What if We Enrich *day-ahead* Solar Irradiance Time Series Forecasting with Spatio-Temporal Context? - *Supplementary material*

Anonymous Author(s)
Affiliation
Address
email

Contents

1 Experimental details

In this section, we present details about training CrossViViT and its Multi-Quantile variant, as well as the time-series baselines.

1.1 Training setup

1.1.1 CrossViViT

CrossViViT integrates two modalities: satellite video data and time-series station data. For both modalities, essential information such as geographic coordinates, elevation, and precise time-stamps is available. In this section, we provide a comprehensive explanation of the encoding process for each feature and conclude by presenting the hyperparameters of the model.

We first start by encoding the timestamps. For each time point, we have access to the following *time features*: The year, the month, the day, the hour and the minute at which the measurement was made. We use a cyclical embedding to encode these time features discarding the year. For a time feature x, its corresponding embedding can be expressed as:

$$\left[\sin\left(\frac{2\pi x}{\omega(x)}\right), \cos\left(\frac{2\pi x}{\omega(x)}\right)\right] \tag{1}$$

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Where $\omega(x)$ is the frequency for time feature x (see Table 1) for the frequency of each time feature). We concatenate these features to form our final time embedding that we simply concatenate to the context and the time-series channels respectively. Furthermore, we incorporate the elevation data for each coordinate in both the context and the time-series. Specifically, the elevation values are concatenated to their corresponding channels in the context and time-series representations.

Regarding the geographic coordinates, we possess information regarding the latitude and longitude for both the context and the station. These coordinates are normalized so as to lie in $[-1, 1]$:

$$\begin{cases} \text{lat} \leftarrow 2\left(\frac{\text{lat}+90}{180}\right) - 1 \\ \text{lon} \leftarrow 2\left(\frac{\text{lon}+180}{360}\right) - 1 \end{cases} \tag{2}$$

ROtary Positional Encoding Next, we embed these coordinates using ROtary Positional Embedding (ROPE) that we provide a PyTorch implementation for:

ROtary Positional Encoding

```python
class AxialRotaryEmbedding(nn.Module):
    def __init__(self, dim, max_freq):
        super().__init__()
        self.dim = dim
        scales = torch.linspace(1.0, max_freq / 2, dim // 4)

        self.register_buffer("scales", scales)

    def forward(self, coords: torch.Tensor):
        """
        Args:
            coords (torch.Tensor): Coordinates of shape [B, 2, height, width]
        """
        seq_x = coords[:, 0, 0, :]
        seq_x = seq_x.unsqueeze(-1)
        seq_y = coords[:, 1, :, 0]
        seq_y = seq_y.unsqueeze(-1)

        scales = self.scales[(*((None, None)), Ellipsis)]
        scales = scales.to(coords)

        seq_x = seq_x * scales * pi
        seq_y = seq_y * scales * pi

        x_sinu = repeat(seq_x, "b i d -> b i j d", j=seq_y.shape[1])
        y_sinu = repeat(seq_y, "b j d -> b i j d", i=seq_x.shape[1])

        sin = torch.cat((x_sinu.sin(), y_sinu.sin()), dim=-1)
        cos = torch.cat((x_sinu.cos(), y_sinu.cos()), dim=-1)

        sin, cos = map(lambda t: rearrange(t, "b i j d -> b (i j) d"), (sin,
        ↪  cos))
        sin, cos = map(lambda t: repeat(t, "b n d -> b n (d j)", j=2), (sin,
        ↪  cos))
        return sin, cos
```

Table 1: Frequency of each time feature.

Time feature	Frequency
Month	12
Day	31
Hour	24
Minute	60

Training configuration CrossViViT was trained on two RTX8000 GPUs, over 17 epochs with early stopping. Its Multi-Quantile variant was also trained on two RTX8000 GPUs, over 12 epochs with early stopping. The remaining settings are identical for both variants: An effective batch size of 20 was utilized for both models; The training process employed the AdamW optimizer (Loshchilov and Hutter, 2019) with a weight decay of 0.05; A cosine warmup strategy was implemented, gradually increasing the learning rate from 0 to 0.0016 over five epochs before starting the decay phase.

