Energy Reconstruction in JUNO

with classical and quantum machine learning methods

Group: *Città Romanze*

- Pietro Cappelli
- Alberto Coppi
- Giacomo Franceschetto
- Nicolò Lai

Supervisors:

- Prof. Alberto Garfagnini
- Prof. Marco Zanetti

Solve a HEP problem using CNNs

Solve a HEP problem using CNNs

goal

Infer the ***deposited energy*** in neutrino interaction events for the **JUNO** experiment

Requires a ***better resolution than 3%***

Solve a HEP problem using CNNs

goal

Infer the ***deposited energy*** in neutrino interaction events for the **JUNO** experiment

Requires a ***better resolution than 3%***

Complex classical CNN (**ResNet**) already explored in the literature [1]

[1] Y. Malyshkin et al., 2021, *Vertex and Energy Reconstruction in JUNO with Machine Learning Methods*, arXiv:2101.04839

Solve a HEP problem using CNNs

goal

Infer the ***deposited energy*** in neutrino interaction events for the **JUNO** experiment

Requires a ***better resolution than 3%***

Complex classical CNN (**ResNet**) already explored in the literature [1]

Simple classical CNN to be compared with its **quantum** counterpart

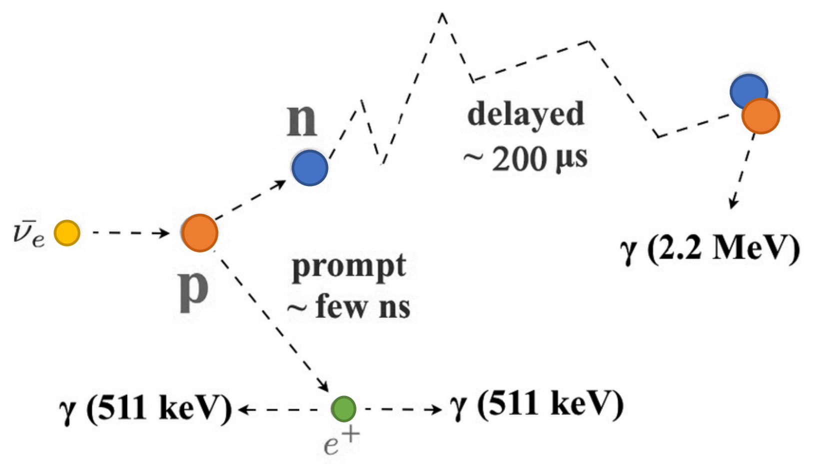
[1] Y. Malyshkin et al., 2021, *Vertex and Energy Reconstruction in JUNO with Machine Learning Methods*, arXiv:2101.04839

Introduction

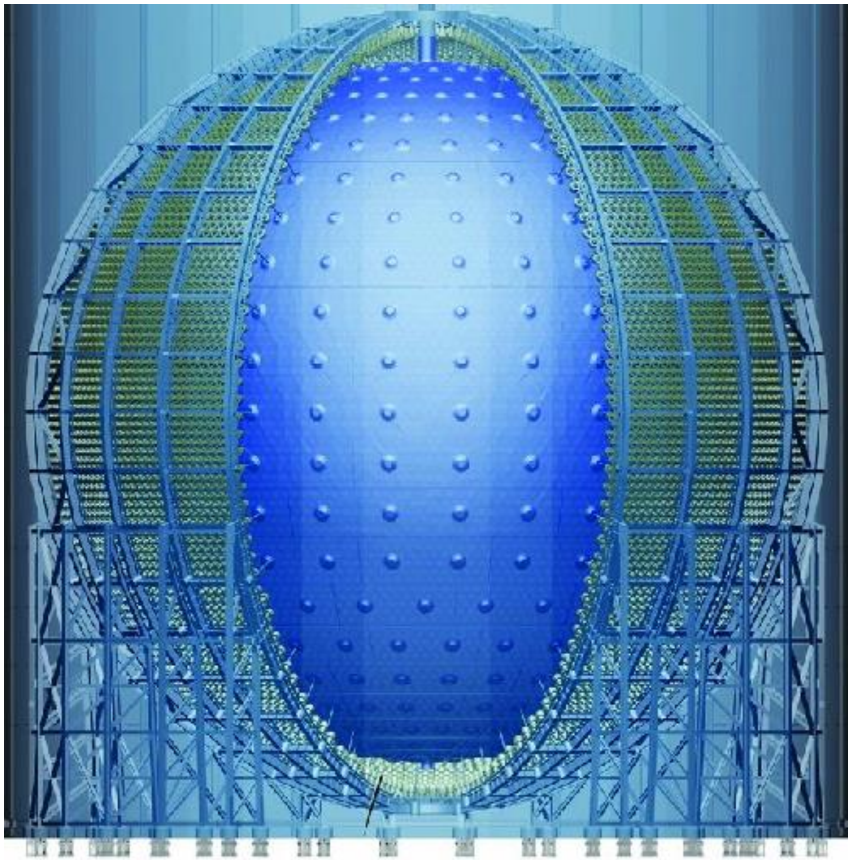
The **Jiangmen Underground Neutrino Observatory** (**JUNO**) experiment for neutrino oscillation studies

The JUNO experiment

- Is a multipurpose neutrino experiment
  - *neutrino mass hierarchy*
- Detects reactor anti-neutrinos
  - *two nuclear power plants at* 53 *km*
  - *mainly* $\bar{\nu}_e$ *with* $\mathcal{E}_{\bar{\nu}_e} < 10\ MeV$
- Measures signal via the inverse beta decay reaction



The JUNO experiment



Central Detector structure:

- Acrylic sphere ~35 m diameter
- Filled with Liquid Scintillator which serves as both interaction and detection medium
- Surrounded by 17612 20'' and 25600 3'' PhotoMultiplier Tubes
- Collected charge and hit time information

From raw data to processed images

An overview of JUNO datasets

Raw data description

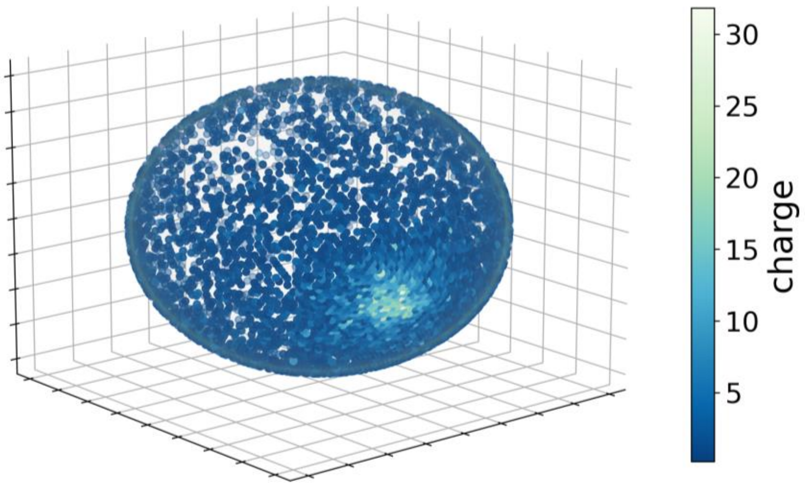
- Source : Monte Carlo simulations with **electronic noise effects**
- Size : 488GB
- Format : .npz files
- Content : ~5 million events
- Features : [*pmt_id, charge, hit_time*]
- Shape : $n_{features} \times n_{events}$

```python
df = pd.DataFrame(train_data.T, columns=("pmt_id", "charge", "hit_time"))
df.head()
```

