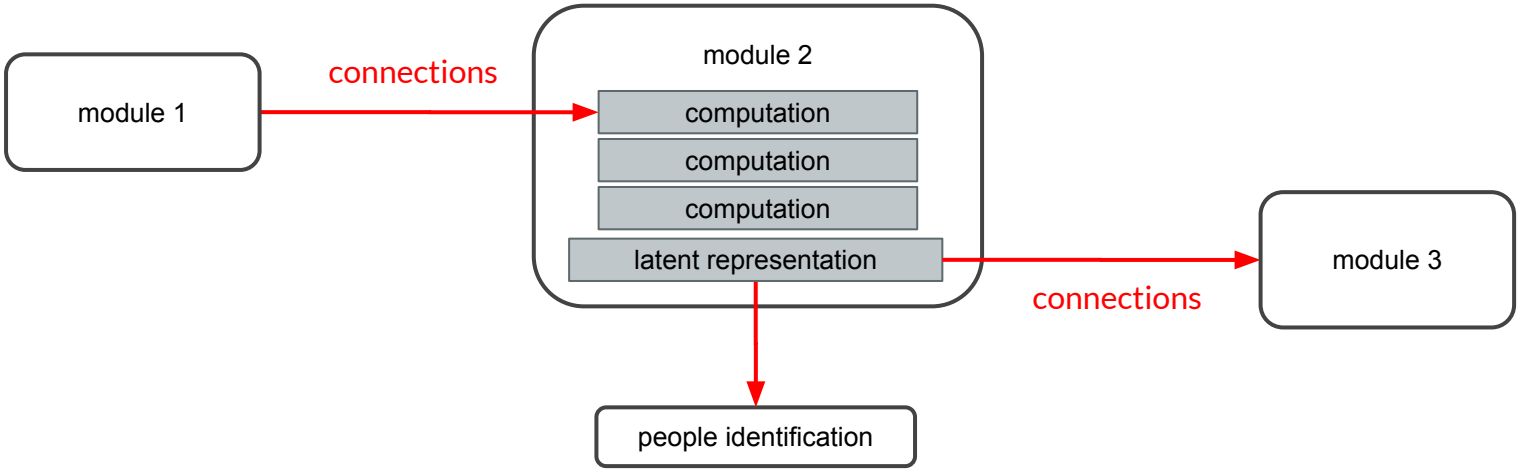
Meaning-Preserving Continual Learning v1

Intuition Pumps

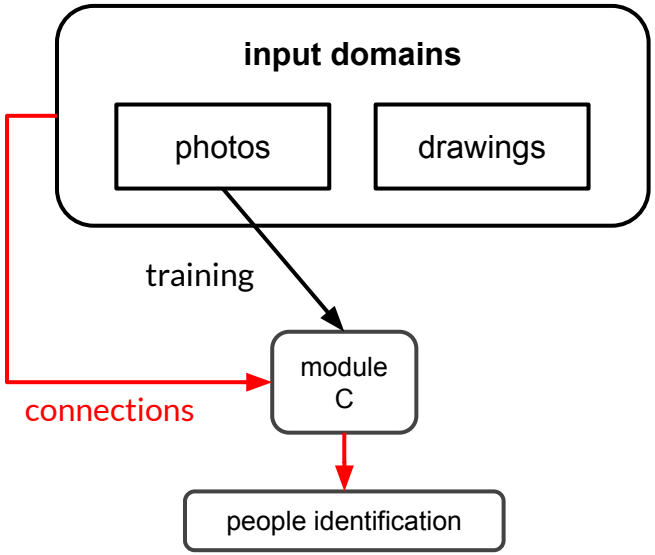
Intuition Pumps (1)

In MPCL, modules compute latent representations, which in turn are connected to:

1. An anchoring function. Below, module 2's function is to identify people ;
2. Other modules.



Intuition Pumps (2)



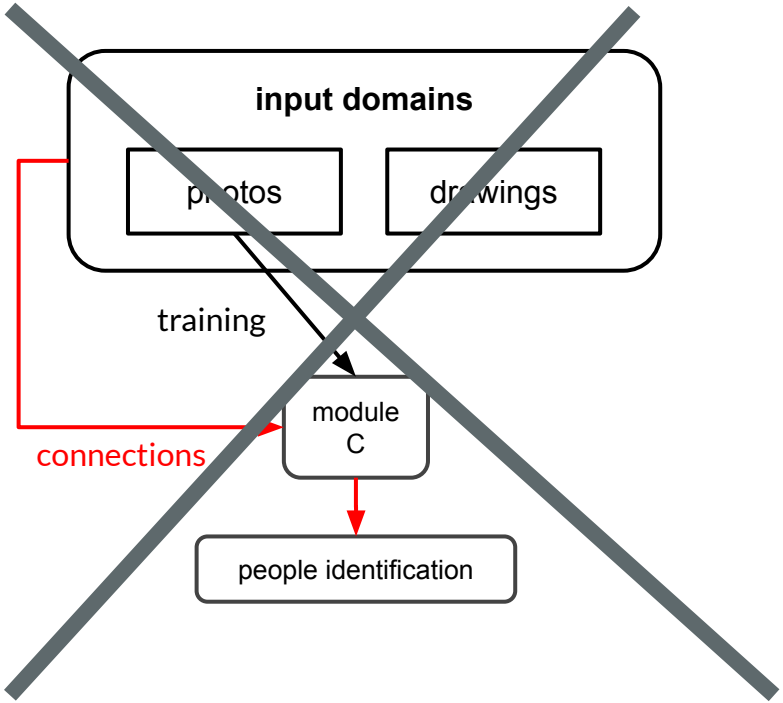
Module C's function is to identify people.