Below, we highlight the relevant model hyperparameters for CrossViViT and Multi-Quantile CrossViViT:

CrossViViT hyperparameters

```yaml
patch_size: [8, 8]
use_glu: True
max_freq: 128
num_mlp_heads: 1

ctx_masking_ratio: 0.99
ts_masking_ratio: 0

# These hyperparameters apply to the encoding transformers and cross-attention
dim: 384
depth: 16
heads: 12
mlp_ratio: 4
dim_head: 64
dropout: 0.4

# These only apply to the decoding transformer
decoder_dim: 128
decoder_depth: 4
decoder_heads: 6
decoder_dim_head: 128
```

Multi-Quantile CrossViViT hyperparameters

```yaml
patch_size: [8, 8]
use_glu: True
max_freq: 128
num_mlp_heads: 11
quantiles: [0.02, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.98]

ctx_masking_ratio: 0.99
ts_masking_ratio: 0

# These hyperparameters apply to the encoding transformers and cross-attention
dim: 256
depth: 16
heads: 12
mlp_ratio: 4
dim_head: 64
dropout: 0.4

# These only apply to the decoding transformer
decoder_dim: 128
decoder_depth: 4
decoder_heads: 6
decoder_dim_head: 128
```

1.1.2 Time-series baselines

We conducted training on a total of nine baseline models, and we emphasize the importance of the hyperparameters used for each of these models. Below, we provide a comprehensive overview of the hyperparameters employed in our study:

- **seq_len**: Input sequence length.
- **label_len**: Start token length.
- **pred_len**: Prediction sequence length.
- **enc_in**: Encoder input size.
- **dec_in**: Decoder input size.
- **e_layers**: Number of encoder layers.
- **d_layers**: Number of decoder layers.
- **c_out**: Output size.
- **d_model**: Dimension of model.
- **n_heads**: Number of attention heads.
- **d_ff**: Dimension of Fully Connected Network.
- **factor**: Attention factor.
- **embed**: Time features encoding, options:[timeF, fixed, learned].
- **distil**: Whether to use distilling in encoder.
- **moving average**: Window size of moving average kernel.

We adapted the majority of the baselines using the Time Series Library (TSlib (Wu et al., 2023)), which served as a valuable resource in our experimentation. We refer the reader to the original papers, which served as a base for the hyperparameters utilized in our study, in order to have a comprehensive understanding of the models and the training settings.

LightTS

```yaml
model:
  enc_in: 10
  seq_len: 48
  pred_len: 48
  d_model: 256
  dropout: 0.05
  chunk_size: 24
```

FiLM

```yaml
model:
  enc_in: 10
  seq_len: 48
  label_len: 24
  pred_len: 48
  e_layers: 2
  ratio: 0.4
```

DLinear

```yaml
model:
  enc_in: 10
  seq_len: 48
  pred_len: 48
  moving_avg: 25
  individual: False
```

Crossformer

```yaml
model:
  enc_in: 10
  seq_len: 48
  pred_len: 48
  d_model: 1024
  n_heads: 2
  e_layers: 4
  d_ff: 2048
  factor: 10
  dropout: 0.01
```

Reformer

```yaml
model:
  enc_in: 10
  c_out: 1
  seq_len: 48
  pred_len: 48
  d_model: 512
  n_heads: 8
  e_layers: 3
  d_ff: 2048
  factor: 5
  dropout: 0.05
  embed: timeF
  activation: gelu
```

PatchTST

```yaml
model:
  enc_in: 10
  c_out: 1
  seq_len: 48
  pred_len: 48
  d_model: 1024
  n_heads: 6
  e_layers: 3
  d_ff: 2048
  factor: 10
  dropout: 0.05
  embed: timeF
  activation: gelu
```

Autoformer

```yaml
model:
  enc_in: 10
  dec_in: 10
  c_out:
  seq_len: 48
  label_len: 24
  pred_len: 48
  moving_avg: 25
  d_model: 1024
  n_heads: 8
  e_layers: 3
  d_layers: 2
  d_ff: 2048
  factor: 10
  dropout: 0.01
  embed: timeF
  activation: gelu
```

Informer

```yaml
model:
  enc_in: 10
  dec_in: 10
  c_out: 1
  label_len: 24
  pred_len: 48
  d_model: 2048
  n_heads: 4
  e_layers: 2
  d_layers: 2
  d_ff: 2048
  factor: 5
  dropout: 0.1
  embed: timeF
  activation: gelu
  distil: True
```

FEDformer

```yaml
model:
  enc_in: 10
  dec_in: 10
  c_out: 1
  seq_len: 48
  label_len: 24
  pred_len: 48
  moving_avg: 25
  d_model: 512
  n_heads: 8
  e_layers: 3
  d_layers: 2
  d_ff: 2048
  dropout: 0.05
  version: fourier
  mode_select: random
  modes: 32
```

The training of the baselines took place on a single RTX8000 GPU over the course of 100 epochs. During training, a batch size of 64 was consistently employed. For model optimization, we utilized the AdamW optimizer (Loshchilov and Hutter, 2019), incorporating a weight decay value set to 0.05. Moreover, we implemented a learning rate reduction strategy known as Reduce Learning Rate on Plateau, which gradually decreased the learning rate by a factor of 0.5 after a patience of 10 epochs.

