Machine Learning Platform: Deploying and Managing Models in the CERN Control System

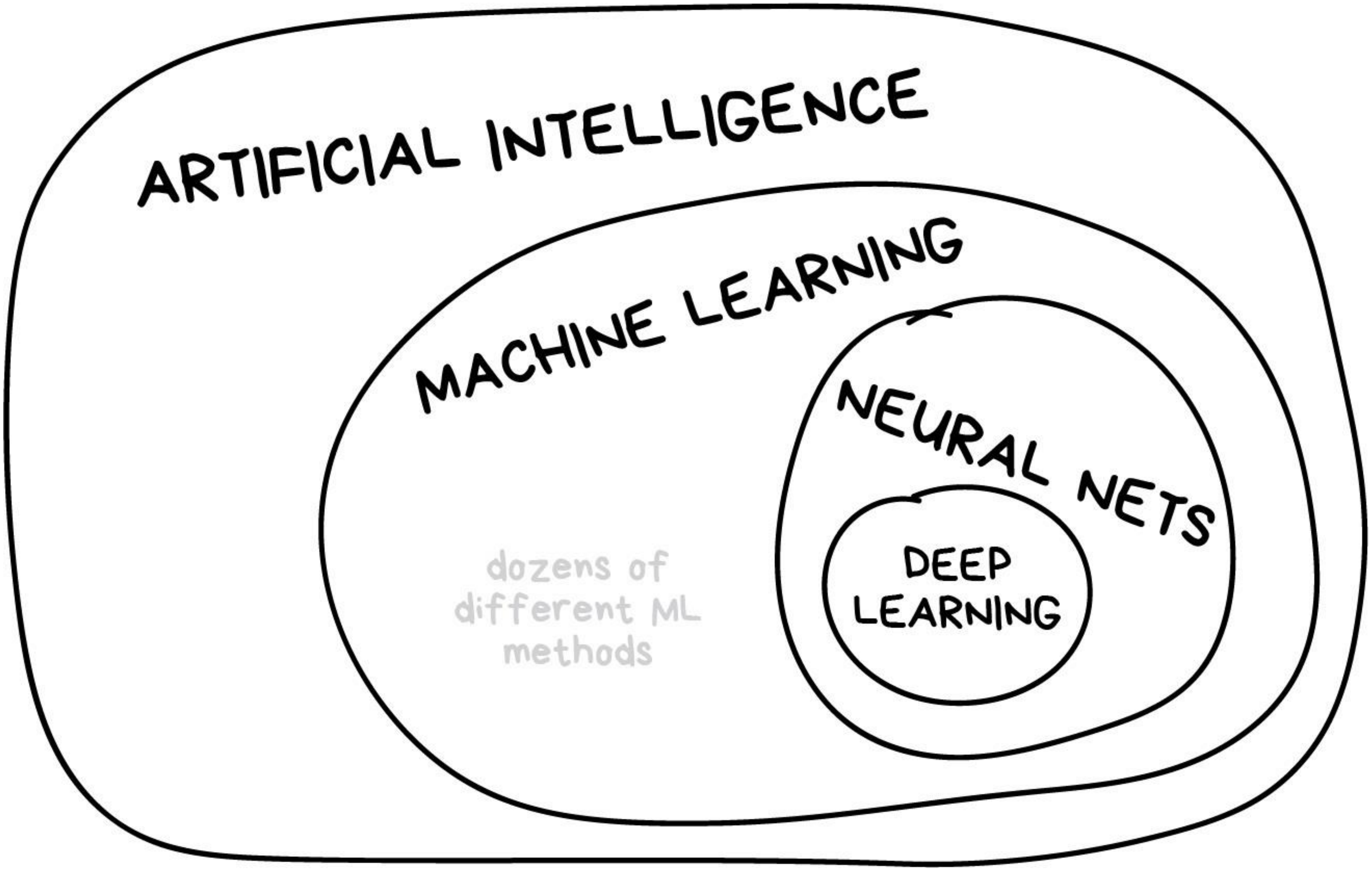
Jean-Baptiste de Martel

10/18/2021 – ICALEPCS 2021

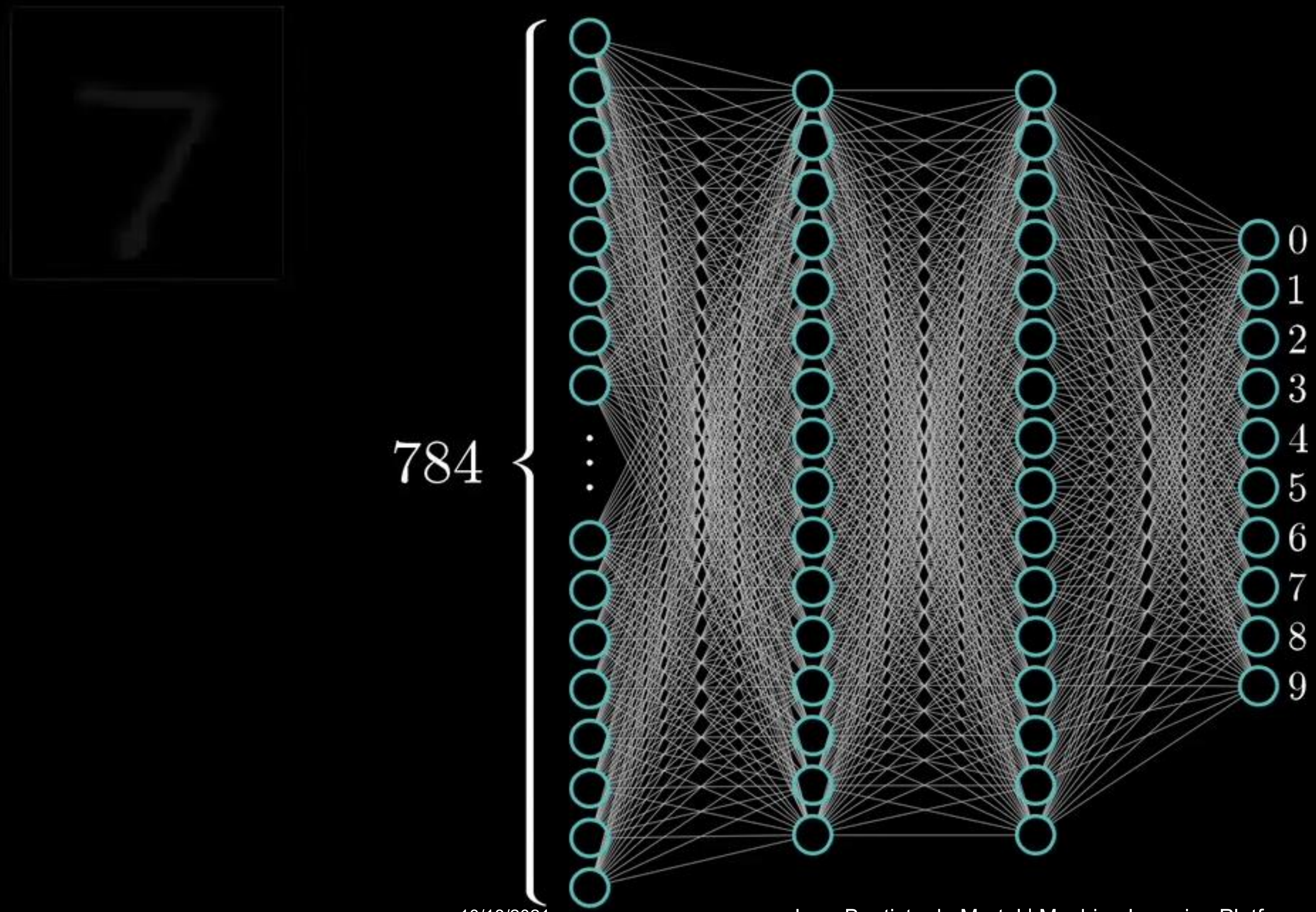
Contents

Introduction

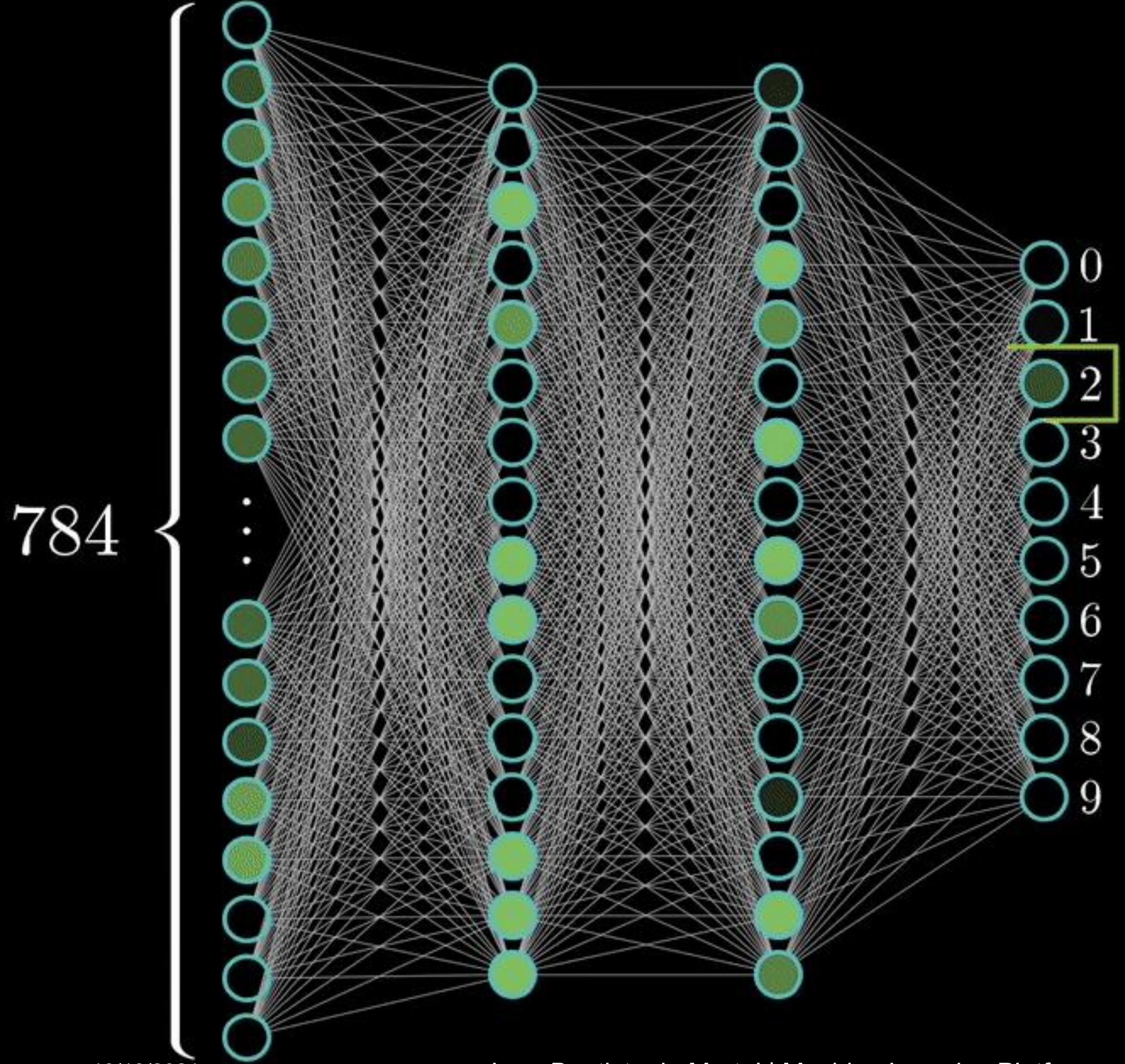
ML for CERN Accelerator Controls



Source: https://vas3k.com/blog/machine_learning/



Source: https://www.youtube.com/watch?v=aircAruvnKk



Model type:
Layout/architecture of the neural network –
i.e., number of neurons, how they are connected, etc…

Model parameters:
“Trained weights”
 - values assigned to the neurons and connections after training

Model:
Combination of a model type and model parameters

Source: https://www.youtube.com/watch?v=aircAruvnKk

ML for accelerator controls

- **Why ML ?**
  - Particle accelerators are complex, time-varying, non-linear systems
  - Large parameter space with many intercorrelated variables
  - Human operators can only process and tune a limited number of parameters at once, act on narrow timescales, and are slow
  - Automated systems lack domain knowledge and deductive reasoning
- **Most of the control system remains based on traditional methods**
- **But certain problems are much easier to solve with ML**
  - Optimization – e.g., trajectory steering at LINAC4
  - Trending and forecasting – e.g., magnet field prediction with hysteresis
  - Computer vision – e.g., beam profile measurements

Finding a compromise

Volatile world of physicists

- Code needs to run once
- Bleeding edge technology
- Used to own tools and comfort, cloud services
- Maintainability is not the main concern

Reliable world of accelerator controls

- Need to run reliably 24/7/365: need reproducibility, robustness, traceability
- Use highly reliable, battle-tested tools
- Constraints of the accelerator network: no internet access, restricted tooling, security precautions
- Standardize and unify to minimize maintenance