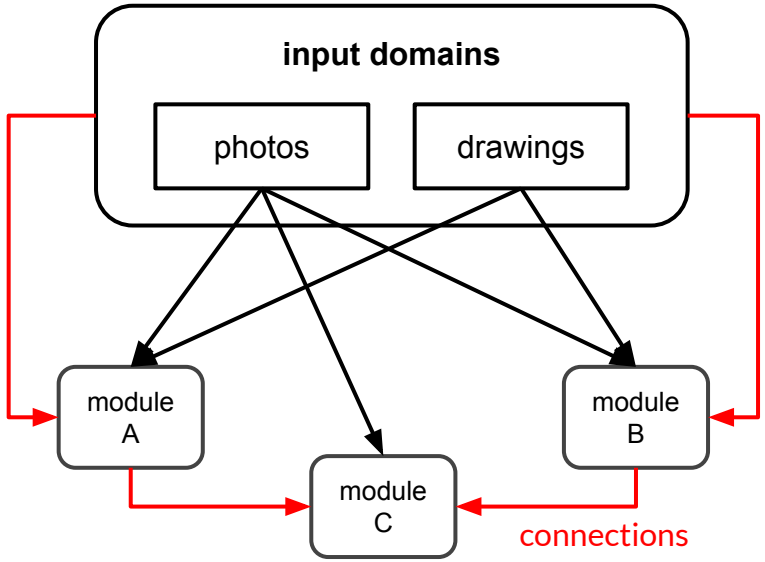
Truism: if module C is trained from photographs only, it is not going to transfer well to drawings.

By expecting module C to generalize to drawings, you tacitly expect C to know what "people" means.

Module C doesn't know what "people" means.
It's not a concept that a basic model can grasp by sitting in a server room.

Intuition Pumps (3)

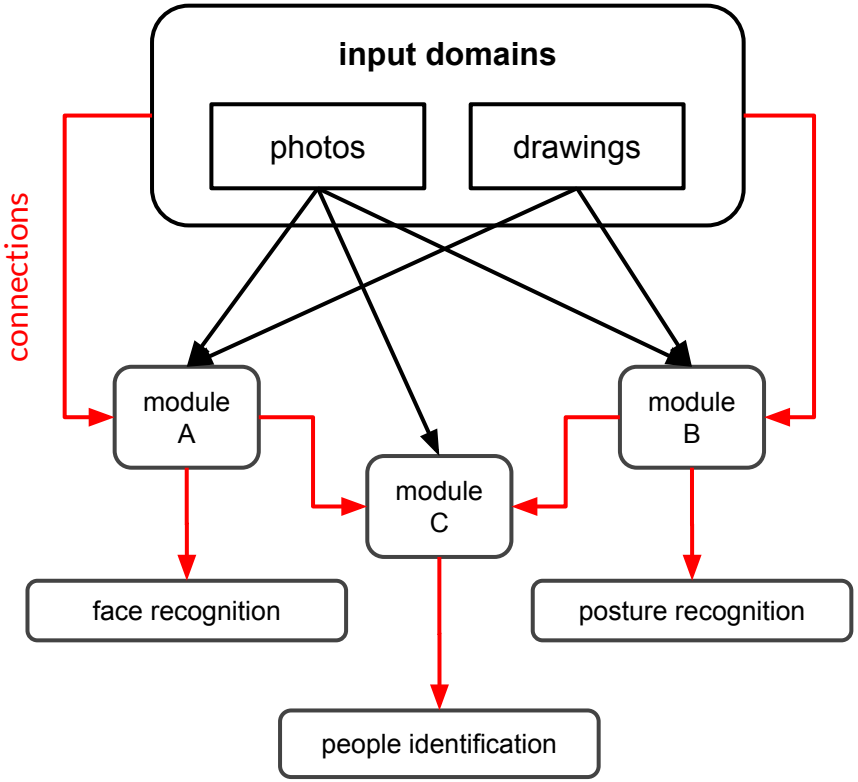




Here too, module C is trained from photographs only.

However, as module A and B were trained from both photos and drawings, module C is a lot more likely to successfully transfer to drawings.

Intuition Pumps (4)



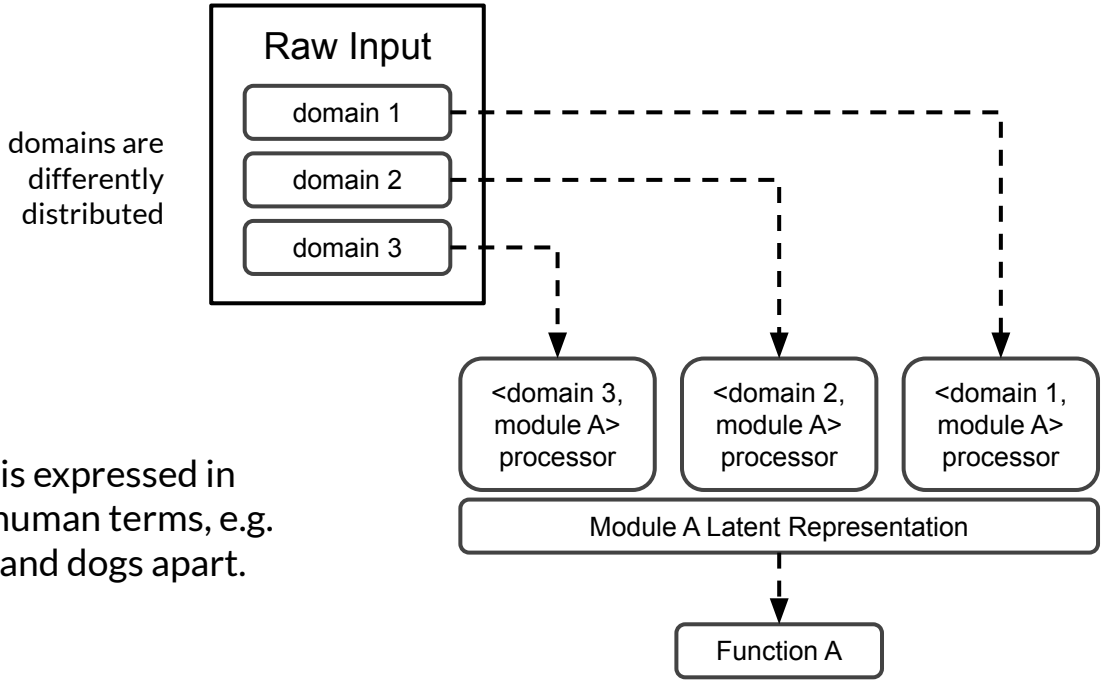
Furthermore, as module A is trained separately via an anchoring face recognition function, and provided it is accurate on both photos and drawings, faces will be represented with the same vectors in photos and drawings (*).
It follows that if module C utilizes John's face and posture to identify him in photos, it will also identify him in drawings.

