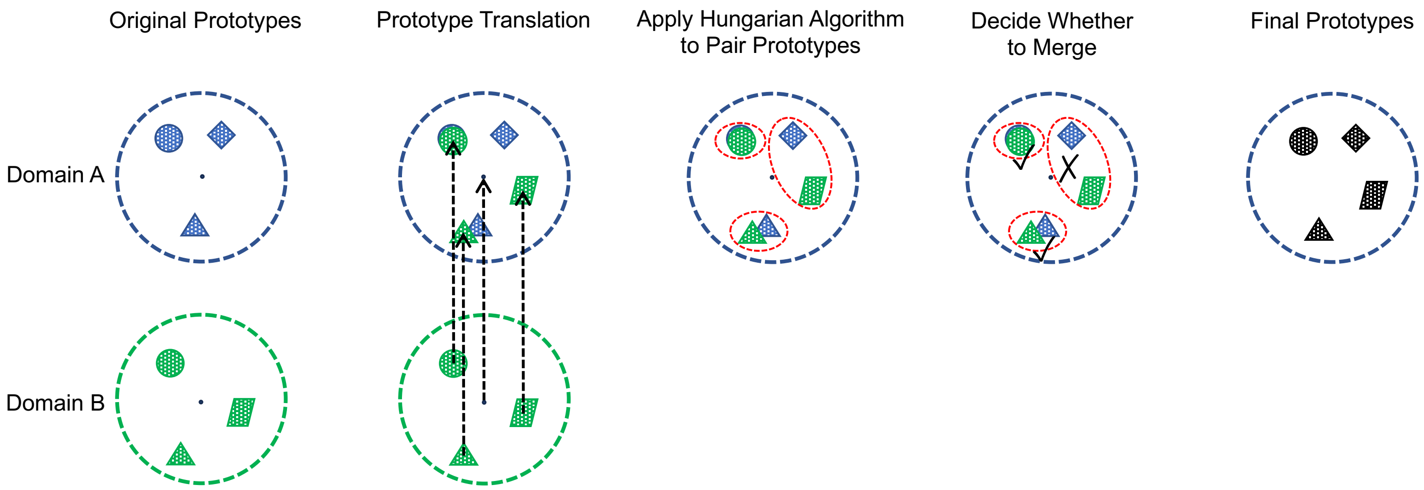

Figure 1: The visualization of Cross-Domain Prototype Conversion in the UEM framework, which involves the following steps: 1) As an example, for prototype conversion of domain A, we first translate all prototypes of domain B along the vector connecting the centers of the two domains to domain A. 2) Next, all prototypes in domain A use the Hungarian algorithm to find the nearest translated domain B prototypes. 3) For each prototype pair determined by the Hungarian algorithm, we check if they satisfy the merging condition (Eq.9 in the paper). If they do, the prototypes are merged by averaging; if not, they remain unchanged. 4) Finally, the prototype set for domain A is composed of the merged prototypes, the unmerged original domain A prototypes, and the unmerged translated domain B prototypes.

