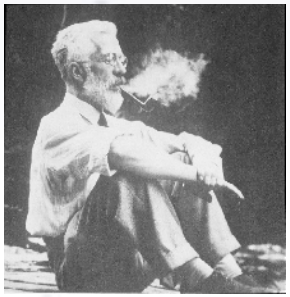
Applied ML

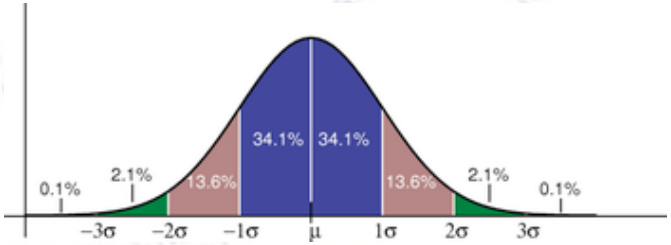
Domain Shifts & Adversarial Training








Troels C. Petersen (NBI)



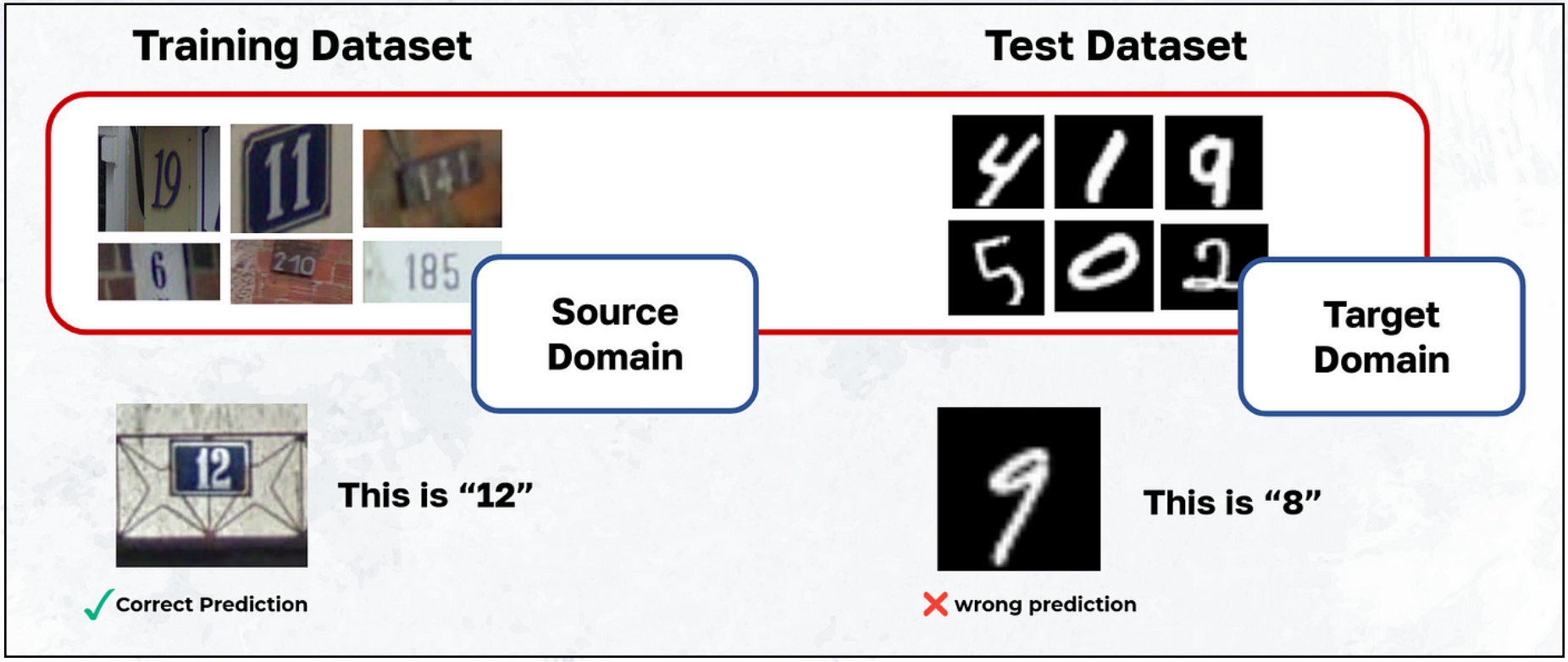
"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