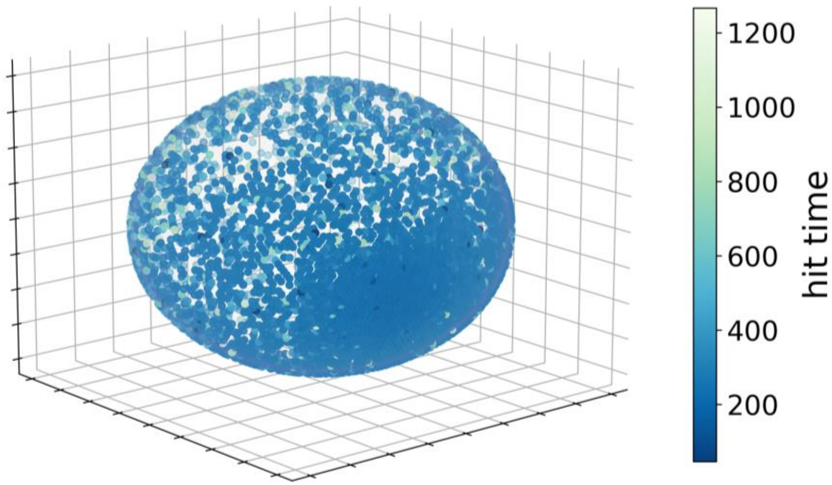
	pmt_id	charge	hit_time
0	[0, 1, 2, 8, 14, 15, 18, 19, 20, 34, 40, 54, 6...	[1.0169096287452863, 1.289895357346751, 1.4573...	[278.5438428571429, 270.03612857142855, 263.31...
1	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[7.69295220634088, 6.316354197252007, 1.649539...	[204.54384285714283, 224.03612857142858, 225.0...
2	[17, 20, 27, 28, 32, 33, 35, 58, 60, 69, 72, 7...	[1.0770311814496256, 1.4405190900405578, 1.112...	[360.4588428571429, 1212.2837, 1169.9381085714...
3	[2, 15, 25, 40, 57, 62, 67, 85, 162, 166, 167,...	[1.481666113625243, 1.0454746500116843, 1.4914...	[288.6028228571429, 344.2614142857143, 568.777...
4	[9, 11, 19, 31, 37, 38, 43, 49, 51, 54, 60, 76...	[1.1243191985536116, 0.7539582292427174, 1.055...	[338.9815371428572, 499.8719857142857, 187.328...