This can be seen as a form of curriculum learning or machine teaching.

(*) under some conditions that are detailed in the LaTex document.

Modules

Module Anatomy

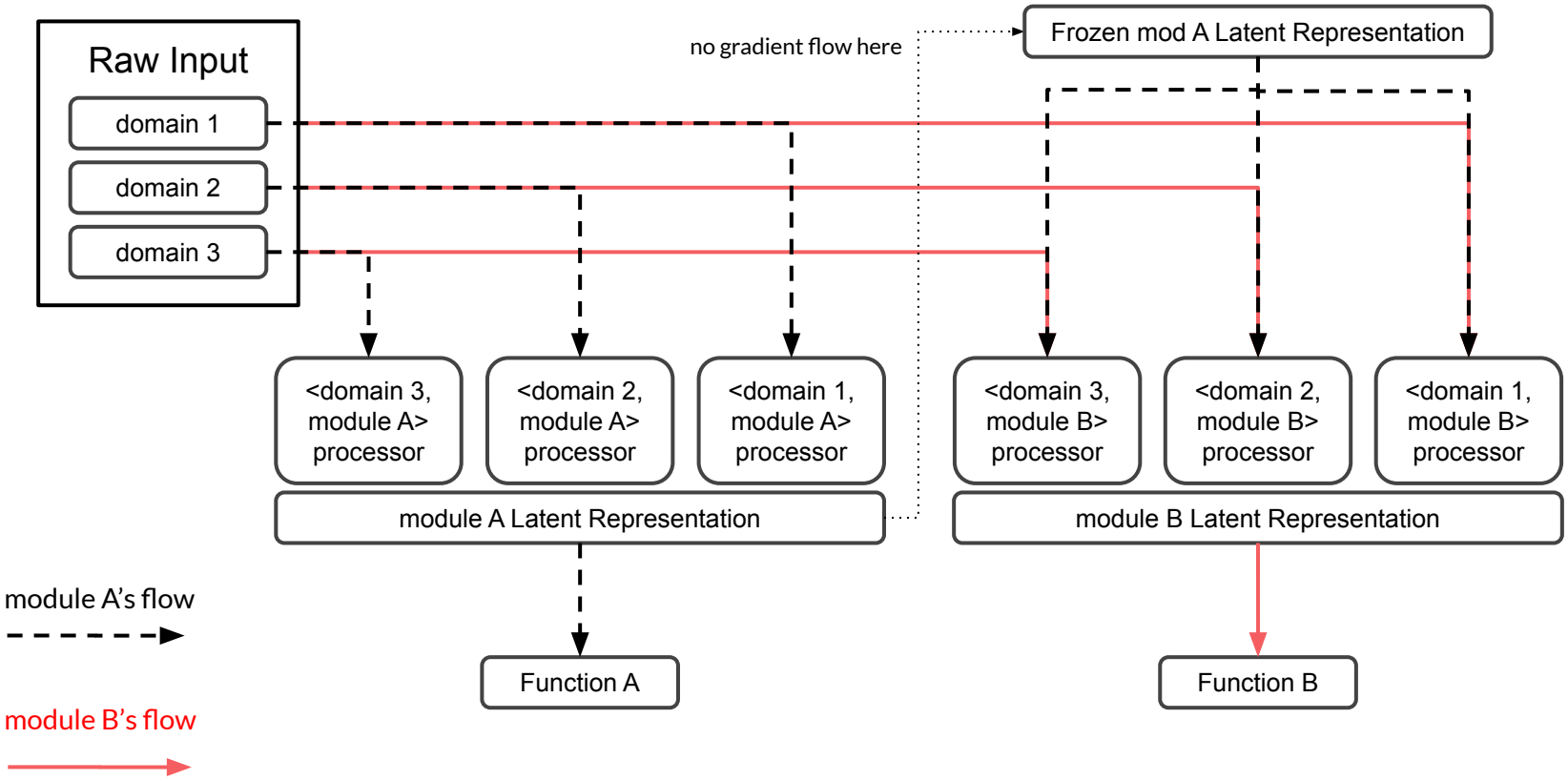


Function A is expressed in subjective human terms, e.g. telling cats and dogs apart.

For *HumanMeaning(ModuleA) = Meaning(Module A's Latent Representation)* to hold true, module A's output classes must be accurately predicted* across many domains/contexts.
If there aren't enough domains, there is no guarantee that *Meaning(Module A's Latent Representation)* aligns with *HumanMeaning(moduleA)*.

