Understanding the Limits of Unsupervised Domain Adaptation via Data Poisoning

Akshay Mehra

Joint-work with **Jihun Hamm**, **Bhavya Kailkhura** & **Pin-Yu Chen**.

Supervised learning

- Aim of supervised learning is to learn a model using a labeled training set such that it generalizes to an unknown test set.
- The main assumptions is that training and test data are i.i.d. samples from a *common* underlying distribution.
- *a.k.a.* single domain setting.

Training data

Test data

Domain adaptation

- In reality, the distributions of the training and test data may not be the same.

- Aim of domain adaptation is to transfer the knowledge from one domain to another using
  - A labeled sample from the source distribution.
  - A sample from the target distribution :
    - If the sample is unlabeled, then we are in the *unsupervised domain adaption* (***UDA***) setting. (Focus of our work)
    - If the sample is labeled, then we are in the *semi-supervised domain adaption* setting.

Source domain data

Target domain data

Successful learning in the UDA setting

- Notation:
  - $\mathcal{X}$ denotes the data domain and $\mathcal{D}$ denotes a distribution on this domain.
  - $f: \mathcal{X} \to [0,1]$ is a deterministic labeling function.
  - $g: \mathcal{X} \to \mathcal{Z}$ is a map from data to the representation space and $h: \mathcal{Z} \to [0,1]$ is a hypothesis in the representation space.
  - $\tilde{p}(z)$ is the density function of the distribution induced by g on $\mathcal{Z}$ and $\tilde{f}(z) \coloneqq \mathbb{E}_{\mathcal{D}}[f(x)|g(x) = z]$ be the induced labeling function.
  - $e(h) = \mathbb{E}_{z\sim\tilde{D}}\left[\left|\tilde{f}(z) - h(z)\right|\right]$ is the misclassification error w.r.t. the induced labeling function.
  - $D_1(\tilde{p}, \tilde{p}') = \int_{\mathcal{Z}} |\tilde{p}(z) - \tilde{p}'(z)|\, dz$ be the total variation distance.

- Upper bound on the target domain error [Ben-David, Shai, et al. 2007, 2010]

$$e_{target}(h) \leq e_{source}(h) + D_1\left(\tilde{p}_{source}, \tilde{p}_{target}\right) + \min\left\{e_{target}\left(\tilde{f}_{source}, \tilde{f}_{target}\right), e_{source}\left(\tilde{f}_{source}, \tilde{f}_{target}\right)\right\}.$$

  - Error on the target domain, $e_{target}(h)$.
  - Error on the source domain, $e_{source}(h)$.
  - Divergence measure between the marginal distributions of the two domains, $D_1\left(\tilde{p}_{source}, \tilde{p}_{target}\right)$.
  - Error of the induced labeling functions across the two domains, $\min\left\{e_{target}\left(\tilde{f}_{source}, \tilde{f}_{target}\right), e_{source}\left(\tilde{f}_{source}, \tilde{f}_{target}\right)\right\}$.

Popular methods for UDA

- Algorithms for UDA aim to minimize the upper bound on the target domain error.

$$e_{target}(h) \leq e_{source}(h) + D_1\left(\tilde{p}_{source}, \tilde{p}_{target}\right) + \min\left\{e_{target}\left(\tilde{f}_{source}, \tilde{f}_{target}\right), e_{source}\left(\tilde{f}_{source}, \tilde{f}_{target}\right)\right\}.$$

- Due to the lack of labels from the target domain data many algorithms solve the following problem

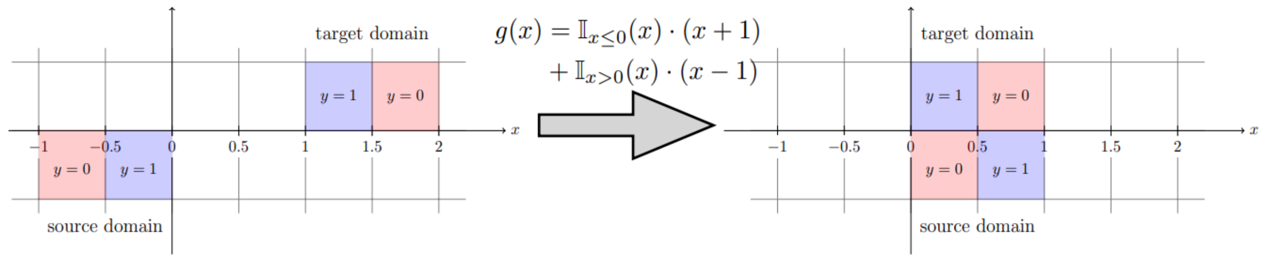
$$\min_{g,h}\ e_{source}(h) + D_1\left(\tilde{p}_{source}, \tilde{p}_{target}\right).$$

- Some popular algorithms
  - DANN [Ganin et al. 2016]
  - DAN [Long et al. 2015]
  - CDAN [Long et al. 2017]
  - MCD [Saito et al. 2018]
  - SSL [Xu et al. 2019]
  - CYCADA [Hoffman et al. 2018]
  - …

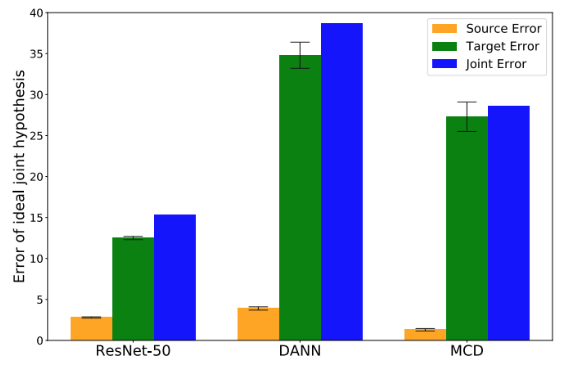
Evidence of failure of UDA methods

- The algorithms that solve $\min_{g,h} e_{source}(h) + D_1(\tilde{p}_{source}, \tilde{p}_{target})$ could increase the error of the ideal joint hypothesis, $\lambda^* \coloneqq \min_{h' \in \mathcal{H}} \epsilon_{source}(h') + \epsilon_{target}(h')$, which increases the upper bound on the target domain error.