For the hyperparameter tuning of the baselines, we employed the Orion package (Bouthillier et al., 2022), which is an asynchronous framework designed for black-box function optimization. Its primary objective is to serve as a meta-optimizer specifically tailored for machine learning models. As an example, we present the details of the Crossformer hyperparameter tuning scheme, showcasing the approach we followed to optimize its performance:

Crossformer hyperparameters tuning

```yaml
params:
    optimizer.lr: loguniform(1e-8, 0.1)
    optimizer.weight_decay: loguniform(1e-10, 1)
    d_model: choices([128,256,512,1024,2048])
    d_ff: choices([1024,2048,4096])
    n_heads: choices([1,2,4,8,16])
    e_layers: choices([1,2,3,4,5])
    dropout: choices([0.01,0.05,0.1,0.2,0.25])
    factor: choices([2,5,10])
    max_epochs: fidelity(low=5, high=100, base=3)
```

2 Additional results and visualisations

2.1 Performance on (unobserved) validation PAY station, on validation and test years

This section presents a comprehensive comparative analysis that assesses the performance of CrossViViT in relation to various timeseries approaches and solar irradiance baselines specifically for the PAY station. The evaluation encompasses both the validation period (2017-2019) and the test period (2020-2022), with the validation period serving as the basis for model selection. The results, as presented in Table 2, offer compelling evidence of the superior performance of CrossViViT compared to the alternative approaches across all evaluation splits. These findings underscore the crucial role of accurately capturing cloud dynamics in solar irradiance forecasting, which is particularly pronounced in the "Hard" splits.

Table 2: Comparison of model performances in the **val station** PAY, during **test years** (2020-2022) and **val years** (2017-2019). We report the MAE and RMSE for the easy and difficult splits (presented in the main paper) along with the number of data points for each split. We add the MAE resulting from the Multi-Quantile CrossViViT median prediction, along with p_t, the probability for the ground-truth to be included within the interval, averaged across time steps.

Models	Parameters	PAY (Test years - 2020-2022)						PAY (Val years - 2017-2019)					
		All (12171)		Easy (8053)		Hard (4118)		All (27166)		Easy (16683)		Hard (10483)	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Persistence	N/A	61.9	140.05	44.98	105.47	94.99	190.31	63.61	141.16	48.57	108.91	87.54	180.99
Fourier$_3$	N/A	69.21	129.29	52.17	91.61	102.53	181.63	71.71	130.58	56.03	94.81	96.68	172.86
Fourier$_4$	N/A	65.67	131.13	48.77	93.68	98.72	183.48	67.35	132.41	51.8	96.71	92.08	174.79
Fourier$_5$	N/A	63.39	132.38	46.14	95.08	97.14	184.71	65.71	133.87	50.03	98.61	90.66	175.98
Clear Sky (Ineichen, 2016)	N/A	65.99	138.78	56.41	116.72	84.72	174.03	62.12	131.69	52.59	109.91	77.28	160.36
ReFormer (Kitaev et al., 2020)	8.6M	59.15	109.75	51.67	91.67	73.79	138.44	59.3	109.3	54.06	94.31	67.64	129.62
Informer (Zhou et al., 2021)	56.7M	79.18	133.47	75.8	126.07	85.78	146.86	77.51	131.72	75.65	126.58	80.46	139.52
FiLM (Zhou et al., 2022b)	9.4M	69.87	124.86	55.93	91.15	97.11	172.72	71.1	125.32	58.35	94.17	91.39	163.06
PatchTST (Nie et al., 2023)	9.6M	63.21	131.64	53.06	113.78	83.06	160.93	63.88	131.14	55.47	115.59	77.25	152.64
LighTS (Zhang et al., 2022)	32K	58.57	111.22	48.93	88.69	77.4	145.53	58.22	110.84	50.72	91.24	70.16	136.33
CrossFormer (Zhang and Yan, 2023)	227M	59.64	111.03	51.04	90.34	76.45	143.08	59.39	111.49	52.55	93.15	70.29	135.66
FEDFormer (Zhou et al., 2022a)	23.6M	63.44	110.62	58.15	97.33	73.79	**132.83**	62.46	109.92	59.88	100.61	66.57	**123.3**
DLinear (Zeng et al., 2022)	4.7K	75.09	128.64	59.42	93.82	105.74	178.04	76.78	129.45	62.64	97.93	99.29	167.81
AutoFormer (Wu et al., 2021)	50.4M	73.36	117.22	68.89	105.52	82.12	137.25	71.39	114.96	69.42	106.79	74.54	126.88
CrossViViT	145M	**51.47**	**107.73**	**41.59**	**85.14**	**70.81**	141.87	**52.02**	**108.77**	**44.71**	**90.47**	**63.65**	132.79
		MAE	p_t	MAE	p_t	MAE	p_t	MAE	p_t	MAE	p_t	MAE	p_t
Multi-Quantile CrossViViT	78.8M	59.06	0.83	54.96	0.83	67.06	0.84	57.65	0.86	51.75	0.87	67.05	0.86