Domain Shift

aka. Domain Gap and the want for Domain Adaptation

Domain Shifts

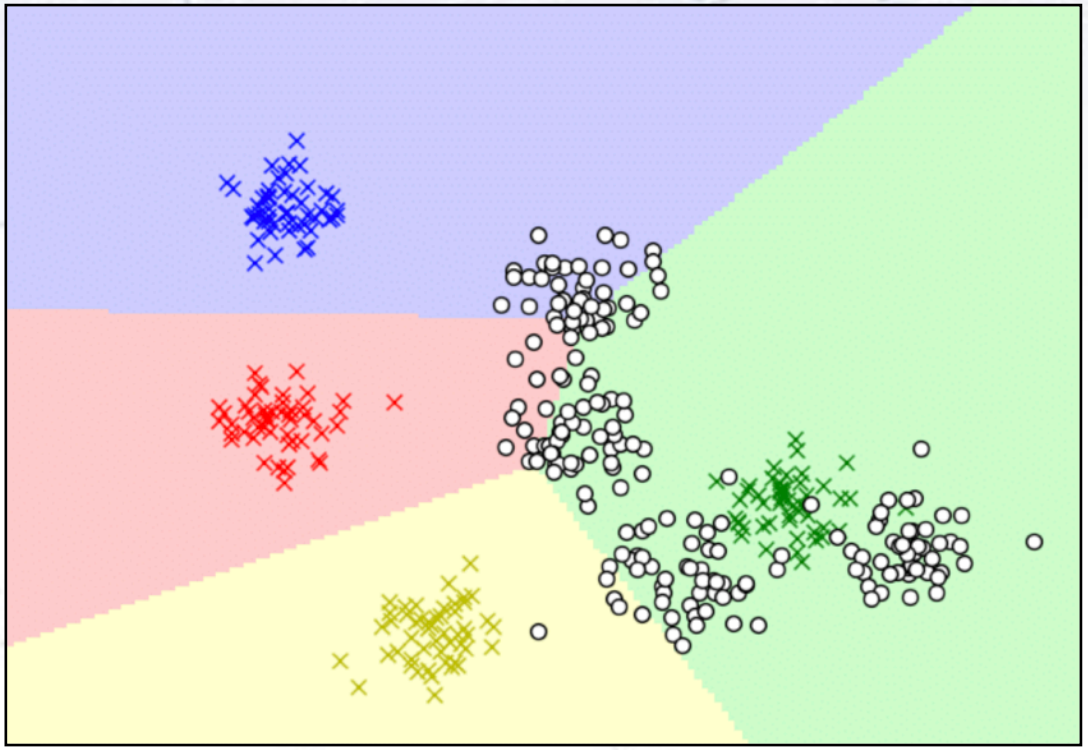
Imagine that you have trained a model to read **house numbers**, which works well. But now you apply it to **hand written numbers**… it it fails!



This problem is called "Domain Shift", i.e. there is **a "shift" between** the data that a model was **trained on** and the data that it was **applied to**.

