Pin the Memory: Learning to Generalize Semantic Segmentation

Jin Kim[1] Jiyoung Lee[2] Jungin Park[1] Dongbo Min[3*] Kwanghoon Sohn[1*]
[1]Yonsei University [2]NAVER AI Lab [3]Ewha Womans University
{kimjin928, newrun, khsohn}@yonsei.ac.kr lee.j@navercorp.com dbmin@ewha.ac.kr

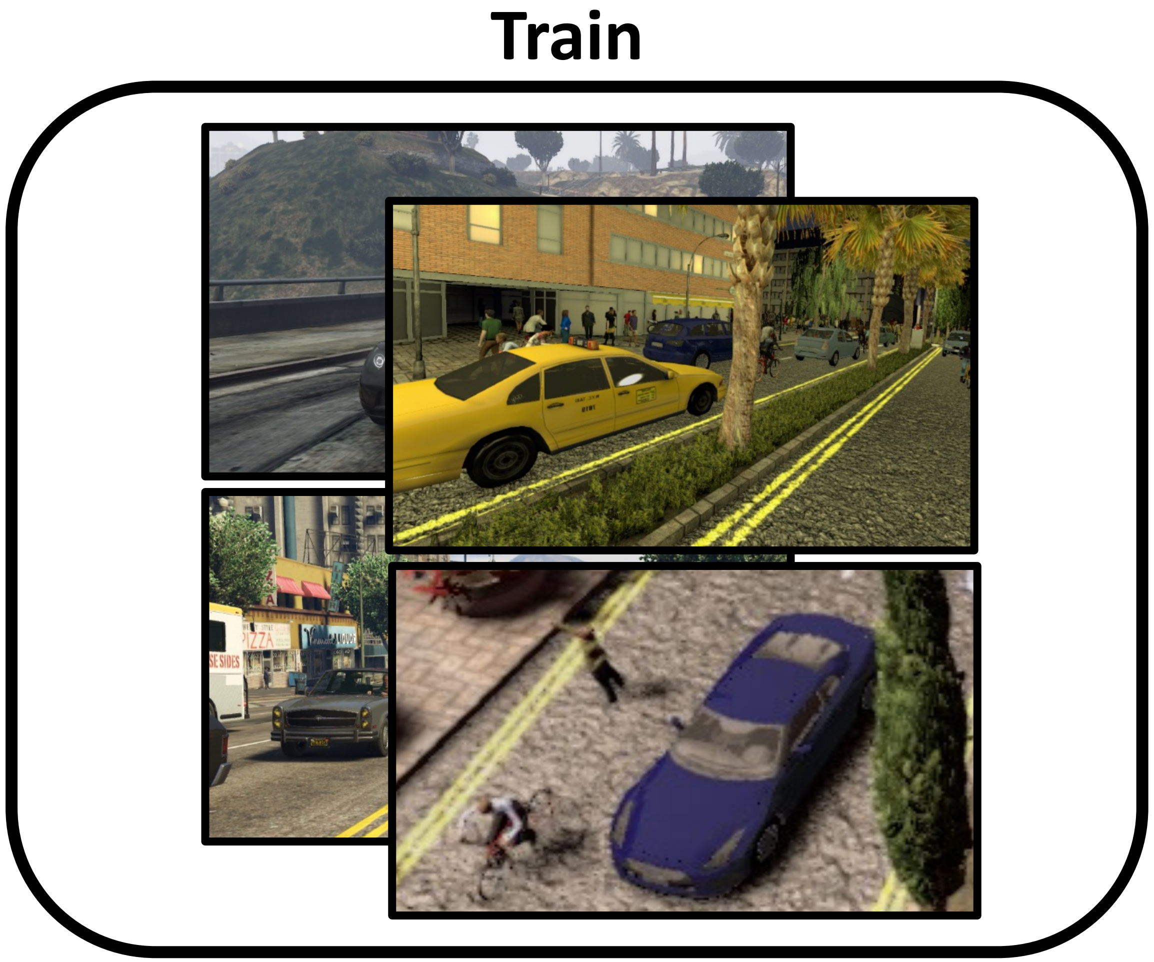
Yonsei University
School of electrical & Electronic Engineering
Digital Image Media Lab

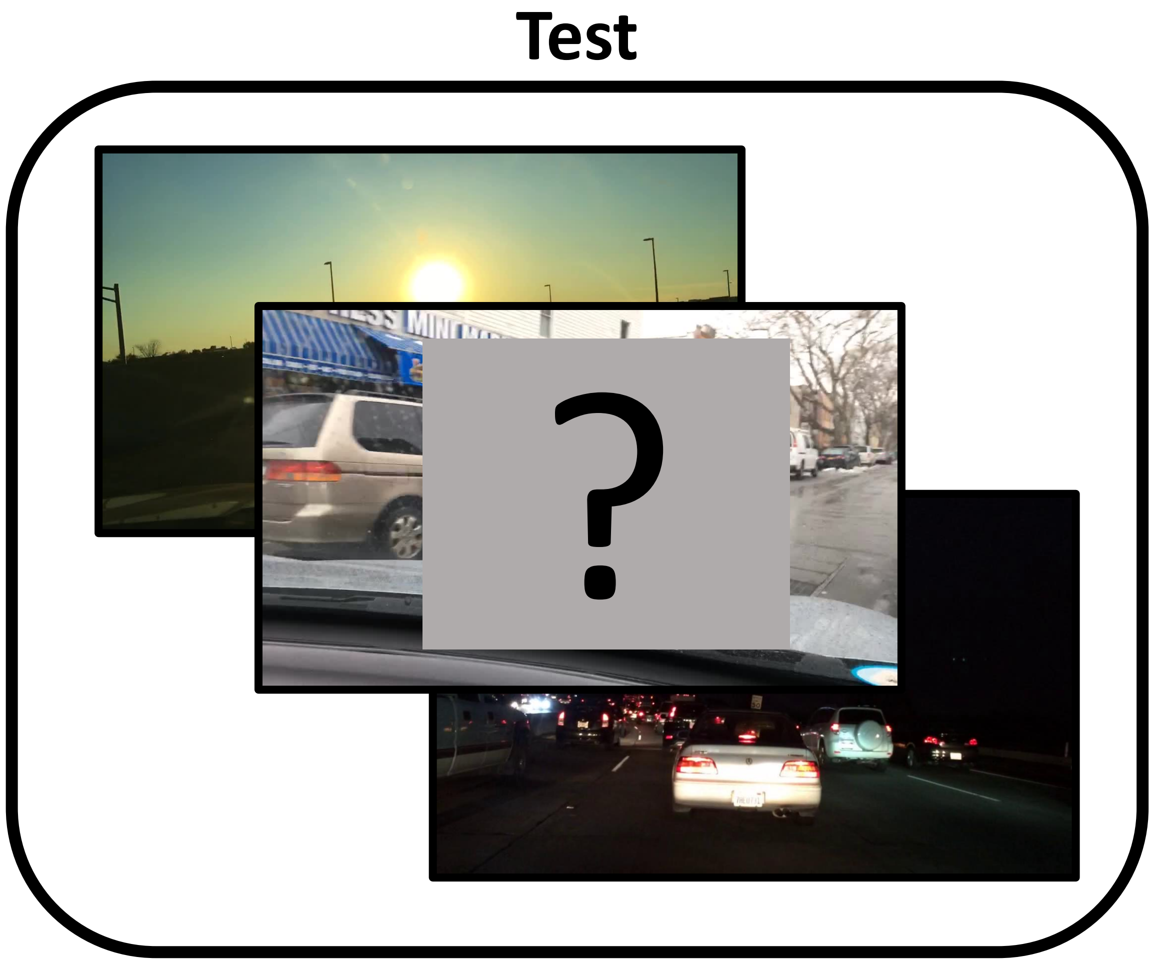
Jin Kim

Hello everyone. My name is Jin Kim from Yonsei University. Today, I'm going to talk about 'Pin the Memory: Learning to Generalize Semantic Segmentation.'

INTRODUCTION

DOMAIN GENERALIZATION

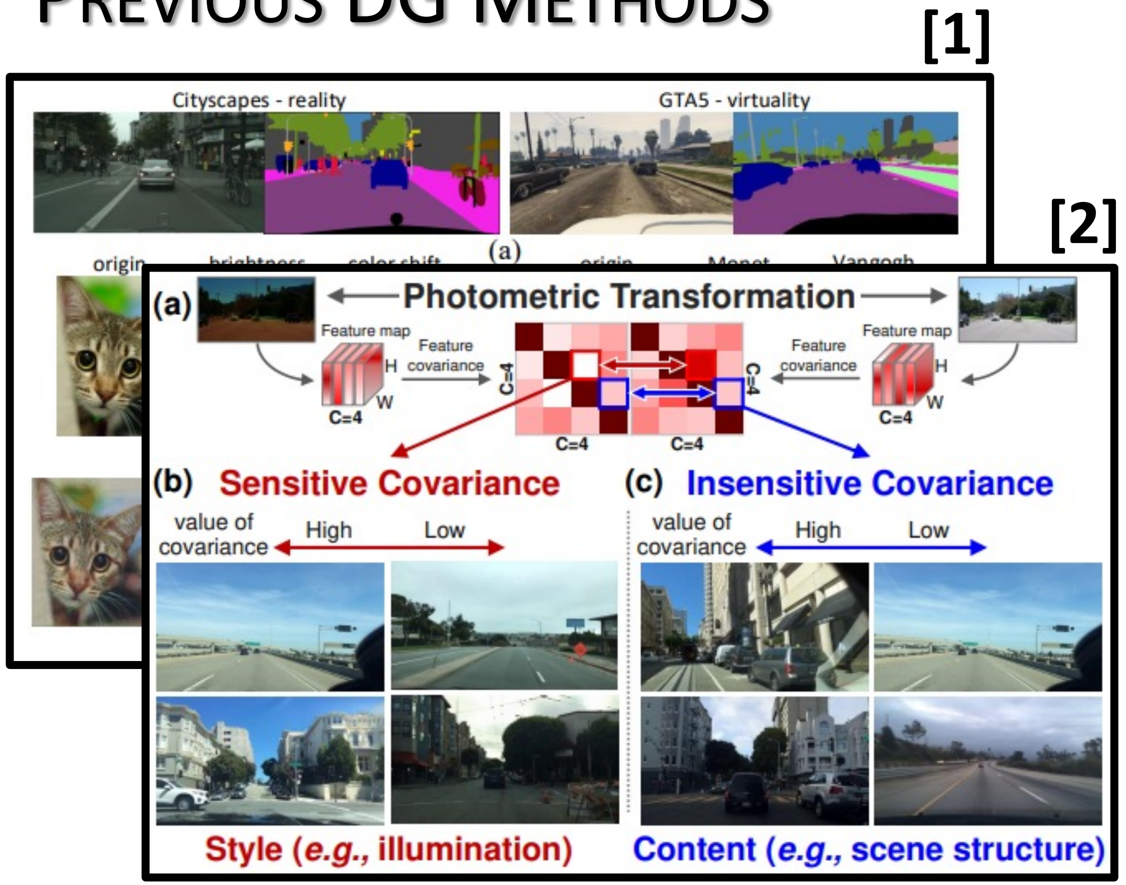


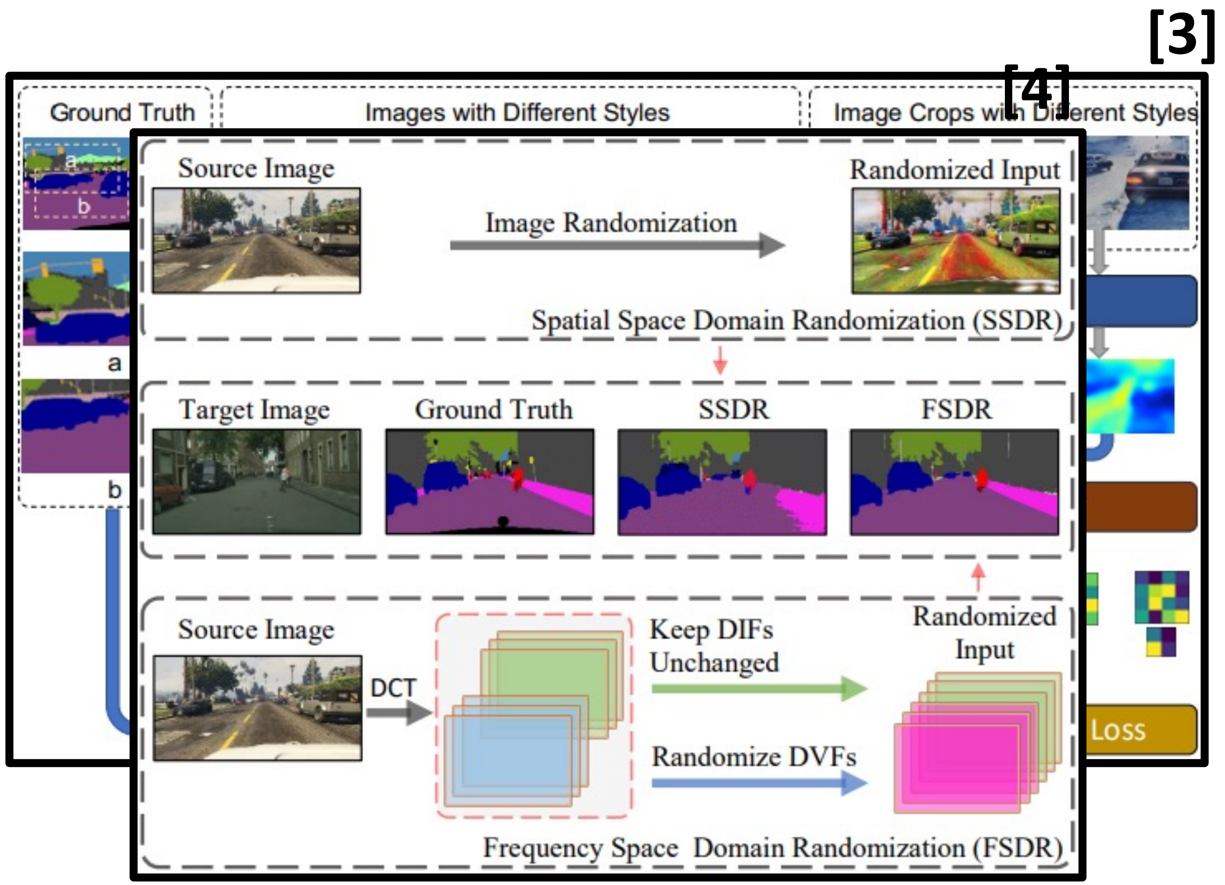


Most approaches are trained under the assumption that training and testing data are sampled from the same distribution.
However they often fail to infer accurate predictions for arbitrary target domain dataset because of the domain shift.