- Zhao et al. 2019, Johansson et al. 2019:



- Liu et al. 2019, Wang et al. 2019:



A necessary condition for learning in the UDA setting

- **First contribution**: New lower bound on the target domain error to provably explain the failure of learning in the UDA setting.

Theorem 1: *Let $\mathcal{H}$ be the hypothesis class and $\mathcal{G}$ be the class of representation maps. Then $\forall h \in \mathcal{H}$ and $g \in \mathcal{G}$,*

$$e_{target}(h) \geq \max\{e_{target}(\tilde{f}_{source}, \tilde{f}_{target}), e_{source}(\tilde{f}_{source}, \tilde{f}_{target})\} - (e_{source}(h) + D_1(\tilde{p}_{source}, \tilde{p}_{target})).$$

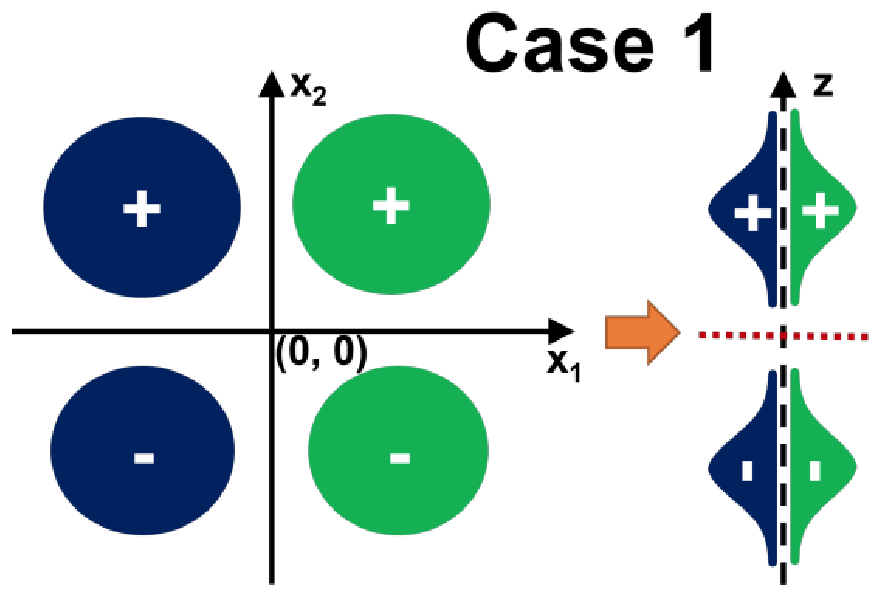
- UDA algorithms which solve, $\min_{g,h} e_{source}(h) + D_1(\tilde{p}_{source}, \tilde{p}_{target})$, increase the lower bound.

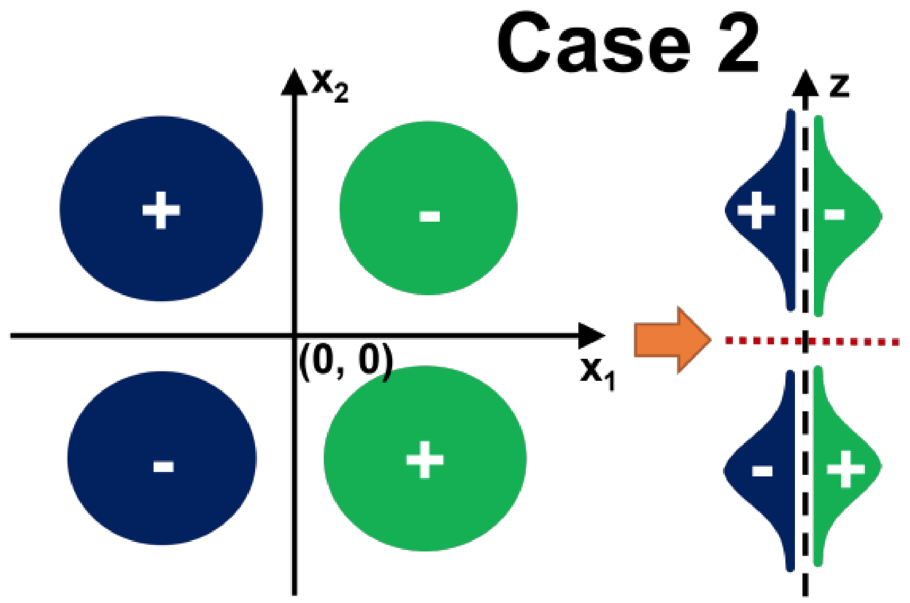
- UDA methods only guarantee $e_{target}(h)$ to be close to the labeling function mismatch $e(\tilde{f}_{source}, \tilde{f}_{target})$.