Visualizing an event on the PMT sphere

Charge

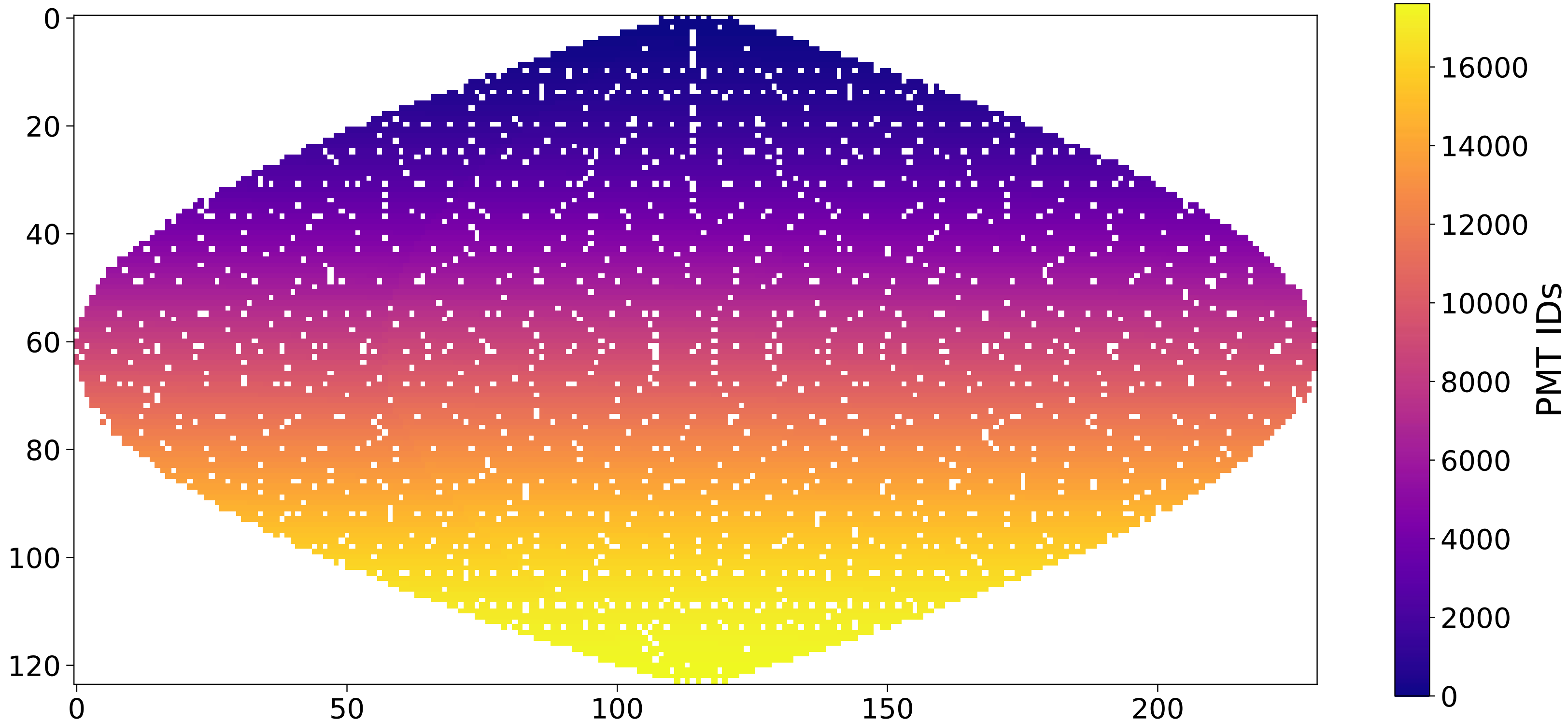


Hit time

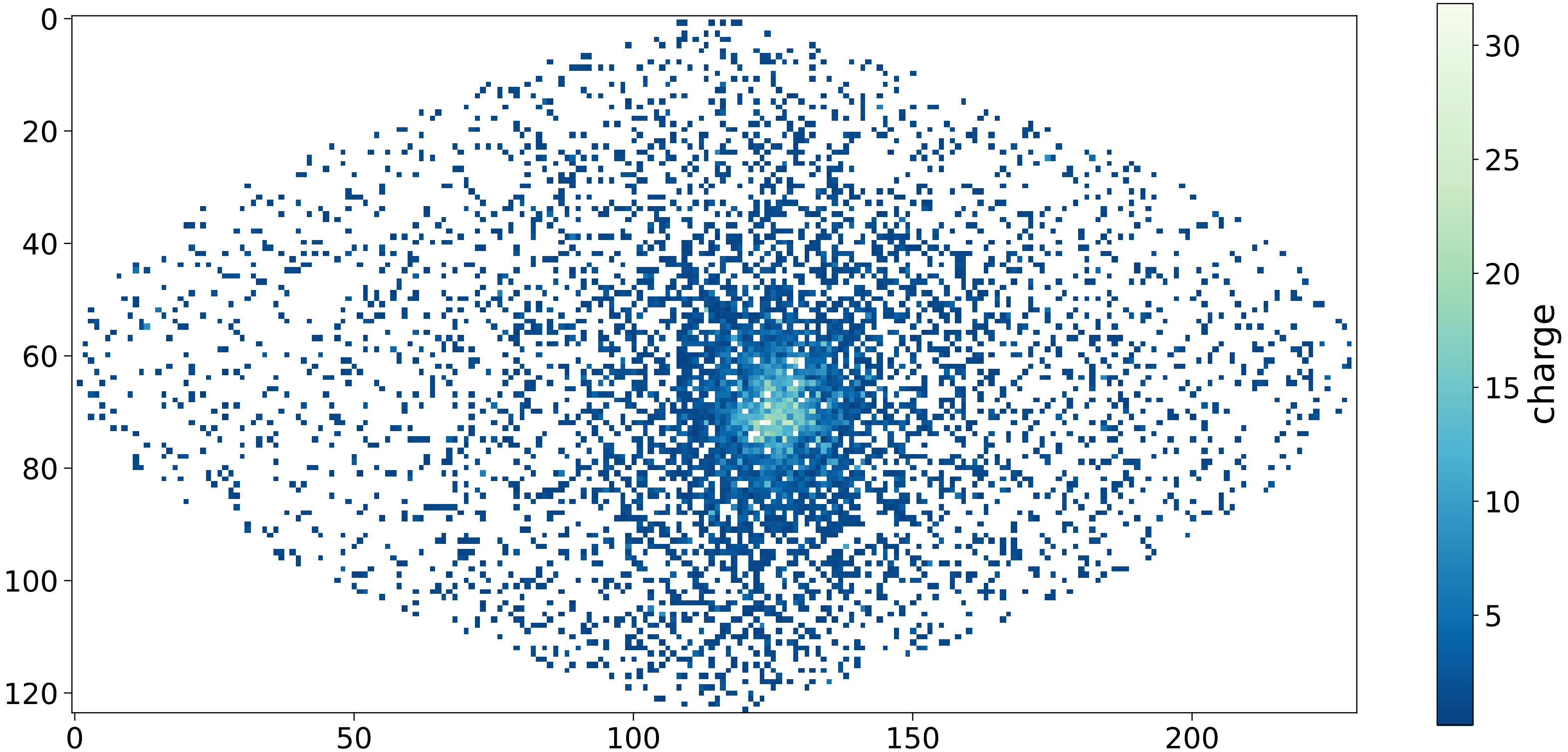


Building the image dataset

The 17612 PMTs on the sphere are *mapped* into a 230x124 image



Example of projected event - charge channel



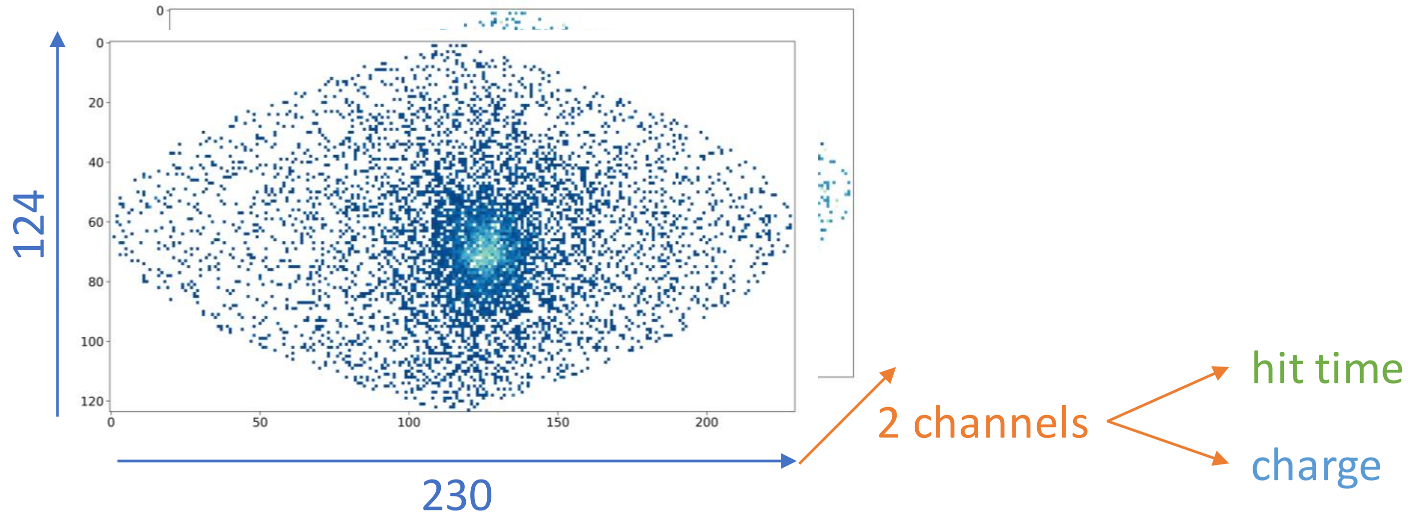
Training and Test datasets

Training set

- 5 million images
  - shaped 230 x 124 x 2
- Energies are uniformly distributed
  $\mathcal{E}_k \in [0, 10]\ MeV$