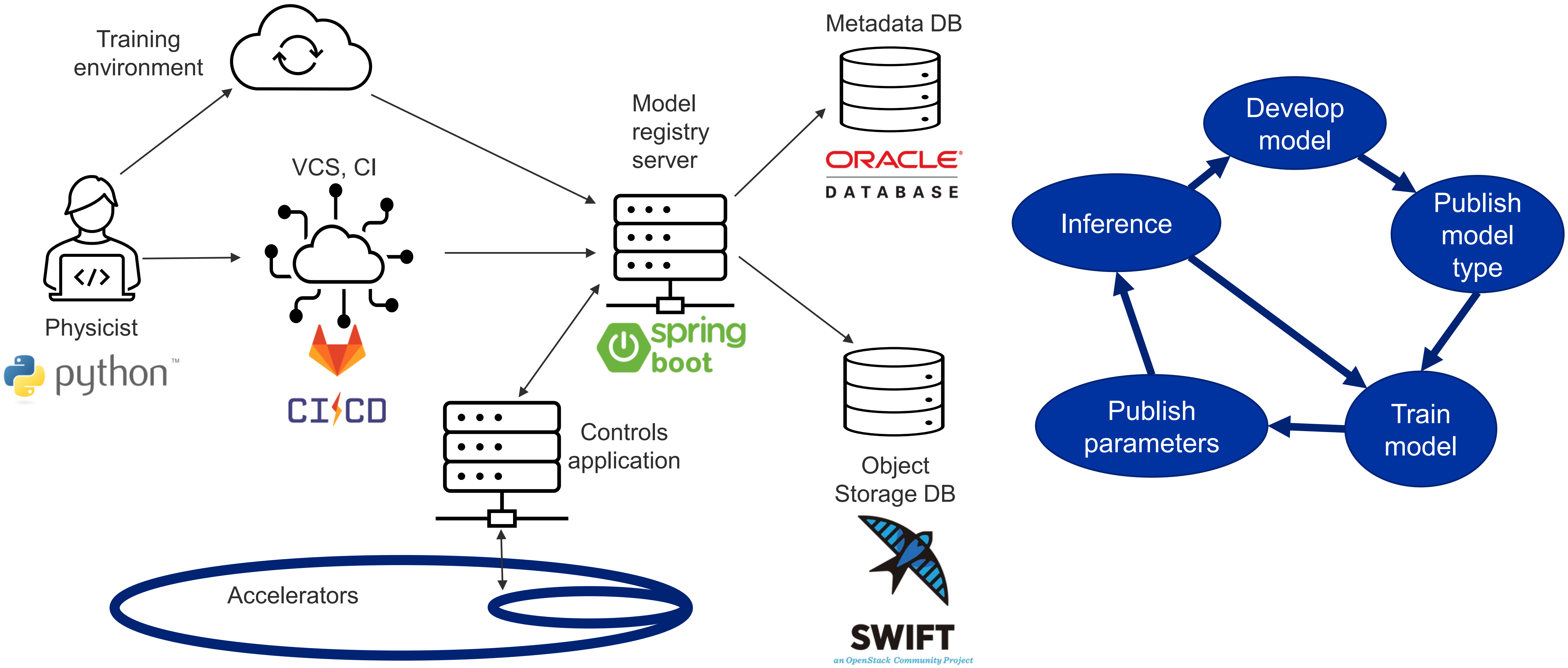
Enabling ML for accelerator controls

MLP aims to bridge the gap between these 2 worlds by providing tooling which:

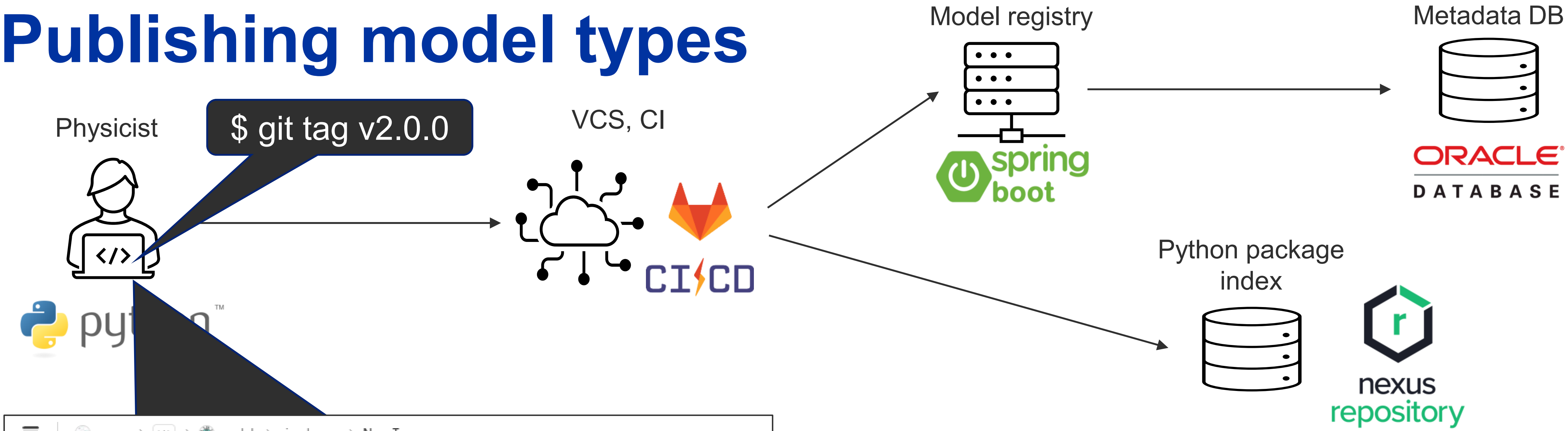
- **helps fulfill the specific needs of the control system**
  - reliability
  - traceability
  - security
  - standardization
- **stays out of the user's way**
  - minimizes impact on model developer's workflow
  - avoids constraining choice of tools
- **facilitates model development by hiding infrastructural concerns**