Corollary 1: $\forall h \in \mathcal{H}$ *and* $g \in \mathcal{G}$,

$$|e_{target}(h) - e_{source}(\tilde{f}_{source}, \tilde{f}_{target})| \leq e_{source}(h) + D_1(\tilde{p}_{source}, \tilde{p}_{target}),$$
$$|e_{target}(h) - e_{target}(\tilde{f}_{source}, \tilde{f}_{target})| \leq e_{source}(h) + D_1(\tilde{p}_{source}, \tilde{p}_{target}).$$

Illustrative cases of learning in the UDA setting



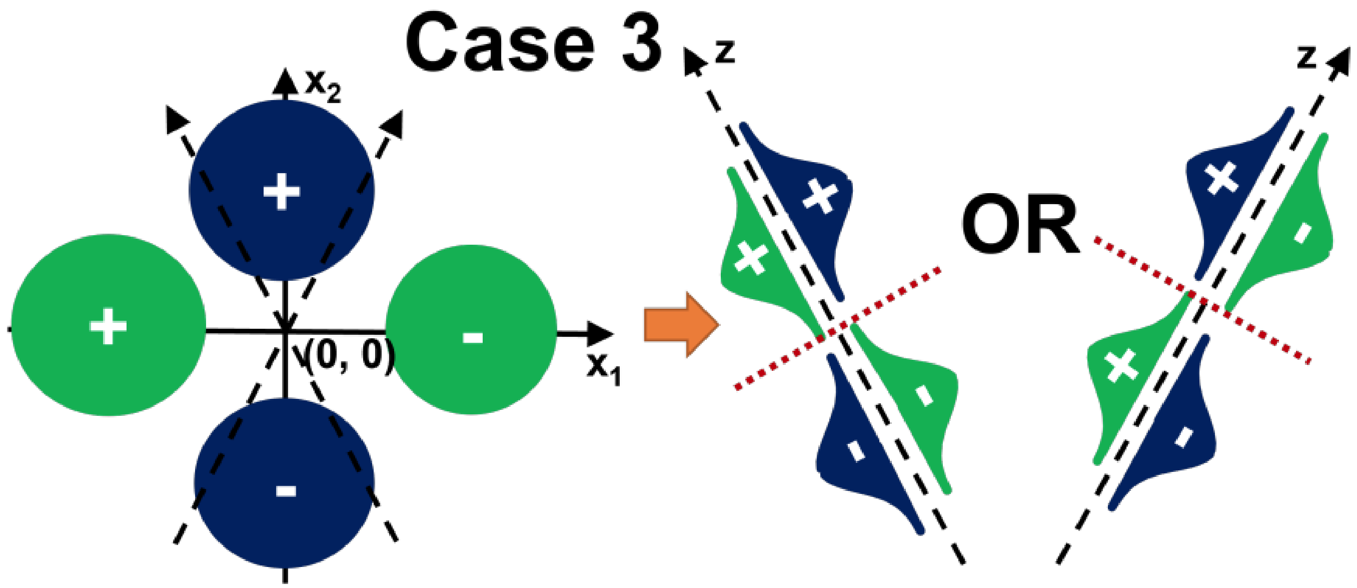


Good Case: Same classes are closer across the two domains aiding UDA methods in aligning the two domains correctly.

Bad Case: Opposite classes are closer across the two domains leading UDA methods to align the incorrect classes.