Domain Shifts

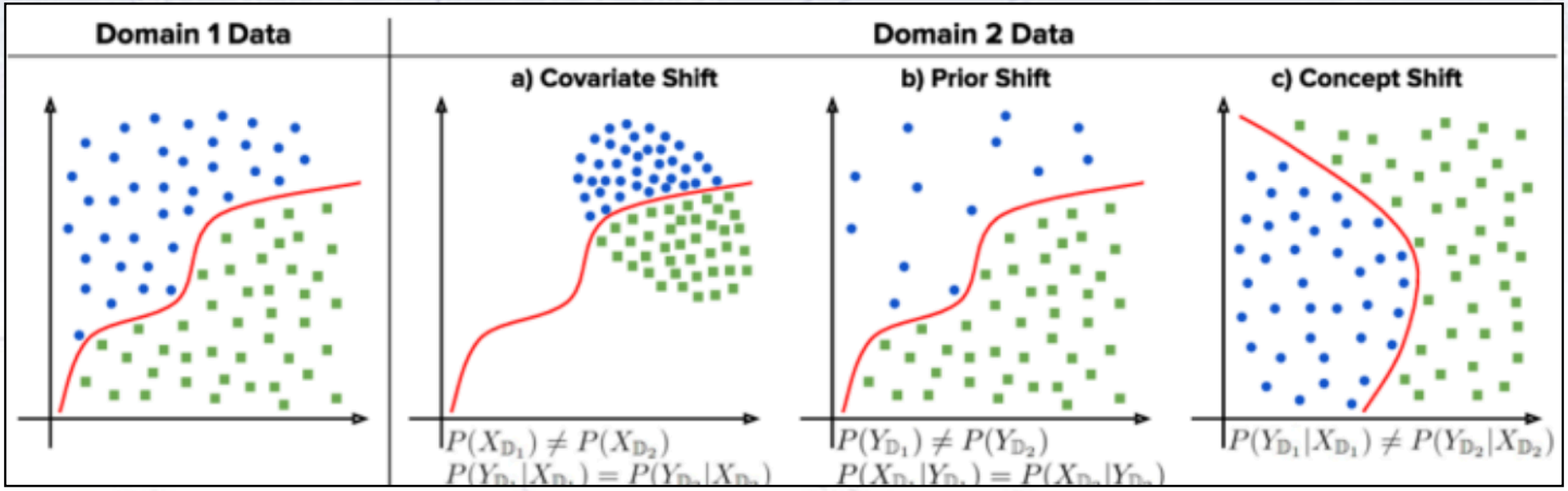
The typical cause is a shift in the input variables. One should therefore **always compare variable distributions between domains**. But even if these are alike, different correlations might still cause changes.



An illustration of domain shift where the distribution of the unlabelled target domain (white) is misaligned with the labelled source domain (color) but both distributions share structural similarities.

Types of Domain Shifts

There are several types of Domain Shifts (Covariate, Prior, and Conceptual):



Typically, real world domain shifts are a mixture of the above types.

A "simple fix" is to re-weight the original domain (1) data to match that of the target domain (2) data. Aim of re-weighting is to get **identical multidimensional distributions**. An effective method is given by **GBReweighter**.

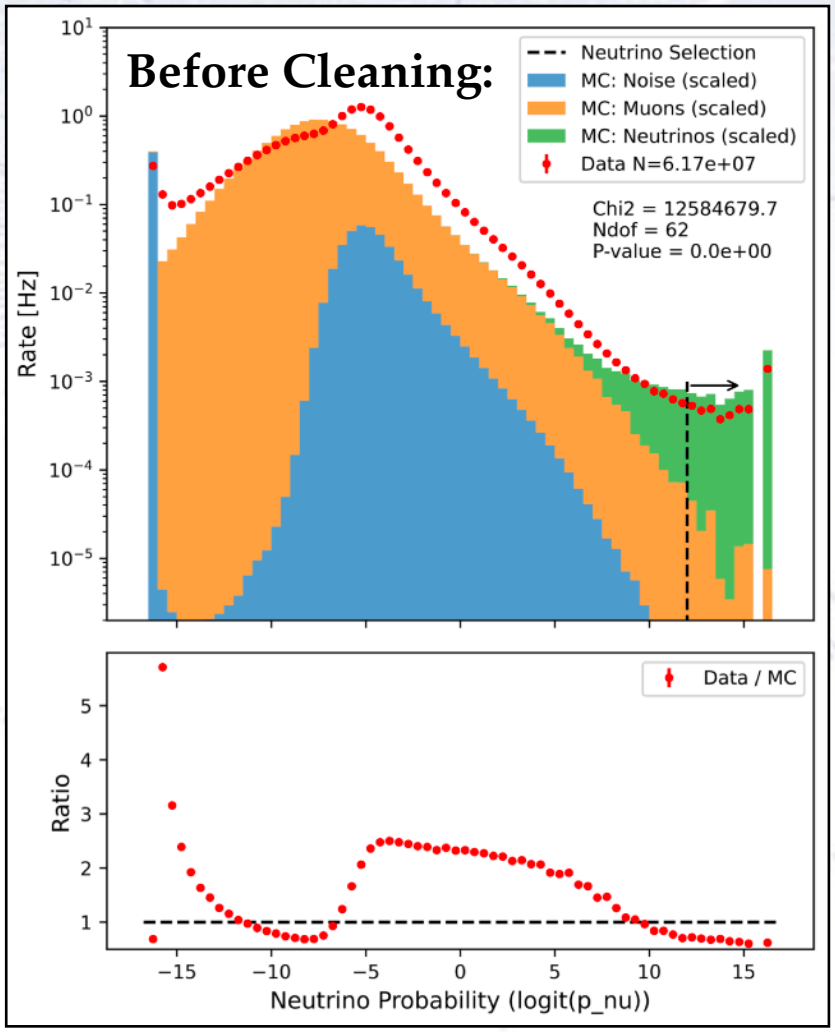
However, note that one can not re-weight events to cover a new domain.

