Met^{2}Net: A Decoupled Two-Stage Spatio-Temporal Forecasting Model for Complex Meteorological Systems

Supplementary Material

Algorithm 1 Pseudocode of Implicit Two-Stage Process in a PyTorch-like Style Integrated Within a Inference Pipeline.

```
# E1, E2: encoders.
# D1, D2: decoders.
# H: translator.

for x in loader: # load a minibatch x with N samples
   x1, x2 = slice(x) # Slicing Operations in Python
   z1_x = E1(x1) # Independent encoding of Variables
   z2_x = E2(x2)
   z_x = torch.stack(z1_x,z2_x)

   # Spatio-Temporal learnning and variable fusion
   z_y = H(z_x)

   z1_y, z2_y = slice(z_y)
   y1 = D1(z1_y) # Independent decoding of Variables
   y2 = D2(z2_y)
   y_pre = torch.stack(y1, y2)
```

6. Appendix

6.1. Experimental Setup for Variable Distributions

Data Distributions. The experiment utilized the 2018 T2M (2-meter air temperature) and TCC (Total Cloud Cover) data from the WeatherBench dataset to analyze spatial and temporal distribution patterns. The data were normalized to the range [0, 1] for comparability. Spatial analysis was based on grid data from a single time step to capture geographic variability, while temporal analysis focused on the time series data of a single grid point.

First-Order Differences. The experiment analyzed first-order differences of 2018 T2M and TCC data from the WeatherBench dataset. Temporal differences were calculated between steps, and spatial differences from adjacent grid points along the h and w dimensions. Differences were standardized to zero mean and unit variance, with outliers exceeding three standard deviations removed.

6.2. Pseudocode of Inference Pipeline

In Algorithm 1, we present the pseudocode of our method within a inference pipeline. For simplicity, we demonstrate the case with only two variables.

6.3. Pseudocode of Trainning Pipeline

In Algorithm 2, we present the pseudocode of our method within a training pipeline. For simplicity, we demonstrate the case with only two variables.

Algorithm 2 Pseudocode of Implicit Two-Stage Process in a PyTorch-like Style Integrated Within a Training Pipeline.

```
# E1, E2, E1_m, E2_m: encoder that applies gradient
     updates and momentum updates to two different
     variables.
# D1, D2, D1_m, D2_m: decoder that applies gradient
     updates and momentum updates to two different
     variables.
# H, H_m: translator that gradient updates and
     momentum updates.
# a: momentum

E1_m.params = E1.params # initialize
E2_m.params = E2.params # initialize
D1_m.params = D1.params # initialize
D2_m.params = D2.params # initialize
H_m.params = H.params # initialize

for x,y in loader: # load a minibatch x,y with N
     samples
   # stage 1
   x1, x2 = slice(x) # Slicing Operations in Python
   z1_x = E1(x1) # Independent encoding of Variables
   z2_x = E2(x2)
   z_x = torch.stack(z1_x,z2_x)

   # Spatio-Temporal learnning and variable fusion
   z_y = H_m(z_x)

   z1_y, z2_y = slice(z_y)
   y1 = D1(z1_y) # Independent decoding of Variables
   y2 = D2(z2_y)
   y_rec = torch.stack(y1, y2)

   # momentum update
   E1_m.params = a*E1_m.params +(1-a)*E1.params
   E2_m.params = a*E2_m.params +(1-a)*E2.params
   D1_m.params = a*D1_m.params +(1-a)*D1.params
   D2_m.params = a*D2_m.params +(1-a)*D2.params

   loss_rec = MSE(y_rec,y)

   # stage 2
   x1, x2 = slice(x) # Slicing Operations in Python
   z1_x = E1_m(x1) # use momentum updates module
   z2_x = E2_m(x2)
   z_x = torch.stack(z1_x, z2_x)

   # Spatio-Temporal learnning and variable fusion
   z_y_pre = H(z_x)

   z1_y, z2_y = slice(z_y)
   y1 = D1_m(z1_y)
   y2 = D2_m(z2_y)
   y_pre= torch.stack(y1, y2)

   y1, y2 = slice(y)
   z1_y = E1_m(y1) # use momentum updates module
   z2_y = E2_m(y2)
   z_y = torch.stack(z1_y, z2_y)

   # momentum update
   H_m.params = a*H_m.params +(1-a)*H.params

   loss_pre = MSE(z_y_pre,z_y)

   loss = loss_rec + loss_pre

   # Adam update: query network
   loss.backward()
```