🟦 SOURCE 🟩 TARGET

Illustrative cases of learning in the UDA setting

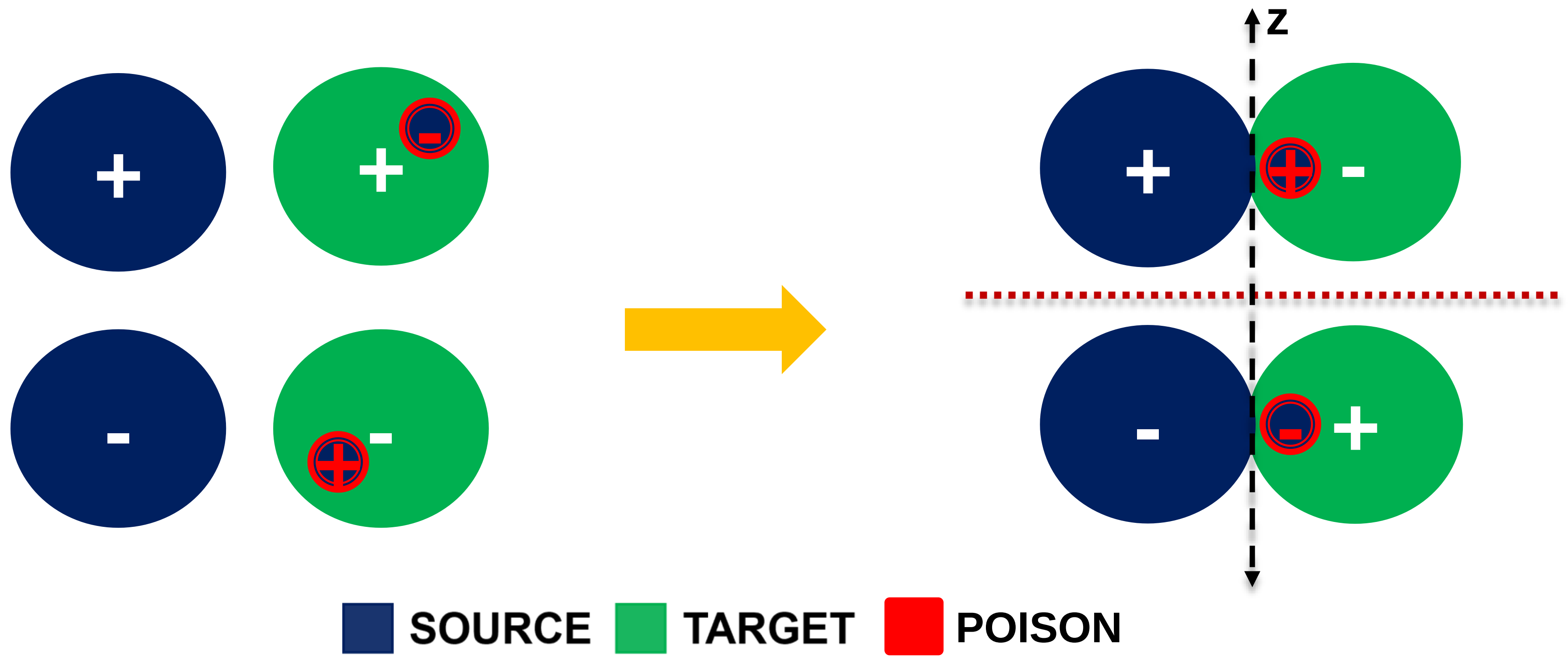


Ambiguous Case: Due to the absence of labeled data from the target domain presence of a small amount of correct or incorrect labels of the target domain can lead UDA methods to produce drastically different representations.

🟦 SOURCE 🟩 TARGET

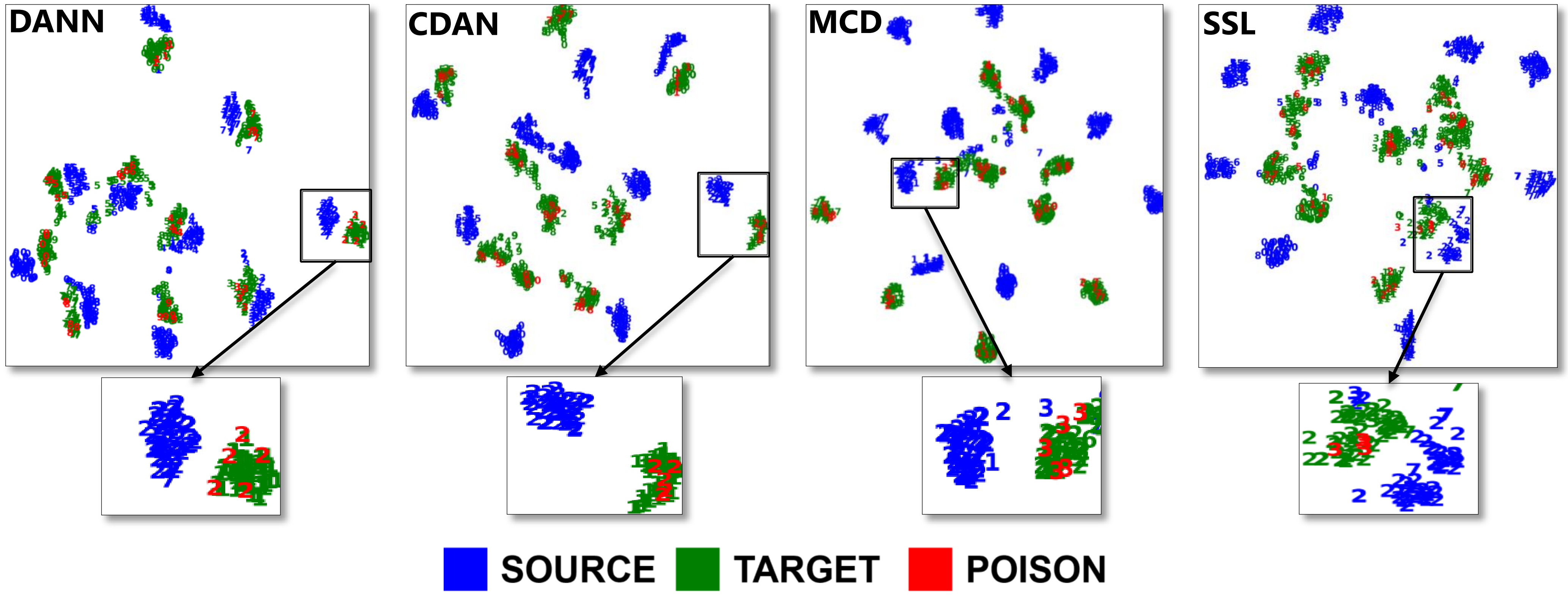
Effect of a small amount of poisoned data

- In this attack, small amount of mislabeled target domain data is added as poison data to the source domain. This leads UDA methods to align the wrong classes from the two domains.



