.6
HandWriting	0.65	0.25
HeartBeat	0.55	0.85
JapaneseVowels	0.6	0.6
PEMS-SF	0.25	0.35
SelfRegulationSCP1	0.5	0.85
SelfRegulationSCP2	1.2	0.9
SpokenArabicDigits	0.65	0.6
UWaveGestureLibrary	0.1	0.2
ArticularyWordRecognition	0.2	0.6
BasicMotions	0.1	0.1
Epilepsy	0.3	0.2
ERing	0.1	0.9
FingerMovements	1.0	0.3
Libras	0.3	0.7
NATOPS	1.0	0.4
RacketSports	1.0	0.4
AtrialFibrillation	0.9	0.6
Cricket	0.2	1.0
StandWalkJump	1.0	0.6
HandMovementDirection	0.5	0.5
LSST	0.3	0.2
DuckDuckGeese	0.6	0.3
MotorImagery	0.2	0.3
AppliancesEnergy	0.4	0.2
BenzeneConcentration	0.2	0.1
BeijingPM10	0.1	0.5
BeijingPM25	0.1	0.2
LiveFuelMoisture	0.1	0.3
IEEEPPG	0.4	0.2

Table 14: The values of β of Attention-BN/BN+SH trained on the 5 tasks of the LRA benchmark (Tay et al., 2021).

Dataset/Model	Attention-BN	Attention-BN+SH
ListOps	0.5	0.2
Text	0.5	0.8
Retrieval	1.0	1.0
Image	0.2	0.2
Pathfinder	0.2	0.4

J HYPERPARAMETERS

In this section, we provide the hyper-parameters for our best models.

J.1 UEA TIME SERIES CLASSIFICATION AND REGRESSION

For these two benchmarks, use the set of downsampling factors $\mathbf{s} = [1, 1, 2, 2, 4, 4, 8, 8]$ for Attention-SH/BN+SH and Linear/Sparse Attention-SH/BN+SH models trained on the UEA benchmark. Table 13 and Table 12 provide the values of β used for our best Attention-BN/BN+SH and Linear Attention-BN/BN+SH, Sparse Attention-BN/BN+SH models trained on subsets of the two benchmarks.

J.2 LONG RANGE ARENA BENCHMARK

For all 5 tasks of the LRA benchmark, we set the downsampling factors **s** of Attention-SH/BN+SH, Linear/Sparse Attention-SH/BN+SH is $[1, 2]$ and kernel size of Attention-Conv1D models is 5. In addition, Table 14 provides the values β of Attention-BN/BN+SH models trained on the benchmark.

J.3 IMAGENET CLASSIFICATION

This task's β of Attention-BN/BN+SH is 1. Attention-SH/BN+SH has the downsampling factor of $[1, 1, 2, 4]$, and the kernel size of Attention-Conv2D is $(2, 2)$.