6.4. Impact of different blocks in translator

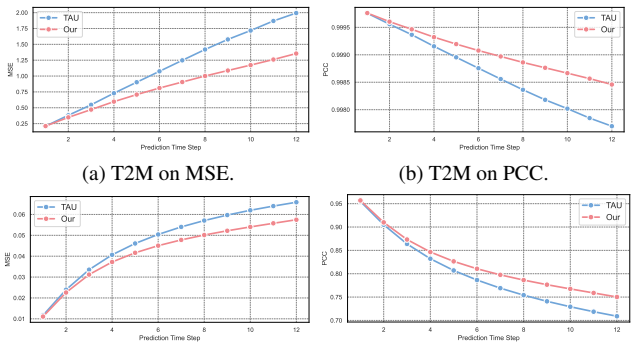
We tested different blocks within the Translator of our framework, as shown in Table 9. The results indicate that while the TAU block remains a competitive choice, our method consistently outperforms the baseline methods across all tested blocks. This demonstrates the robustness of our framework in handling various blocks. Regardless of the block selected, our method maintains superior performance, validating the effectiveness of the proposed framework across different configurations.

Table 9. Impact of using different **Blocks** in the translator on T2M and TCC prediction performance. The light gray background indicates results not applied in our framework. The white background indicates results obtained using different translators within our framework.

Method	T2M			TCC		
	MSE	MAE	RMSE	MSE	MAE	RMSE
HorNet	1.2010	0.6906	1.0960	0.0469	0.1475	0.2166
TAU	1.1620	0.6707	1.0780	0.0472	0.1460	0.2173
Wast	1.0980	0.6338	1.0440	-	0.1452	0.2150
ConvNext	1.0238	0.6598	1.0105	0.0440	0.1426	0.2096
SimVPv2	0.9215	0.6148	0.9588	0.0425	**0.1367**	0.2061
PoolFormer	0.9493	0.6271	0.9730	0.0435	0.1426	0.2085
Hornet	0.8778	0.5987	0.9358	0.0423	0.1388	0.2055
Moga	1.0314	0.6643	1.0141	0.0445	0.1455	0.2109
TAU	**0.8271**	**0.5770**	**0.9094**	**0.0422**	0.1370	**0.2054**

6.5. Performance evolves with time steps.

As shown in Figure 7, the performance of both models (TAU and our method) varies with the prediction time steps for different variables (T2M and TCC). Although the performance of both models declines as the time step increases, our method consistently outperforms TAU, exhibiting slower growth in MSE and more stable PCC.



(a) T2M on MSE. (b) T2M on PCC.

(c) TCC on MSE. (d) TCC on PCC.

Figure 7. Performance comparison of T2M and TCC prediction using MSE and PCC across different prediction time steps.

6.6. Single meteorological variables prediction

To validate the applicability and effectiveness of our method, we conducted single-variable prediction experiments. Table 10 presents quantitative comparison results for UV10 and T2M, showing that our method outperforms existing models across all key metrics. Although single-variable accuracy is lower than multi-variable predictions (Table 1), this highlights the effectiveness of our multi-variable fusion approach and the importance of considering multiple variables in meteorological forecasting.

Table 10. Quantitative comparison on the UV10 and T2M variables. The subscript S in Baselines indicates the single-variable model.

Method	UV10		T2M	
	MAE	RMSE	MAE	RMSE
ConvLSTM	0.9215	1.3775	0.7949	1.2330
PredRNN++	0.9019	1.3685	0.7866	1.2070
SimVP	0.9510	1.4091	0.7037	1.1130
ConvNeXt	0.8698	1.3006	0.7220	1.1300
TAU	0.8426	1.2619	0.6607	1.0780
Met^{2}Net$_S$	0.8197	1.2518	0.6536	1.0753

6.7. Additional metrics and resource comparison

We report both the anomaly correlation coefficient (ACC) and the resource consumption of different models on the cropped ERA5 dataset, as presented in Table 11. All experiments are conducted under the same setting with fp32 precision and batch size 16 on a single NVIDIA RTX 4090 GPU.