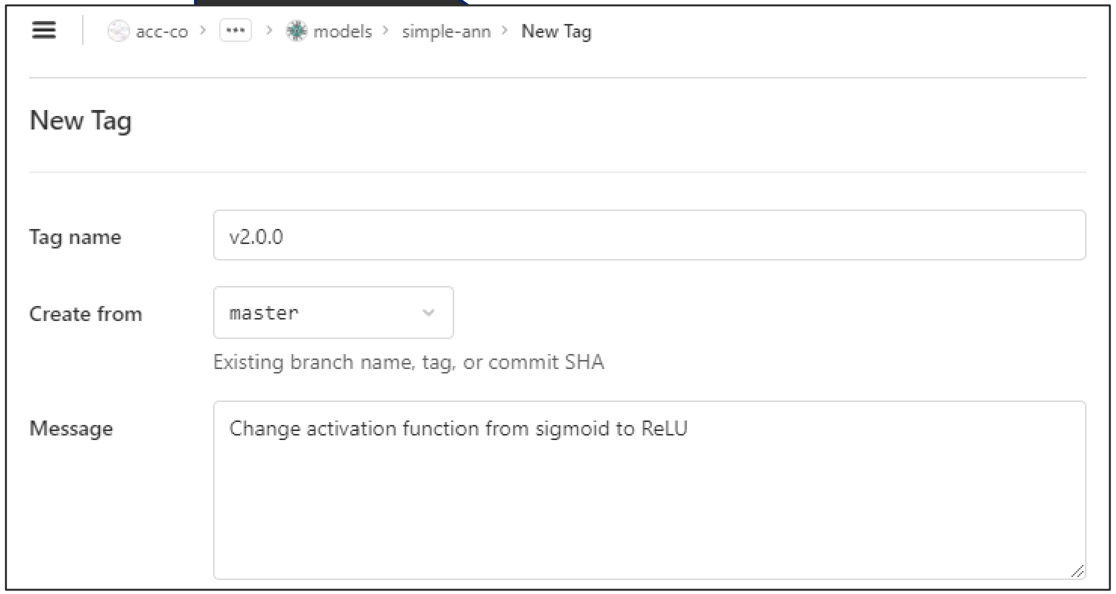
Development to production with MLP

Development workflow



Publishing model types




acc-co › ••• › models › simple-ann › New Tag

New Tag

Tag name: v2.0.0

Create from: master

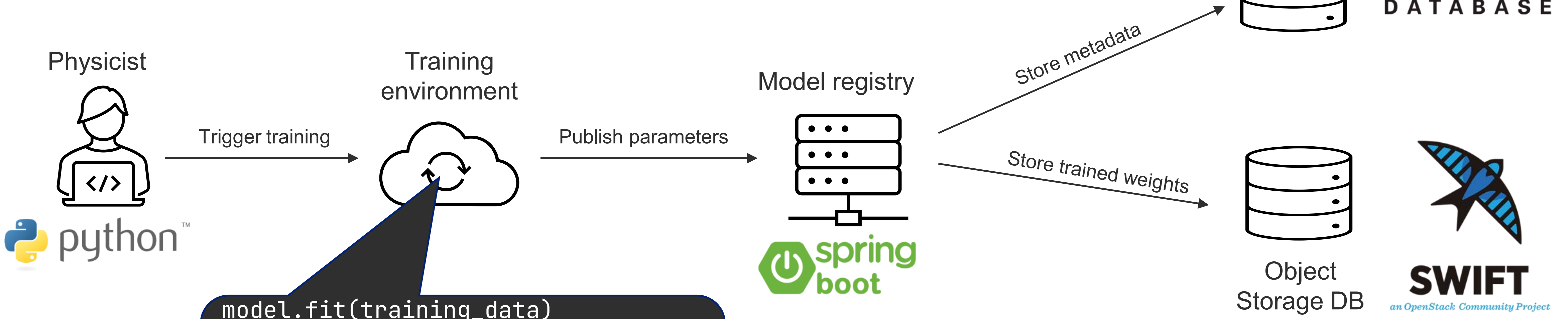
Existing branch name, tag, or commit SHA

Message: Change activation function from sigmoid to ReLU

Advantages

- **Access control and traceability for model types**
- **Quick & easy, no need to learn new tools, complexity is hidden**
- **Minimal constraints on use of git**

Publishing model parameters




```
model.fit(training_data)
client.publish_parameters_version(
    model,
    name = "proton_beam_config",
    version = "2.0.3"
)
```

