A Optimization Algorithm of Cascaded Teaching

In this section, we develop an optimization algorithm to solve the multi-level optimization problem. We first approximate $T_1^*(A)$ using one-step gradient descent w.r.t $L\left(A, T_1, D^{(\text{trn})}\right)$:

$$T_1^*(A) \approx T_1^{'} = T_1 - \eta \nabla_{T_1} L\left(A, T_1, D^{(\text{trn})}\right) \quad . \tag{13}$$

We substitute $T_1^{'}$ into $L\left(T_2, D^{(\text{pse})}(T_1^*(A))\right)$ and get an approximated objective. Then we approximate $T_2^{'}$ using one-step gradient descent w.r.t the approximated loss as $T_2^* \approx T_2^{'} = T_2 - \eta \nabla_{T_2} L\left(T_2, D^{(\text{pse})}(T_1^{'})\right)$. For the nested function $T_k^*(T_{k-1}^*(\cdots T_1^*(A)))$, we approximate it using one-step gradient descent $T_k^* \approx T_k^{'} = T_k - \eta \nabla_{T_k} L\left(T_k, D^{(\text{pse})}(T_{k-1}^{'})\right)$. In this way, we can plug $\{T_i^{'}\}_{i=1}^K$ into the validation loss:

$$L\left(A, T_1^*(A), D^{(\text{val})}\right) + \lambda \sum_{i=2}^{K} L\left(T_i^*, D^{(\text{val})}\right) \quad .$$

We update A by gradient descend w.r.t the approximated validation loss:

$$A \leftarrow A - \eta \nabla_A \left(L(A, T_1^{'}, D^{(\text{val})}) + \lambda \sum_{i=2}^{K} L(T_i^{'}, D^{(\text{val})}) \right) \quad . \tag{14}$$

The above derivation makes it possible to generalize our approach in Section 3 to the K-learner case.

B Details analysis

To better understand why our cascaded framework works, we have to use the Lipschitz condition of the transformer-based model. As discussed in Kim et al. (2021), the scaled dot-product self-attention is Lipschitz if the input space is compact. In our case, the compact input space of the time-series dataset can be written as : $[-M, M]^{B \times L \times D}$ where M is the upper bound of the whole dataset and B, L, D represent respectively batch size, input length and hidden dimension. Most of the variants of transformer including Informer and QS-Selector are still continuously differentiable, so the Lipschitz condition applies. Assume that the Lipschitz constant of the cascaded models is K.

In the case of two learners, the teacher model and the student model can be simplified as follows:

$$\begin{aligned} F_T(T, X_i) &= Y_i^T \quad , \\ F_S(S, X_i) &= Y_i^S \quad . \end{aligned} \tag{15}$$

where F_T denotes the teacher model and F_S denotes the student model. For simplicity, we assume that the batch size is 3, γ=0 and after several iterations the models are close to convergence so that their gradients are a lot smaller than their parameters. Now we do one-step gradient descent on the teacher model in Eq.(4) and use the new parameters to generate pseudo label for a specific sample e_j in the unlabeled dataset E. This process can be written as :

$$F_T(T', e_j) = F_T(T + p_1 t_1 + p_2 t_2 + p_3 t_3, e_j) \quad , \tag{16}$$

where $t_i = -\eta_T \frac{\partial}{\partial T} l(T, X_i)$ denotes the parameters update brought by sample X_i . Since $t_i \ll 1$ we can have the following approximation:

$$F_T(T', e_j) = F_T(T, e_j) + (p_1 t_1 + p_2 t_2 + p_3 t_3) \frac{\partial}{\partial T} F_T(T, e_j) \quad . \tag{17}$$

Therefore, when we train student model on the generated pseudo-labeled dataset $D^{(\text{pse})}$, the MSE loss function can be written as :

$$
\begin{aligned}
l_j(S, e_j) =& \left(F_S(S, e_j) - F_T(T', e_j)\right)^2 \\
=& (F_S(S, e_j) - F_T(T, e_j) - \\
& (p_1 t_1 + p_2 t_2 + p_3 t_3) \frac{\partial}{\partial T} F_T(T, e_j))^2 \quad ,
\end{aligned} \tag{18}
$$

After omitting the second-order terms, we have:

$$
\begin{aligned}
& l_j(S, e_j) = l_{j0} + p_1 l_{j1} + p_2 l_{j2} + p_3 l_{j3} \quad , \\
s.t. \quad & l_{j0} = \left(F_S(S, e_j) - F_T(T, e_j)\right)^2 \\
& l_{ji} = -2 t_i \frac{\partial}{\partial T} F_T(T, e_j) \left(F_S(S, e_j) - F_T(T, e_j)\right)
\end{aligned} \tag{19}
$$

So the one-step gradient descend for student model in Eq.(5) can be rewritten as :

$$
\begin{aligned}
S' =& S - \eta_S \frac{\partial}{\partial S} l_j(S, e_j) \\
=& S + g_{j0} + p_1 g_{j1} + p_2 g_{j2} + p_3 g_{j3} \quad , \\
s.t. \quad g_{j0} =& -\eta_S \frac{\partial}{\partial S} l_{j0} \\
g_{ji} =& -\eta_S \frac{\partial}{\partial S} l_{ji}
\end{aligned} \tag{20}
$$

Now we have proven that the dataset-weights directly control the size of the corresponding update of student model parameters.