Met^{2}Net achieves the highest forecasting accuracy across all variables while maintaining moderate parameter count and competitive efficiency in terms of computation and memory usage.

Table 11. ACC and resource comparison on cropped ERA5.

Method	Params (M)	FLOPs (G)	Mem (MiB)	Time (Min)	MSL	U10	V10	T2M
					ACC			
ConvLSTM	7.44	135.0	4398	2:42	0.9671	0.9073	0.9427	0.9293
MogaNet	12.83	18.9	14408	1:37	0.9690	0.9181	0.9492	0.9533
TAU	12.29	18.3	11942	1:21	0.9652	0.9097	0.9432	0.9510
Met^{2}Net	8.90	119.0	23078	**2:24**	**0.9803**	**0.9340**	**0.9590**	**0.9711**

Note: All experiments were conducted on a single NVIDIA RTX 4090 GPU. **Mem** indicates the peak GPU memory usage with fp32 and a batch size of 16; **Time** refers to the training time per epoch.

6.8. Scalability under increased variable input

To evaluate the scalability of the proposed method, we expand the number of input meteorological variables in the $Weather_L$ setting by introducing three additional physical fields: total precipitation (TP), geopotential height (Z), and top-of-atmosphere incoming shortwave radiation (TISR). Correspondingly, we increase the encoder–decoder pairs

from 4 to 8, while maintaining the same translator architecture. We compare the forecasting performance of TAU and Met^{2}Net under varying numbers of encoder–decoder pairs. The results are summarized in Table 12.

Table 12. Forecasting performance with increased variables (UV10 and TCC) under different encoder–decoder configurations.

Method	# P	UV10		TCC	
	(M)	MSE	RMSE	MSE	RMSE
TAU_1	12.2	1.5925	1.2619	0.0472	0.2173
Met^2Net_4	8.7	1.5055	1.2270	0.0422	0.2054
TAU_8	12.2	1.7345	1.3161	0.0444	0.2107
Met^2Net_8	8.9	**1.4740**	**1.2129**	**0.0417**	**0.2043**

The results show that Met^{2}Net maintains superior forecasting accuracy while scaling to more variables, with only a marginal increase in parameter count (from 8.65M to 8.87M). In contrast, TAU exhibits a performance drop despite using the same number of encoders. This demonstrates that Met^{2}Net is well-suited for scalable spatiotemporal modeling in multi-variable settings.

6.9. Cross-Variable Attention Analysis

To better understand the inter-variable dependencies captured by the translator module, we analyze the cross-variable attention weights learned during the forecasting process. In our model, each meteorological variable is encoded independently as a token, and the translator performs self-attention over these variable tokens to enable dynamic information aggregation. Figure 8 presents the averaged attention map across all samples and heads. Each row represents the target variable being predicted, and each column indicates the source variable being attended to. The values are normalized attention weights, reflecting how much each variable contributes to the others.

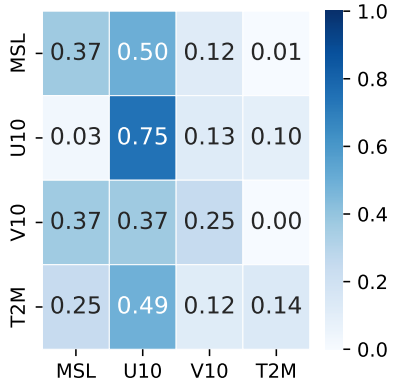
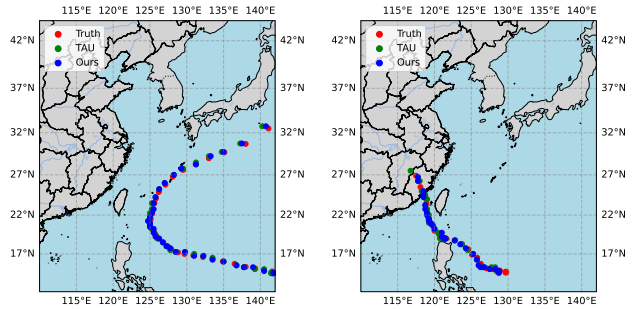


Figure 8. Cross-variable attention map extracted from the translator module. Brighter colors indicate stronger attention. Variables include: MSL, U10, V10, T2M, TP, Z, TISR, and TCC.