2.2 Visualisations for day ahead time series predictions

This section presents visualizations of predictions generated by CrossViViT and CrossFormer on the PAY station for both the validation period (2017-2019) (see Figure 2) and the test period (2020-2022) (see Figure 1). We also present visualizations of predictons generated by the Multi-Quantile version, for the two periods, on the PAY station. The predictions depicted in red are the median (50% quantile) estimation from the model, and the generated prediction interval is defined as the interval between the two predefined quantiles: $[q_{0.02}, q_{0.98}]$

3 Vision Transformers (ViT) and Video Vision Transformers (ViViT)

ViT The Vision Transformer (ViT) model (Dosovitskiy et al., 2020) leverages self-attention mechanisms inspired by the popular transformer architecture (Bahdanau et al., 2015; Vaswani et al., 2017) to efficiently process images. The input image of dimensions $H \times W$ is divided into non-overlapping patches $x_i \in \mathbb{R}^{h \times w}$, which are linearly projected and transformed into d-dimensional vector tokens $z_i \in \mathbb{R}^d$ using a learned weight matrix $\mathbf{E}$ that applies 2D convolution. The sequence of patches is defined as $\mathbf{z} = [z_{\text{class}}, \mathbf{E}x_1, ..., \mathbf{E}x_P] + \mathbf{E}_{\text{pos}}$, where z_{class} is an optional learned classification token representing the class label, and $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{P \times d}$ is a 1D learned positional embedding that encodes position information.

To extract global features from the image, the embedded patches undergo K transformer layers. Each transformer layer k consists of Multi-Headed Self-Attention (MSA) (Vaswani et al., 2017), layer normalization (LN) (Ba et al., 2016), and MLP blocks with residual connections. The MLPs consist of two linear layers separated by the Gaussian Error Linear Unit (GELU) activation function (Hendrycks and Gimpel, 2016). The output of the final layer can be used for image classification, either by directly utilizing the classification token or by applying global average pooling to all tokens $\mathbf{z}^L$ if the classification token was not used initially.

ViViT The Video Vision Transformer (Arnab et al., 2021) extends the ViT model to handle video classification tasks by incorporating both spatial and temporal dimensions within a transformer-like architecture. The authors propose multiple versions of the model, with a key consideration being how to embed video clips for attention computation.

Two approaches are presented: (1) *Uniform frame sampling* and (2) *Tubelet embedding*. The first method involves uniformly sampling n_t frames from the video clip, applying the same ViT embedding method Dosovitskiy et al. (2020) to each 2D frame independently, and concatenating the resulting tokens into a single sequence. The second method extracts non-overlapping spatio-temporal "tubes"