Poisoning using mislabeled data

- t-SNE visualization of the effect of poisoning
  - Incorrect classes from the two domains are aligned



Breaking UDA methods using data poisoning

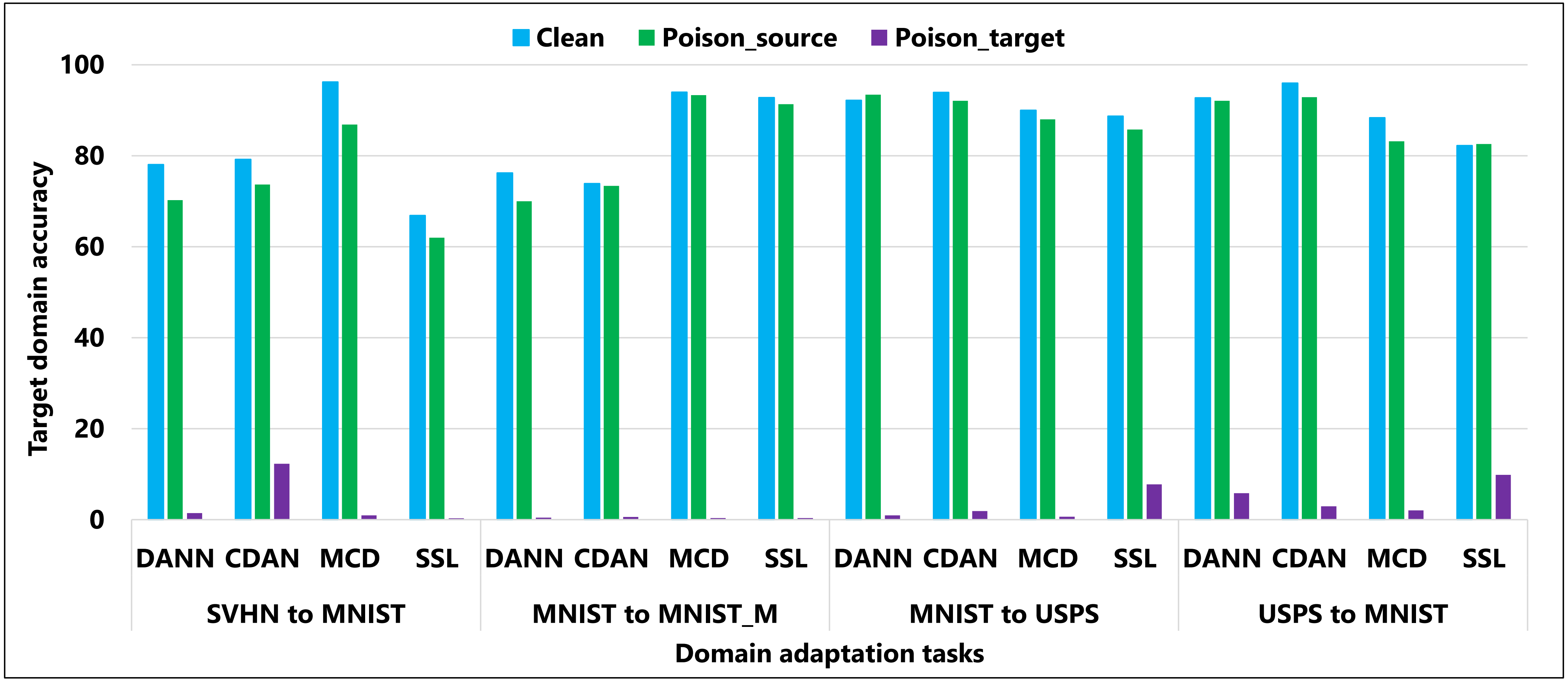
- **Second Contribution:** Novel data poisoning attacks to evaluate the effectiveness of UDA methods.

- Data poisoning attacks augment victim's training data with malicious data to affect the performance of the victim's model after training.

- Our attacks comprise of
  - Using mislabeled source/target domain data as poisons
  - Using watermarked data as poisons
  - Using correctly labeled data as poisons

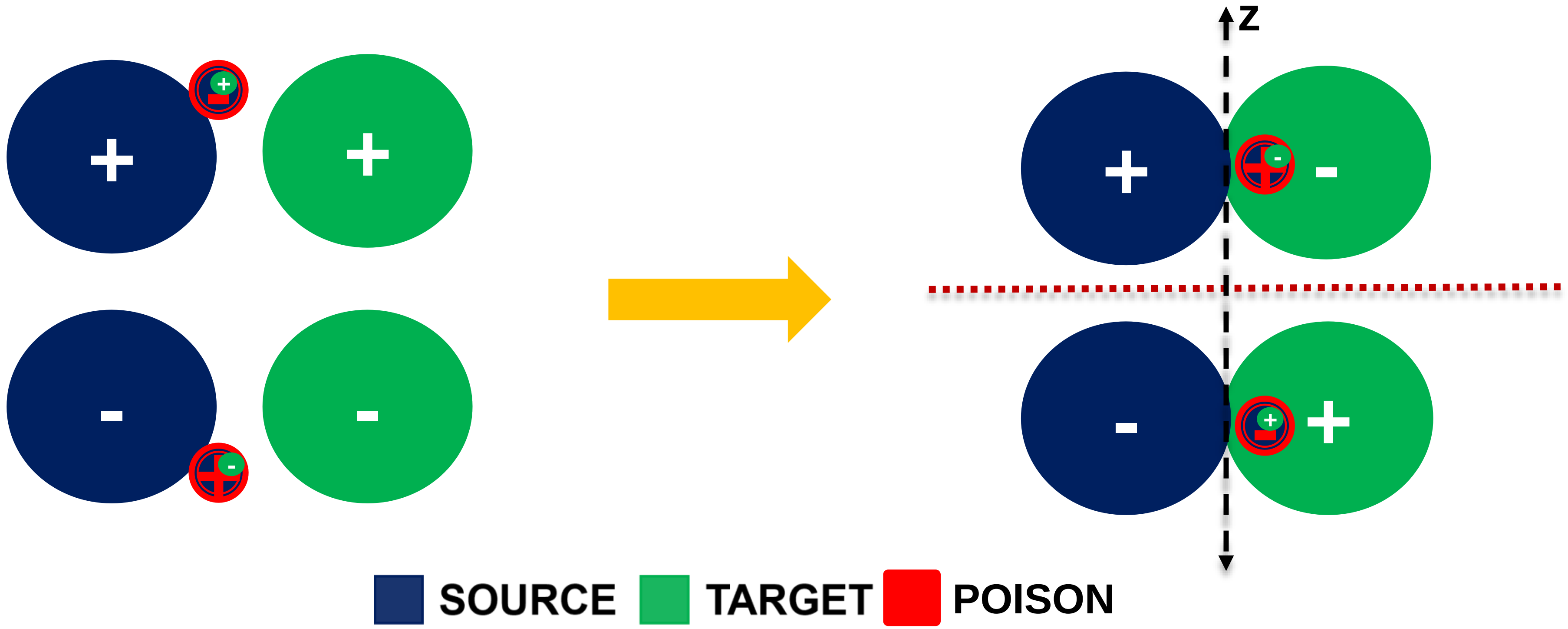
- Poisoning in the UDA setting is extremely effective, even with a small amount of poisoned data.