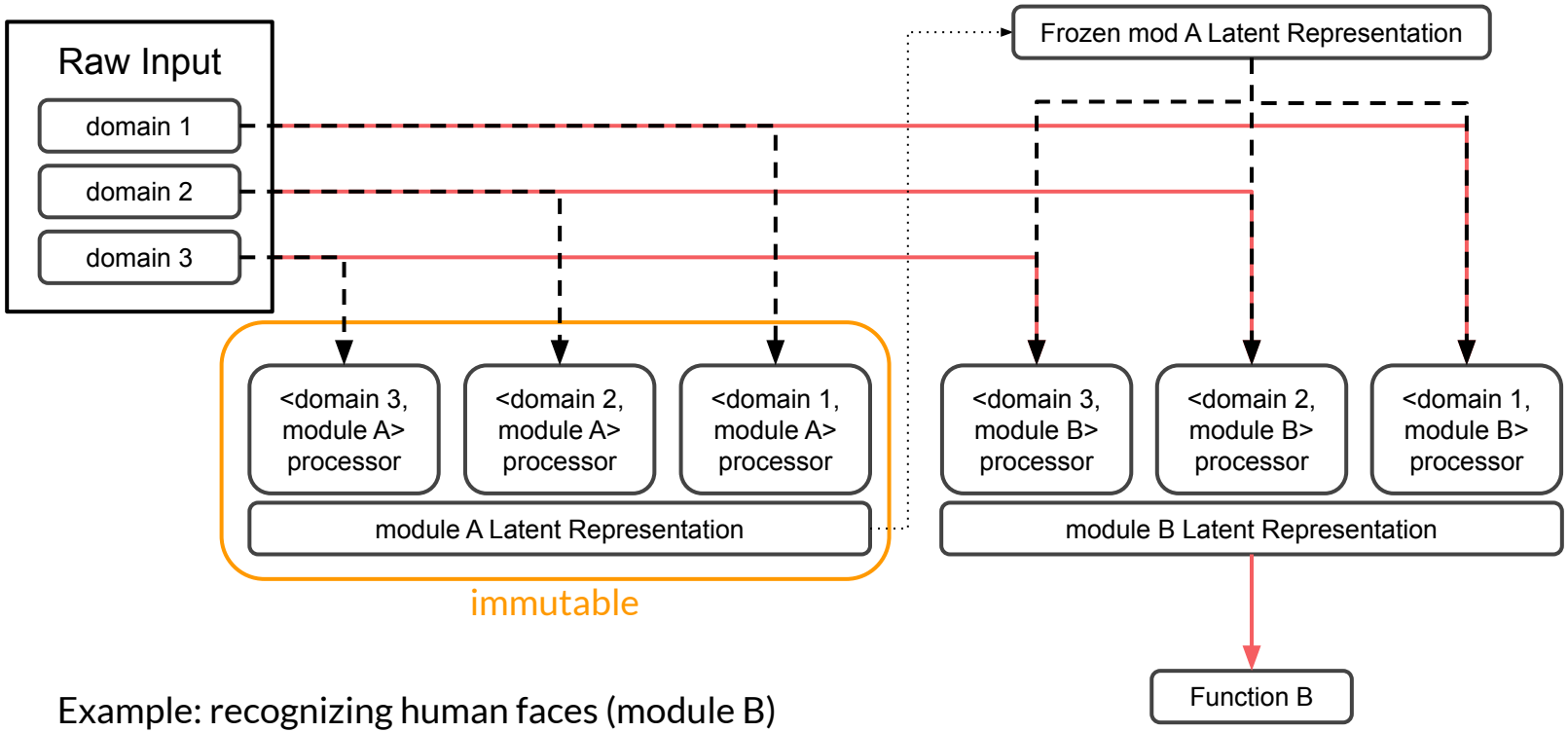
*prediction of classes or numerical values, or rewards from motor goals.

Two-module Scenario



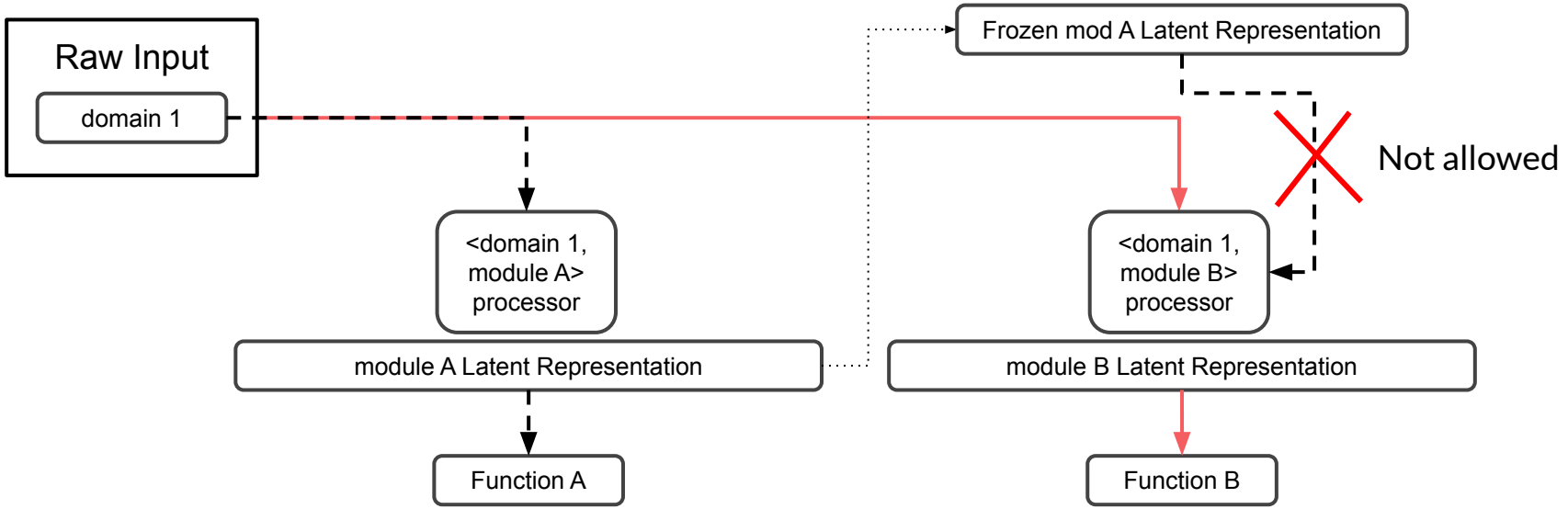
Rules

Rule 1: module A is frozen when training module B



Example: recognizing human faces (module B) cannot interfere with the function of telling cats and dogs apart (function A).