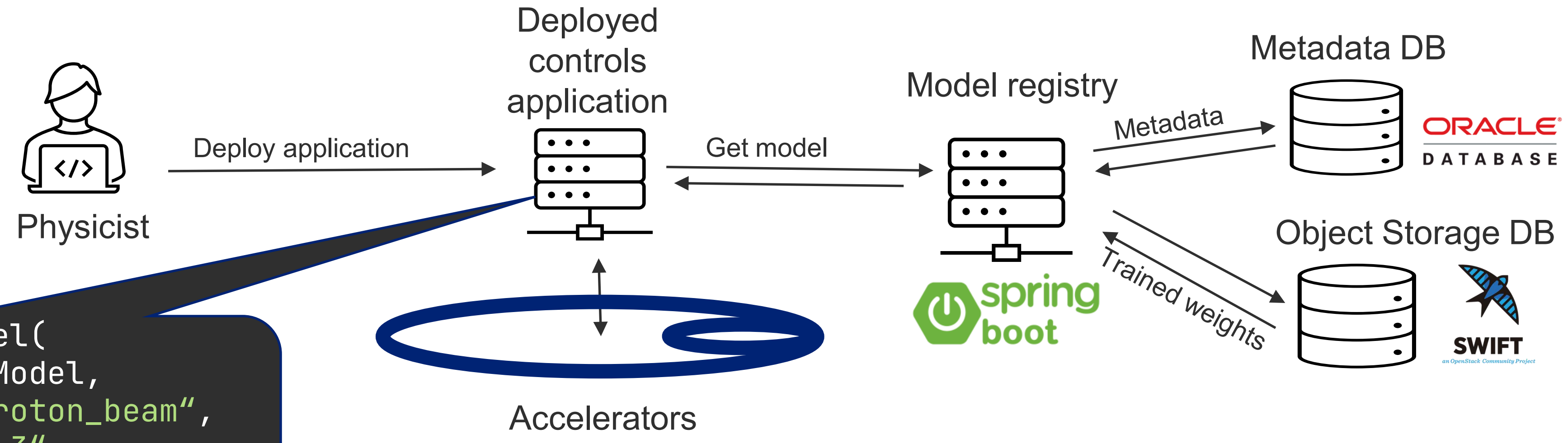
Usage

- **Choose parameters name and version**
- **Use the client library to publish**

Advantages

- **All parameters stored centrally and reliably**
- **Compatibility is fully managed**

Inference
(Deployment)




```
model = client.create_model(
    model_type = BeamLineModel,
    model_parameters = “proton_beam“,
    params_version = “2.0.3“
)
result = model.predict(input)
```

Usage

- **Use the MLP client library to instantiate the model**
- **Provide model type, parameters name and version**