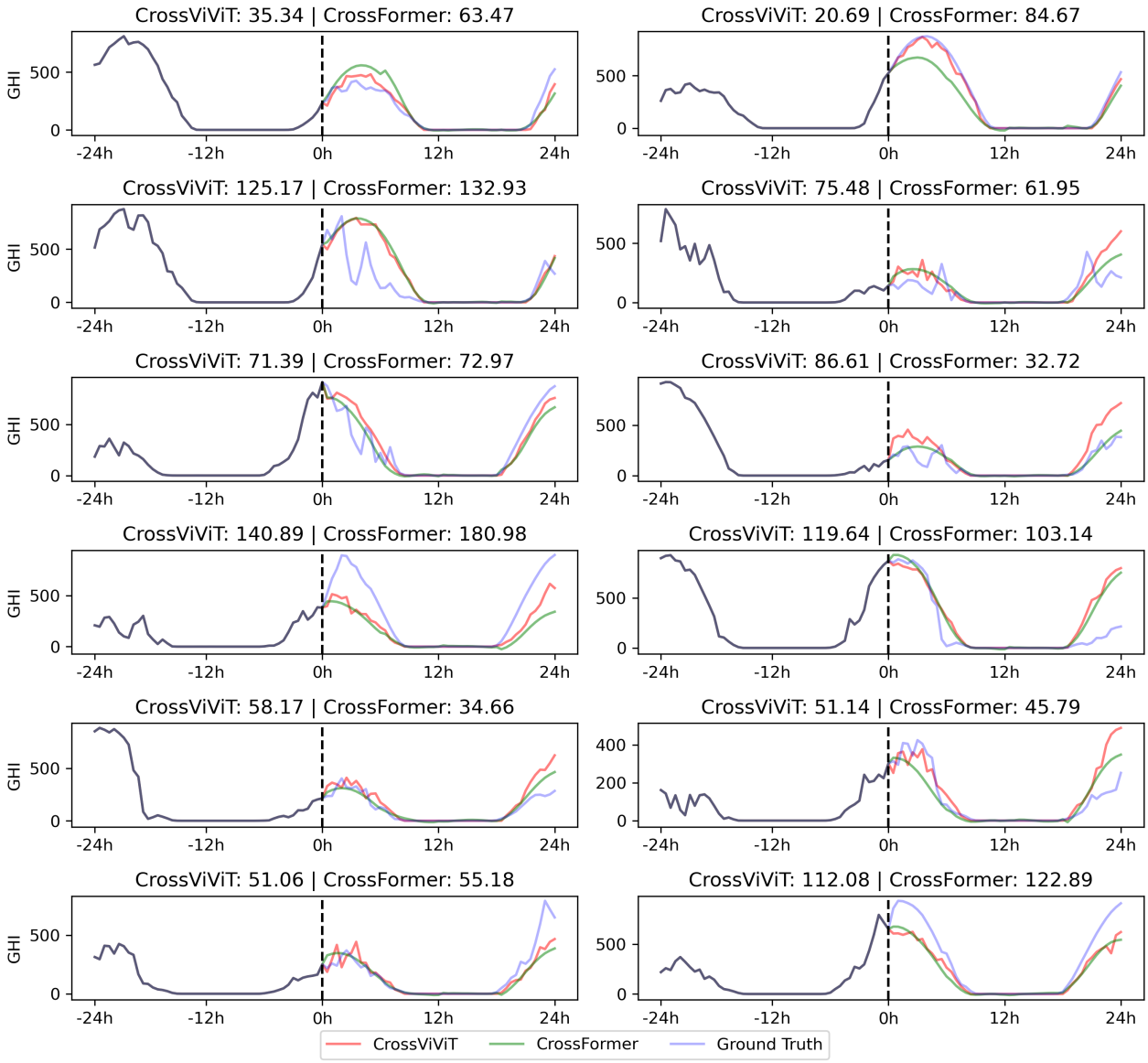

Figure 1: Visualizations of CrossViViT and CrossFormer are presented for a subset of 12 randomly selected days from the "Hard" split on the PAY station during the test period spanning from 2020 to 2022.