C Extra experimental results

C.1 SCINet and Reformer

In this section, we show the performance of the cascaded framework applied to Reformer Kitaev et al. (2020) and SCINet Liu et al. (2021), a CNN-based neural network for time-series forecasting. The results are summarized in table 5. As we can see, the usage of the cascaded framework helps the two models to have 13% of improvement. This proves the scalability of the cascaded framework on convolutional neural networks and that data-reweighting does not affect the sparse attention mechanism within the Transformer-based models. We also notice that the student SCINet model performs slightly better than the teacher model. Nevertheless, since the self-study rate is set to 0.2, the two models have a very similar improvement compared to the baseline model.

C.2 Dataset-weight generator

We have evaluated the Cas-Informer with different dataset-weight generator. Due to limited time, we could only test normal-distribution-based dataset-weight generator on three datasets, among which the Exchange dataset (financial) records the daily exchange rates of eight different countries ranging from 1990 to 2016. As we can see in table6, the Cas-Informer with Fourier dataset-weight generator perform better on almost all settings except the Exchange dataset. The results confirm the statement about financial datasets in section 3.2 of the paper.

C.3 Hyperparameters

The structure of the time-series forecasting model within our cascaded framework is the same as the original setting in Informer Zhou et al. (2021), Query Selector Klimek et al. (2021), Reformer Kitaev et al. (2020) and SCINet Liu et al. (2021). By default, the input length of the encoder is set to 96 and the input length of the decoder is set to 48. We use the Adam optimizer to train for 10 epochs with an initial learning rate of 0.0001 which is halved every 4 epochs. The dataset-weights are also trained by the Adam optimizer with an initial learning rate of 0.0002. Due to GPU memory limitations, we adopt a batch size of 32 and a hidden dimension of 512. Unless otherwise specified, the self-study

Table 5: Multivariate time-series forecasting results on ETT dataset with SCINet and Reformer.

Methods		Cas-SCINet				SCINet		Cas-Reformer				Reformer	
Role		Student		Teacher		Baseline		Student		Teacher		Baseline	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	0.311	0.357	**0.305**	**0.350**	0.332	0.375	**0.845**	**0.713**	0.862	0.733	0.991	0.754
	48	**0.360**	**0.392**	0.367	0.398	0.408	0.430	**1.123**	**0.835**	1.158	0.840	1.313	0.906
	168	**0.428**	**0.433**	0.434	0.436	0.471	0.468	1.703	1.082	**1.672**	**1.075**	1.824	1.138
	336	**0.706**	**0.578**	0.711	0.583	0.738	0.601	**1.916**	**1.193**	1.937	1.215	2.117	1.280
	720	**0.717**	**0.584**	0.722	0.589	0.761	0.628	2.163	1.512	**2.156**	**1.501**	2.415	1.520
$ETTh_2$	24	**0.186**	**0.289**	0.189	0.292	0.223	0.315	1.403	1.567	**1.387**	**1.549**	1.531	1.613
	48	**0.287**	**0.354**	0.292	0.358	0.563	0.546	**1.650**	**1.687**	1.678	1.707	1.871	1.735
	168	**0.502**	**0.489**	0.519	0.494	0.586	0.543	**4.254**	**1.513**	4.289	1.545	4.660	1.846
	336	**0.604**	**0.541**	0.621	0.558	0.702	0.603	**3.271**	**1.245**	3.338	1.274	4.028	1.688
	720	1.384	0.775	**1.376**	**0.768**	1.493	0.879	**4.946**	**1.827**	4.991	1.882	5.381	2.015
$ETTm_1$	24	**0.238**	**0.301**	0.242	0.302	0.277	0.329	0.668	0.591	**0.652**	**0.576**	0.724	0.607
	48	0.384	0.403	**0.380**	**0.399**	0.414	0.416	**0.923**	**0.618**	0.937	0.623	1.098	0.777
	96	**0.336**	**0.355**	0.343	0.366	0.375	0.401	**1.364**	**0.882**	1.371	0.890	1.433	0.945
	288	**0.502**	**0.526**	0.513	0.535	0.582	0.556	**1.615**	**0.965**	1.648	0.982	1.820	1.094
	672	**0.735**	**0.658**	0.757	0.675	0.892	0.727	1.904	1.198	**1.838**	**1.172**	2.187	1.232
Count		24		6		0		20		10		0	

Table 6: Experimental results on dataset-weight generator.

Methods		Cas-Informer(fourier)		Cas-Informer(normal)		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	**0.473**	**0.492**	0.526	0.524	0.577	0.549
	48	**0.620**	**0.586**	0.664	0.609	0.685	0.625
	168	**0.877**	**0.714**	0.912	0.735	0.931	0.752
	336	**0.957**	**0.725**	1.077	0.821	1.128	0.873
	720	**1.035**	**0.805**	1.133	0.857	1.215	0.896
$ETTh_2$	24	**0.633**	**0.638**	0.735	0.673	0.720	0.665
	48	**1.146**	**0.875**	1.418	0.988	1.457	1.001
	168	**1.915**	**1.046**	2.732	1.348	3.489	1.515
	336	**2.318**	**1.167**	2.611	1.290	2.723	1.340
	720	**2.479**	**1.336**	3.124	1.422	3.467	1.473
Exchange	96	0.834	0.747	**0.816**	**0.741**	0.856	0.758
	192	1.009	0.802	**0.975**	**0.788**	1.221	0.905
	336	1.613	0.995	**1.582**	**0.976**	1.633	1.014
	720	**2.275**	**1.302**	2.298	1.312	2.496	1.352
Count		22		6		0	

rate $\gamma = 0.2$ and the sigmoid temperature $\mathcal{T} = 5$. The early stop algorithm is also adopted to prevent the model from overfitting. We have implemented the cascaded framework in Python 3.8 with Pytorch 1.10 so that the recently released distributed data parallel package can be used.