The attention map reveals several meaningful patterns. For example, the model places strong attention between U10 and V10, and between T2M and TCC, which are physically correlated. This indicates that the translator can adaptively capture variable-specific influences, enhancing both forecasting performance and model interpretability.

6.10. Additional tracking tropical cyclones

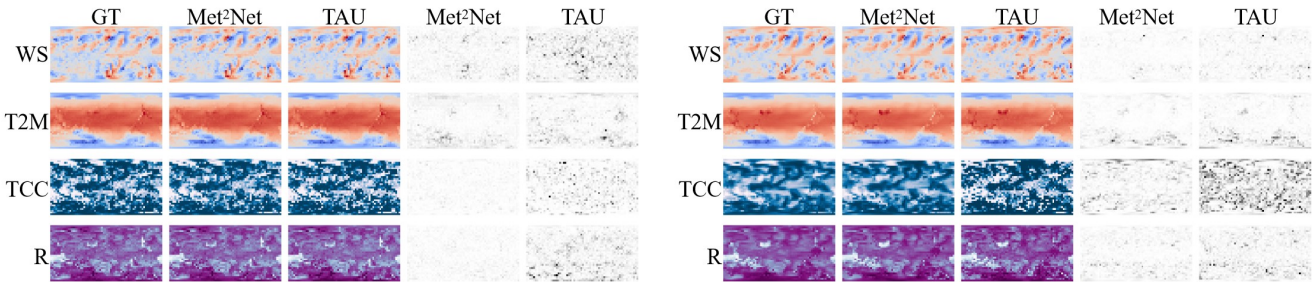


(a) Predicted and ground truth tracks of Typhoon MAWAR (1-Hour lead time).

(b) Predicted and ground truth tracks of Typhoon DOKSURI (3-Hour lead time).

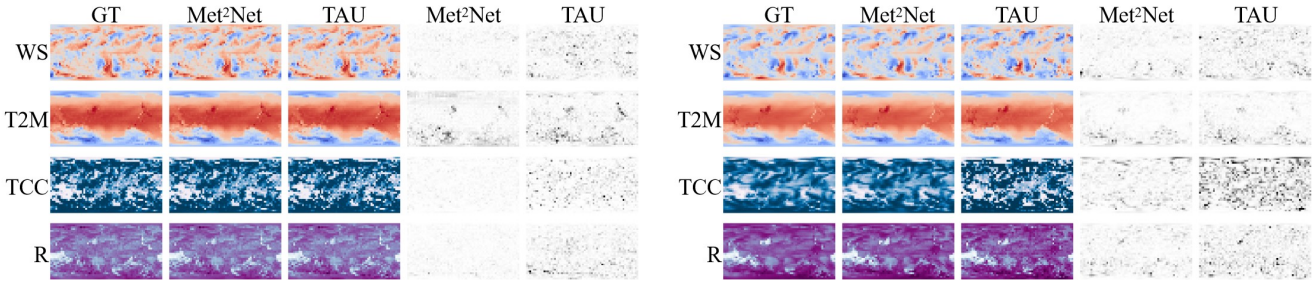
Figure 9. Tracking tropical cyclones.

