About the algorithm part:

Meta-learning is a learning to learn approach that enables the learning model to adapt to new tasks by leveraging previous experience from related tasks. In this framework, tasks are drawn from a specific distribution, denoted as $\tau\sim\mathcal{P}(\tau)$, and each task includes a support set for training and a query set for test. In an $N$-way $k$-shot classification problem, a task consists of $N$ classes, each with $k$ samples. In the meta-training stage, $M$ training tasks, $\{\tau\_{i}\}\_{i=1}^{M}\sim\mathcal{P}(\tau)$, are sampled from the distribution and the corresponding datasets are made available to the model. In the meta-testing stage, a new test task $T\sim\mathcal{P}(\tau)$ is presented, consisting of a small support set and a query set. The objective of meta-learning is to train a model on the $M$ training tasks, such that it can quickly adapt to the new test task using the small support set and perform well on the query set.

Model-agnostic meta-learning (MAML) does so by learning a set of initial parameters $\bm{\theta}\_{MAML}$ for neural networks that enable good performance on a new task with only a few gradient descent steps. In the meta-training stage, MAML formulates a meta-optimization problem to find $\bm{\theta}\_{MAML}$ as:

\begin{equation} \bm{\theta}\_{MAML}=\mathop{\arg\min}\limits\_{\bm{\theta}}\sum\_{i=1}^{M}\mathcal{L}\_{\tau\_{i}}\left(\bm{\theta}-\alpha\nabla\_{\bm{\theta}}\hat{\mathcal{L}}\_{\tau\_{i}}(\bm{\theta})\right),

\end{equation}

where it contains two task-specific loss functions $\hat{\mathcal{L}}\_{\tau\_{i}}$ and

$\mathcal{L}\_{\tau\_{i}}$ computed based on the support set and query set of the training task $\tau\_{i}$, respectively. Then the meta-parameters are updated via stochastic gradient descent (SGD):

\begin{equation}

\bm{\theta}\_{MAML}\gets\bm{\theta}\_{MAML}-\beta\nabla\_{\bm{\theta}}\sum\_{i=1}^{M}\mathcal{L}\_{\tau\_{i}}\left(\bm{\theta}-\alpha\nabla\_{\bm{\theta}}\hat{\mathcal{L}}\_{\tau\_{i}}(\bm{\theta})\right),

\label{eq:outer\_loop}

\end{equation}

where $\alpha$ and $\beta$ denote the step size of the inner loop and outer loop, respectively. During the meta-test stage, the meta-parameters $\bm{\theta}\_{MAML}$ are fine-tuned to obtain the parameters $\bm{\theta}\_{T}$ for the neural network used in the test task $T$. This is achieved by updating the meta-parameters using the gradient of the loss function $\hat{\mathcal{L}\_{T}}\left(\bm{\theta}\_{MAML}\right)$ computed based on the support set of the test task, as follows:

\begin{equation}\bm{\theta}\_{T}\gets\bm{\theta}\_{MAML}-\alpha\nabla\_{\bm{\theta}}\hat{\mathcal{L}\_{T}}\left(\bm{\theta}\_{MAML}\right),\end{equation}

\begin{algorithm}[ht]

\caption{Vanilla MAML}\label{alg:alg1}

\begin{algorithmic}

\STATE \textbf{Require:} \\$\mathcal{P}(\tau)$: distribution over tasks;

\\$\alpha$: step size of the inner loop; \\$\beta$: step size of the outer loop;

\STATE \textbf{Meta-training Stage (in the historical environments):}

\STATE 1: Randomly initialize $\bm{\bm{\theta}}$;

\STATE 2: For $ite$ in iterations do:

\STATE 3: \hspace{0.5cm}Sample training tasks $\{\tau\_{i}\}\_{i=1}^{M}\sim\mathcal{P}(\tau) $;

\STATE 4: \hspace{0.5cm}For each $i$ in $\{1,2,\dots,M\}$ do:

\STATE 5: \hspace{1.0cm}$ \bm{\bm{\theta}}\_{i}^{\prime}=\bm{\bm{\theta}} - \alpha\nabla\_{\bm{\bm{\theta}}}\mathcal{L}\_{\tau\_{i}}(f\_{\bm{\theta}};D\_{\tau\_{i}}^{s})$;

\STATE 6: \hspace{0.5cm}$ \bm{\theta}\gets\bm{\theta}-\beta\nabla\_{\bm{\theta}}\sum\_{\tau\_{i}}\mathcal{L}\_{\tau\_{i}}(f\_{\bm{\theta}\_{i}^{\prime}};D\_{\tau\_{i}}^{q})$;

\STATE 7: \textbf{return} $\bm{\theta}^\*\gets\bm{\theta}$ when it converges.