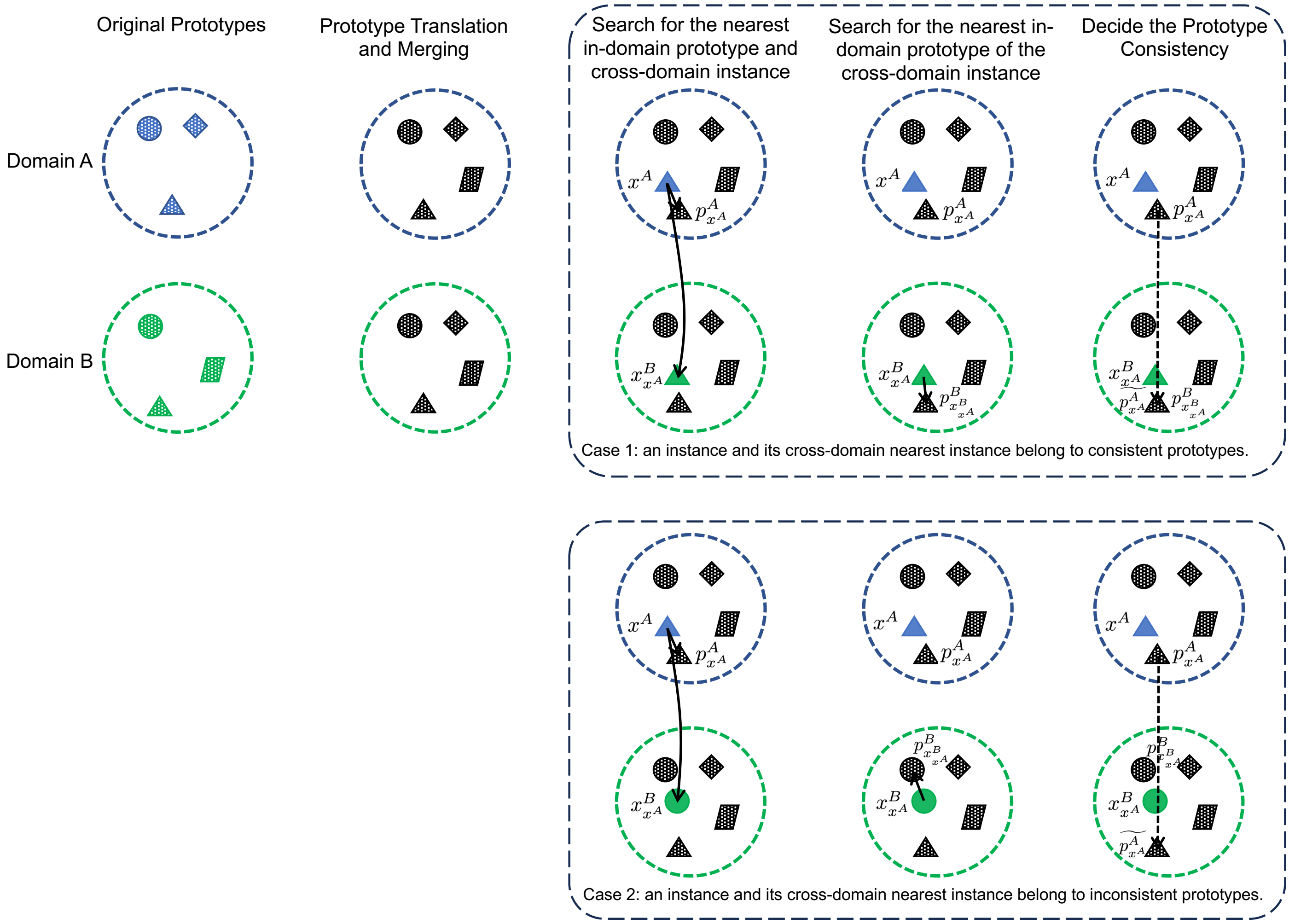

Figure 2: The visualization of Switchable Nearest Neighboring Match in the UEM framework, which involves the following steps: 1) The prototypes of domains A and B are transformed according to the previously described prototype conversion strategy. 2) For an instance x^A in domain A, we first find its nearest domain A prototype $p^A_{x^A}$ based on the product of cosine similarity and Euclidean distance. We also find x^A's nearest domain B instance $x^B_{x^A}$ using the same criteria (refer to Eq.15 and Eq.16 in the paper). 3) For the nearest domain B instance $x^B_{x^A}$, we then find its nearest domain B prototype $p^B_{x^B_{x^A}}$. 4) If $p^B_{x^B_{x^A}}$ matches $p^A_{x^A}$ across domains (i.e., if we translate $p^A_{x^A}$ from domain A to B, the translated $\widetilde{p^A_{x^A}}$ is the same as $p^B_{x^B_{x^A}}$), we consider x^A and $x^B_{x^A}$ as a positive pair in contrastive learning (Eq.17). Otherwise, we only consider x^A and $\widetilde{p^A_{x^A}}$ as a positive pair in contrastive learning.