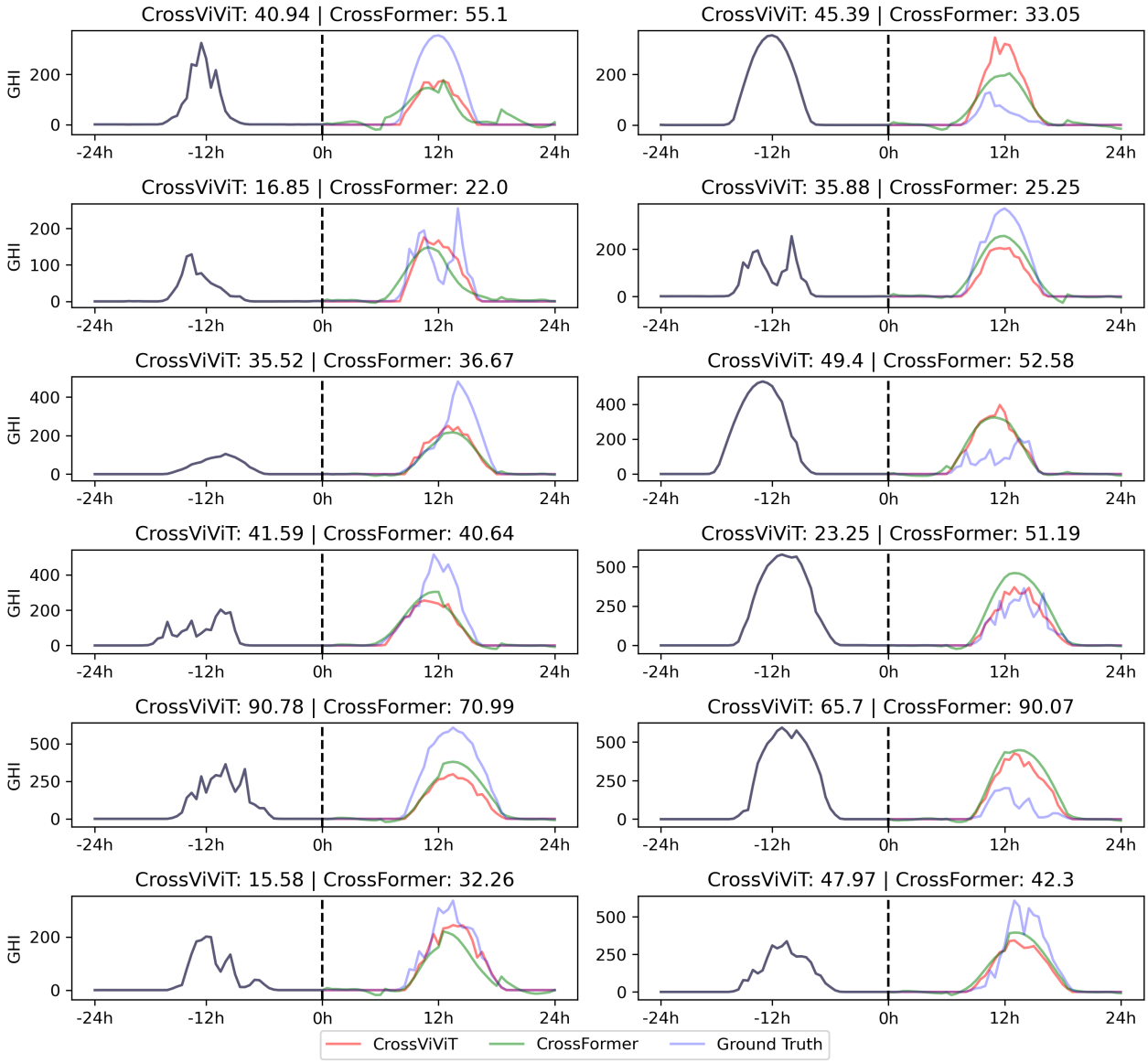

Figure 2: Visualizations of CrossViViT and CrossFormer are presented for a subset of 12 randomly selected days from the "Hard" split on the PAY station during the validation period spanning from 2017 to 2019.