Poisoning using mislabeled data



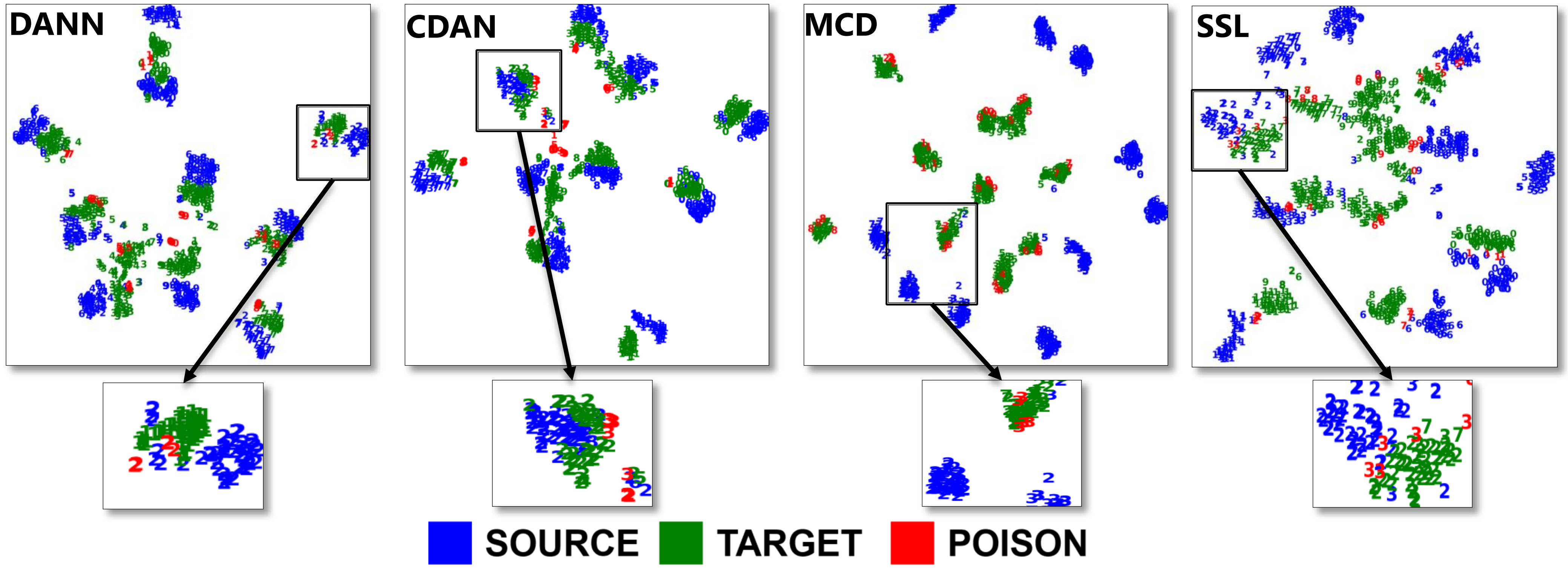
Poisoning using mislabeled watermarked data

- Attack with mislabeled watermarked data as poison data also leads UDA methods to align the wrong classes from the two domains.