Advantages

- **Parameters retrieved and loaded transparently**
- **Parameter traceability**

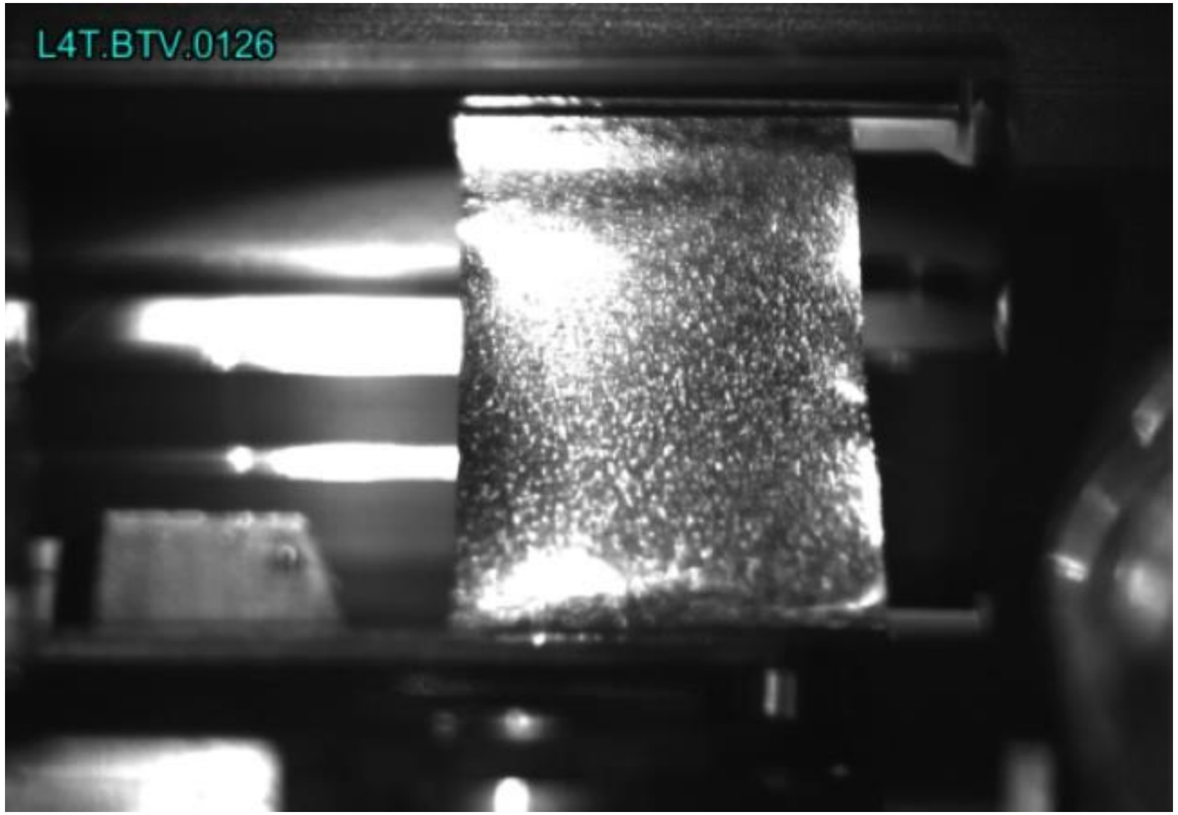
Continuous retraining

Continuous retraining - motivation

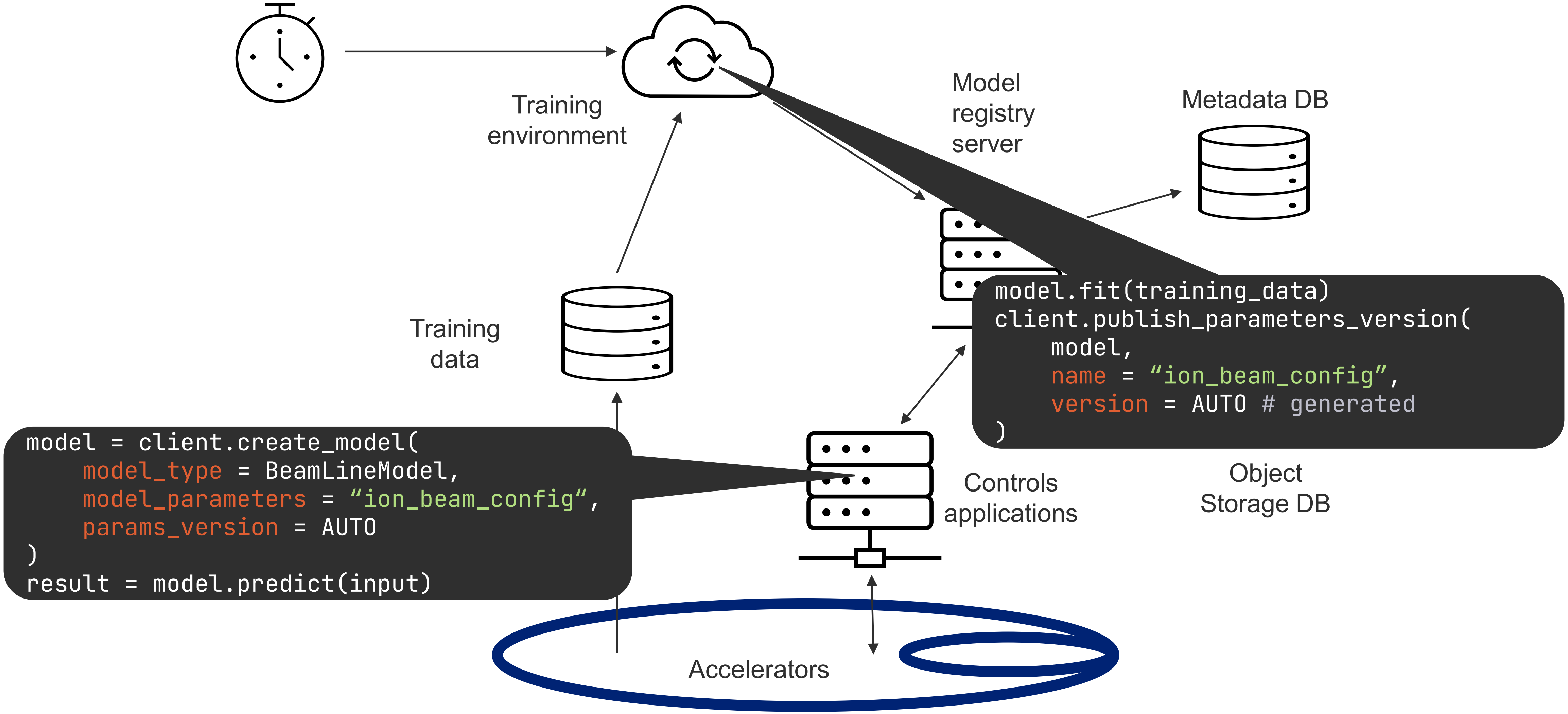
Example: stripper foil degradation

- The stripper foil is an essential component of our linacs
- It degrades over time and is replaced regularly
- Beam characteristics vary
- Machine parameters need to adapt