6.11. Additional visualization results



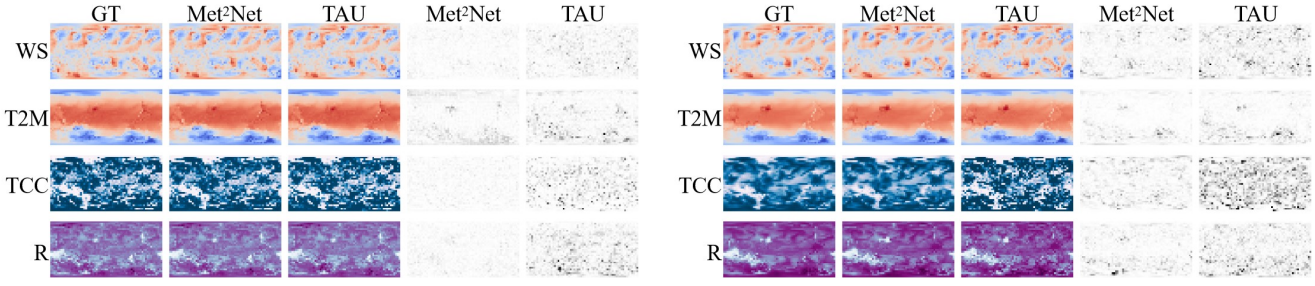
(a) t=1. (b) t=12.

Figure 10. Visualization of prediction results for different lead times. (a) Results at a forecast time of 1 hour. The background in white represents the absolute error (| GT-Prediction |) for each model. (b) Results at a forecast time of 12 hours.



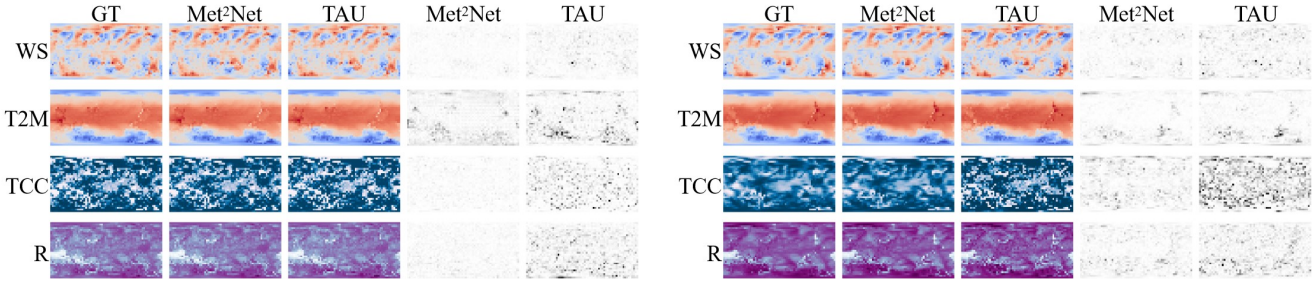
(a) t=1. (b) t=12.

Figure 11. Visualization of prediction results for different lead times. (a) Results at a forecast time of 1 hour. The background in white represents the absolute error (| GT-Prediction |) for each model. (b) Results at a forecast time of 12 hours.



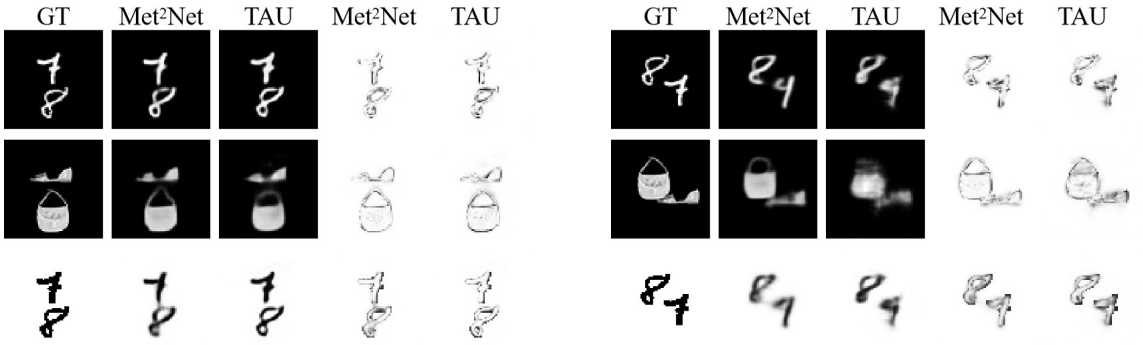
(a) t=1. (b) t=12.