Test set

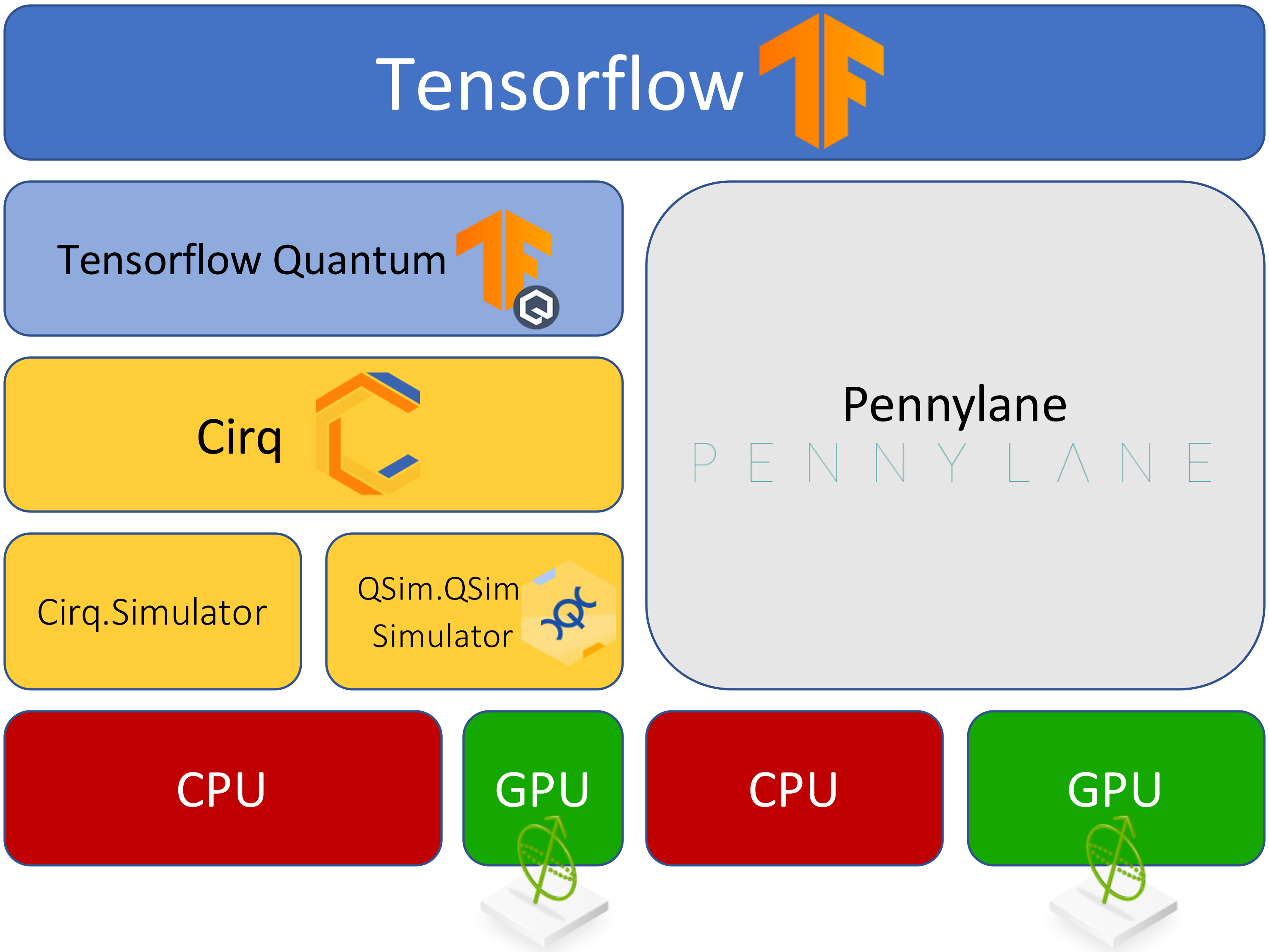
- 1.4 million images in total
  - shaped 230 x 124 x 2
- 14 subsets with fixed energy
  - 100 thousands images each



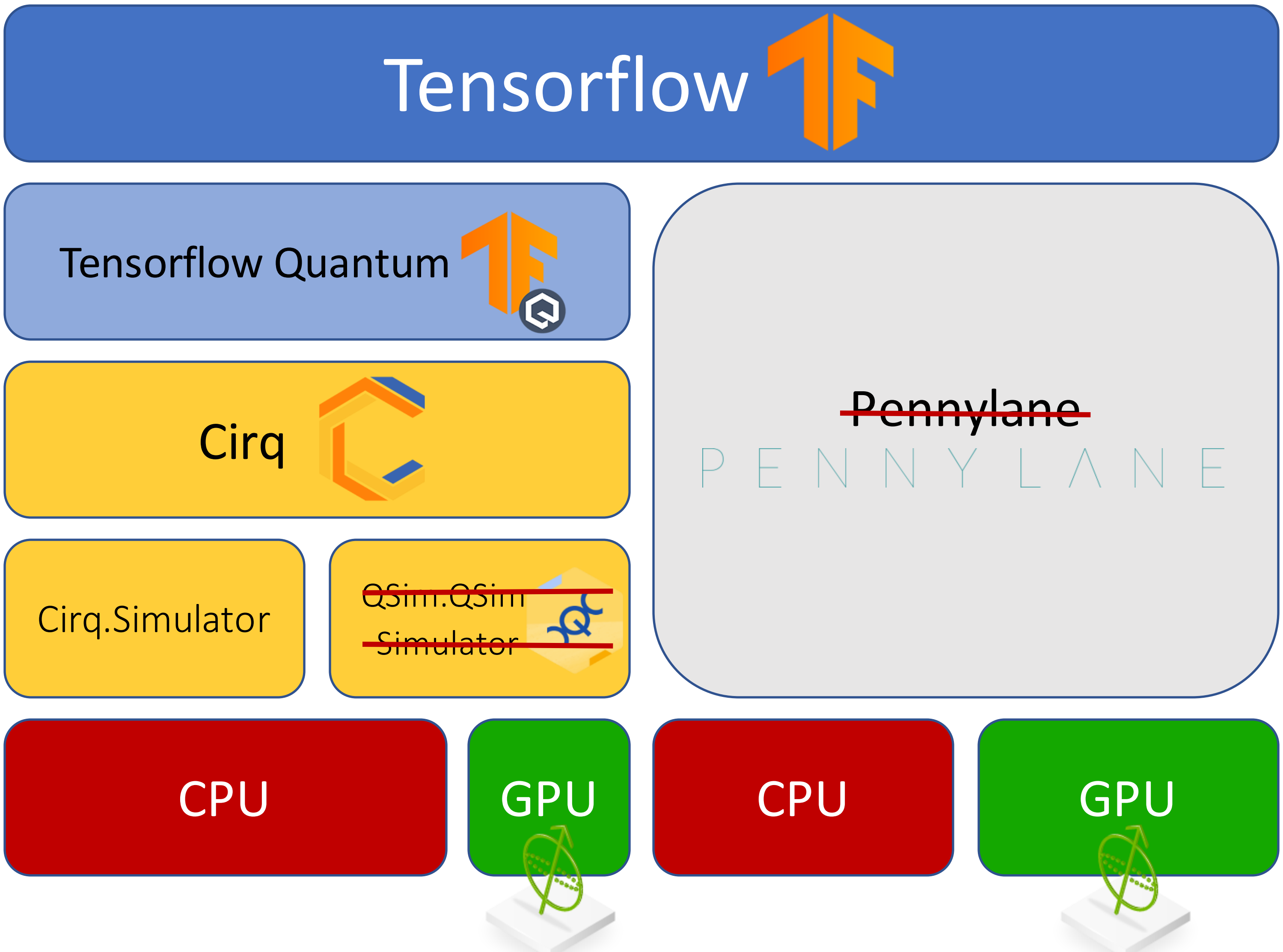
Tools and Resources

Quantum simulation tools, APIs, hardware, and optimizations

Tools and **libraries** for ML and QML



Tools and **libraries** for ML and QML



Hardware

cloudveneto

PhysicsOfData-students

HPC-Physics

LCPB-CittaRomanze

- Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz, 15 cores
- 90 GB RAM
- 5 TB storage
- NVIDIA Tesla T4, 16 GB

qmlJUNO

- Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz, 18 cores
- 56 GB RAM
- 5 TB storage
- NVIDIA Tesla V100, 32 GB