\STATE \textbf{Meta-test Stage (in the new environment):}

\STATE 8: Sample a test task $T\sim\mathcal{P}(\tau) $;

\STATE 9: $\bm{\theta}\_{T}\gets\bm{\theta}^\*-\alpha\nabla\_{\bm{\theta}}\mathcal{L}\_{T}(f\_{\bm{\theta}^\*};D\_{T}^{s})$;

\STATE 10: \textbf{return} $\bm{\theta}\_{T}^\*\gets\bm{\theta}\_T$ when it converges.

\end{algorithmic}

\end{algorithm}

For the vanilla MAML algorithm, we only have a theoretical guarantee (an upper bound) that the algorithm will converge on different domains, shown as follows:

\noindent\textbf{Theorem 1.} \textit{Suppose $\mathcal{L}\_{T}(f\_{\bm{\theta}},D\_{T}^s)$ is G-Lipschitz continuous and $W$-smooth with respect to the parameters $\bm{\theta}$, and $\alpha$ satisfies $\alpha\leq\frac{1}{W}$. Setting $\rho = 1+2\alpha W$, then for any $T\sim \mathcal{P}(\tau)$ with $D\_T^{s}=\left\{\left(x\_i, y\_i\right)\right\}\_{i=1}^{k\_{spt}} \sim T$, we have}

\begin{small}\begin{align\*}

ER(\bm{\theta}\_{T}^{Q}) &\leq \frac{2G^{2}(\rho^{Q}-1)}{k\_{spt}\*W}+E\_{T\sim\tau}E\_{D\_{T}^s}[\mathcal{L}\_{T}(\bm{\theta}\_{T}^{Q};D\_{T}^s)-\mathcal{L}\_{T}(\bm{\theta}\_{T}^\*)] \\

&\leq \frac{2G^{2}(\rho^{Q}-1)}{k\_{spt}\*W}+\frac{1}{2\alpha}E\_{T\sim\tau}[||\bm{\theta}^\*-\bm{\theta}\_{T}^\*||\_{2}^{2}].

\end{align\*}\end{small}

However, since our research is based on Widar dataset, which contains different domains, we address this issue by proposing model-agnostic meta-

learning with domain generalization (MAML-DG), Unlike the original MAML approach which trains environment-specific meta-parameters in isolation for each environment, MAML-DG allows the environments to share the environment-specific meta-parameters, while still protecting the raw data of each environment.

The concrete algorithm of MAML-DG is shown below:

\begin{algorithm}

\caption{MAML-DG}\label{alg:alg2}

\begin{algorithmic}

\STATE \textbf{Require:} \\$\{\mathcal{P}^{(i)}(\tau)\}\_{i=1}^{S}$: distributions over tasks in $S$ domains; \\$\alpha$: step size of the inner loop; \\$\beta$: step size of the outer loop; \\$w$: weight of the loss function of the second training domain $D\_{II}$;

\STATE \textbf{Meta-training Stage (in the historical environments):}

\STATE 1: Randomly initialize $\bm{\theta}$;

\STATE 2: For $ite$ in iterations do:

\STATE 3: \hspace{0.5cm}Sample two training domains $D\_{I}$ and $D\_{II}$ uniformly from $\{1,2,...,S\}$;

\STATE 4: \hspace{0.5cm}Sample tasks $\{\tau\_{i}^{(D\_{I})}\}\_{i=1}^{M}\sim\mathcal{P}^{(D\_{I})}(\tau) $ in domain $D\_{I}$;

\STATE 5: \hspace{0.5cm}For $i$ in range ($M$) do:

\STATE 6: \hspace{1.0cm}$ \bm{\theta}\_{i}^{(D\_I)}=\bm{\theta} - \alpha\nabla\_{\bm{\theta}}\mathcal{L}\_{\tau\_{i}^{(D\_I)}}(f\_{\bm{\theta}};D\_{\tau\_{i}^{(I)}}^{s})$;

\STATE 7: \hspace{0.5cm}$ \bm{\theta}^{\prime}=\bm{\theta}-\beta\nabla\_{\bm{\theta}}\sum\_{\tau\_{i}^{(D\_I)}}\mathcal{L}\_{\tau\_{i}^{(D\_I)}}(f\_{\bm{\theta}\_{i}^{(D\_I)}};D\_{\tau\_{i}^{(D\_I)}}^{q}) $;