INTRODUCTION

PREVIOUS DG METHODS





Normalization Based

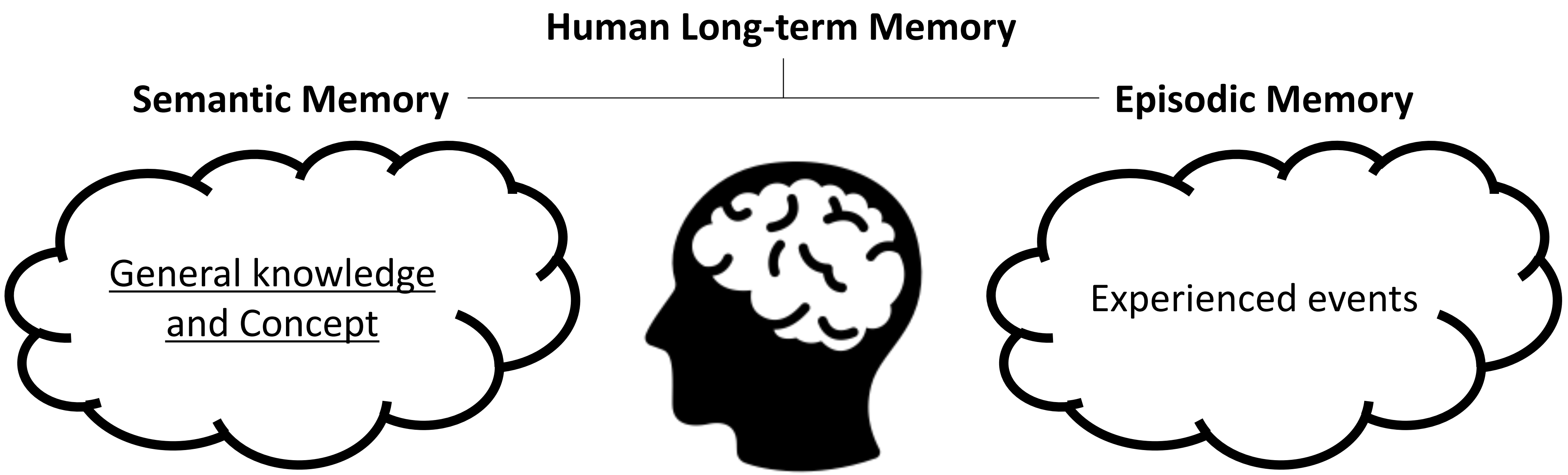
Randomization Based

[1] Pan et al. "Two at once: Enhancing learning and generalization capacities via ibn-net." ECCV'18.
[2] Choi et al. "Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening." CVPR'21.
[3] Yue et al. "Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data." *CVPR'*19.
[4] Huang et al. "Fsdr: Frequency space domain randomization for domain generalization." *CVPR'21*.

To overcome this problem, Some DG methods heuristically define domain-biased information as style and erase. Other methods explicitly augment the style information.

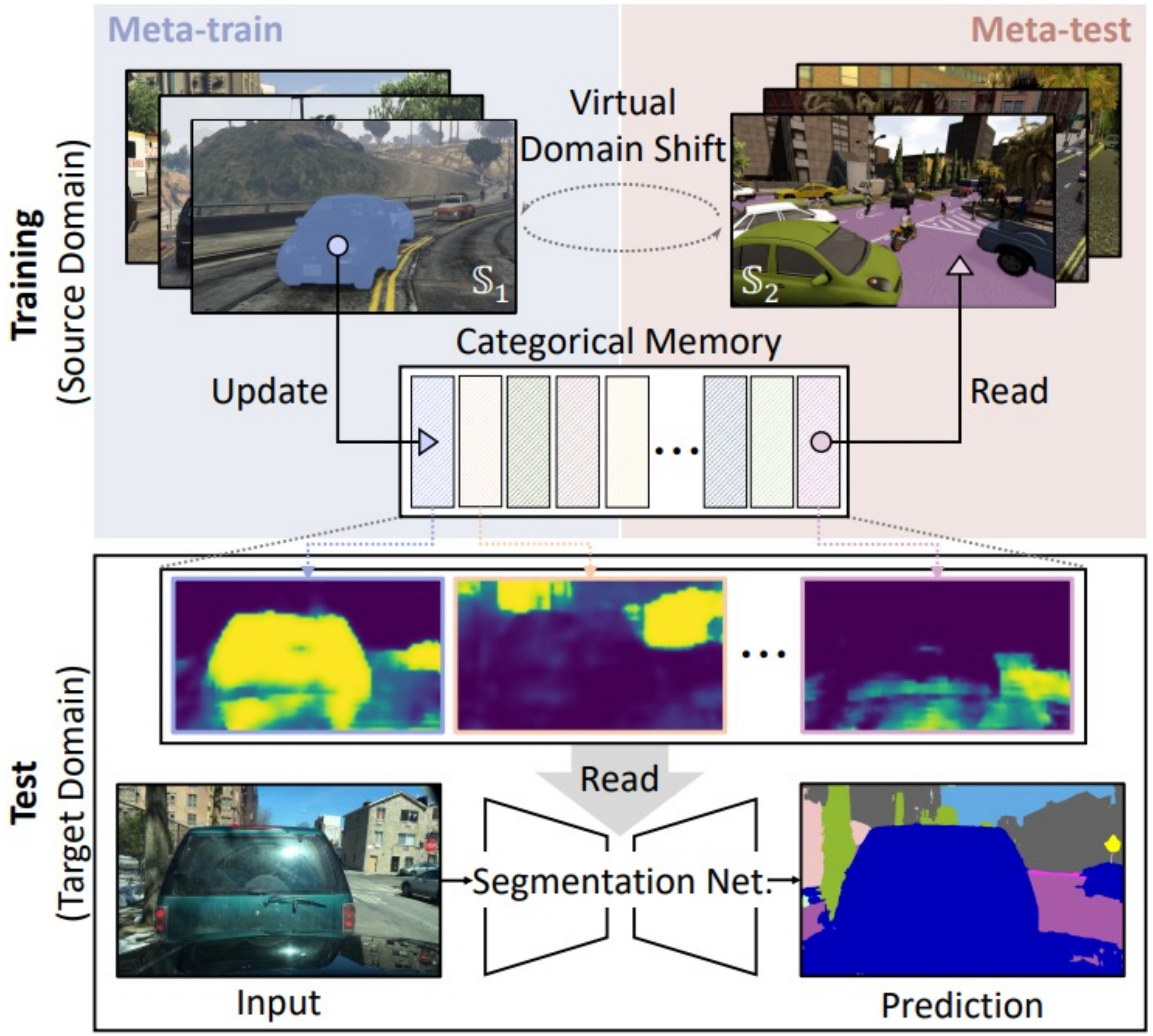
INTRODUCTION

SEMANTIC MEMORY



Despite their efforts, the existing method is still different from the human method that leverage semantic memory.

INTRODUCTION

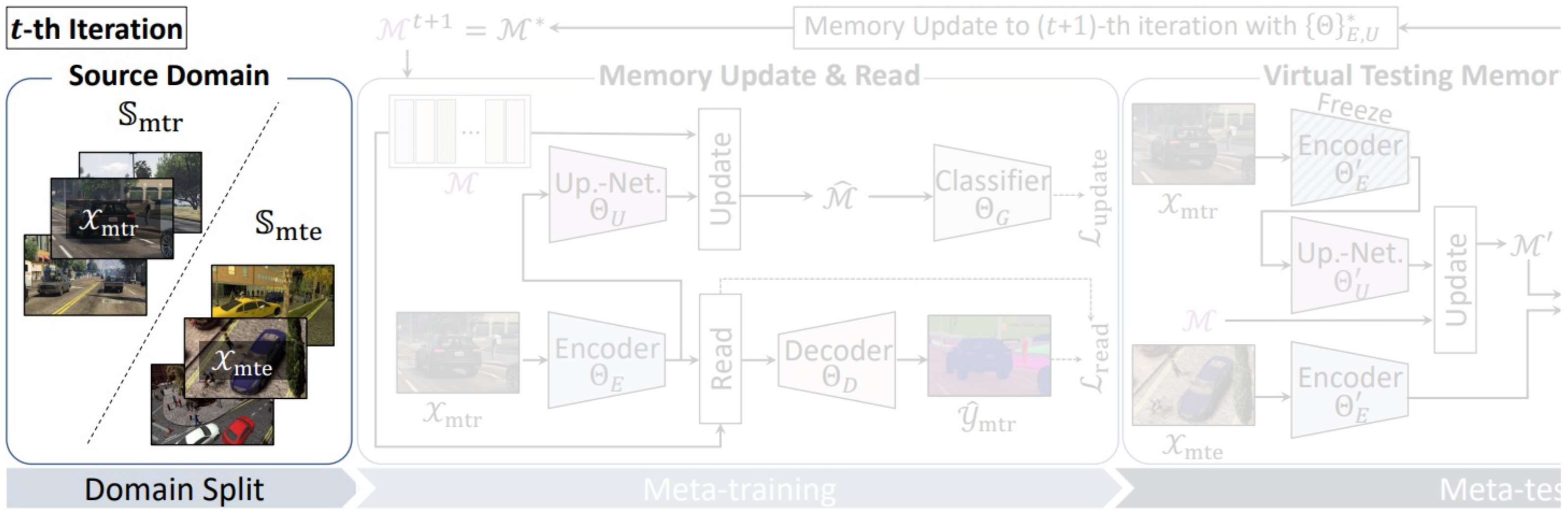


- We propose a novel **memory-guided meta-learning framework** to capture and memorize **co-occurrent categorical knowledge** across domains.
- To this end, we **split source domain datasets** into meta-training and meta-testing sets to **explicitly mimic domain shift**, allowing the network to store and invoke memory corresponding to **domain-agnostic prototypes of class patterns**.
- We introduce a **memory divergence loss** and a **feature cohesion loss** which boost discriminative power of memory and make more domain-invariant representations, respectively.

In this work, we propose a novel memory-guided meta-learning framework to capture and memorize co-occurrent categorical knowledge across domains.

METHOD

OVERALL PROCEDURE



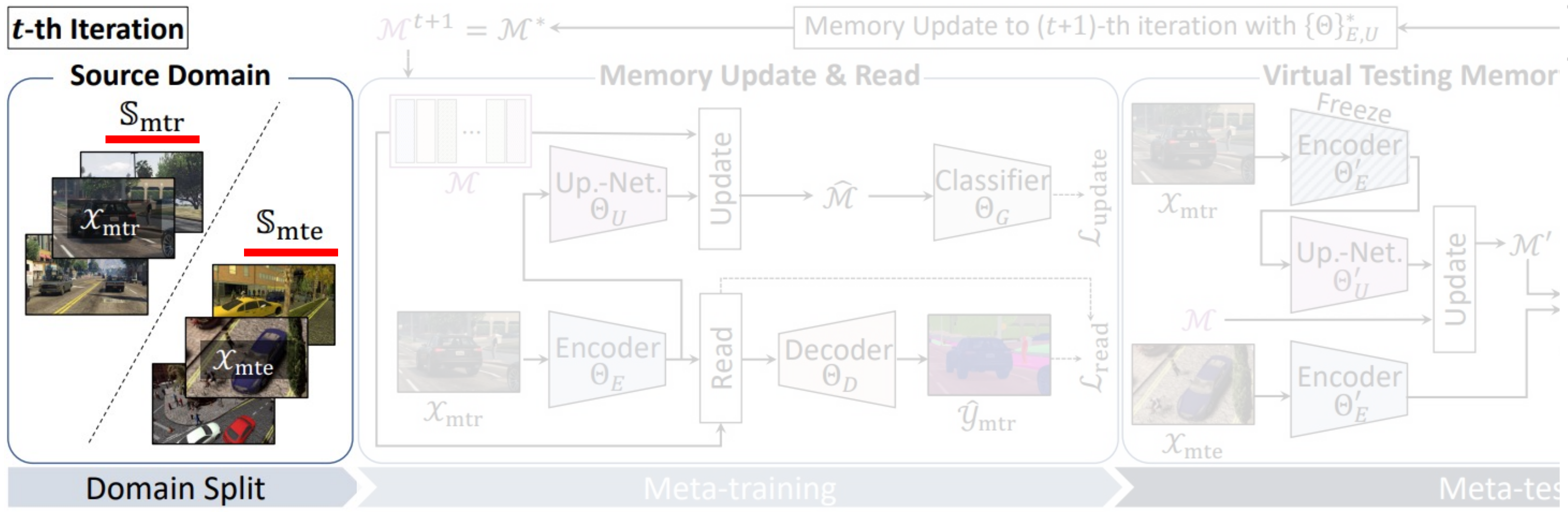
Algorithm 1: Overall Training Procedure

```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{mtr}$ and $\mathbb{S}_{mte}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{mtr} = \{\mathcal{X}^b_{mtr}\}^B_{b=1}$ from $\mathbb{S}_{mtr}$
        Compute $\mathcal{L}_{read}$ with $(\mathcal{X}^t_{mtr}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{mtr}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{update}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{mtr}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{mte} = \{\mathcal{X}^b_{mte}\}^B_{b=1}$ from $\mathbb{S}_{mte}$
        Compute $\mathcal{L}_{read}$ with $(\mathcal{X}^t_{mte}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{mtr}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

The overall training procedure is as follows. As preliminary step, we initialize our memory module with mean feature vector for each class in the source datasets.

Method

Overall Procedure



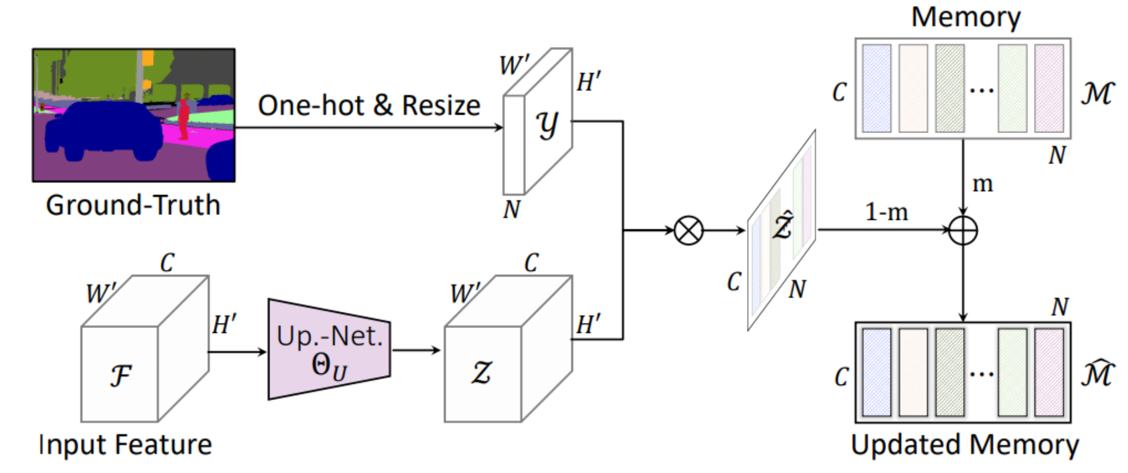
t-th Iteration
Source Domain
$\mathbb{S}_{\text{mtr}}$
$\mathcal{X}_{\text{mtr}}$
$\mathbb{S}_{\text{mte}}$
$\mathcal{X}_{\text{mte}}$
Domain Split
$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
$\mathcal{X}_{\text{mtr}}$
Encoder Θ_E
Read
Decoder Θ_D
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Meta-training
Virtual Testing Memor
Freeze
Encoder Θ'_E
$\mathcal{X}_{\text{mtr}}$
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
$\mathcal{M}$
$\mathcal{X}_{\text{mte}}$
Encoder Θ'_E
Meta-tes