Running NNs on **large** datasets

```
def get_data_from_filename(filename):
    # Read the corresponding label file
    labelfile = '../../juno_data/data/real/train/targets/targets_train_{}.csv'\
        .format(re.findall('\d+', filename.decode())[0])
    labeldata = pd.read_csv(labelfile)
    labeldata = labeldata['edep'].to_numpy()

    npdata = np.load(filename, mmap_mode='r')

    return (npdata, labeldata)

def get_data_wrapper(filename):
    # Assuming here that both your data and label is double type
    features, labels = tf.numpy_function(
        get_data_from_filename, [filename], (tf.float64, tf.float64))
    return tf.data.Dataset.from_tensor_slices((features, labels))

# Create dataset of filenames.
ds = tf.data.Dataset.from_tensor_slices(filelist)
# Retrieve .npy files
ds = ds.flat_map(get_data_wrapper)
# Optimizations
ds = ds.apply(tf.data.experimental.prefetch_to_device("/GPU:0"))
ds = ds.batch(BATCH_SIZE, num_parallel_calls=tf.data.AUTOTUNE,
              deterministic=False)
```

488 GB dataset to train the ResNet on **doesn't fit** into RAM

Need **lazy load** to RAM, i.e. read data from file only when needed

Exploit both **`tf.data.Dataset`** API and **`.npy`** **memory-mapping**

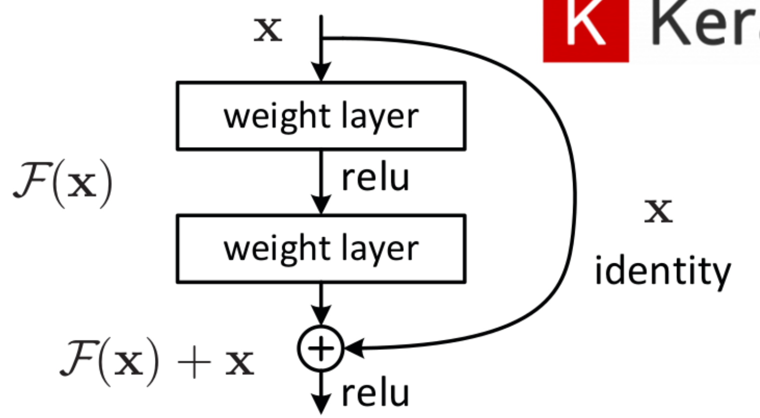
Constant data reading from storage at ~**270 MB/s**, small RAM and VRAM usage, training speed-up (**GPU load ~90%**)

The classical **ResNet** machine learning model

Energy reconstruction with the **Residual Network** (**ResNet**)

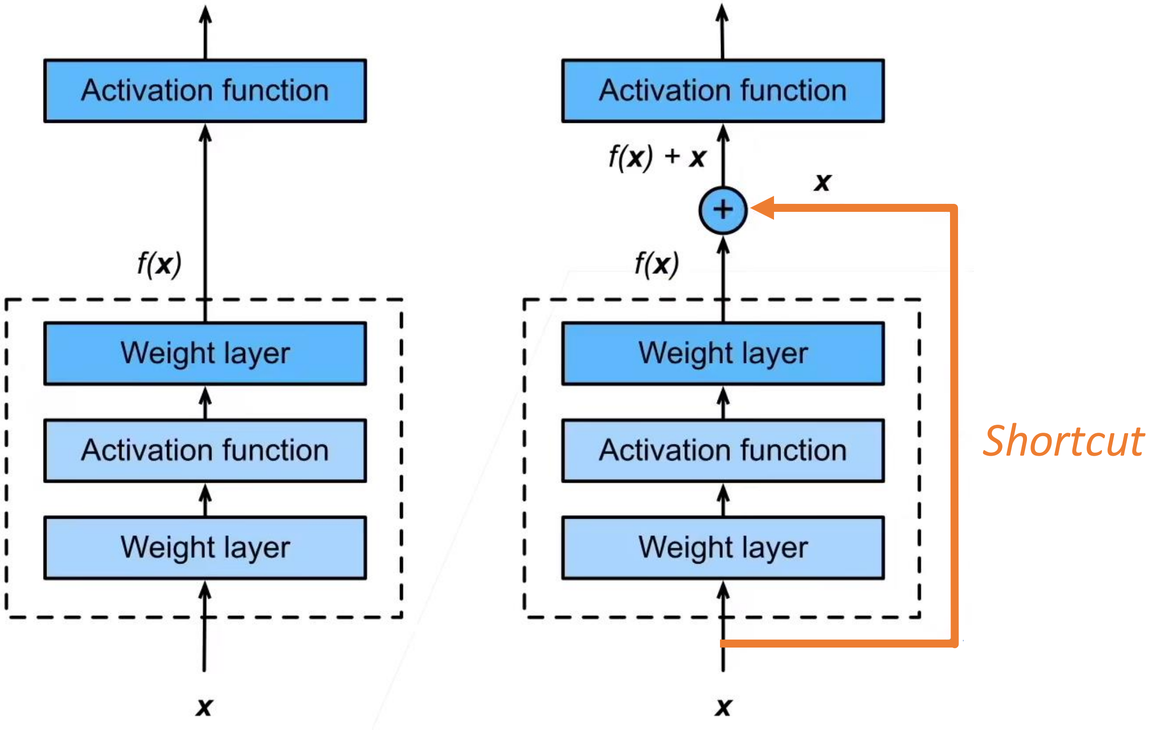
Residual Network (ResNet)

- Classical *deep* neural network model
- Key feature: *Residual Block* (or *Skip Connection*)
- Allows deeper network architectures
  - avoid *vanishing gradients*
  - mitigate *degradation* (accuracy saturation)
  - maximize reconstruction performance



