Masked Representation Modeling for Domain-Adaptive Segmentation

Supplementary Material

Algorithm 1 Training with Masked Representation Modeling (MRM).

Require: Source images $\{x_s, y_s\}$, target images $\{x_t\}$, segmentation model $F = D \circ E$, Rebuilder R

for each training step **do**
  $\#$ *Supervised loss on source domain*
  $f_s \leftarrow E(x_s)$
  $\hat{y}_s \leftarrow D(f_s)$
  $\mathcal{L}_{\text{sup}} \leftarrow \text{CrossEntropy}(\hat{y}_s, y_s)$
  $\#$ *Unsupervised loss (e.g., pseudo-labeling)*
  $f_t \leftarrow E(x_t)$
  $\hat{y}_t \leftarrow D(f_t)$
  $\tilde{y}_t \leftarrow \text{PseudoLabel}(\hat{y}_t)$
  $\mathcal{L}_{\text{uda}} \leftarrow \text{CrossEntropy}(\hat{y}_t, \tilde{y}_t)$
  $\#$ **Masked Representation Modeling (ours)**
  $f_t^{\text{recon}} \leftarrow R(f_t^{\text{mask}})$
  $\mathcal{L}_{\text{mrm}} \leftarrow \text{CrossEntropy}(D(f_t^{\text{recon}}), \tilde{y}_t)$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{uda}} + \lambda \cdot \mathcal{L}_{\text{mrm}}$
  Update model parameters via backpropagation
end for

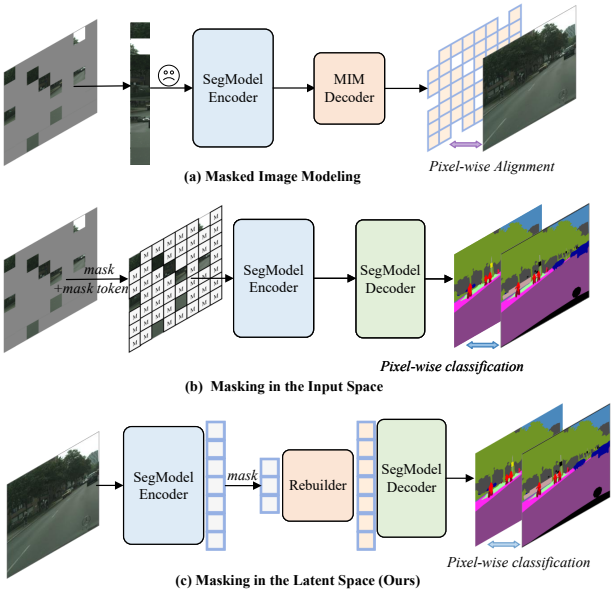

Figure 6. Different masking locations in the network.

6. Pseudo Code

The pseudo code in Algorithm 1 outlines the integration of our proposed MRM into a typical domain adaptation training pipeline. MRM is designed to be modular and can be seamlessly incorporated into existing frameworks that combine supervised learning on source data and unsupervised learning on target data via pseudo-labeling [25].

At each training step, conventional losses are computed: a supervised segmentation loss on the source domain, and an unsupervised loss on the target domain. Our method introduces an additional auxiliary branch, where masked representation are reconstructed and then regularized using the pseudo-label supervision. This not only enriches the representation but also enhances generalization across domains.

Importantly, MRM does not interfere with the original optimization objectives but complements them with minimal architectural changes. Its plug-and-play nature allows it to be effectively combined with a wide range of existing domain adaptation or semi-supervised segmentation methods.

7. Further Ablation Study

Different Masking Locations in the Network. MAE [16] removes image patches and processes only the visible ones through the encoder, which makes it incompatible with semantic segmentation encoders that rely on full spatial inputs. To address this issue, MRM performs masking and reconstruction directly in the latent space, enabling compatibility with arbitrary backbone models as long as the input and output shapes of the Rebuilder are aligned. However, alternative designs are also possible. Inspired by BERT [12], one could insert masked tokens before the encoder by concatenating them with visible patches, allowing the encoder to maintain a standard forward pipeline. In this section, we explore the effectiveness of this design choice. The results are presented in Table 6.

Masking Position	mIOU	Training FLOPs (G)
w/o MRM	52.1	1.00× (182.34)
Input Space	54.0 (+1.9)	2.00× (364.68)
Latent Space (ours)	55.9 (+3.8)	1.13× (207.74)

Table 6. Ablation on masking locations using a DACS[43] evaluated on the GTA → Cityscapes Benchmark.

Experiment demonstrate that applying masking in the input space can also lead to performance improvements. Notably, since the Rebuilder is removed during training, this indirectly validates the importance of using the main task loss as the optimization objective for the auxiliary task. However, masking in the latent space still yields greater performance gains compared to input-space masking.

To provide a more intuitive comparison, we also introduce Training Floating Point Operations (FLOPs) as a metric to evaluate the computational cost of the two masking strategies in MRM. Specifically, this FLOPs count reflects the cost incurred when processing the target image. For reference, the baseline method (i.e., DACS w/o MRM) requires 182.34G FLOPs. In the input-space design, after masking and inserting mask tokens, the resulting input is no longer identical to the original target image. As a result, the processed input must be passed through the full network again, effectively doubling the training FLOPs compared to the baseline.

In contrast, MRM maintains the original input (i.e., the target image), allowing it to reuse the encoder's representation already computed in the UDA pipeline. This avoids redundant computations and leads to substantial savings in training cost. The additional overhead comes only from passing these encoder features through the lightweight Rebuilder and Decoder modules, which are significantly more efficient than the encoder itself, resulting in only a marginal increase in FLOPs.