```
Algorithm 1: Overall Training Procedure
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

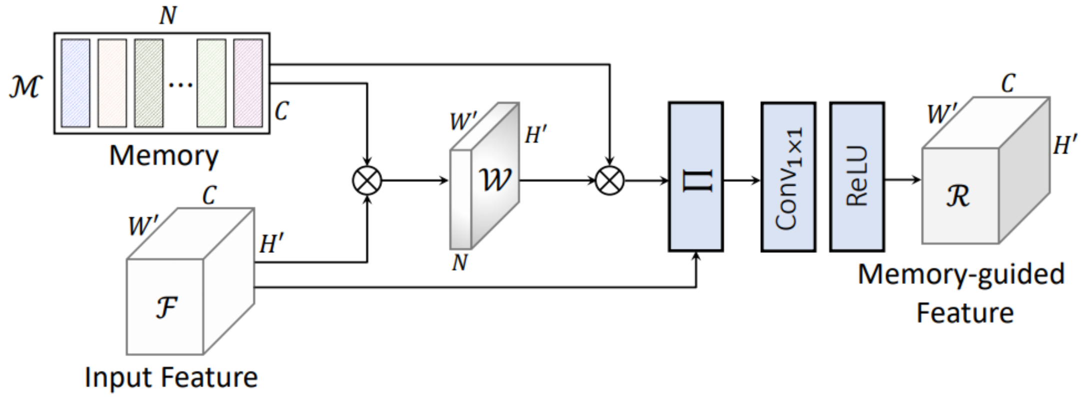
Then, we randomly split the available source domains S into meta-train domains Smtr and meta-test domains Smte at every iteration step. Note that the meta-train mimic the training, and the meta-test mimic the inference.

METHOD

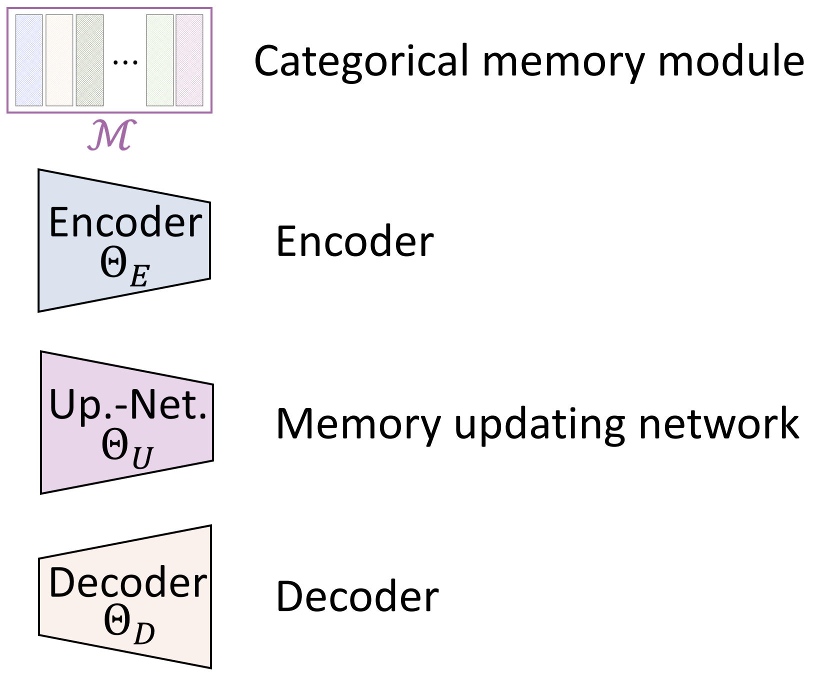
OVERALL PROCEDURE



(a) Memory Update



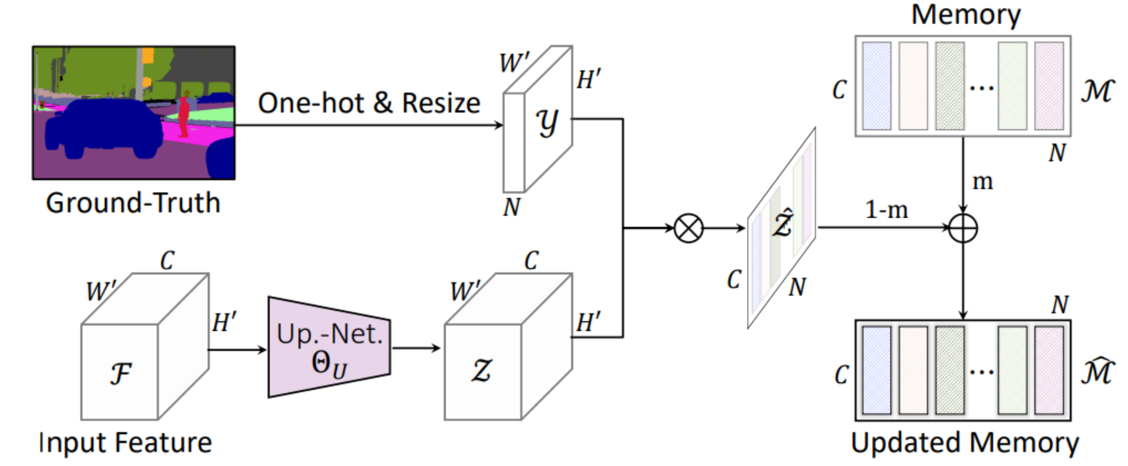
(b) Memory Read



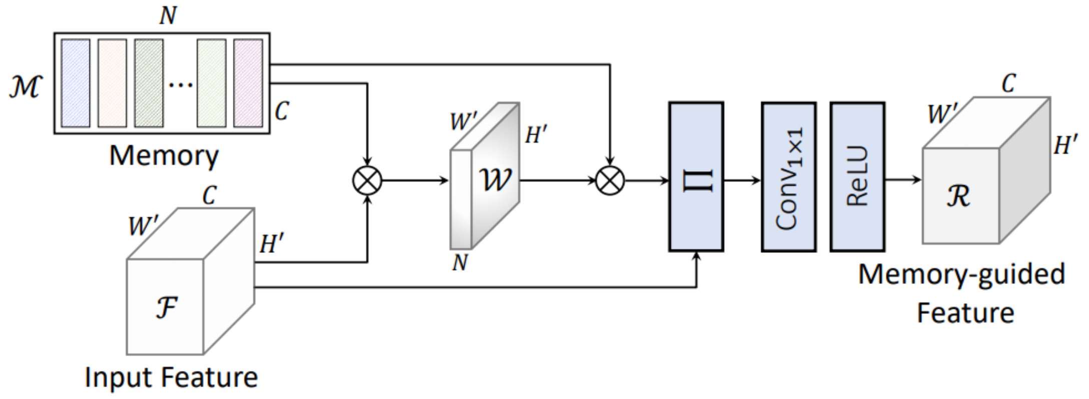
Our memory stores domain-agnostic prototype vectors for each class. Memory update and read operate on the feature map from the encoder.

Method

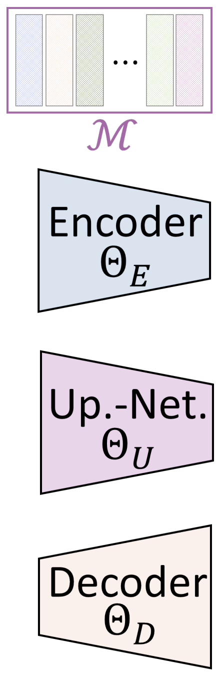
Overall Procedure



(a) Memory Update



(b) Memory Read

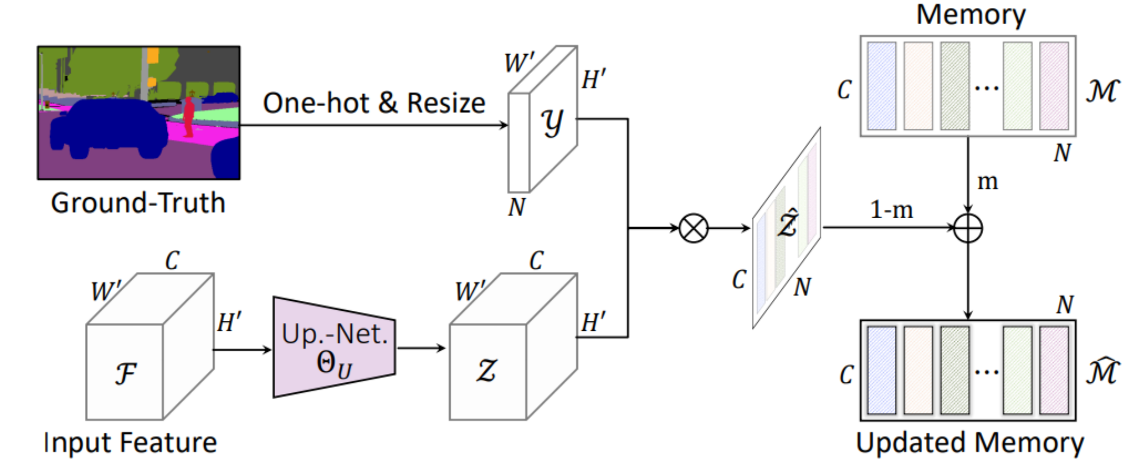


	Categorical memory module
	Encoder
	Memory updating network
	Decoder

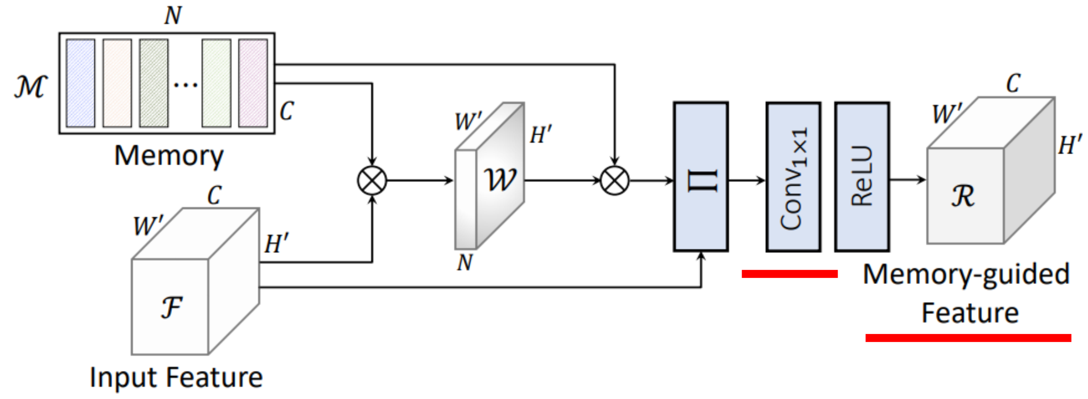
We update the memory by performing an masked average pooling for each class.
We read the memory by computing a memory weight matrix via cosine similarity and normalize it with the gumbel softmax function.

METHOD

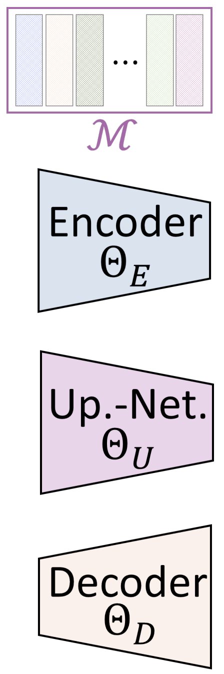
OVERALL PROCEDURE



(a) Memory Update



(b) Memory Read

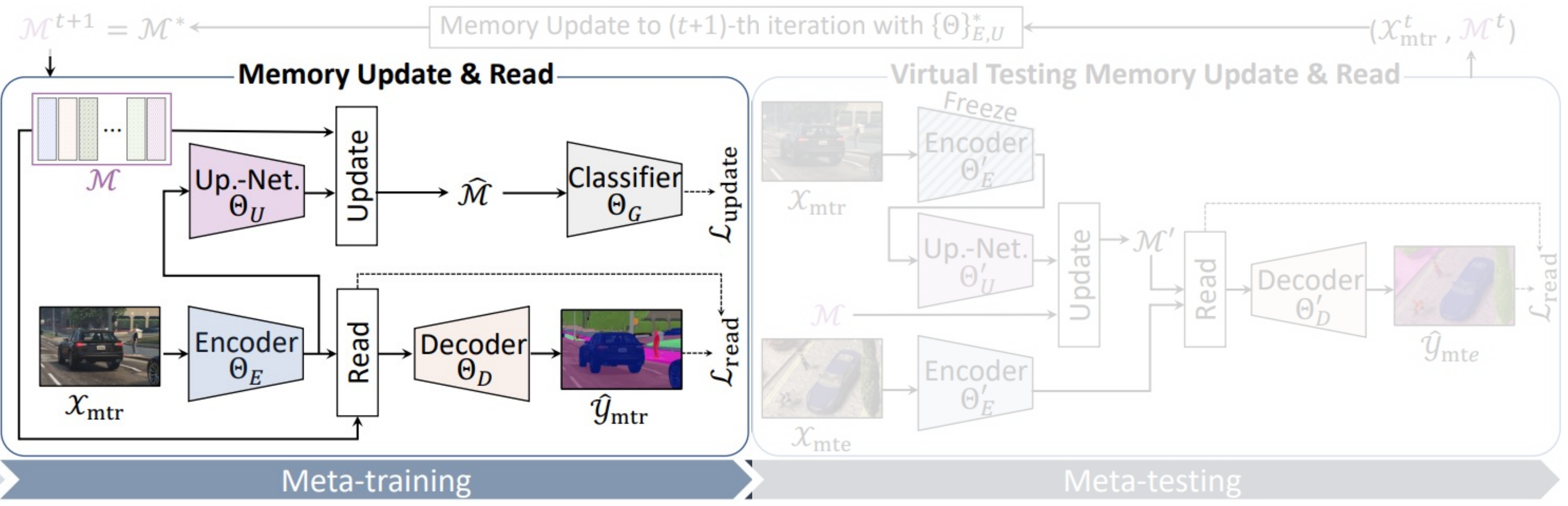


▭	Categorical memory module
▭	Encoder
▭	Memory updating network
▭	Decoder

Then, the memory-guided feature map R is obtained by fusing the original feature map F and weighted memory feature.
The 1 by 1 convolution is to match the channel dimension.