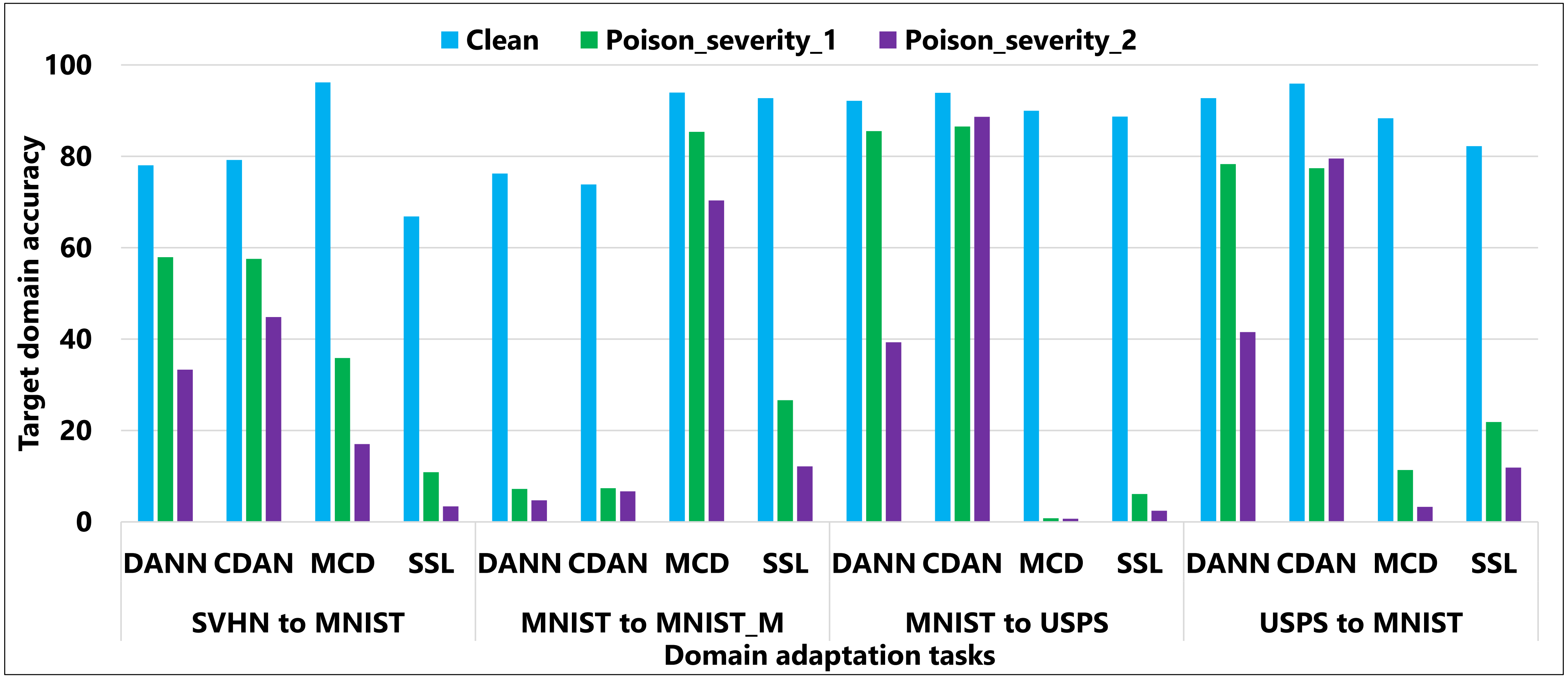


Poisoning using mislabeled watermarked data

- t-SNE visualization of the effect of poisoning
  - Incorrect classes from the two domains are aligned except for CDAN whose robustness can be attributed to the use of pseudo labels in the discriminator.