8. Discussion

8.1. Limitations and Future Work

While MRM demonstrates strong performance across diverse architectures and datasets, several limitations remain:

- **Limited capacity of Rebuilder**: To maintain lightweight design, the Rebuilder employs only a small number of Transformer blocks. Although sufficient for current benchmarks, its ability to model complex feature dependencies may be limited in more diverse domains.
- **Training stability**: We observed that scaling up the Rebuilder or increasing the masking ratio can lead to unstable training. This reflects the difficulty of integrating auxiliary Transformer components with pre-trained segmentation models.
- **Task specificity**: MRM is specifically designed for pixel-wise classification tasks. Its direct extension to other dense prediction tasks such as depth estimation or panoptic segmentation requires further investigation.

In future work, we aim to explore more powerful and stable Rebuilder architectures, potentially leveraging pre-trained or distilled representations to improve semantic reconstruction, as well as incorporating diverse masking strategies to enhance robustness. Moreover, we plan to generalize the MRM paradigm to other structured prediction tasks, and to study its synergy with advanced pseudo label and consistency regularization techniques. Finally, we are interested in investigating the theoretical foundations of task-aligned auxiliary learning and its generalization benefits in domain adaptation.

8.2. What makes the Rebuilder pluggable?

To enable MRM to be compatible with arbitrary architectures and simultaneously incorporate the decoder into auxiliary task training, we introduce a Rebuilder module between the encoder and decoder to perform reconstruction-based decoding during training. This results in an inference process of $D(R(E(x)))$. Notably, although this training strategy modifies the original pipeline, the Rebuilder can be seamlessly removed during inference, restoring the original inference flow $D(E(x))$.

The ability to remove the Rebuilder during inference without compromising performance is attributed to hybrid training strategy. Specifically, during training, the decoder is jointly exposed to both the reconstructed features $R(E(x))$ and the original features $E(x)$ derived from both source and target domain samples. This allows the decoder to learn from a joint distribution encompassing both types of features. Consequently, at inference time, even in the absence of reconstructed inputs, the decoder remains effective, as the distribution of original features is subsumed within the training-time joint distribution. This ensures that removing the Rebuilder has minimal to no adverse impact on the inference process.

8.3. Difference from MIC

Although both MIC [20] and our Masked Representation Modeling use masking to improve domain-adaptive segmentation, they differ in where masking is applied and how the masked information is used.

MIC operates in the image space: masked images are passed through the network and the model is trained with a consistency objective between masked and unmasked predictions. In this sense, MIC is conceptually similar to a high-ratio CutOut-style augmentation, where large portions of the input are removed and the model learns robustness to severe input corruption.

In contrast, MRM performs masking in the latent representation space. Instead of masking pixels, we mask encoder features and reconstruct the missing representations from the visible ones, and the reconstructed features are optimized using the standard segmentation objective.

These mechanisms regularize the model at different levels. MIC primarily improves robustness to input-level corruption, while MRM enforces semantic consistency in feature space. As a result, the two approaches are conceptually distinct and often complementary in practice.

8.4. Why does MRM work?

While prior work has demonstrated that masked image modeling (MIM) [2, 16, 50] enhances the representation capacity of the backbone by training with masked inputs and strengthens contextual modeling—findings also supported

by previous qualitative analyses—MRM operates differently. Unlike MIM, which masks input tokens in the image space, MRM performs occlusion and reconstruction directly in the latent space. As a result, the backbone processes the full input signal, rather than only the visible tokens as in MAE [16] for vision or BERT [12] for NLP. This distinction suggests that the effectiveness of MRM may stem from additional factors beyond those commonly attributed to MIM.

We observe that MRM exhibits notable similarities with feature-level mixup [46]. Specifically, both employ a loss function aligned with the primary task during training, and their feature fusion strategies are remarkably analogous.

$$f^r = M^s \odot f^o + (1 - M^s) \odot f^t \tag{8}$$

$$f^{mix} = \lambda_{mix} \cdot f^i + (1 - \lambda_{mix}) \cdot f^j \tag{9}$$

Equation (8) specifies the MRM-based representation reconstruction operation, whereas Equation (9) represents feature-level mixup. Two primary distinctions can be observed: (i) MRM conducts fusion in a local region of the feature map, in contrast to the global fusion strategy used in standard feature-level mixup. Notably, feature-level mixup can also be extended to support local fusion. (ii) The sources of fusion differ—mixup leverages features from a different sample, while MRM generates a fusion target from the features of the same input. Given their structural similarity, MRM can be interpreted as a special case of feature-level mixup. This perspective enables us to analyze the effectiveness of MRM through the lens of feature-level mixup.

Beyond its structural resemblance to feature-level mixup, MRM can also be interpreted through the lens of the Information Bottleneck (IB) principle [42]. The IB framework posits that a good representation Z should capture the minimal sufficient statistics of the input X with respect to the target Y, by maximizing the mutual information $I(Z;Y)$ while minimizing $I(Z;X)$. In this context, MRM implicitly acts as an information bottleneck by enforcing localized stochastic fusion in the feature space, effectively injecting structured noise into the intermediate representations. Unlike mixup—which introduces global perturbations across samples—MRM perturbs features locally and within the same instance. This intra-sample stochasticity encourages the network to retain task-relevant, spatially consistent features while suppressing instance-specific redundancy. As such, MRM serves as a self-supervised regularizer that biases the learned representation toward compression and abstraction.

From an IB perspective, MRM reduces $I(Z;X)$ by removing local pixel-level details, while preserving or even enhancing $I(Z;Y)$, particularly when the reconstruction loss aligns with the downstream task. This enables MRM to compress input information in a manner that preserves its task-relevant aspects, improving generalization performance in downstream tasks.