Method

Overall Procedure


$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
$\mathcal{X}_{\text{mtr}}$
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{M}$
Encoder Θ'_E
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

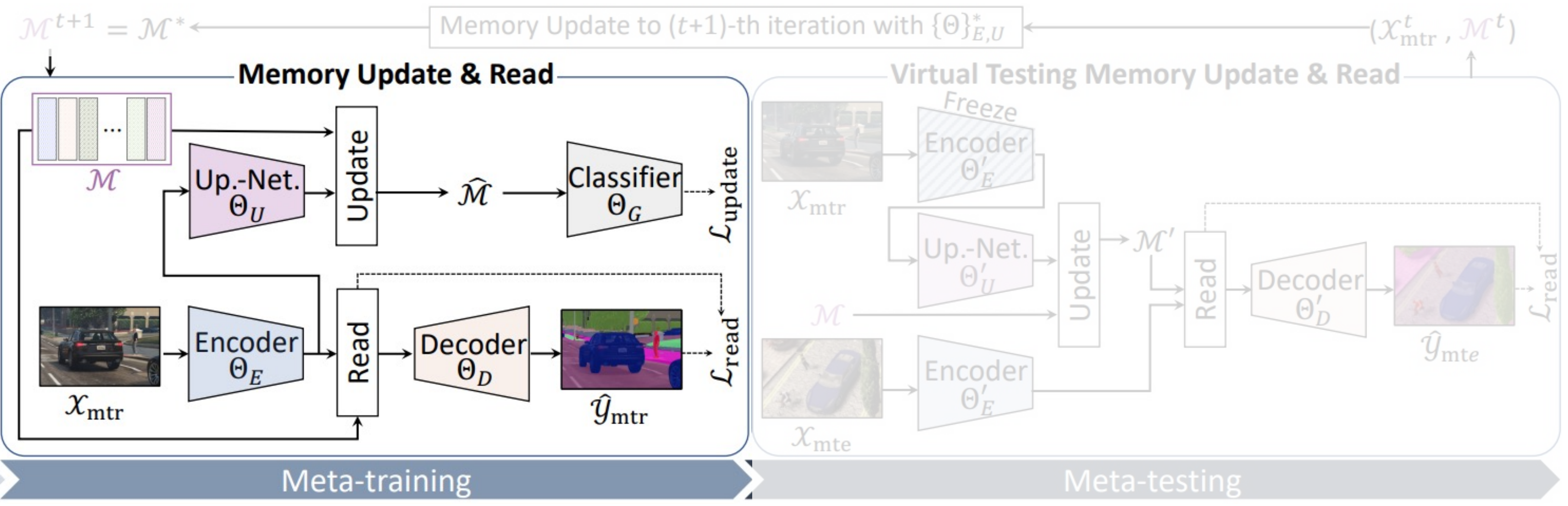
Algorithm 1: Overall Training Procedure

```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

Every iteration consists of a meta training step and a meta testing step. The meta training steps are as follows.

Method

Overall Procedure



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{M}$
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

Read Loss

$$\mathcal{L}_{\text{read}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}\,; \{\Theta\}_{E,D}) = \mathcal{L}_{\text{seg}} + \lambda_1 \underline{\mathcal{L}_{\text{coh}}},$$

$$\underline{\mathcal{L}_{\text{coh}}} = \frac{1}{H'W'} \sum_{j=1}^{H'W'} -\mathcal{Y}_{\text{mtr}}^{\top}[j] \log(\mathcal{W}_{\text{mtr}}[j])$$

Memory update loss

$$\mathcal{L}_{\text{update}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}\,; \{\Theta\}_{E,U,G}) = \lambda_2 \underline{\mathcal{L}_{\text{div}}},$$

$$\underline{\mathcal{L}_{\text{div}}} = \sum_{n=1}^{N} (-\mathcal{I}[n] \log(G(\hat{\mathcal{M}}[n]^{\top})) + 2 \cdot \sum_{n' \neq n}^{N} \frac{\max(\hat{\mathcal{M}}[n]\hat{\mathcal{M}}[n']^{\top}, 0)}{N(N-1)})$$

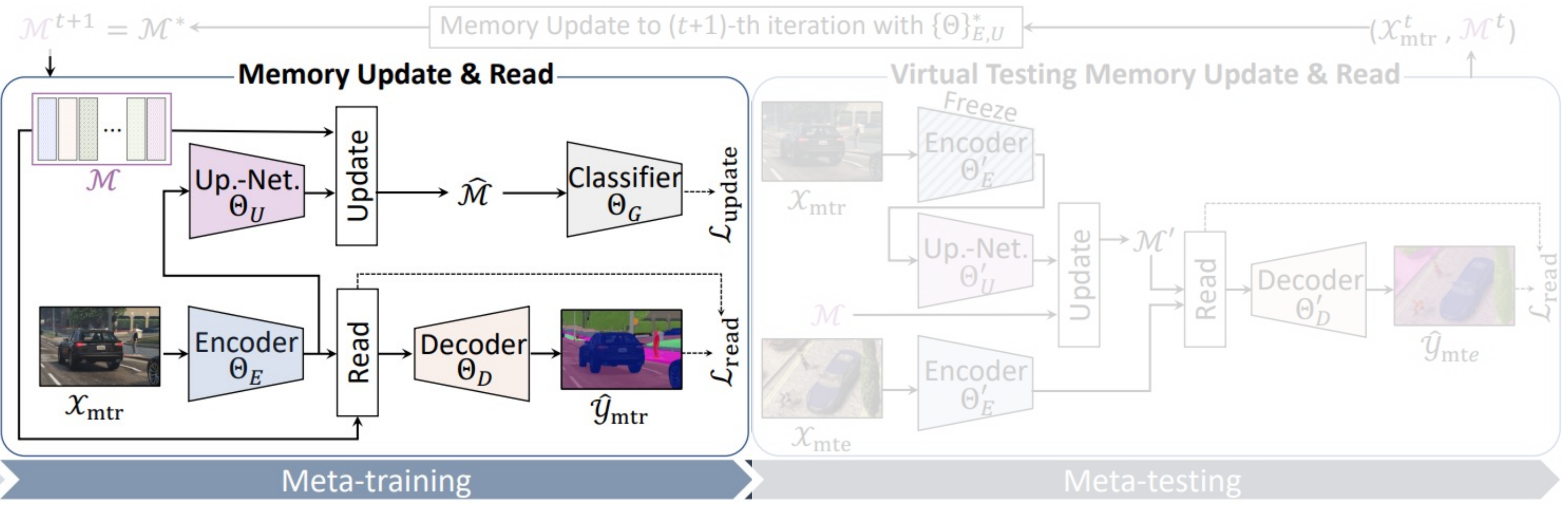
Algorithm 1: Overall Training Procedure

```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

Given an input image Xmtr in Smtr domain, the encoder computes a feature map and augments it by using the memory M through the reading operation. We calculate a per-pixel cross-entropy loss, Lseg.

Method

Overall Procedure



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{L}_{\text{read}}$
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
$\mathcal{M}'$
Decoder Θ'_D
$\mathcal{X}_{\text{mte}}$
$\hat{\mathcal{Y}}_{\text{mte}}$
Meta-training
Meta-testing

```
Algorithm 1: Overall Training Procedure
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

Read Loss

$$\mathcal{L}_{\text{read}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}; \{\Theta\}_{E,D}) = \mathcal{L}_{\text{seg}} + \lambda_1 \underline{\mathcal{L}_{\text{coh}}},$$

$$\underline{\mathcal{L}_{\text{coh}}} = \frac{1}{H'W'} \sum_{j=1}^{H'W'} -\mathcal{Y}^{\top}_{\text{mtr}}[j] \log(\mathcal{W}_{\text{mtr}}[j])$$

Memory update loss

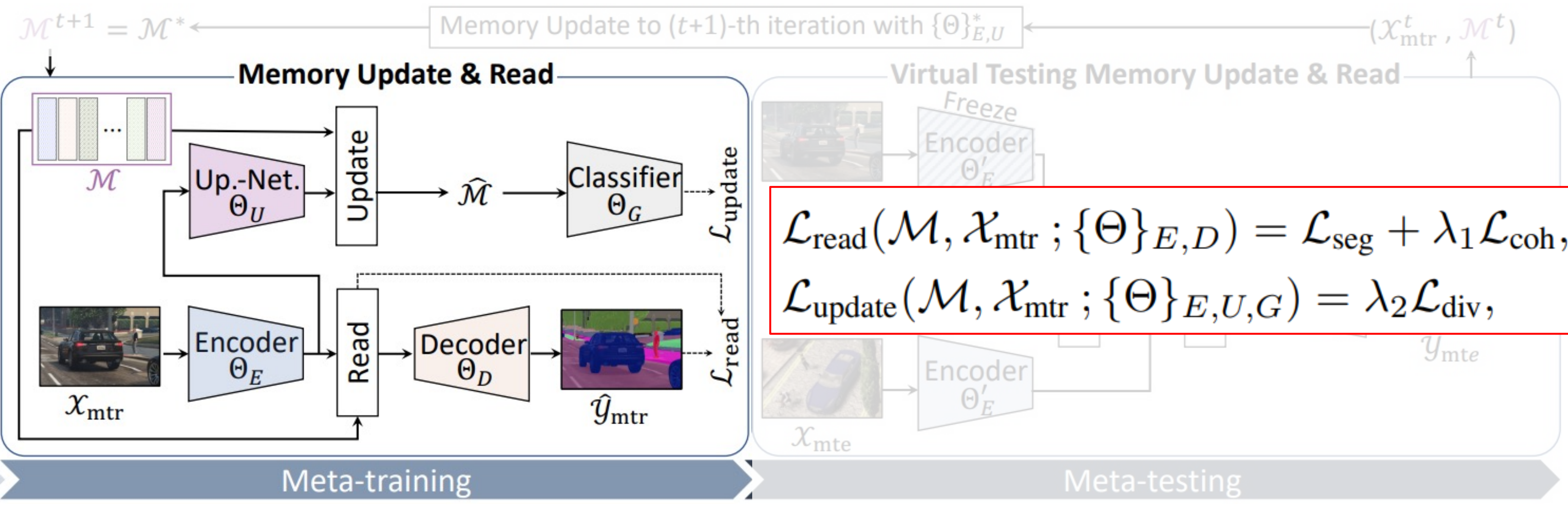
$$\mathcal{L}_{\text{update}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}; \{\Theta\}_{E,U,G}) = \lambda_2 \underline{\mathcal{L}_{\text{div}}},$$

$$\underline{\mathcal{L}_{\text{div}}} = \sum_{n=1}^{N} (-\mathcal{I}[n] \log(G(\hat{\mathcal{M}}[n]^{\top})) + 2 \cdot \sum_{n' \neq n}^{N} \frac{\max(\hat{\mathcal{M}}[n]\hat{\mathcal{M}}[n']^{\top}, 0)}{N(N-1)})$$

We also calculate a feature cohesion loss Lcoh to encourage semantic features to be locally assembled based on each memory item and a memory divergence loss Ldiv to increase the distance between memory items, as well as maximizes the decision margin.

METHOD

OVERALL PROCEDURE



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to $(t+1)$-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{L}_{\text{read}}$
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
Encoder Θ'_E
$\mathcal{Y}_{\text{mte}}$
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

$$\mathcal{L}_{\text{read}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}\,; \{\Theta\}_{E,D}) = \mathcal{L}_{\text{seg}} + \lambda_1 \mathcal{L}_{\text{coh}},$$
$$\mathcal{L}_{\text{update}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}\,; \{\Theta\}_{E,U,G}) = \lambda_2 \mathcal{L}_{\text{div}},$$

Algorithm 1: Overall Training Procedure

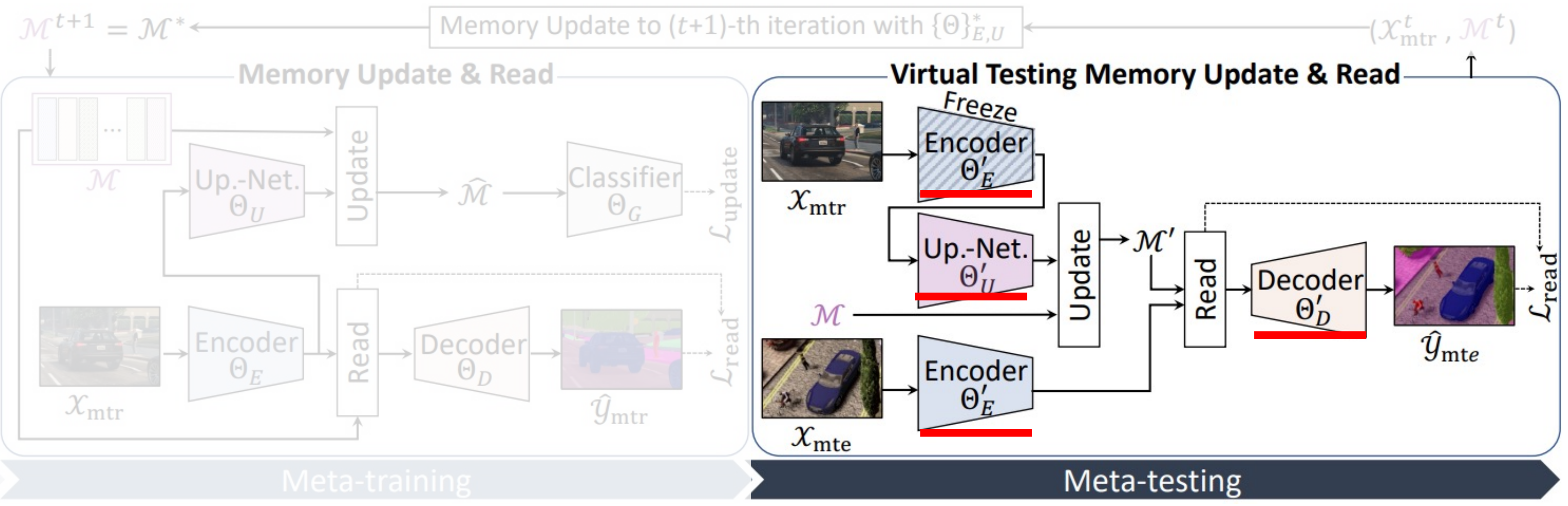
```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\underline{\{\Theta\}'_{E,U,D}, \Theta^*_G}$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

$$\begin{aligned}\underline{\{\Theta\}'_{E,U,D}}, \Theta^*_G \leftarrow \{\Theta\}_{E,U,D,G} &- \alpha \nabla_\Theta \mathcal{L}_{\text{read}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}\,; \{\Theta\}_{E,D}) \\ &- \alpha \nabla_\Theta \mathcal{L}_{\text{update}}(\mathcal{M}, \mathcal{X}_{\text{mtr}}\,; \{\Theta\}_{E,U,G})\end{aligned}$$

Note that we calculate the reading loss before memory update so that the segmentation mask information does not affect the segmentation loss. Consequently, the updated network parameters are obtained. We denote these parameters with prime.