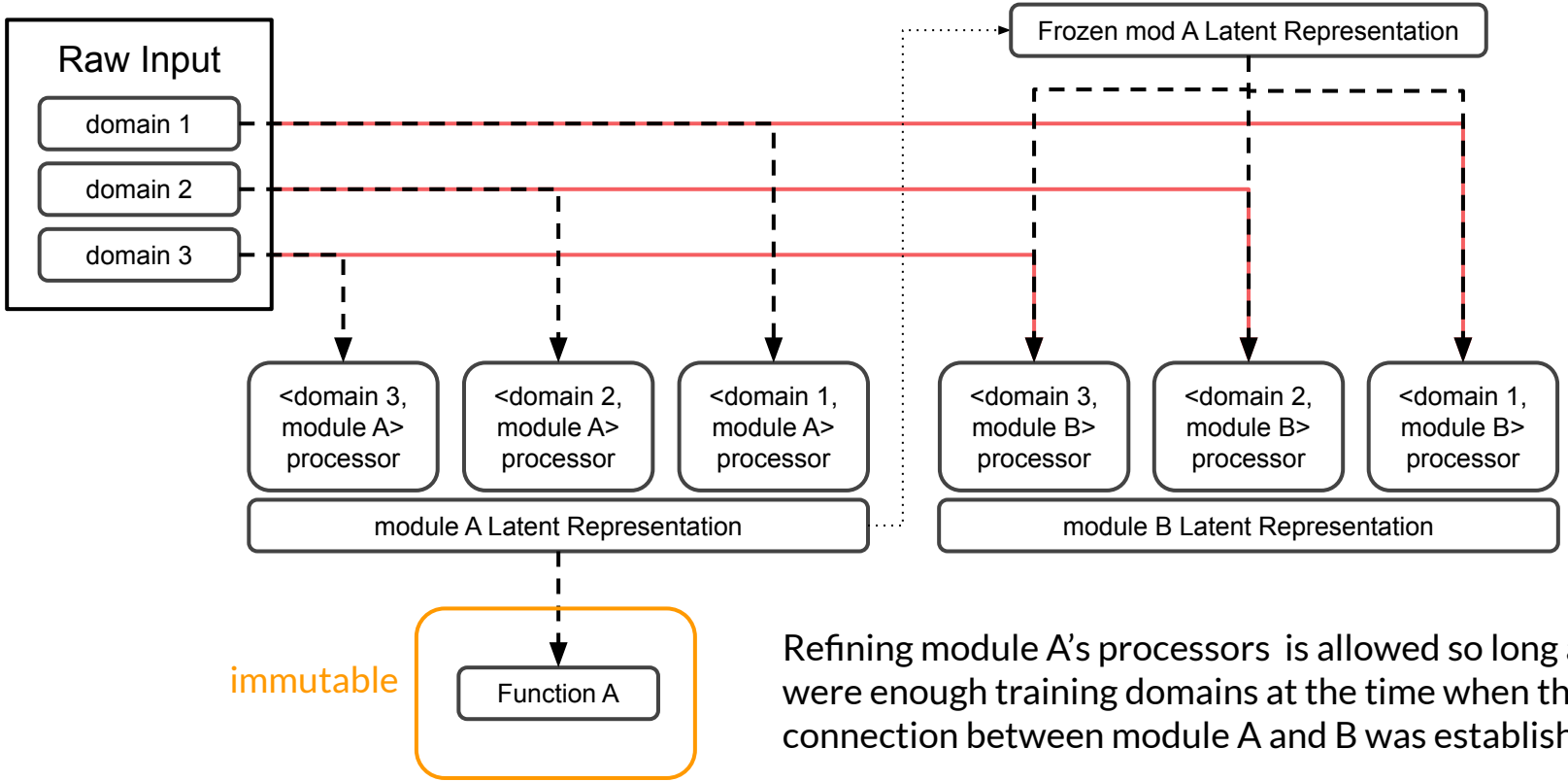
Rule 2: module B cannot utilize module A if training domains were scarce



If there is a misalignment due to the lack of training domains, i.e. *HumanMeaning(moduleA) != Meaning(module A's Latent Representation)*, then the system might find a correlation between module A and module B that doesn't exist in our reality.

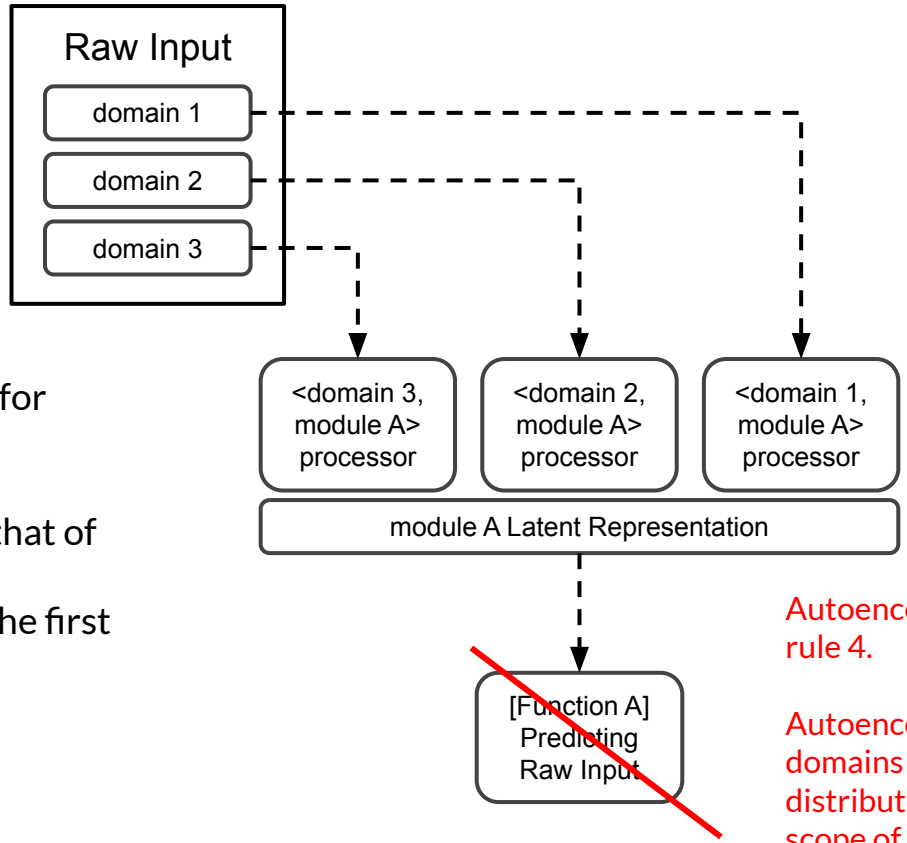
Example: without this rule, the system might mistakenly connect cats (module A) to arctic foxes (module B) if white cats were the only kind of cats seen by the system. If a connection between module A and B were to be drawn, nothing would stop module A from interfering with module B in a destructive way.

Rule 3: module A is allowed to interfere with module B if it doesn't break Rule 2



Refining module A's processors is allowed so long as there were enough training domains at the time when the connection between module A and B was established.