D. Lazar, 2020, *Building a ResNet in Keras*, *link to web page*

The *Residual Block*

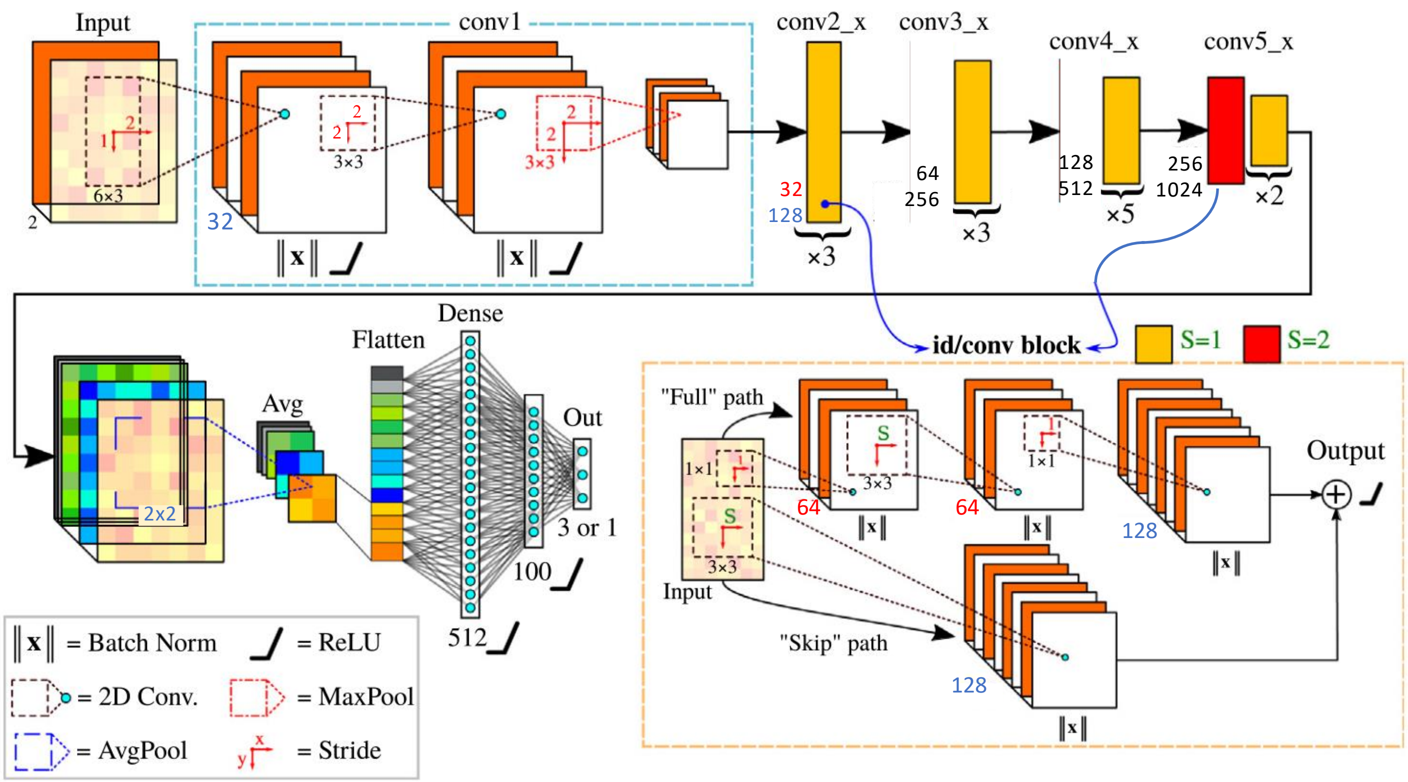


Regular block Residual block

A. Zhang et al., 2021, *Dive into Deep Learning, arXiv:2106.11342*

JUNO ResNet architecture

Input image size: (230,124,2)



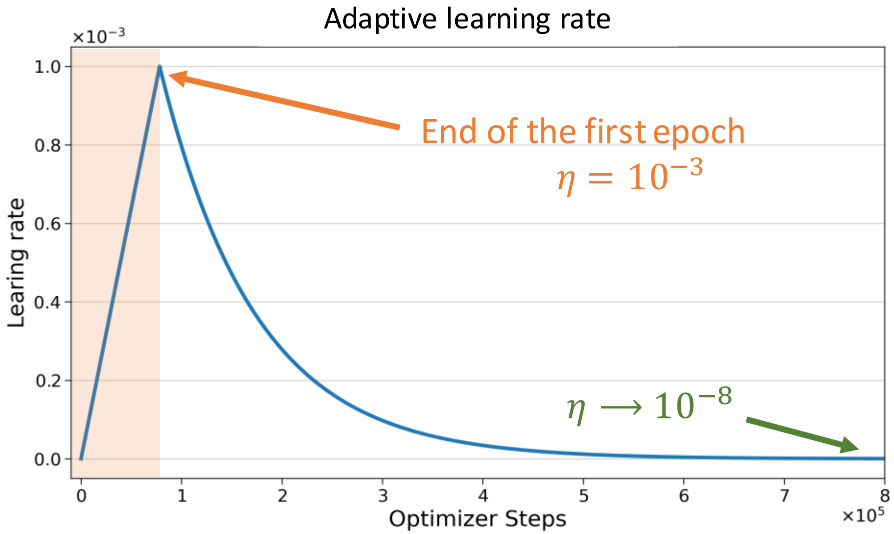
Total parameters	: 56 653 309
Trainable parameters	: 56 649 187
Non-trainable parameters	: 4 122

Y. Malyshkin et al., 2021, *Vertex and Energy Reconstruction in JUNO with Machine Learning Methods*, arXiv:2101.04839

Training the ResNet

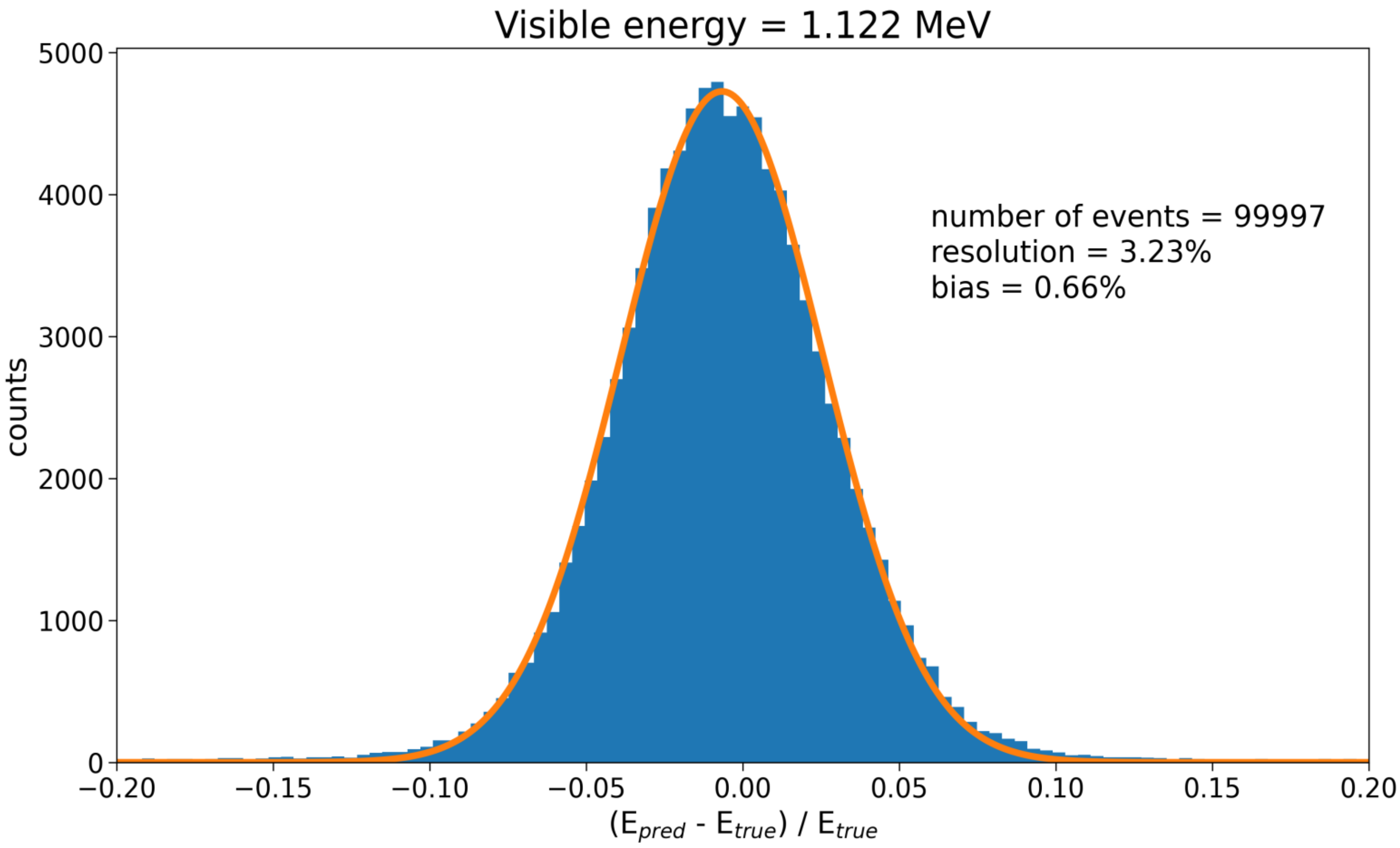
- Hardware: qmlJUNO VM
  - NVIDIA Tesla V100 GPU
- Training time: 2h15min (1d10h in total)
- Training parameters:
  - Number of epochs : 15
  - Batch size : 64
  - Optimizer : Adam
  - Loss : mean squared error
  - **Adaptive learning rate**:

$$\begin{cases} \mathrm{m} * step, & step \leq \mathrm{n°steps/epoch}^{(*)} \\ \mathrm{c} * \mathrm{rate}^{(step)}, & step > \mathrm{n°steps/epoch} \end{cases}$$