Notes on Domain Shifts

Sometimes (often?), domain shifts are caused by the (lacking) **cleaning of data**.
An example of this (where we knew this) is from the IceCube experiment:

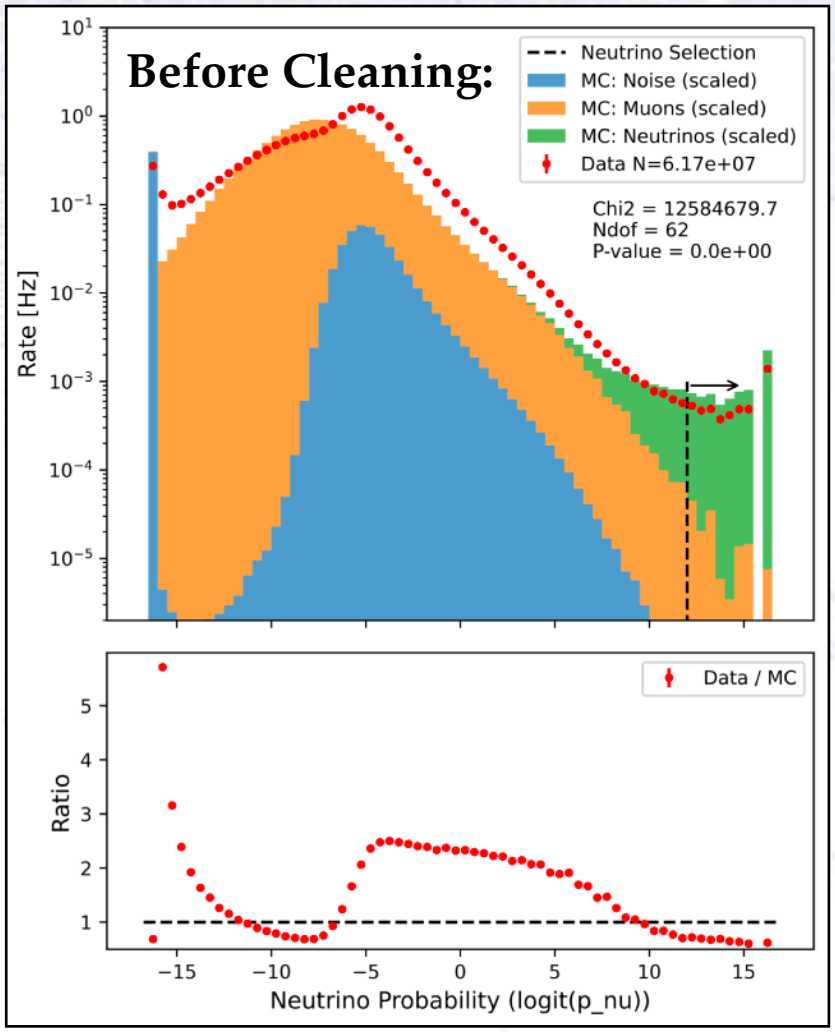
Notes on Domain Shifts

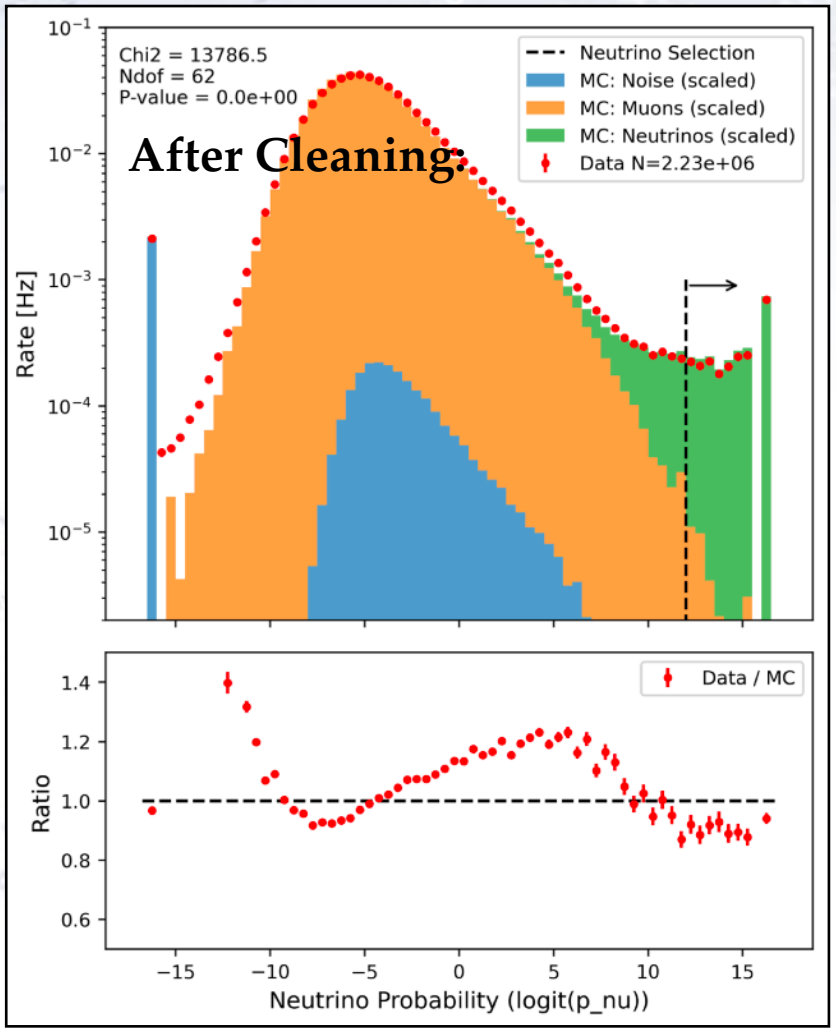
Sometimes (often?), domain shifts are caused by the (lacking) **cleaning of data**.
An example of this (where we knew this) is from the IceCube experiment:



Notes on Domain Shifts

Sometimes (often?), domain shifts are caused by the (lacking) **cleaning of data**.
An example of this (where we knew this) is from the IceCube experiment:

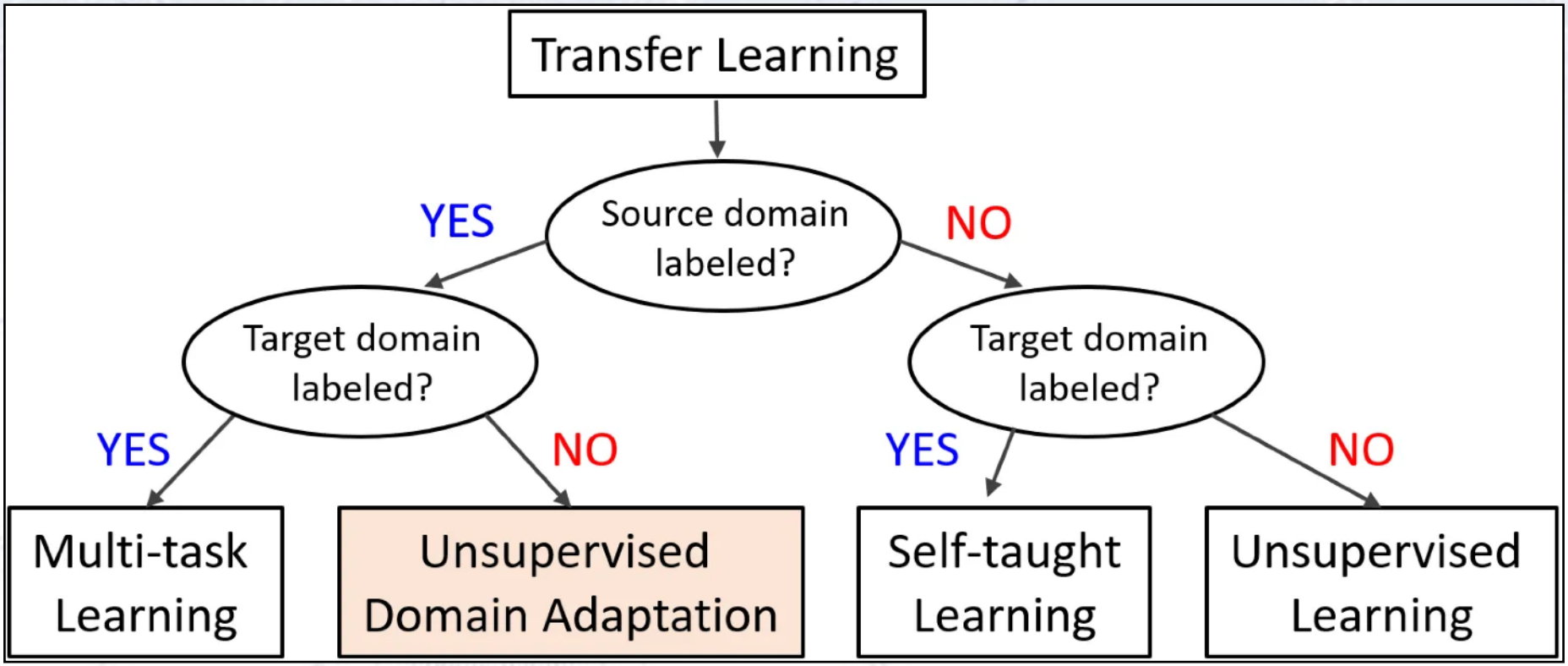




Adversarial Training

Domain Adaptation

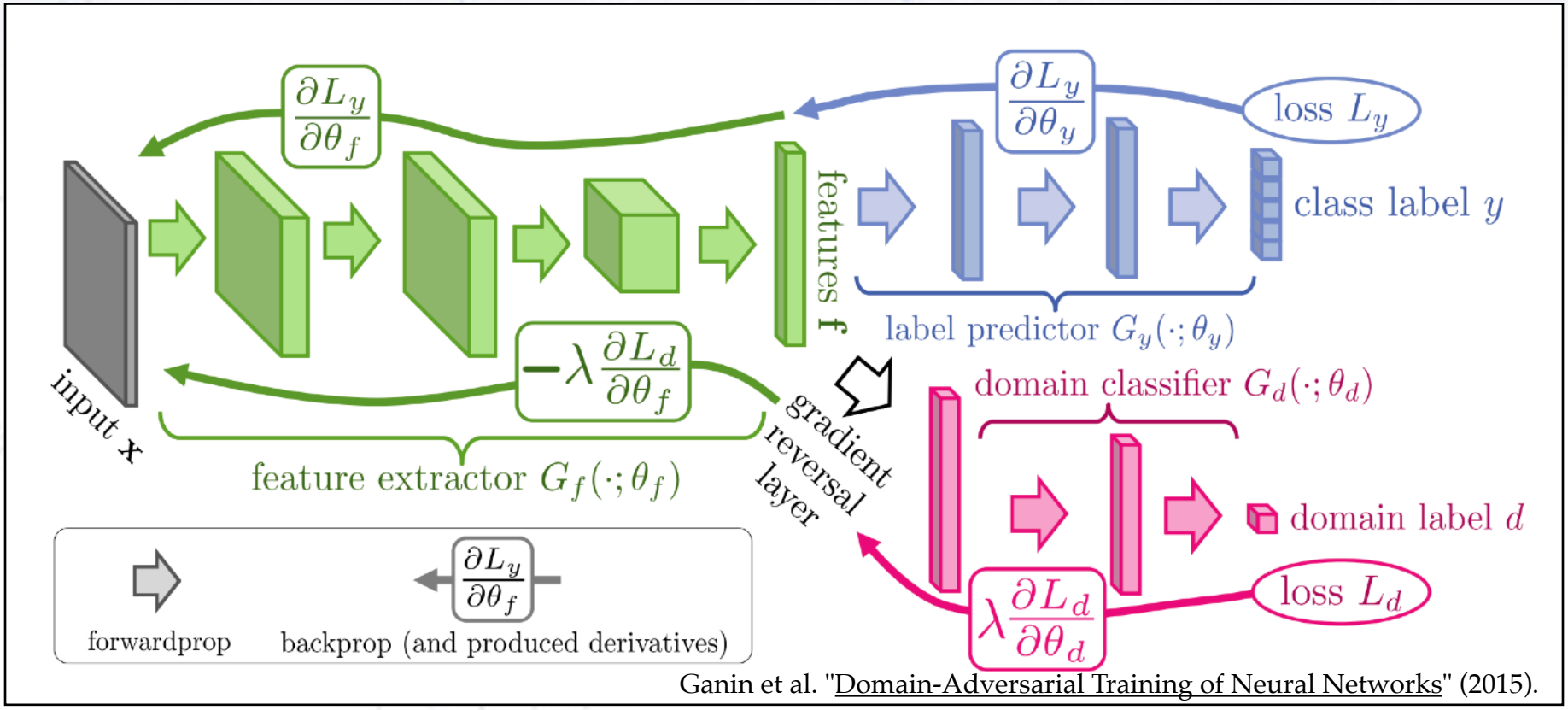
The



Adversarial Training

What if we could **make** the training (simulated) and testing (real) data alike?

Adversarial training is the idea to use an adversarial network to "enforce" this alikeness in a latent space in the network (typically in the last part).