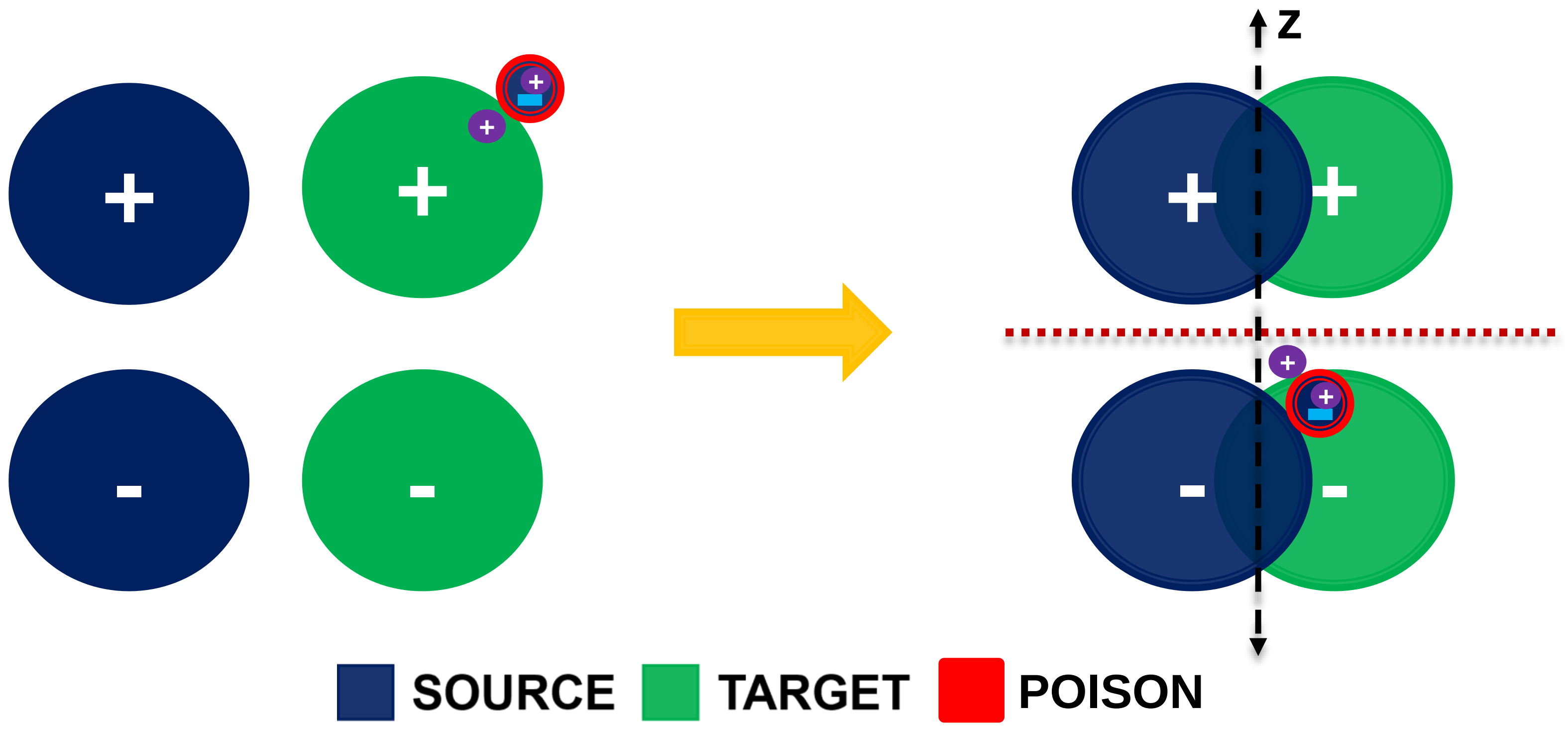


Poisoning using mislabeled watermarked data

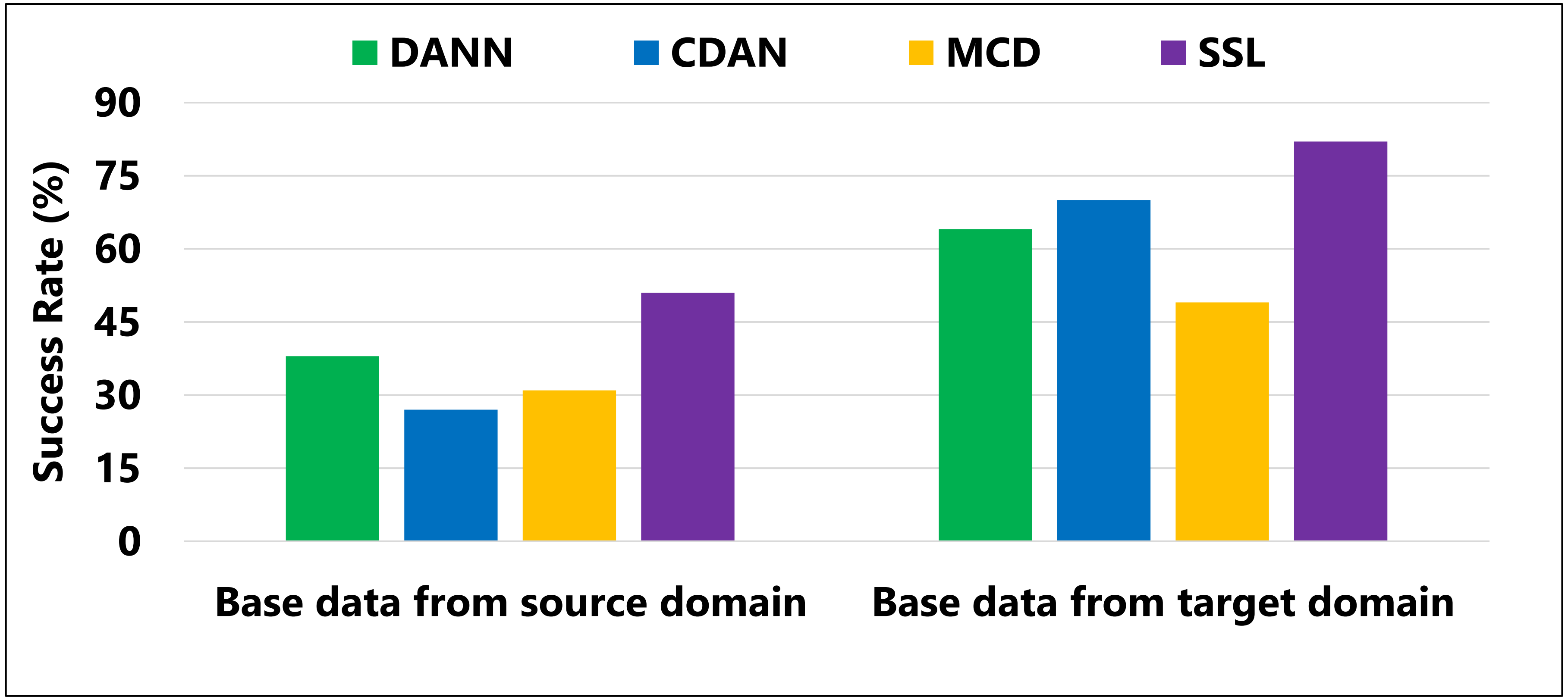


Poisoning using clean-label data

- Poison data is initialized by watermarking points from the base class (opposite to the class of the target point) with the target point and then optimized.



Poisoning using clean-label data



Attack success rate with a binary (3 and 8) domain adaptation problem on MNIST→ MNIST_M.

Conclusion

- We studied the problem of learning in the UDA setting and proposed a necessary condition for successful learning in form of a lower bound on the target domain error.
- We demonstrated different cases where current UDA methods are successful and when they fail to adapt the two domains. We also presented the ambiguous case where success and failure of UDA methods are equally likely.
- We presented novel data poisoning attacks which demonstrate the vulnerability of UDA methods to poisoning.
- Considering the sensitivity of the success of UDA methods to the data distribution we believe, future research in this direction should consider robustness to adversarial attacks as a measure of their performance.