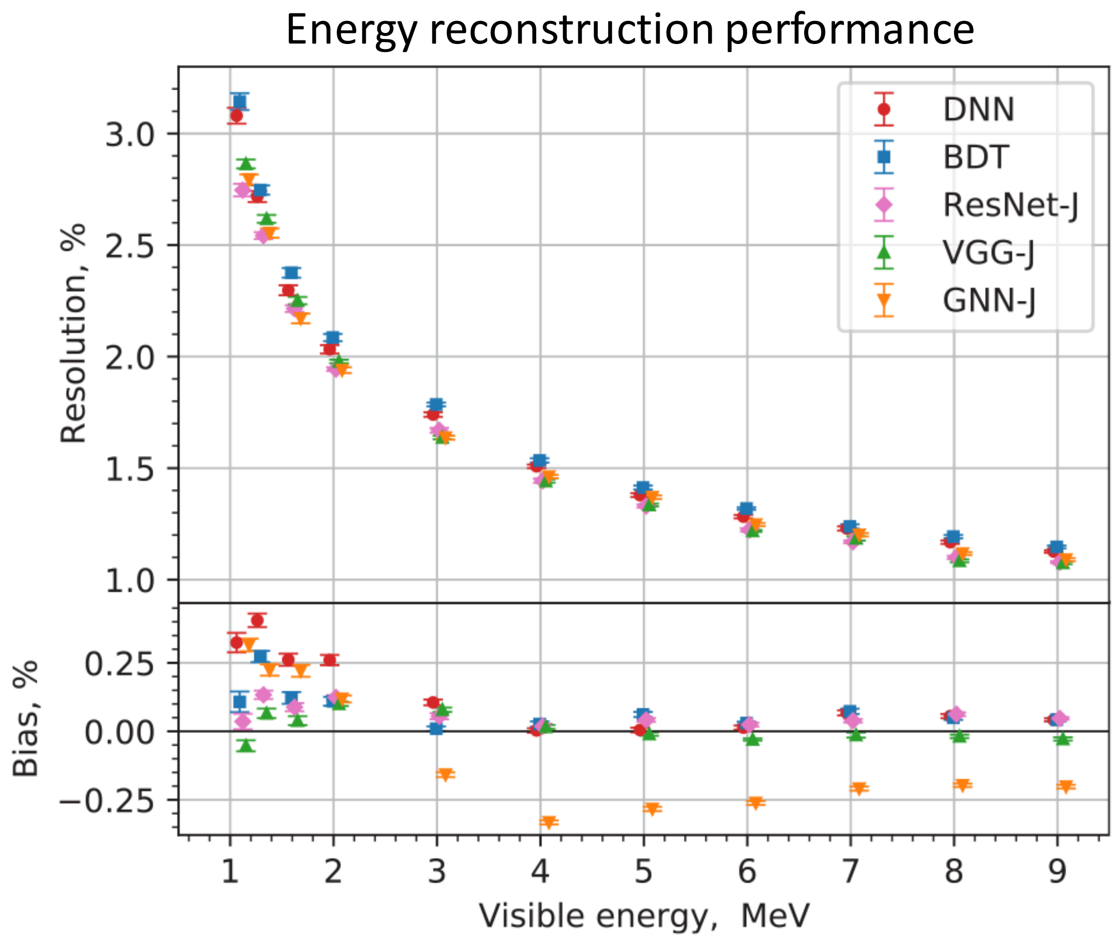


(*) n°steps/epoch = 78 282

Energy reconstruction distribution

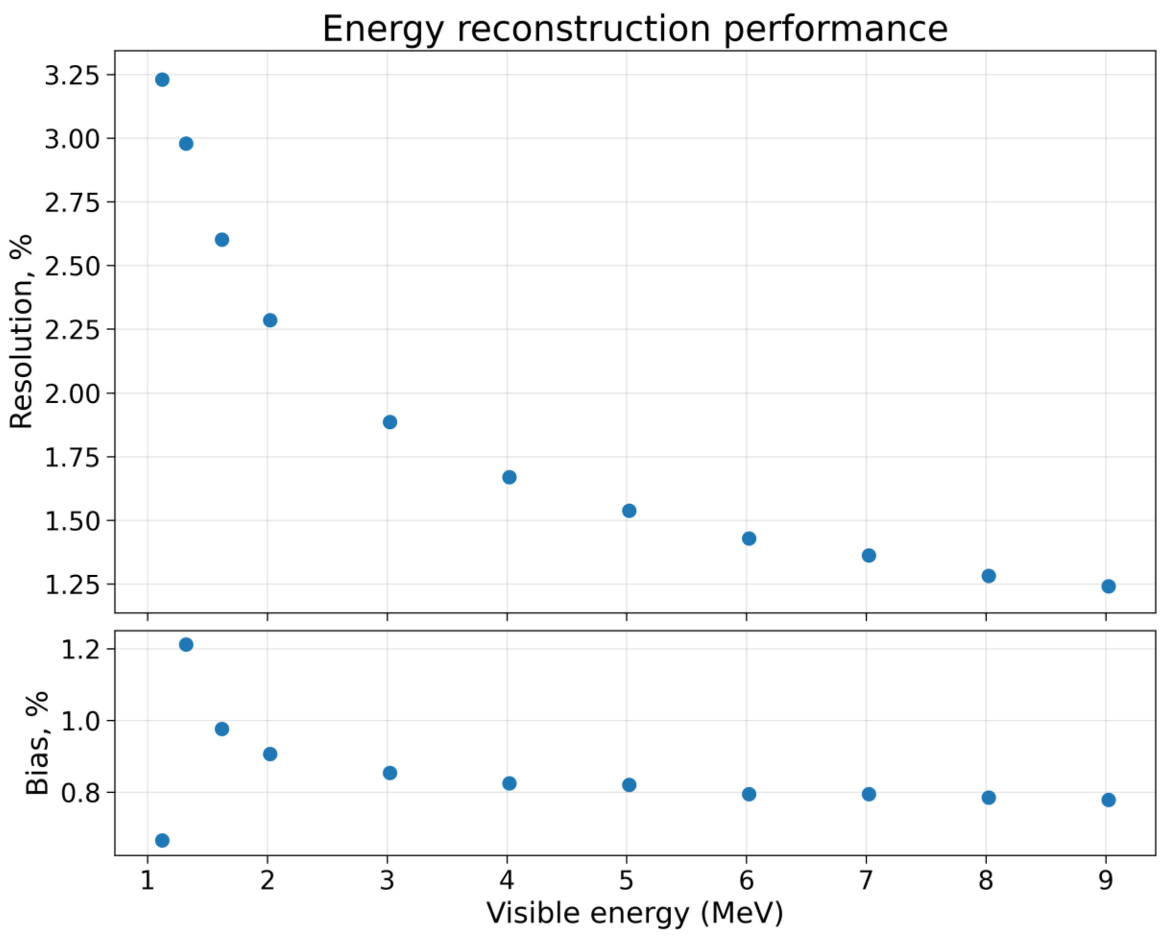


Prediction performance



Y. Malyshkin et al., 2021, *Vertex and Energy Reconstruction in JUNO with Machine Learning Methods*, arXiv:2101.04839

Prediction performance



Final considerations on ResNet

- The reconstruction resolution is ***acceptable***
- The prediction bias is ***higher*** than the one in previous studies