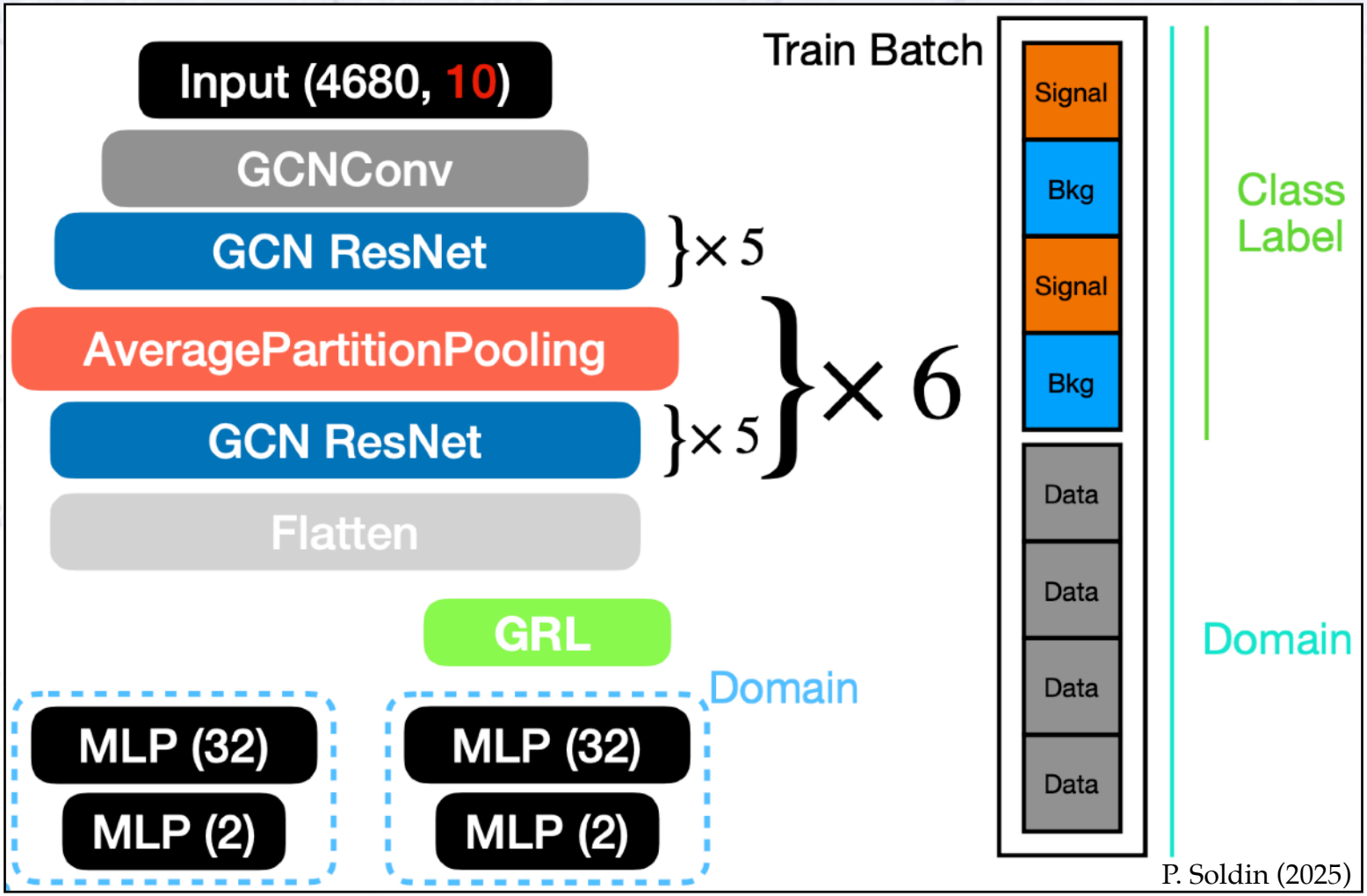


Ganin et al. "Domain-Adversarial Training of Neural Networks" (2015).

The adversarial "forces" to learn from features that are common in domains.

Example case

Using an advanced architecture (Graph NN++) **the flattened latent space is forced to be alike between MC and data** with a small and shallow MLP.



Domain Adaptation

Advantages of Unsupervised Domain Adaptation:

- **Cost-Effective:** It enables models to perform well in new domains without requiring expensive labelling efforts.
- **Adaptability:** It helps models generalise better, which is particularly valuable for applications that need to function in dynamic environments.
- **Broad Applicability:** Works well in scenarios where labeled data is readily available in one domain but not in another, such as simulated and real data.

Drawbacks of Unsupervised Domain Adaptation (UDA):

- **Feature Space Assumptions:** It assumes that features extracted from both domains are similar enough for alignment, which may not always hold.
- **Training Complexity:** Some methods, like adversarial training, require careful tuning. Otherwise, the training process can become unstable.
- **Limited Scope Without Label Feedback:** Since no labels are available in the target domain, the model can only rely on indirect signals. It is thus vital to have methods to cross check the performance in the target domain.

Personally, I had a “Damn, that is brilliant” experience learning of this.

Supervised Domain Adaptation

In Supervised Domain Adaptation, the model leverages labeled data from both the source and target domains to minimise the differences between the two.

The assumptions are thus:

- The model has **access to labeled data** in both the *source* and *target domains*.
- The features in source and target domains may have **slight differences**, but the model can use labeled target data to **adapt and improve performance**.

The model can be fine-tuned or retrained to perform well on the target data by:

- **Combining Training Data:** The model is trained on a **combination** of source and target data. This helps it learn general patterns from the source domain while also adapting to specific characteristics of the target domain.
- **Fine-Tuning:** If a pre-trained model exists, it can be fine-tuned using the labeled data from the target domain. This allows the model to adapt to new data without starting the training process from scratch (Foundation Model).