Figure 12. Visualization of prediction results for different lead times. (a) Results at a forecast time of 1 hour. The background in white represents the absolute error (| GT-Prediction |) for each model. (b) Results at a forecast time of 12 hours.



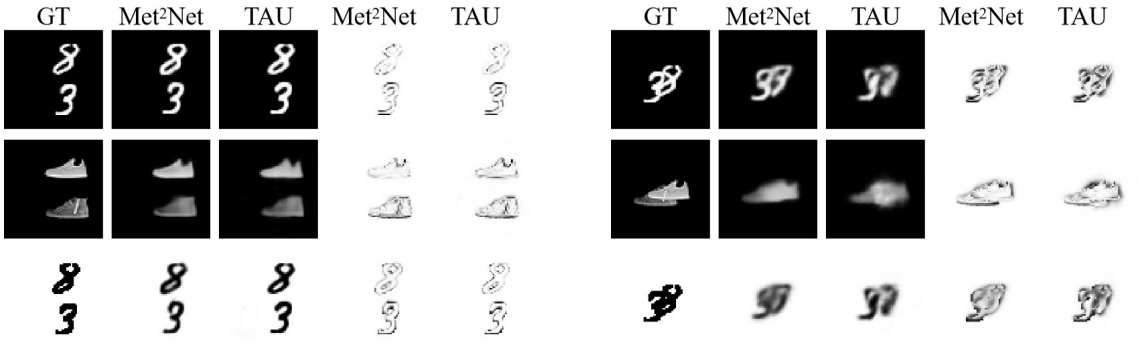
(a) t=1. (b) t=12.

Figure 13. Visualization of prediction results for different lead times. (a) Results at a forecast time of 1 hour. The background in white represents the absolute error (| GT-Prediction |) for each model. (b) Results at a forecast time of 12 hours.



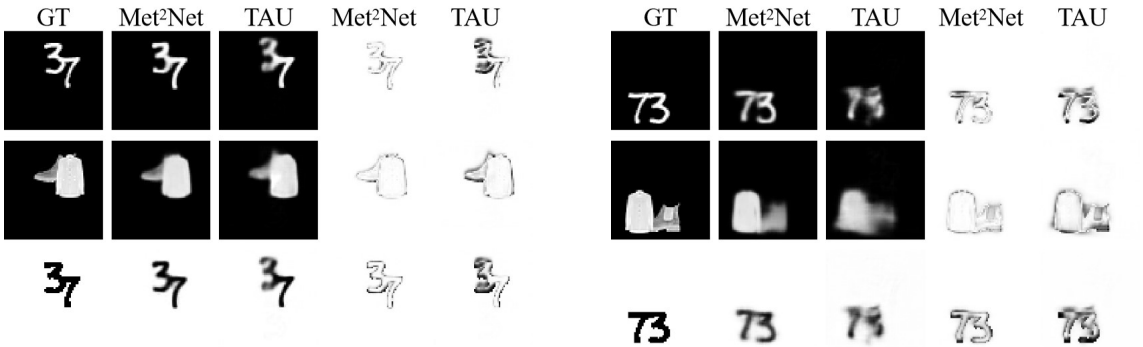
(a) t=1. (b) t=10.

Figure 14. Visualization of prediction results for different lead times on the Mv_Mmfnist dataset. The last two columns represent the absolute error (| GT - Prediction |) for each model. (a) Results at a forecast time of 1 frame. (b) Results at a forecast time of 10 frame.



(a) t=1. (b) t=10.

Figure 15. Visualization of prediction results for different lead times on the Mv_Mmfnist dataset. The last two columns represent the absolute error (| GT - Prediction |) for each model. (a) Results at a forecast time of 1 frame. (b) Results at a forecast time of 10 frame.



(a) t=1. (b) t=10.

Figure 16. Visualization of prediction results for different lead times on the Mv_Mmfnist dataset. The last two columns represent the absolute error (| GT - Prediction |) for each model. (a) Results at a forecast time of 1 frame. (b) Results at a forecast time of 10 frame.