Rule 4: modules must interpret/abstract their input

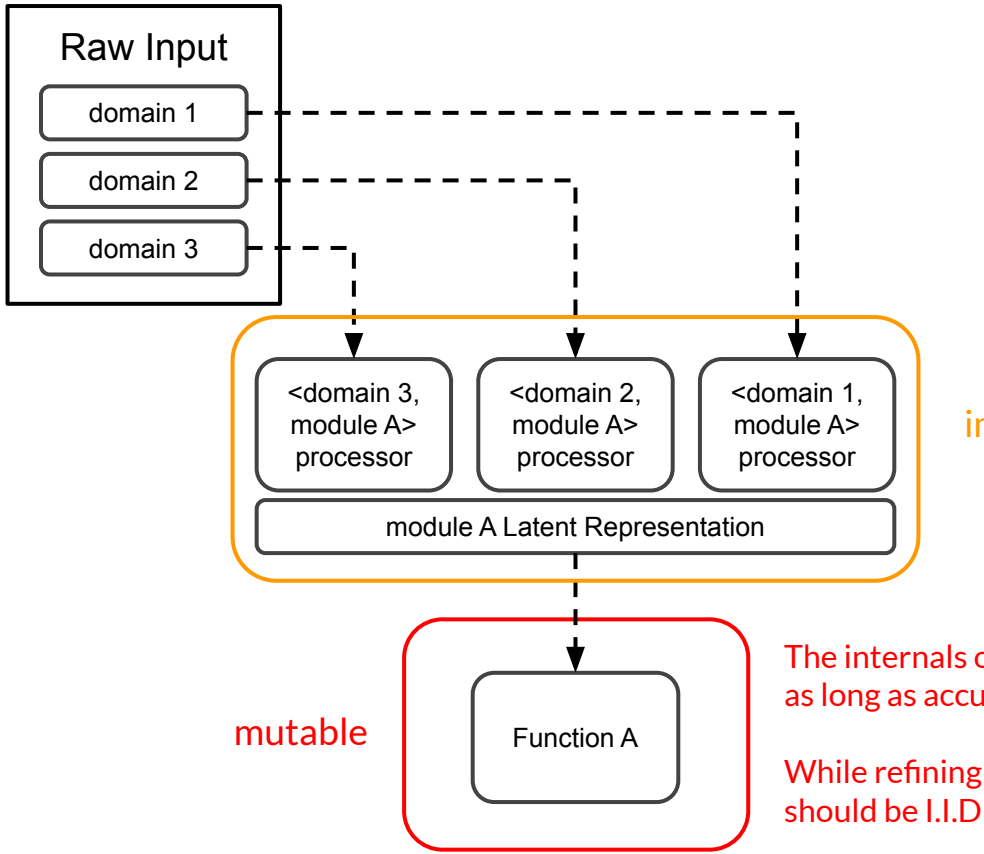


Rule 4 is necessary for domain 1's latent representations to be congruent with that of domain 2 and 3.
(like John's face in the first slides).

Autoencoding is incompatible with rule 4.

Autoencoding might work if domains are identically distributed, but that's beyond the scope of MPCL.

Rule 5: functions' internals can be improved when representations are fixed



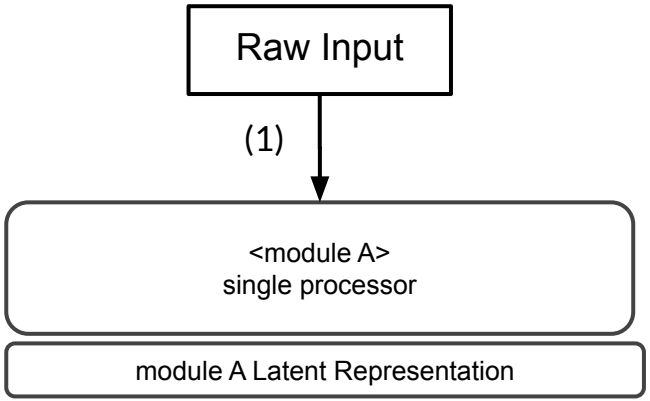
The internals of Function A can be altered as long as accuracy doesn't decrease.

While refining Function A, goals/targets should be I.I.D to avoid forgetting.

Domain boundaries and proto-MPCL

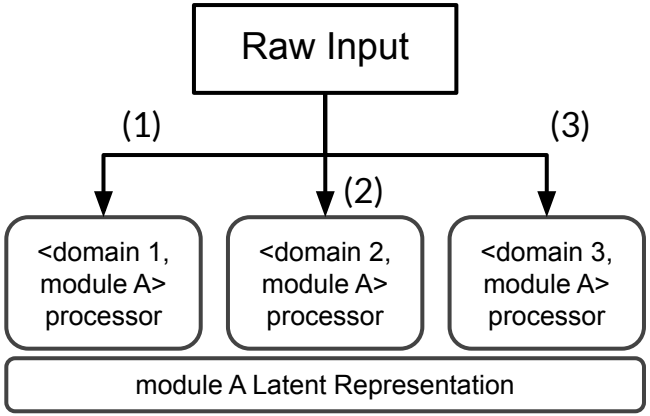
Inside Modules: The Routing Problem

Conventional Continual Learning



It is common for CL algorithms to train a single processor for all domains.
There is no issue detecting domain boundaries and mutualising knowledge, but there is a risk of catastrophic forgetting.

MPCL

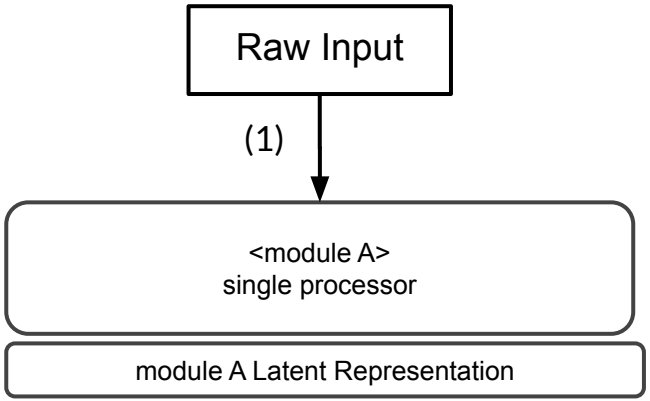


MPCL modules don't forget but there is no easy way to know where to route the raw input between (1), (2) and (3).