\STATE 8: \hspace{0.5cm}Sample tasks $\{\tau\_{j}^{(D\_{II})}\}\_{j=1}^{M}\sim\mathcal{P}^{(D\_{II})}(\tau)$ in $D\_{II}$;

\STATE 9: \hspace{0.5cm}For $j$ in range ($M$) do:

\STATE 10: \hspace{1.0cm}$ \bm{\theta}\_{j}^{(D\_{II})}=\bm{\theta} - \alpha\nabla\_{\bm{\theta}}\mathcal{L}\_{\tau\_{j}^{(D\_{II})}}(f\_{\bm{\theta}^{\prime}};D\_{\tau\_{j}^{(D\_{II})}}^{s})$;

\STATE 11:\hspace{0.5cm}$\bm{\theta}\gets\bm{\theta}^{\prime}-w\beta\nabla\_{\bm{\theta}}\sum\_{\tau\_{j}^{(D\_{II})}}\mathcal{L}\_{\tau\_{j}^{(D\_{II})}}(f\_{\bm{\theta}\_{j}^{(D\_{II})}};D\_{\tau\_{j}^{(D\_{II})}}^{q}) $;

\STATE 12: \textbf{return} $\bm{\theta}^\*\gets\bm{\theta}$ when it converges.

\STATE \textbf{Meta-test Stage (in the new environment):}

\STATE 13: Sample a test task $T\sim\mathcal{P}(\tau) $;

\STATE 14: $\bm{\theta}\_{T}\gets\bm{\theta}^\*-\alpha\nabla\_{\bm{\theta}}\mathcal{L}\_{T}(f\_{\bm{\theta}^\*

};D\_{T}^{s})$;

\STATE 15: \textbf{return} $\bm{\theta}\_{T}^\*\gets\bm{\theta}\_T$ when it converges.

\end{algorithmic}

\end{algorithm}

As outlined in Algorithm~\ref{alg:alg2}, MAML-DG is designed to train a deep learning model with parameters $\bm{\theta}$ across $S$ training domains, which may have different statistical distributions but share the same label and input features space. During each meta-training iteration, MAML-DG randomly selects two training domains $D\_{I}, D\_{I}=1,2,\dots,S$ and $D\_{II}, D\_{II}=1,2,\dots,S, D\_{I}\neq D\_{II}$ and generates tasks in these two domains. The full steps are as follows.

\textbf{Step \ding{172}}: We virtually train a domain-specific meta-parameters $\bm{\theta}^{\prime}$ on the tasks generated from the training domain $D\_{I}$ using the vanilla MAML algorithm. We derive the first domain-specific loss function as

\begin{equation}

\small

F(\cdot)=\sum\_{\tau\_{i}^{(D\_{I})}}\mathcal{L}\_{\tau\_{i}^{(D\_{I})}}\left(f\_{\bm{\theta}\_{i}^{(D\_{I})}}\right)=\sum\_{i=1}^{M}\mathcal{L}\_{\tau\_{i}^{(D\_{I})}}\left(f\_{\bm{\theta}-\alpha\nabla\_{\bm{\theta}}\mathcal{L}\_{\tau\_{i}^{(D\_{I})}}}\right).

\end{equation}

\textbf{Step \ding{173}}: With the initialization $\bm{\theta}^{\prime}$ obtained in the previous step, we derive a second domain-specific loss function for the tasks generated from the training domain $D\_{II}$ using vanilla MAML once again, which is shown as \begin{equation}

\small

G(\cdot)=\sum\_{\tau\_{j}^{(D\_{II})}}\mathcal{L}\_{\tau\_{j}^{(D\_{II})}}(f\_{\bm{\theta}\_{j}^{(D\_{II})}})=\sum\_{j=1}^{M}\mathcal{L}\_{\tau\_{j}^{(D\_{II})}}\left(f\_{\bm{\theta}^{\prime}-\alpha\nabla\_{\bm{\theta}^{\prime}}\mathcal{L}\_{\tau\_{j}^{(D\_{II})}}}\right).

\end{equation}

\textbf{Step \ding{174}}: We sum up the two domain-specific losses $F(\cdot)$ and $G(\cdot)$, and then update the meta-parameters $\bm{\theta}$, which emulates the real-time train-test domain shifts and helps the model generalize faster after a few iterations.

Since we add several different domains into the meta-learning stage, and we do fine-tuning on the testing dataset, MAML-DG can alleviate overfitting and achieve good generalization across different training domains compared to the vanilla MAML.