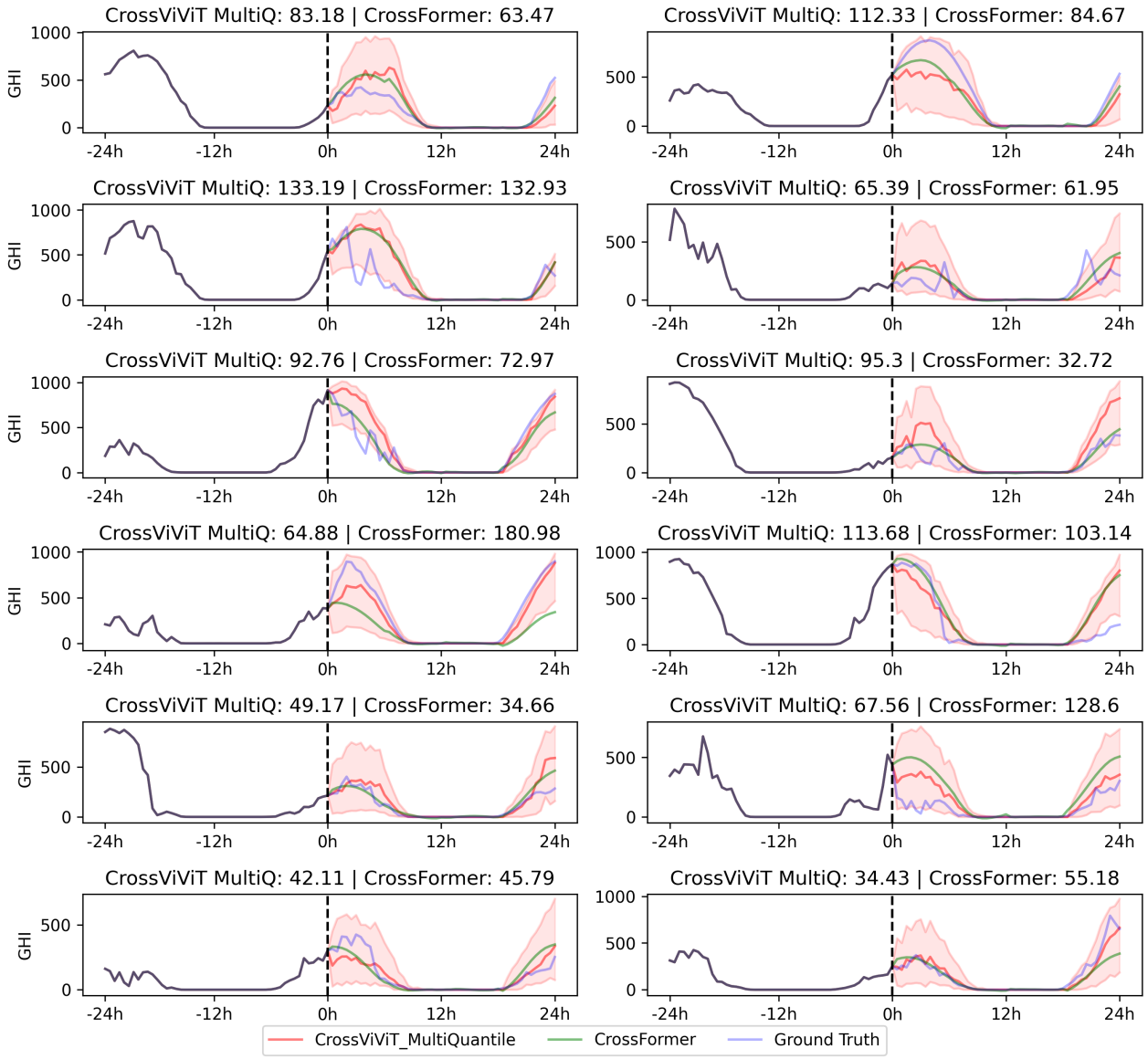

Figure 3: Visualizations of Multi-Quantile CrossViViT (median prediction and $[q_{0.02}, q_{0.98}]$ prediction interval) and CrossFormer predictions are presented for a subset of 12 randomly selected days from the "Hard" split on the PAY station during the validation period spanning from 2020 to 2022.