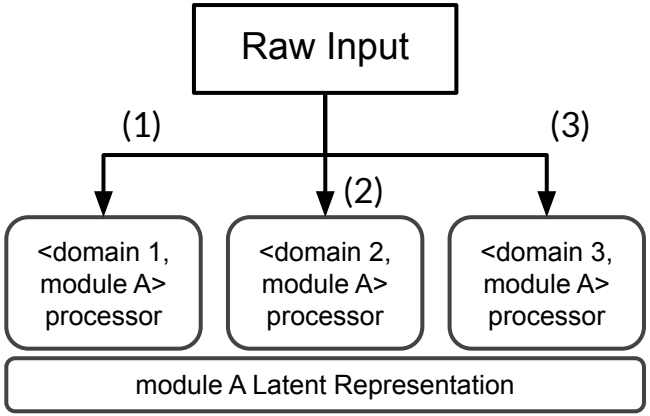
As for knowledge mutualisation, refer to soft-parameter sharing.

Inside Modules: The Routing Problem

Conventional Continual Learning

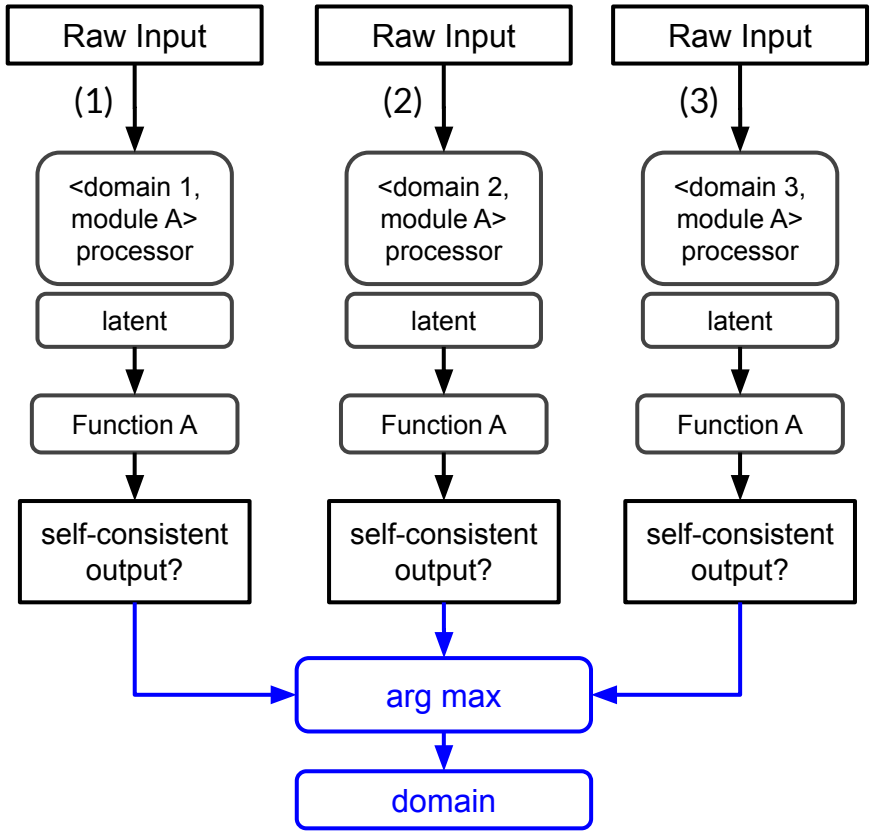


MPCL



MPCL is not strongly committed to one way or the other, but the multi-processor approach makes it easier to define generalization, abstraction and transferability.

Proto-MPCL



Proto-MPCL is that building block.

For MPCL to work at a larger scale, we need proto-MPCL to do quite well on isolated continual-learning tasks.

On paper, it doesn't need to be perfect because the more proto-MPCL blocks you combine to realize various goals, the easier it gets to find ways of detecting inconsistencies between the functions' outputs.

Meaning

Meaning != Representation

I mean “representation” in the ML sense, as in “representation learning”.
I do not mean “mental representation”.

If Representation-Preserving Continual Learning (RPCL) were a thing, it would not be the same thing as MPCL. It would be more limited and limiting.

- Not every representation vector needs stability ;
- Meaning always needs stability ;
- Representation vectors can be more fine-grained than meaning.

Meaning != Representation

- In a classification setting, each class' representation vectors need stability. Meaning is not a particularly useful concept in such a setting because it is conceivable to enumerate all the representation vectors that need stability without resorting to any other concept.

- In regression and motor settings, however, it becomes harder to identify the representations that need stability* and it is useful to look at the problem from the meaning angle, i.e. the link between representations and goals. Goal-realizing functions need to remain *accurate*. They need not remain *stable* at all times.