-> need to re-train model continuously to keep it up to date



Continuous retraining - implementation



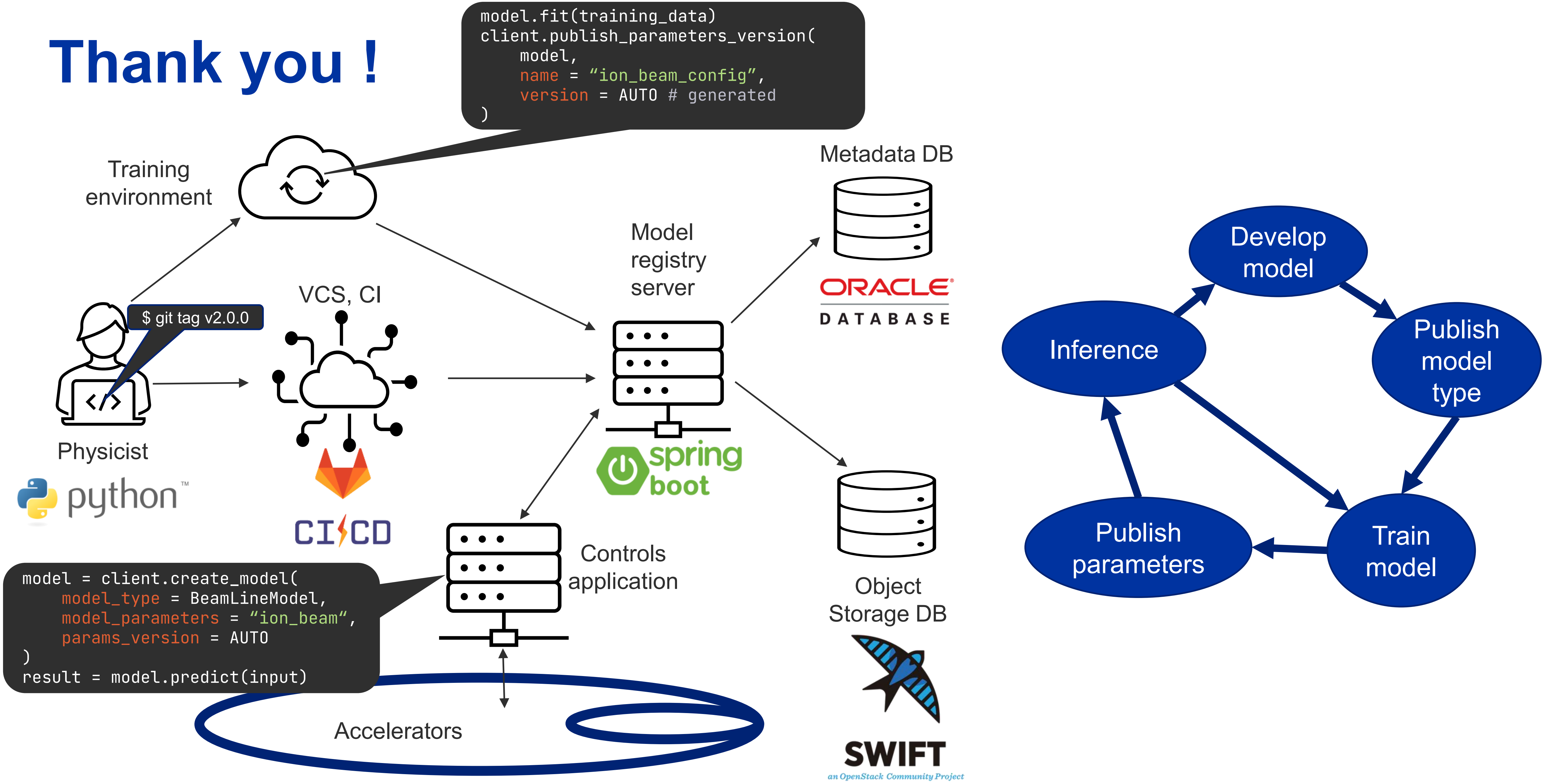

```
model.fit(training_data)
client.publish_parameters_version(
    model,
    name = “ion_beam_config”,
    version = AUTO # generated
)
```

```
model = client.create_model(
    model_type = BeamLineModel,
    model_parameters = “ion_beam_config“,
    params_version = AUTO
)
result = model.predict(input)
```

Conclusion

- **The number of ML applications for controls is growing exponentially**
- **We want to help physicists develop models faster and unburden them from infrastructural concerns while minimizing constraints**
- **We also want to apply software engineering best practices to ensure reliability and maintainability of the control system**
- **MLP provides a basis to achieve these goals and is now being adopted**
- **Could not cover everything, simplified a lot – please see paper or contact me offline!**
  - jean-baptiste.de.martel@cern.ch

Thank you !


model.fit(training_data)
client.publish_parameters_version(
model,
name = “ion_beam_config”,
version = AUTO # generated
)
Training environment
Metadata DB
ORACLE DATABASE
Model registry server
VCS, CI
$ git tag v2.0.0
Physicist
python
CI/CD
spring boot
Controls application
Object Storage DB
model = client.create_model(
model_type = BeamLineModel,
model_parameters = “ion_beam“,
params_version = AUTO
)
result = model.predict(input)
Accelerators
SWIFT
an OpenStack Community Project
Develop model
Publish model type
Inference
Train model
Publish parameters

CERN