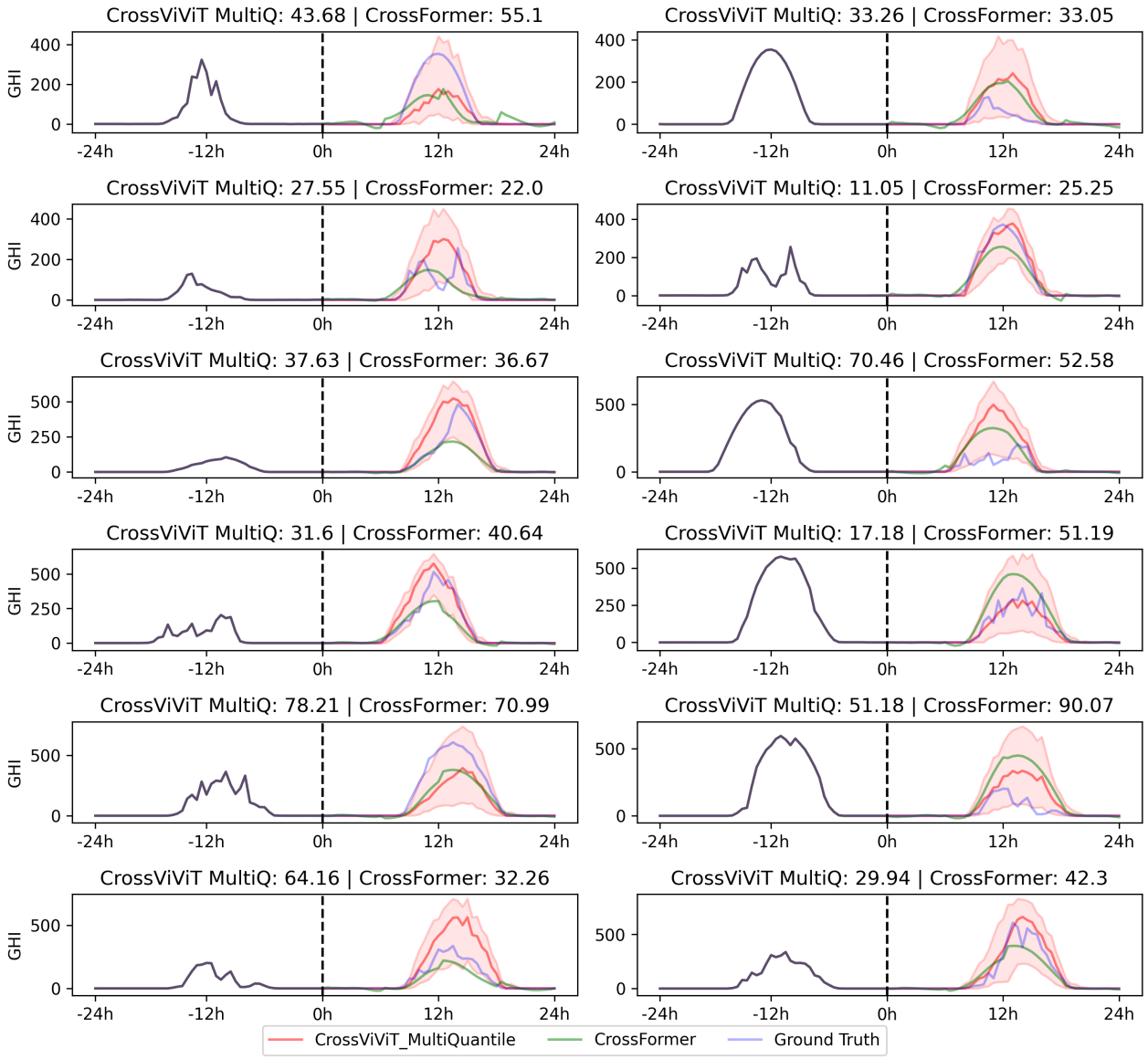

Figure 4: Visualizations of Multi-Quantile CrossViViT (median prediction and $[q_{0.02}, q_{0.98}]$ prediction interval) and CrossFormer predictions are presented for a subset of 12 randomly selected days from the "Hard" split on the PAY station during the validation period spanning from 2017 to 2019.

from the input volume and linearly projects them into $\mathbb{R}^d$, effectively extending ViT's embedding technique to 3D data, akin to performing a 3D convolution. Intuitively, the second method allows for the fusion of spatio-temporal information during tokenization, whereas the first method requires independent temporal fusion post-tokenization. Experimental results, however, show only slightly superior performance for the second method in specific scenarios, despite its significantly higher computational complexity (Arnab et al., 2021).

References

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. (2021). Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bouthillier, X., Tsirigotis, C., Corneau-Tremblay, F., Schweizer, T., Dong, L., Delaunay, P., Normandin, F., Bronzi, M., Suhubdy, D., Askari, R., Noukhovitch, M., Xue, C., Ortiz-Gagné, S., Breuleux, O., Bergeron, A., Bilaniuk, O., Bocco, S., Bertrand, H., Alain, G., Serdyuk, D., Henderson, P., Lamblin, P., and Beckham, C. (2022). Epistimio/orion: Asynchronous Distributed Hyperparameter Optimization.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Ineichen, P. (2016). Validation of models that estimate the clear sky global and beam solar irradiance. *Solar Energy*, 132:332–344.

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. (2023). Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430.

Zeng, A., Chen, M.-H., Zhang, L., and Xu, Q. (2022). Are transformers effective for time series forecasting? *ARXIV.ORG*.

Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., and Li, J. (2022). Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures.

Zhang, Y. and Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022a). Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting.

Zhou, T., Ma, Z., xue wang, Wen, Q., Sun, L., Yao, T., Yin, W., and Jin, R. (2022b). FiLM: Frequency improved legendre memory model for long-term time series forecasting. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.