METHOD

OVERALL PROCEDURE



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{L}_{\text{read}}$
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
$\mathcal{X}_{\text{mtr}}$
Up.-Net. Θ'_U
$\mathcal{M}$
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
Encoder Θ'_E
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

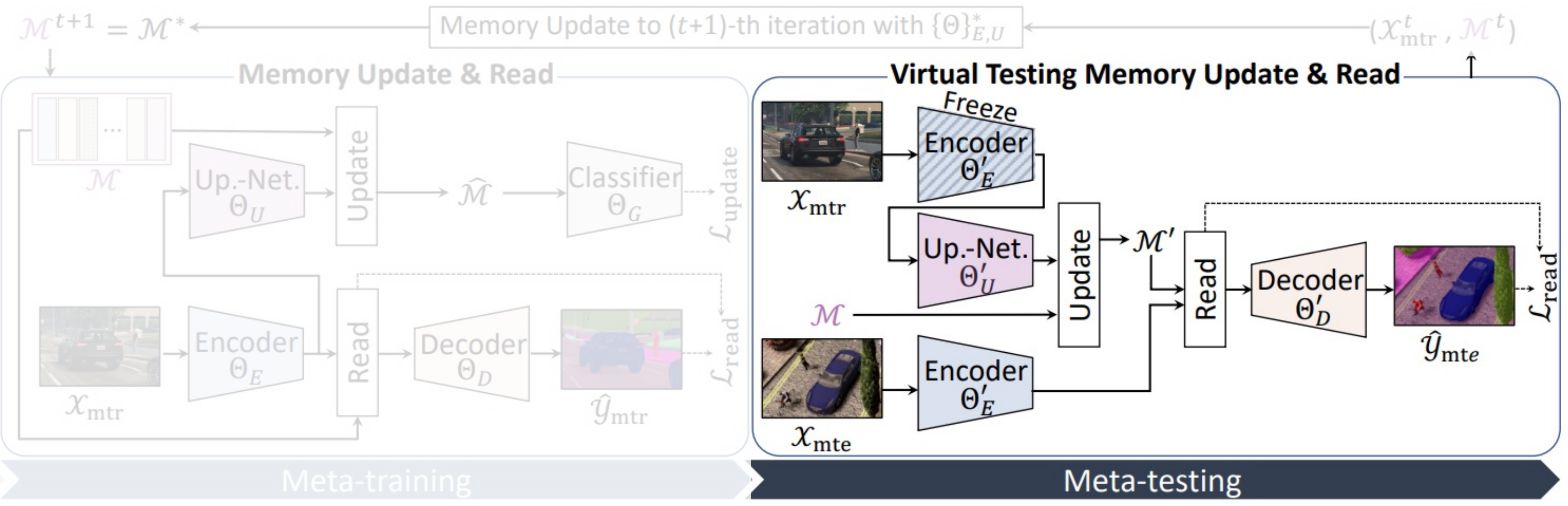
Algorithm 1: Overall Training Procedure

```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow \textbf{update}(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

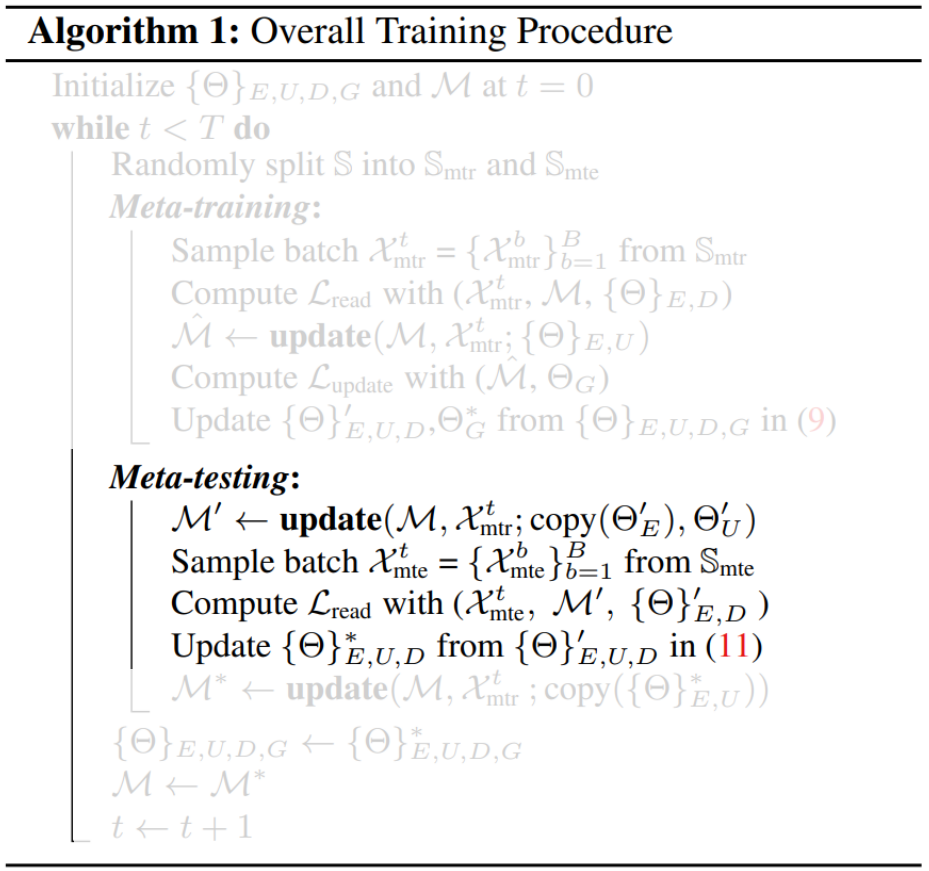
We perform meta testing step using the temporarily updated network parameters in the meta training step.

Method

Overall Procedure



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{L}_{\text{read}}$
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
$\mathcal{X}_{\text{mtr}}$
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
$\mathcal{M}$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
Encoder Θ'_E
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

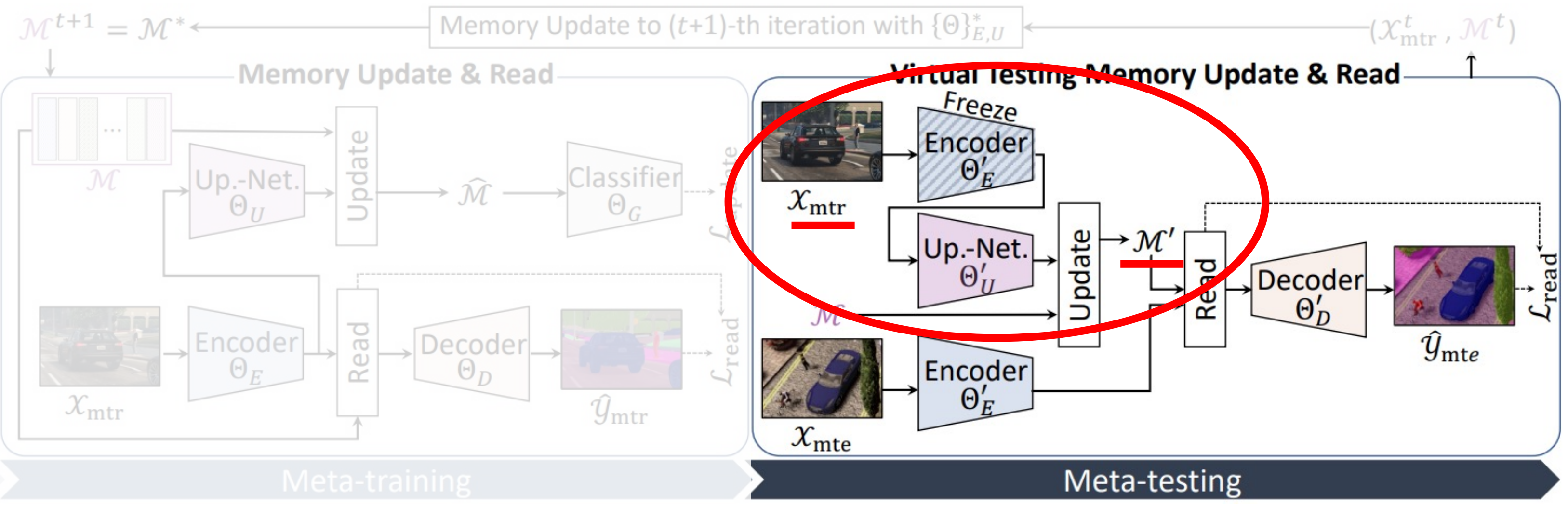


```
Algorithm 1: Overall Training Procedure
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

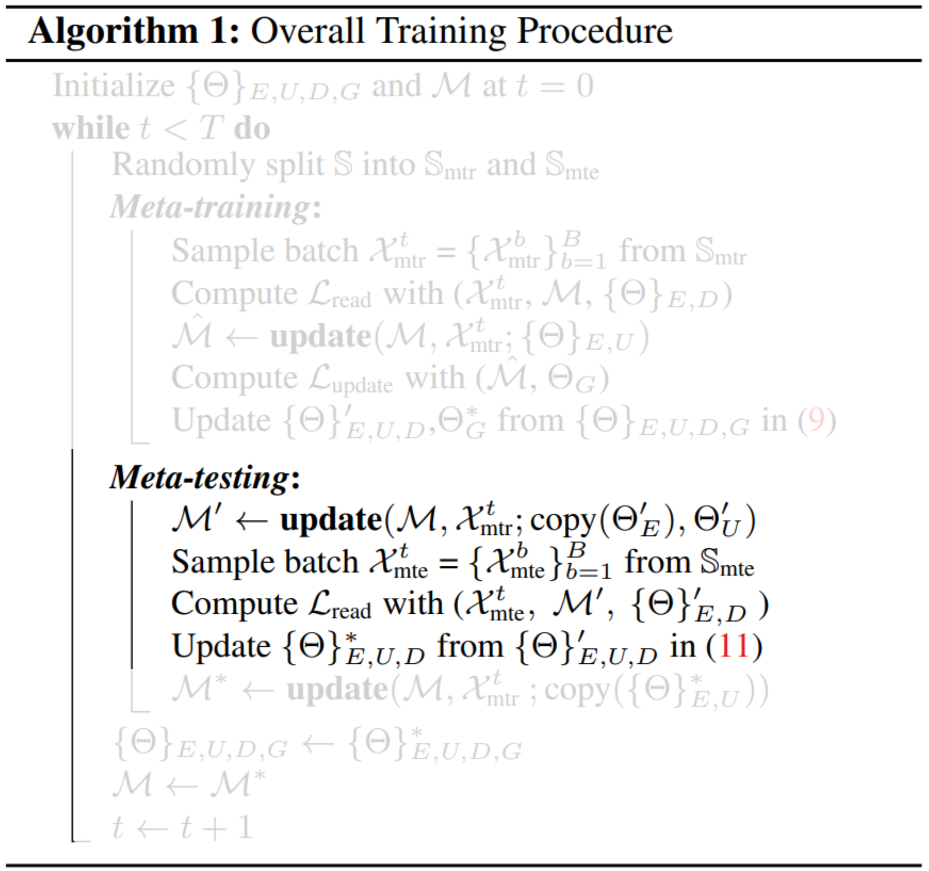
The goal of meta-testing in our method is to not only virtually simulate testing the networks on new data statistics but also characterize learning to update categorical memory to work well across the domains.

METHOD

OVERALL PROCEDURE

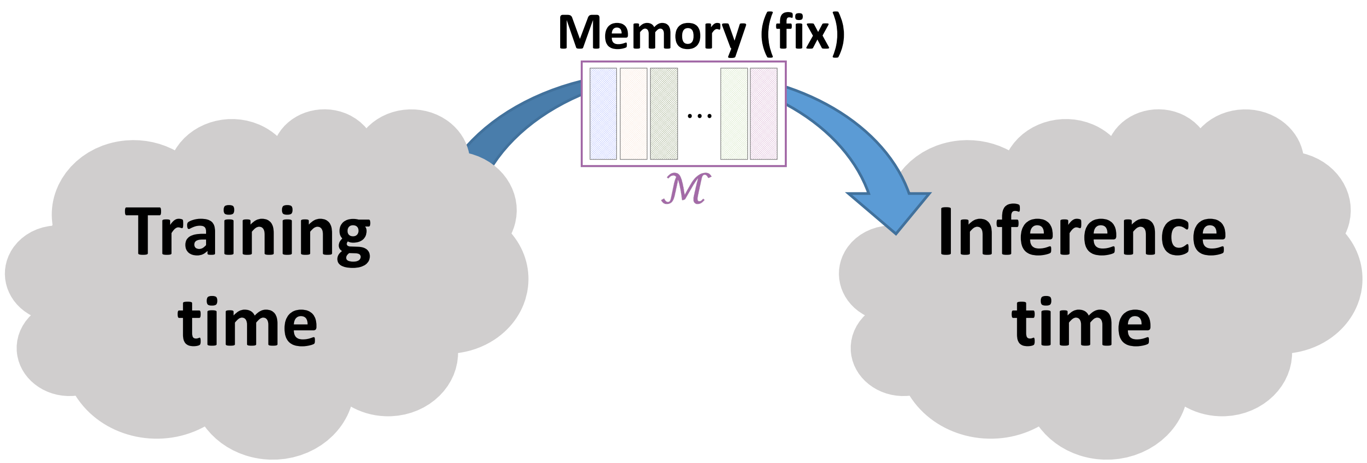


$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
Virtual Testing Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Freeze
Encoder Θ'_E
$\mathcal{X}_{\text{mtr}}$
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{M}$
Encoder Θ'_E
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing



Algorithm 1: Overall Training Procedure

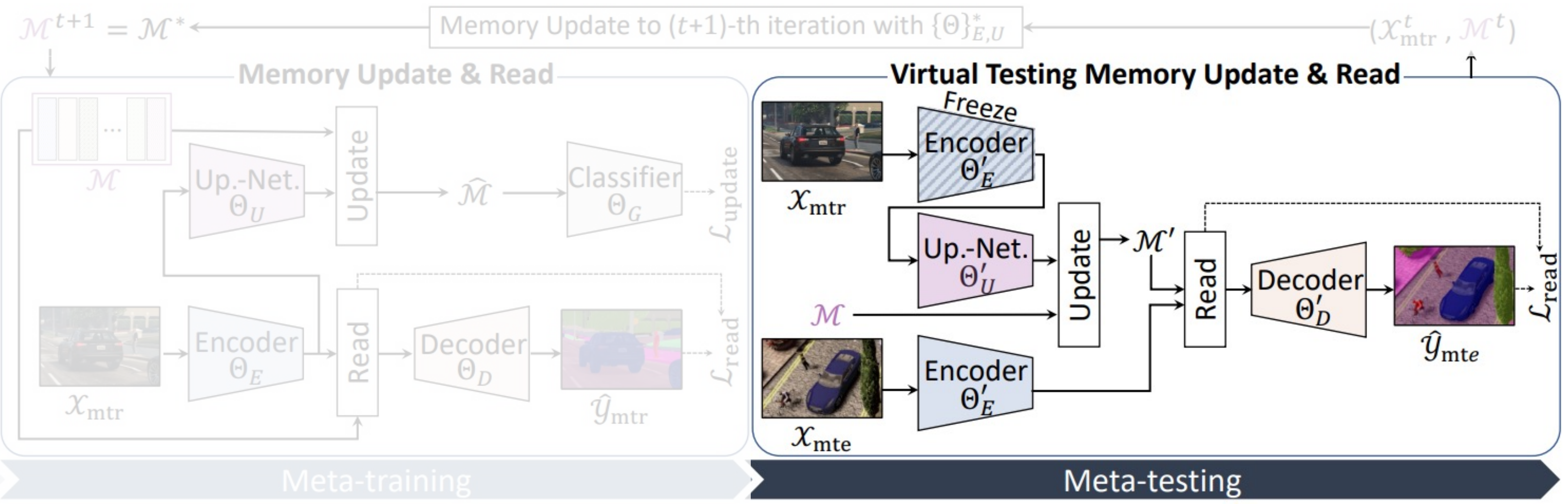
```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```



However in the true inference time, we will just reuse the trained memory without updating memory.
So we obtain the memory once again with meta-train data, not meta-test data.

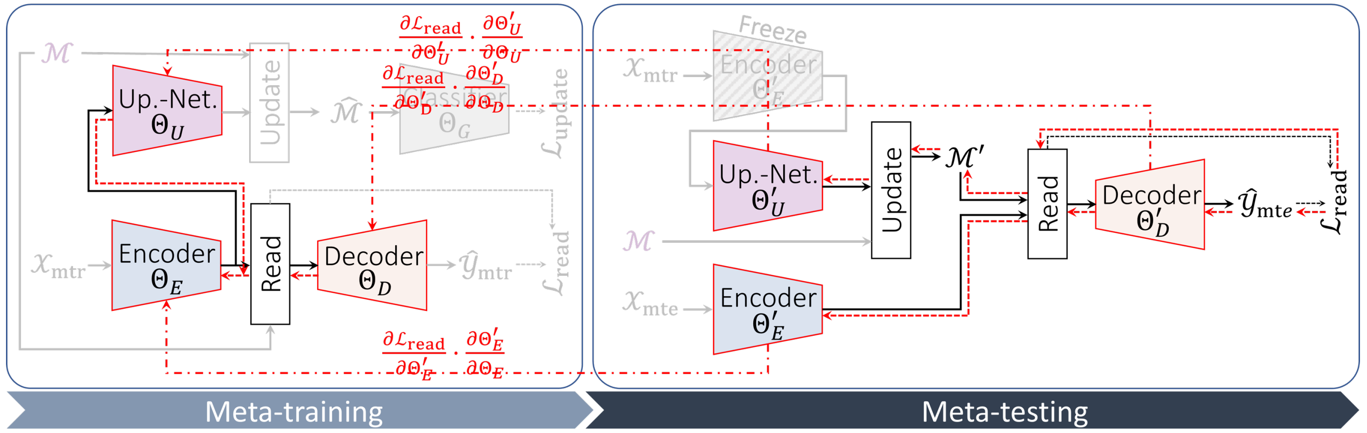
METHOD

OVERALL PROCEDURE



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{X}_{\text{mtr}}$
$\hat{y}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{y}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{M}$
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

Second gradient flow



$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_U} \cdot \frac{\partial \Theta'_U}{\partial \Theta_U}$
$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_D} \cdot \frac{\partial \Theta'_D}{\partial \Theta_D}$
$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_E} \cdot \frac{\partial \Theta'_E}{\partial \Theta_E}$
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
$\mathcal{X}_{\text{mtr}}$
Encoder Θ_E
Read
Decoder Θ_D
$\hat{y}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{y}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

Algorithm 1: Overall Training Procedure

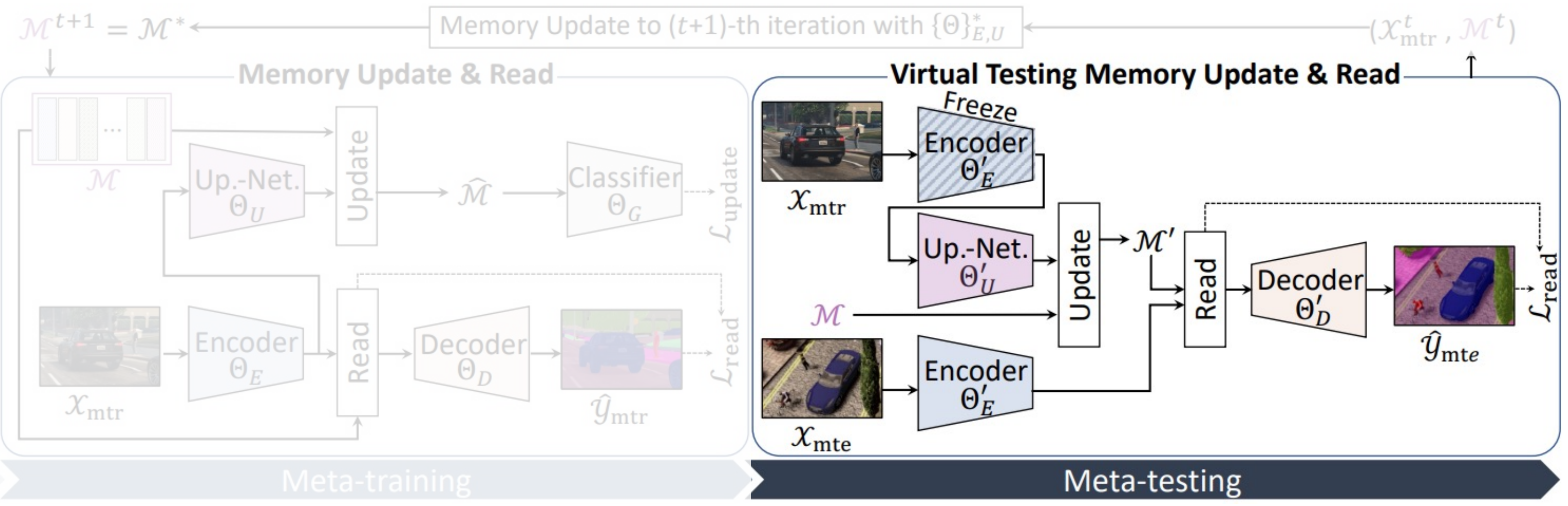
```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
    $t \leftarrow t + 1$
```

$$\{\Theta\}^*_{E,U,D} \leftarrow \{\Theta\}_{E,U,D} - \beta \nabla_{\Theta} \mathcal{L}_{\text{read}}(\mathcal{M}', \mathcal{X}_{\text{mte}}; \{\Theta\}'_{E,U,D}), \quad (11)$$

Lastly, Guided by updated memory M’, the network parameters are updated with the reading loss for the image Xmte from meta-test domain Smte.

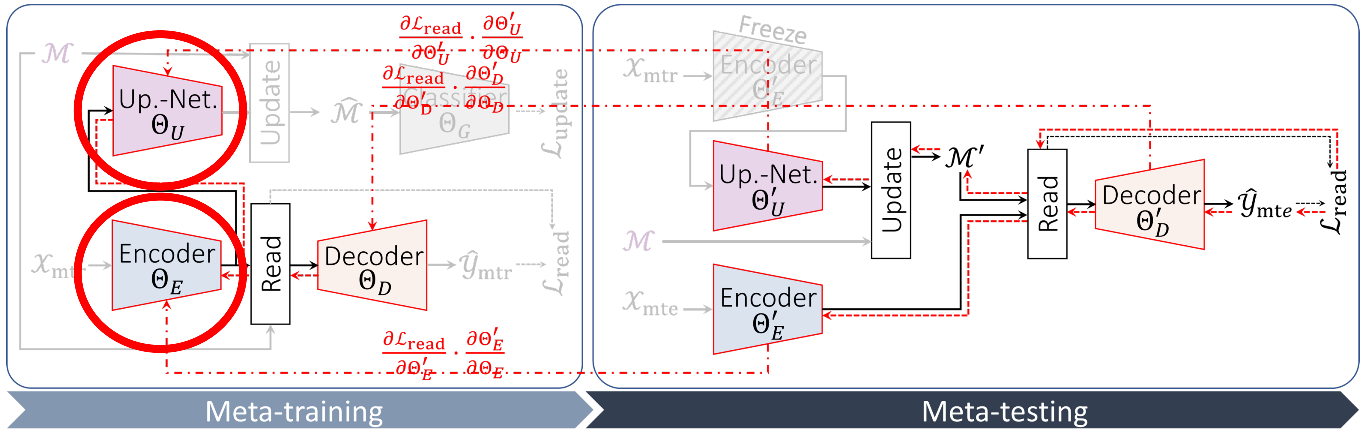
METHOD

OVERALL PROCEDURE



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
$\mathcal{X}_{\text{mtr}}$
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{M}$
Encoder Θ'_E
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

Second gradient flow



$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_U} \cdot \frac{\partial \Theta'_U}{\partial \Theta_U}$
$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_D} \cdot \frac{\partial \Theta'_D}{\partial \Theta_D}$
$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_E} \cdot \frac{\partial \Theta'_E}{\partial \Theta_E}$
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
$\mathcal{X}_{\text{mtr}}$
Encoder Θ_E
Read
Decoder Θ_D
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Freeze
$\mathcal{X}_{\text{mtr}}$
Encoder Θ'_E
Up.-Net. Θ'_U
Update
$\mathcal{M}'$
Read
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{L}_{\text{read}}$
$\mathcal{M}$
$\mathcal{X}_{\text{mte}}$
Encoder Θ'_E
Meta-training
Meta-testing

Algorithm 1: Overall Training Procedure

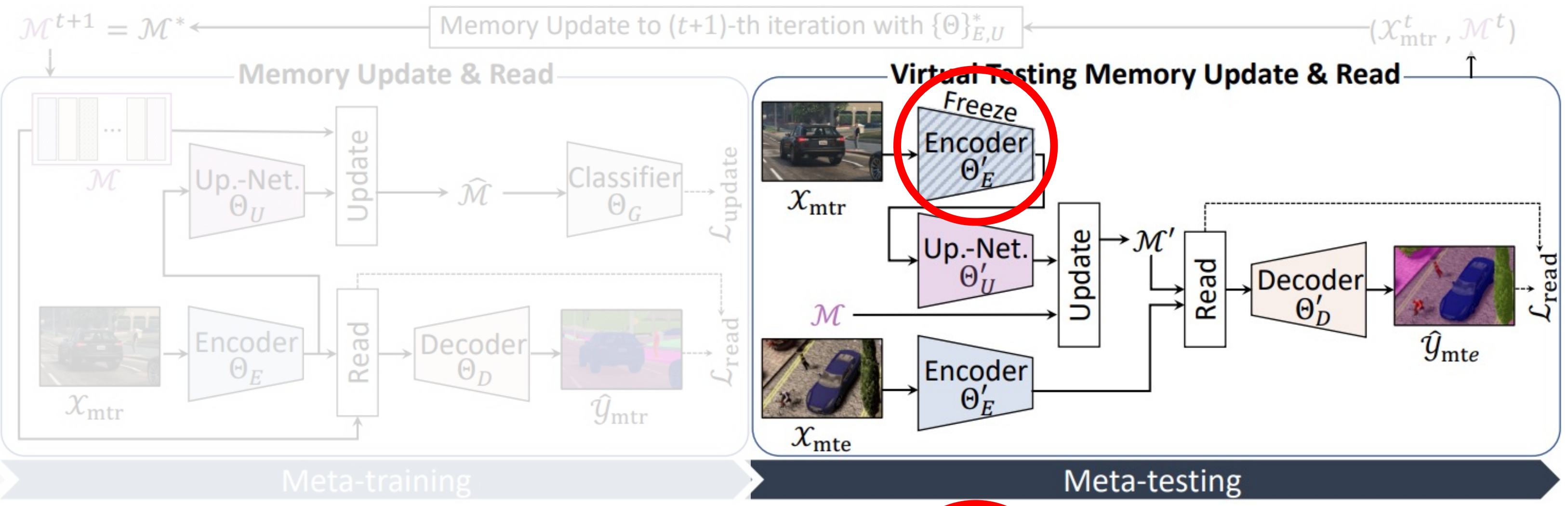
```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
    $t \leftarrow t + 1$
```

$$\{\Theta\}^*_{E,U,D} \leftarrow \{\Theta\}_{E,U,D} - \beta \nabla_\Theta \mathcal{L}_{\text{read}}(\mathcal{M}', \mathcal{X}_{\text{mte}}; \{\Theta\}'_{E,U,D}), \quad (11)$$

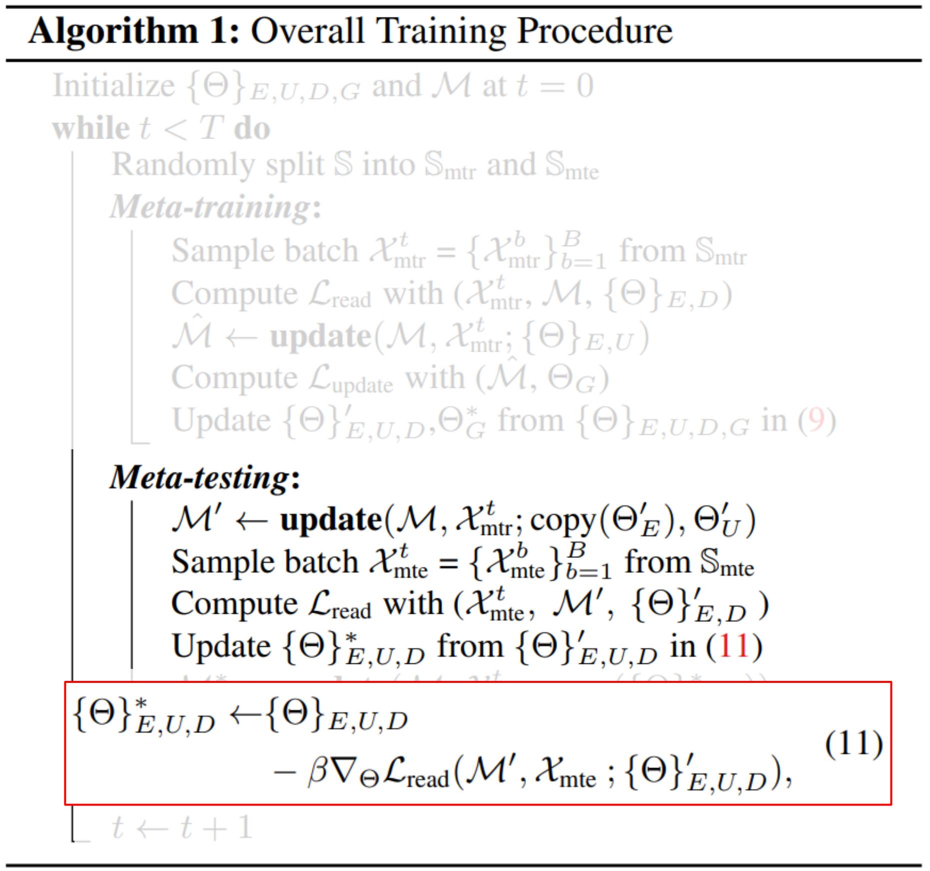
Since this memory M' is used to segment meta-test data Xmte, this novel step allows the memory updating network's parameters ΘU to receive the second-order gradient feedback on whether the updated memory M' is applicable on different domains.

METHOD

OVERALL PROCEDURE



$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{X}_{\text{mtr}}$
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Virtual Testing Memory Update & Read
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
$\mathcal{M}'$
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

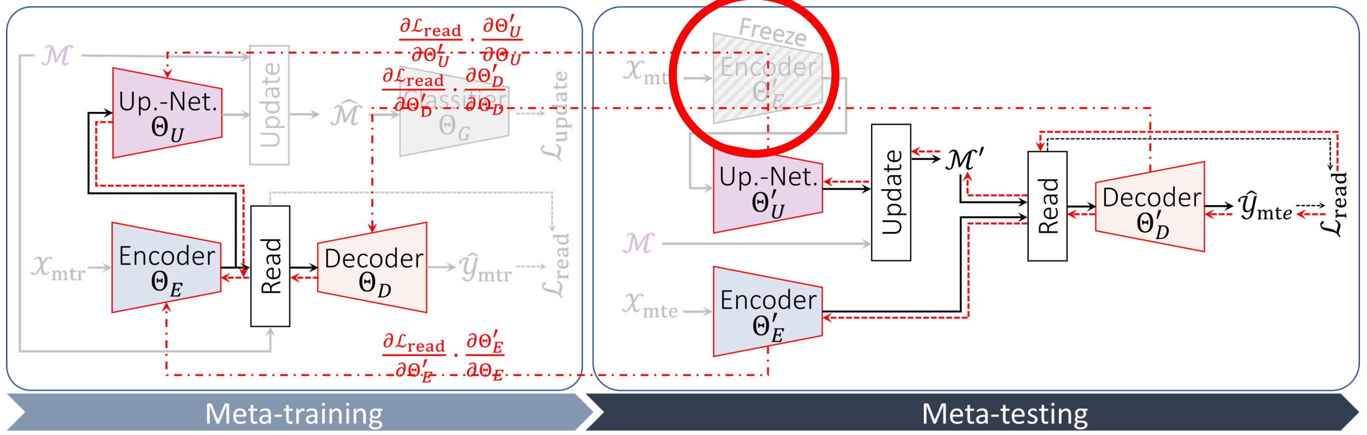


Algorithm 1: Overall Training Procedure