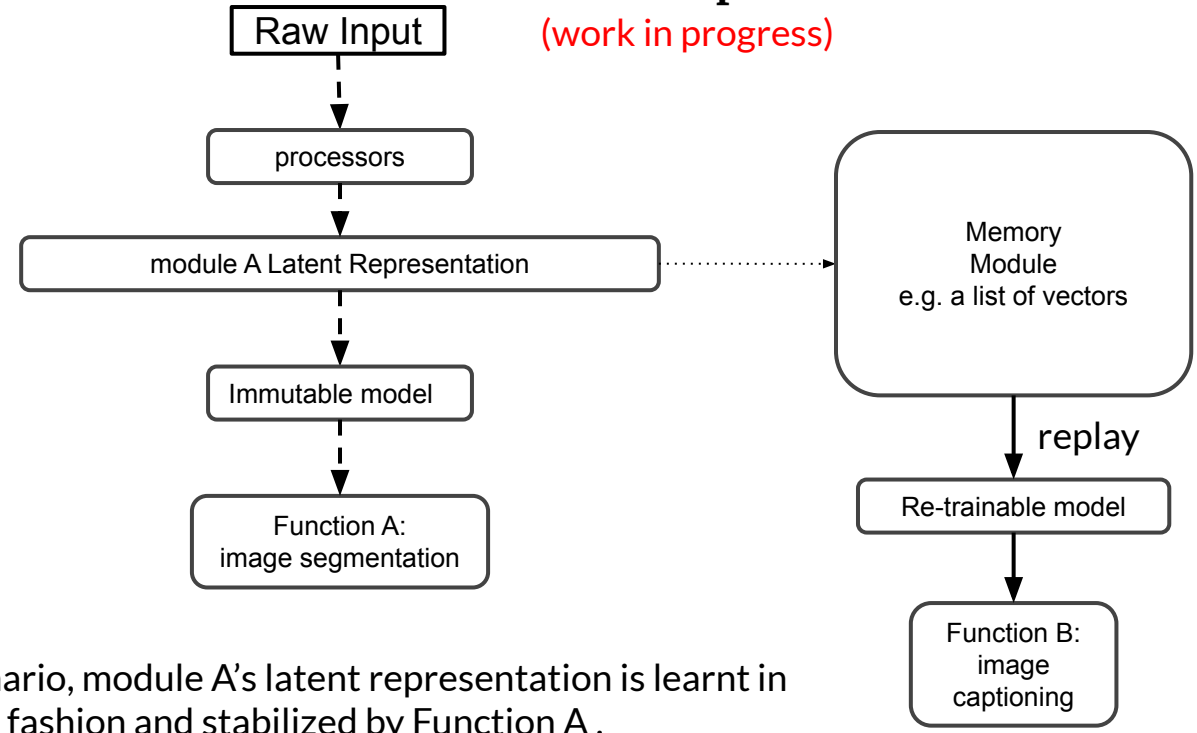
* I'm still debating whether it would be practical or not to require stability for every single representation vector of the training set, thereby doing away with meaning.

Real Examples

(work in progress)

Two-module scenario: example

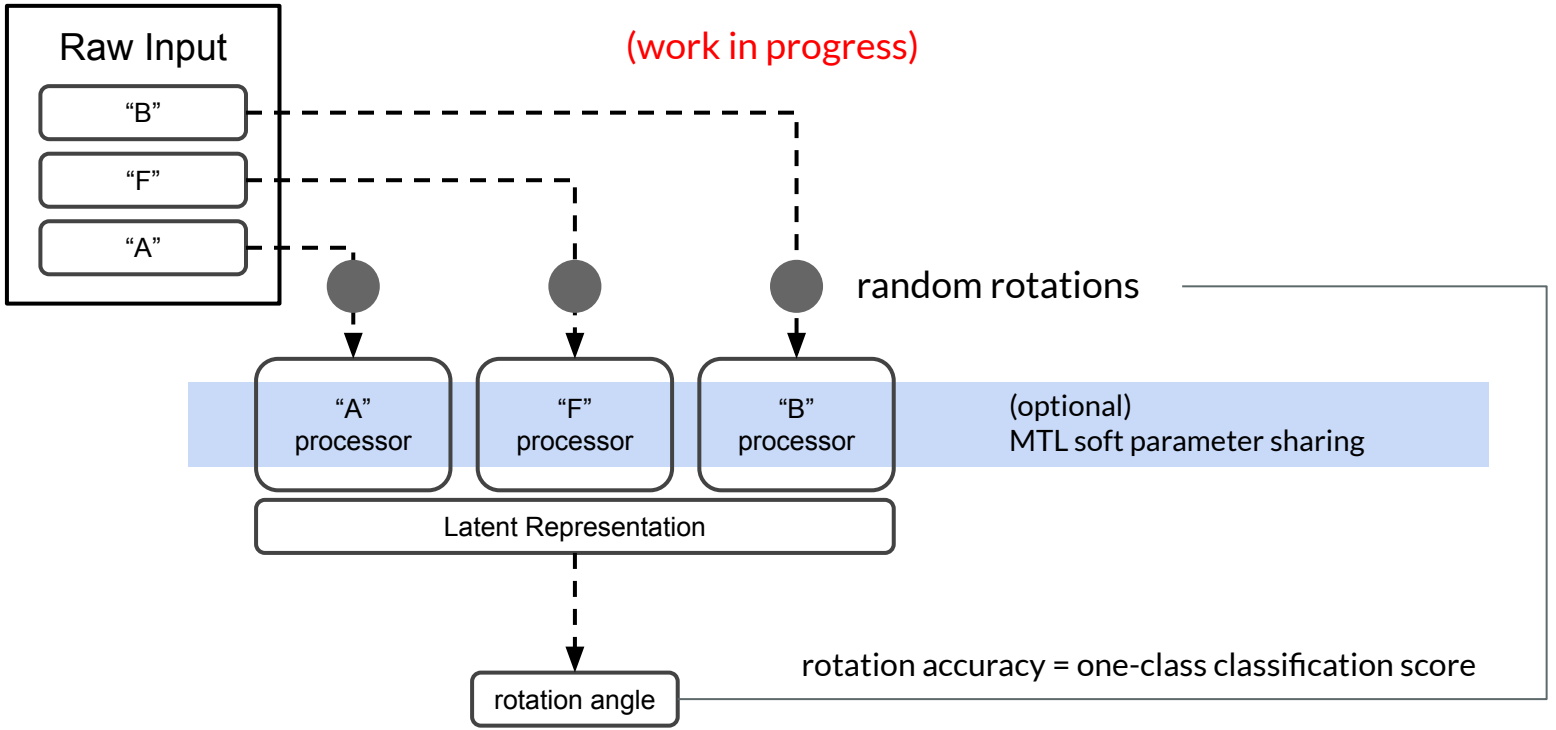
(work in progress)



In this scenario, module A's latent representation is learnt in a continual fashion and stabilized by Function A .

The re-trainable model utilizes A's representations to realize Function B.

One-module scenario: EMNIST



This works because “rotation” is a subjective concept whose meaning is not carried by the input alone. This is why the system works better with a custom processor for each letter, and this is why it is an adequate function for detecting discrepancies.

Example of degenerate module: if the function is to denoise the images or to count the number of black pixels, it can be done without knowing the letter, thus it doesn’t help us predict the letter class.