```
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
    $t \leftarrow t + 1$
```

$$\{\Theta\}^*_{E,U,D} \leftarrow \{\Theta\}_{E,U,D} - \beta \nabla_{\Theta} \mathcal{L}_{\text{read}}(\mathcal{M}', \mathcal{X}_{\text{mte}}; \{\Theta\}'_{E,U,D}), \tag{11}$$

Second gradient flow

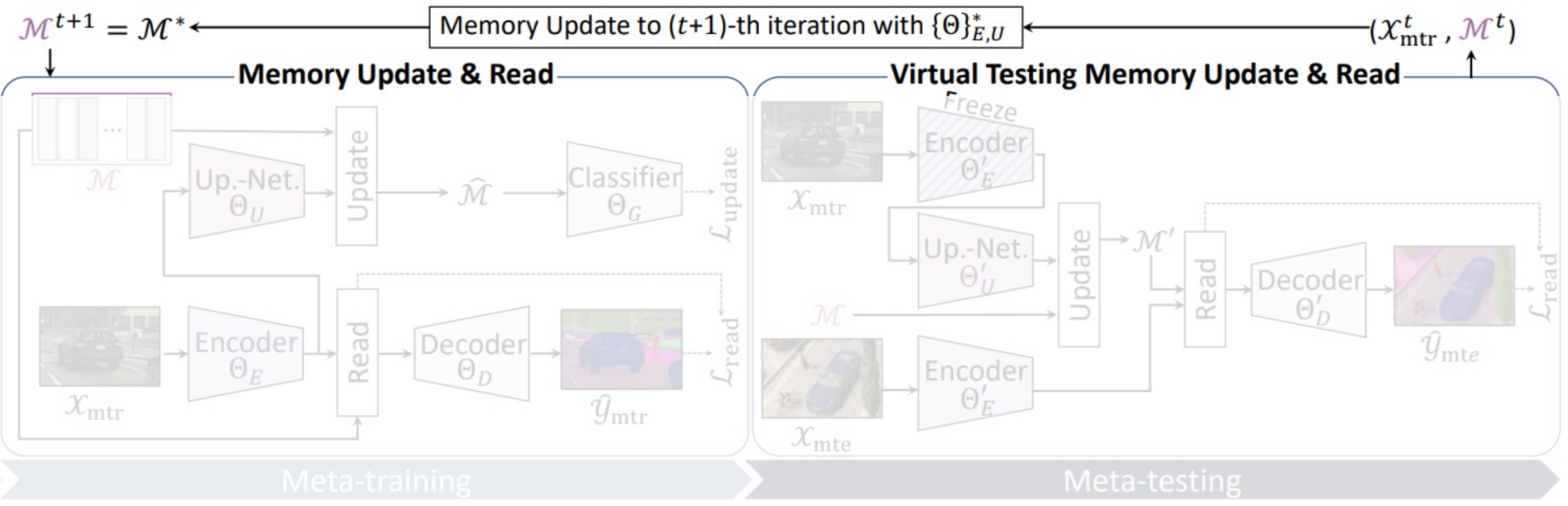


$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_U} \cdot \frac{\partial \Theta'_U}{\partial \Theta_U}$
$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_D} \cdot \frac{\partial \Theta'_D}{\partial \Theta_D}$
$\frac{\partial \mathcal{L}_{\text{read}}}{\partial \Theta'_E} \cdot \frac{\partial \Theta'_E}{\partial \Theta_E}$
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
$\mathcal{X}_{\text{mtr}}$
Encoder Θ_E
Read
Decoder Θ_D
$\hat{\mathcal{Y}}_{\text{mtr}}$
$\mathcal{L}_{\text{read}}$
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
$\mathcal{M}'$
Decoder Θ'_D
$\hat{\mathcal{Y}}_{\text{mte}}$
$\mathcal{X}_{\text{mte}}$
Meta-training
Meta-testing

Also, the effectiveness of the memory divergence loss for the updating network U can be tested within the meta-testing process
And by freezing the encoder's parameter Θ' E, we can avoid unstable meta-learning caused by the asynchronous gradient update between the encoder and the other networks.

METHOD

OVERALL PROCEDURE


$\mathcal{M}^{t+1} = \mathcal{M}^*$
Memory Update to (t+1)-th iteration with $\{\Theta\}^*_{E,U}$
$(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}^t)$
Memory Update & Read
Virtual Testing Memory Update & Read
$\mathcal{M}$
Up.-Net. Θ_U
Update
$\hat{\mathcal{M}}$
Classifier Θ_G
$\mathcal{L}_{\text{update}}$
Encoder Θ_E
Read
Decoder Θ_D
$\mathcal{L}_{\text{read}}$
$\mathcal{X}_{\text{mtr}}$
$\mathcal{Y}_{\text{mtr}}$
Freeze
Encoder Θ'_E
Up.-Net. Θ'_U
$\mathcal{M}'$
Decoder Θ'_D
$\mathcal{X}_{\text{mte}}$
$\mathcal{Y}_{\text{mte}}$
Meta-training
Meta-testing

```
Algorithm 1: Overall Training Procedure
Initialize $\{\Theta\}_{E,U,D,G}$ and $\mathcal{M}$ at $t = 0$
while $t < T$ do
    Randomly split $\mathbb{S}$ into $\mathbb{S}_{\text{mtr}}$ and $\mathbb{S}_{\text{mte}}$
    Meta-training:
        Sample batch $\mathcal{X}^t_{\text{mtr}} = \{\mathcal{X}^b_{\text{mtr}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mtr}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mtr}}, \mathcal{M}, \{\Theta\}_{E,D})$
        $\hat{\mathcal{M}} \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \{\Theta\}_{E,U})$
        Compute $\mathcal{L}_{\text{update}}$ with $(\hat{\mathcal{M}}, \Theta_G)$
        Update $\{\Theta\}'_{E,U,D}, \Theta^*_G$ from $\{\Theta\}_{E,U,D,G}$ in (9)
    Meta-testing:
        $\mathcal{M}' \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\Theta'_E), \Theta'_U)$
        Sample batch $\mathcal{X}^t_{\text{mte}} = \{\mathcal{X}^b_{\text{mte}}\}^B_{b=1}$ from $\mathbb{S}_{\text{mte}}$
        Compute $\mathcal{L}_{\text{read}}$ with $(\mathcal{X}^t_{\text{mte}}, \mathcal{M}', \{\Theta\}'_{E,D})$
        Update $\{\Theta\}^*_{E,U,D}$ from $\{\Theta\}'_{E,U,D}$ in (11)
        $\mathcal{M}^* \leftarrow$ update$(\mathcal{M}, \mathcal{X}^t_{\text{mtr}}; \text{copy}(\{\Theta\}^*_{E,U}))$
    $\{\Theta\}_{E,U,D,G} \leftarrow \{\Theta\}^*_{E,U,D,G}$
    $\mathcal{M} \leftarrow \mathcal{M}^*$
    $t \leftarrow t + 1$
```

Finally, Using the second order optimized network parameters, we initialize the memory M∗ that will be used in the next training iteration step.

EXPERIMENTS

SEMANTIC SEGMENTATION DATASETS

- ***Synthetic***

  Synthia (S) : 9k images with different weather and illumination conditions from multiple viewpoints.
  GTAV (G) : 25k driving scene image dataset.
- ***Real***

  Cityscapes (C) : 3,450 finely-annotated images collected from 50 different citiess, primarily Germany.
  BDD100K (B) : 8K diverse urban driving scene images collected from various locations in the US.
  Mapillary (M) : street-view dataset including 25K images collected from all around the world.
  IDD (I) : 10,004 images captured from Indian roads which are significantly different from the existing datasets mainly collected in Europe or US. (we used this as source dataset)

NETWORK ARCHITECTURE

- DeepLab V3+ with ResNet50, DeepLab V2 with ResNet101

EVALUATION PROTOCOLS

- Single-Source DG : **G** → **C**, **B**, **M**
- Multi-Source DG : 1. **G** + **S** → **C**, **B**, **M** 2. **G** + **S** + **I** → **C**, **B**, **M**

We conduct the experiments on six different datasets to prove the generalization ability of our method.
Synthia and gta5 datasets are synthetic datasets, and the rest are real datasets.

EXPERIMENTS

SEMANTIC SEGMENTATION DATASETS

- ***Synthetic***

  Synthia (S) : 9k images with different weather and illumination conditions from multiple viewpoints.
  GTAV (G) : 25k driving scene image dataset.

- ***Real***

  Cityscapes (C) : 3,450 finely-annotated images collected from 50 different citiess, primarily Germany.
  BDD100K (B) : 8K diverse urban driving scene images collected from various locations in the US.
  Mapillary (M) : street-view dataset including 25K images collected from all around the world.
  IDD (I) : 10,004 images captured from Indian roads which are significantly different from the existing datasets mainly collected in Europe or US. (we used this as source dataset)

NETWORK ARCHITECTURE

- DeepLab V3+ with ResNet50, DeepLab V2 with ResNet101

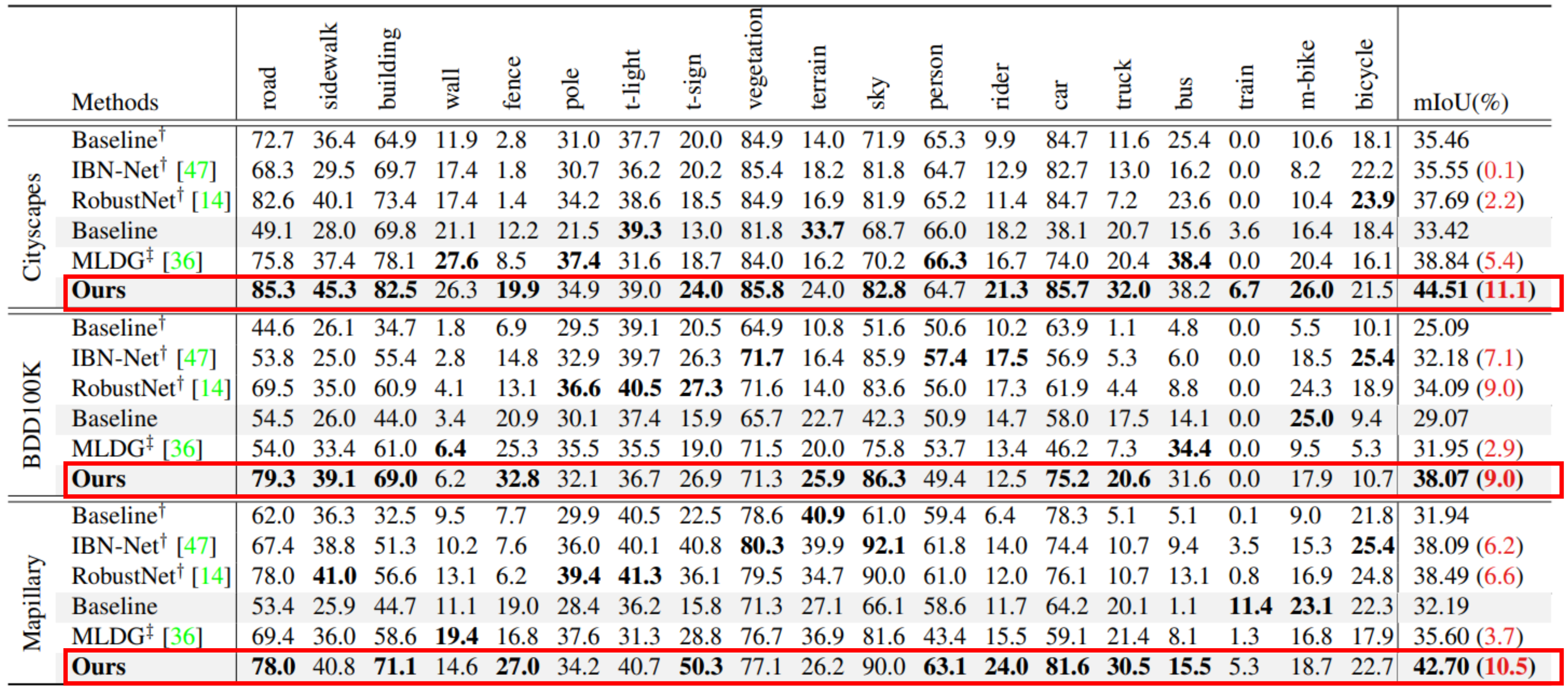
EVALUATION PROTOCOLS

- Single-Source DG : **G** → **C**, **B**, **M**
- Multi-Source DG : 1. **G** + **S** → **C**, **B**, **M** 2. **G** + **S** + **I** → **C**, **B**, **M**

We measured the domain generalization performance by setting the training and testing datasets differently.

EXPERIMENTS

MULTI-SOURCE DG ON SEMANTIC SEGMENTATION



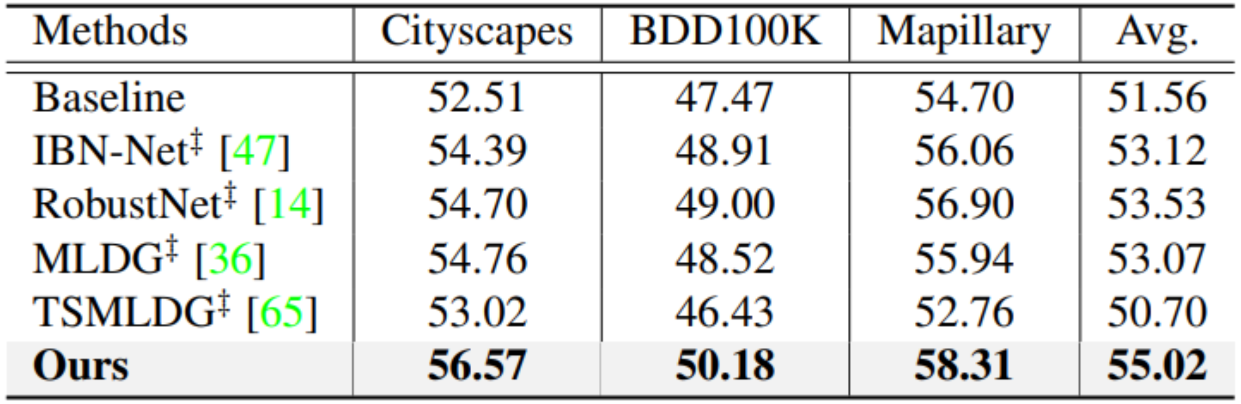
	Methods	road	sidewalk	building	wall	fence	pole	t-light	t-sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	m-bike	bicycle	mIoU(%)
Cityscapes	Baseline†	72.7	36.4	64.9	11.9	2.8	31.0	37.7	20.0	84.9	14.0	71.9	65.3	9.9	84.7	11.6	25.4	0.0	10.6	18.1	35.46
	IBN-Net† [47]	68.3	29.5	69.7	17.4	1.8	30.7	36.2	20.2	85.4	18.2	81.8	64.7	12.9	82.7	13.0	16.2	0.0	8.2	22.2	35.55 (0.1)
	RobustNet† [14]	82.6	40.1	73.4	17.4	1.4	34.2	38.6	18.5	84.9	16.9	81.9	65.2	11.4	84.7	7.2	23.6	0.0	10.4	**23.9**	37.69 (2.2)
	Baseline	49.1	28.0	69.8	21.1	12.2	21.5	**39.3**	13.0	81.8	**33.7**	68.7	66.0	18.2	38.1	20.7	15.6	3.6	16.4	18.4	33.42
	MLDG‡ [36]	75.8	37.4	78.1	**27.6**	8.5	**37.4**	31.6	18.7	84.0	16.2	70.2	**66.3**	16.7	74.0	20.4	**38.4**	0.0	20.4	16.1	38.84 (5.4)
	Ours	**85.3**	**45.3**	**82.5**	26.3	**19.9**	34.9	39.0	**24.0**	**85.8**	24.0	**82.8**	64.7	**21.3**	**85.7**	**32.0**	38.2	**6.7**	**26.0**	21.5	**44.51 (11.1)**
BDD100K	Baseline†	44.6	26.1	34.7	1.8	6.9	29.5	39.1	20.5	64.9	10.8	51.6	50.6	10.2	63.9	1.1	4.8	0.0	5.5	10.1	25.09
	IBN-Net† [47]	53.8	25.0	55.4	2.8	14.8	32.9	39.7	26.3	**71.7**	16.4	85.9	**57.4**	**17.5**	56.9	5.3	6.0	0.0	18.5	**25.4**	32.18 (7.1)
	RobustNet† [14]	69.5	35.0	60.9	4.1	13.1	**36.6**	**40.5**	**27.3**	71.6	14.0	83.6	56.0	17.3	61.9	4.4	8.8	0.0	24.3	18.9	34.09 (9.0)
	Baseline	54.5	26.0	44.0	3.4	20.9	30.1	37.4	15.9	65.7	22.7	42.3	50.9	14.7	58.0	17.5	14.1	0.0	**25.0**	9.4	29.07
	MLDG‡ [36]	54.0	33.4	61.0	**6.4**	25.3	35.5	35.5	19.0	71.5	20.0	75.8	53.7	13.4	46.2	7.3	**34.4**	0.0	9.5	5.3	31.95 (2.9)
	Ours	**79.3**	**39.1**	**69.0**	6.2	**32.8**	32.1	36.7	26.9	71.3	**25.9**	**86.3**	49.4	12.5	**75.2**	**20.6**	31.6	0.0	17.9	10.7	**38.07 (9.0)**
Mapillary	Baseline†	62.0	36.3	32.5	9.5	7.7	29.9	40.5	22.5	78.6	**40.9**	61.0	59.4	6.4	78.3	5.1	5.1	0.1	9.0	21.8	31.94
	IBN-Net† [47]	67.4	38.8	51.3	10.2	7.6	36.0	40.1	40.8	**80.3**	39.9	**92.1**	61.8	14.0	74.4	10.7	9.4	3.5	15.3	**25.4**	38.09 (6.2)
	RobustNet† [14]	78.0	**41.0**	56.6	13.1	6.2	**39.4**	**41.3**	36.1	79.5	34.7	90.0	61.0	12.0	76.1	10.7	13.1	0.8	16.9	24.8	38.49 (6.6)
	Baseline	53.4	25.9	44.7	11.1	19.0	28.4	36.2	15.8	71.3	27.1	66.1	58.6	11.7	64.2	20.1	1.1	**11.4**	**23.1**	22.3	32.19
	MLDG‡ [36]	69.4	36.0	58.6	**19.4**	16.8	37.6	31.3	28.8	76.7	36.9	81.6	43.4	15.5	59.1	21.4	8.1	1.3	16.8	17.9	35.60 (3.7)
	Ours	**78.0**	40.8	**71.1**	14.6	**27.0**	34.2	40.7	**50.3**	77.1	26.2	90.0	**63.1**	**24.0**	**81.6**	**30.5**	**15.5**	5.3	18.7	22.7	**42.70 (10.5)**

Table 1. **Source (G+S)→Target (C, B, M):** Mean IoU(%) and per-class IoU(%) comparison of other SOTA DG methods for semantic segmentation. We report the mIoU improvement as red text. The networks are DeepLabV3+ with ResNet50 and results with † are from [14].

This is Mean IoU(%) and per-class IoU(%) comparison table of other SOTA DG methods for semantic segmentation. Source domains were gta5 and synthia. Our approach consistently outperformed the other models by a large margin on all real-world datasets.

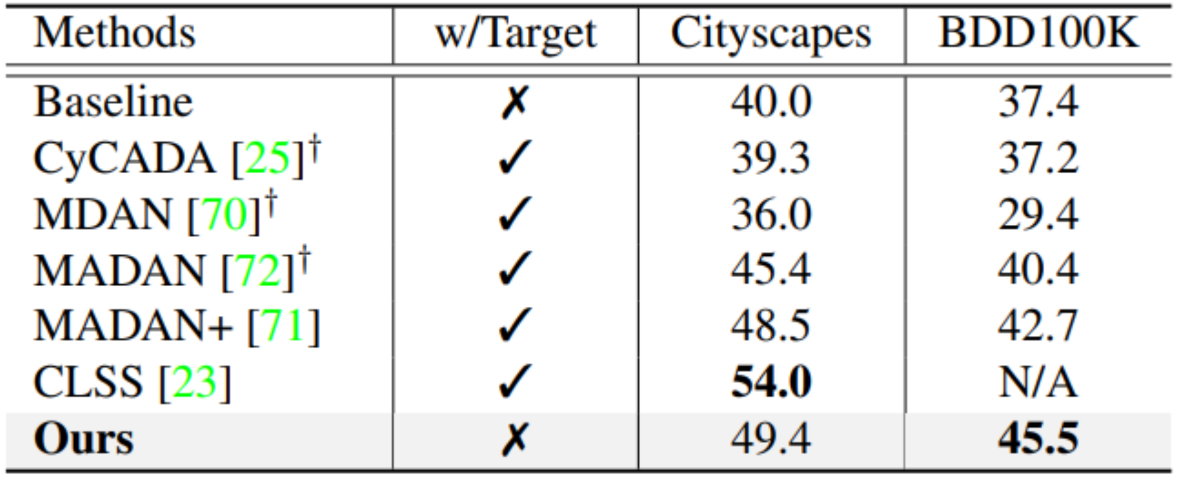
EXPERIMENTS

MULTI-SOURCE DG ON SEMANTIC SEGMENTATION



Methods	Cityscapes	BDD100K	Mapillary	Avg.
Baseline	52.51	47.47	54.70	51.56
IBN-Net‡ [47]	54.39	48.91	56.06	53.12
RobustNet‡ [14]	54.70	49.00	56.90	53.53
MLDG‡ [36]	54.76	48.52	55.94	53.07
TSMLDG‡ [65]	53.02	46.43	52.76	50.70
Ours	**56.57**	**50.18**	**58.31**	**55.02**

Table 2. **Source (G+S+I)→Target (C, B, M):** Mean IoU(%) comparison of other state-of-the-art DG methods, where all networks are trained with two synthetic (GTAV, Synthia) and one real (IDD) datasets. All methods adopt DeepLabV3+ with ResNet50.



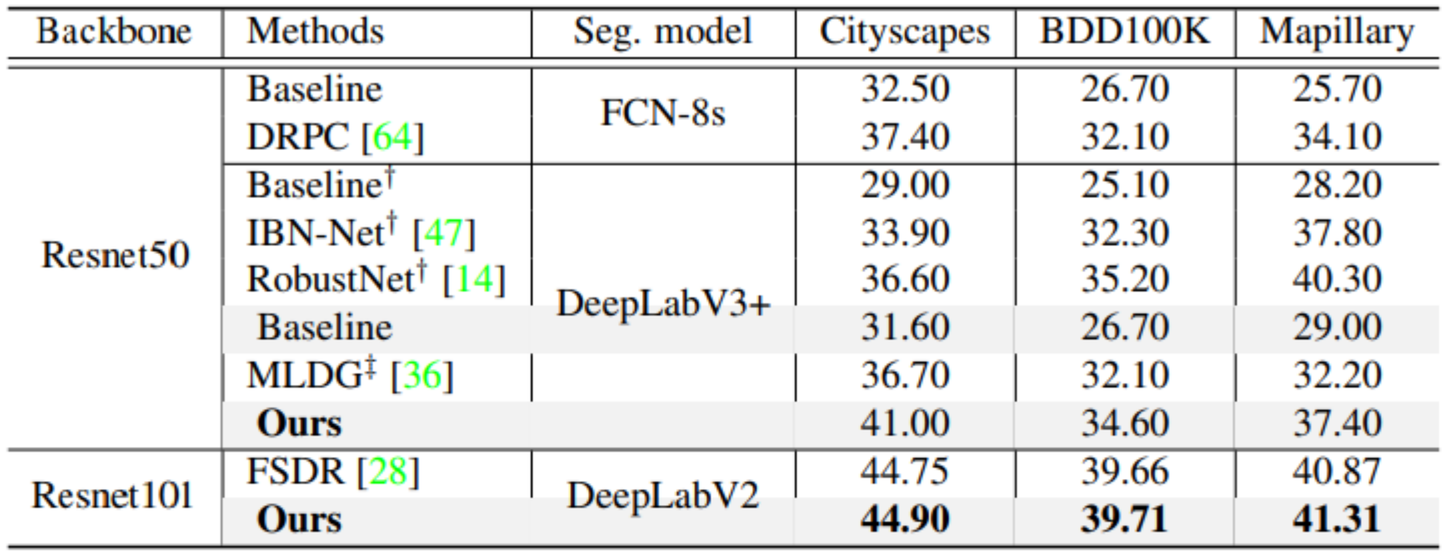
Methods	w/Target	Cityscapes	BDD100K
Baseline	✗	40.0	37.4
CyCADA [25]†	✓	39.3	37.2
MDAN [70]†	✓	36.0	29.4
MADAN [72]†	✓	45.4	40.4
MADAN+ [71]	✓	48.5	42.7
CLSS [23]	✓	**54.0**	N/A
Ours	✗	49.4	**45.5**

Table 4. **Source (G+S)→Target (C, B):** Mean IoU(%) comparison of other multi-source UDA methods. The segmentation models are all DeepLabV2 with ResNet101. Results with † are from [71].

Even when three source domains was used, our model achieved SOTA.
Also, compared to UDA methods using target domain information, comparable performance was achieved.

EXPERIMENTS

SINGLE-SOURCE DG ON SEMANTIC SEGMENTATION



Backbone	Methods	Seg. model	Cityscapes	BDD100K	Mapillary
Resnet50	Baseline	FCN-8s	32.50	26.70	25.70
	DRPC [64]		37.40	32.10	34.10
	Baseline†	DeepLabV3+	29.00	25.10	28.20
	IBN-Net† [47]		33.90	32.30	37.80
	RobustNet† [14]		36.60	35.20	40.30
	Baseline		31.60	26.70	29.00
	MLDG‡ [36]		36.70	32.10	32.20
	Ours		41.00	34.60	37.40
Resnet101	FSDR [28]	DeepLabV2	44.75	39.66	40.87
	Ours		**44.90**	**39.71**	**41.31**

Table 12. **Source (G)→Target (C, B, M):** Mean IoU(%) comparison of other SOTA methods using various segmentation models and backbones. MLDG [36] is re-implemented. Results with † are from [14].

Although it did not come out well for all domains in the single source DG setting, we confirmed that our method gave comparable performance.

Q&A

THANK YOU!

MORE DETAILS AND EXPERIMENTS CAN BE FOUND ON HTTPS://ARXIV.ORG/ABS/2204.03609

Thank you for listening.
Please find more details and experimental results on our main paper.