Our dataset is more ***realistic*** due to the simulation of electronic noise effects

- Data is harder to learn

The ResNet model has shown to be *highly sensible* to the ***training set size***

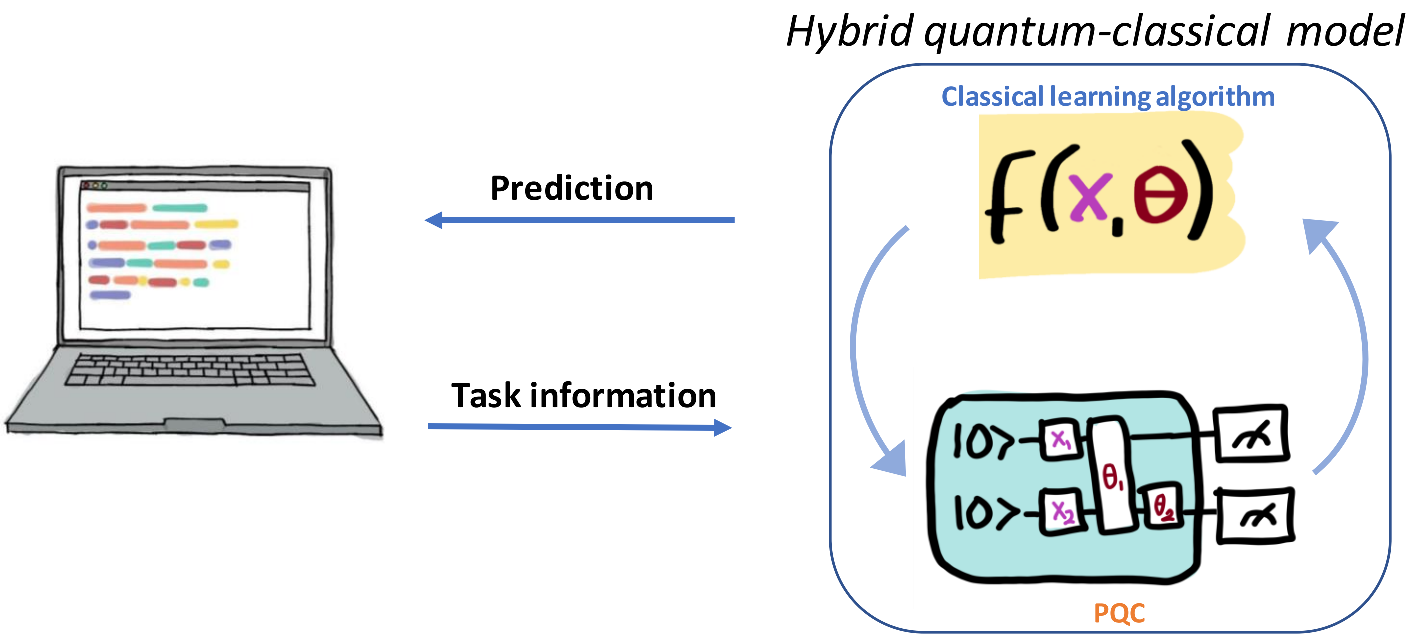
- Larger datasets are necessary

Classical and Quantum convolutional models

Energy reconstruction with a **CNN** and its **quantum** twin

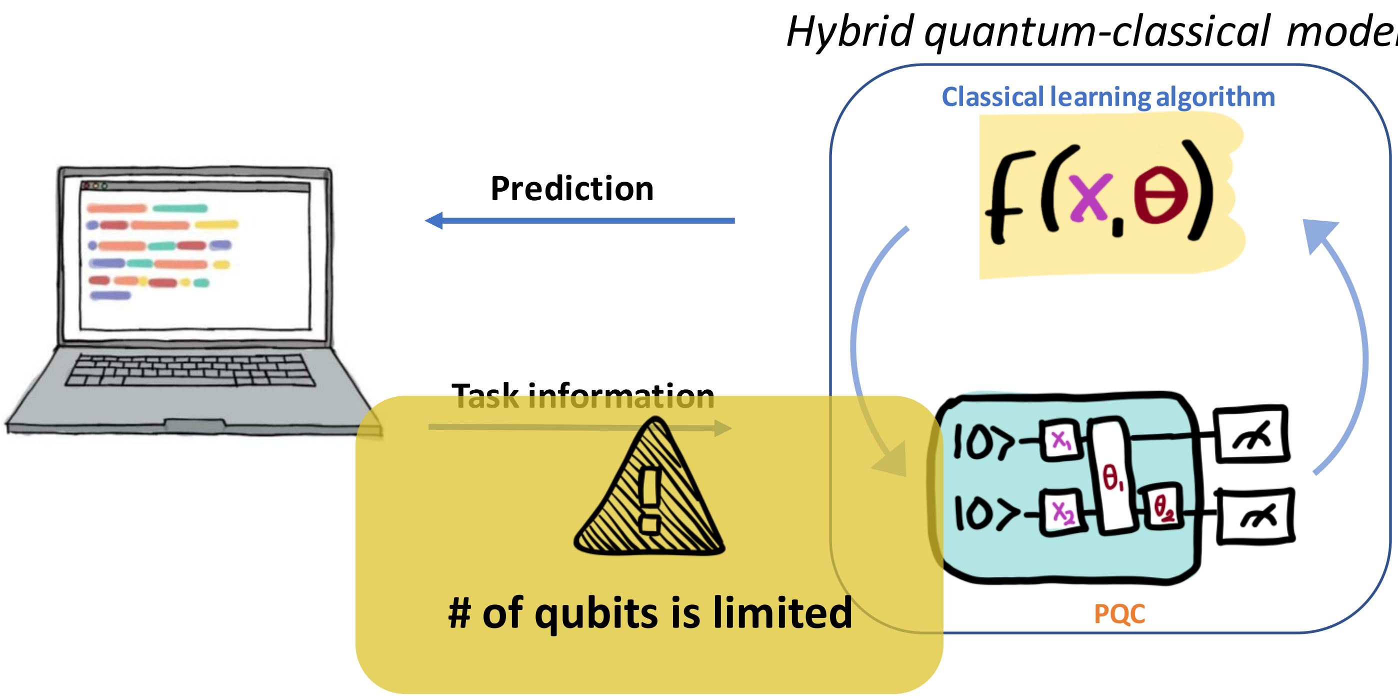
Quantum Machine Learning

- Use the advantages of **quantum computing** in order to improve **machine learning algorithms**
- Parametrized Quantum Circuits (**PQCs**) as machine learning model:

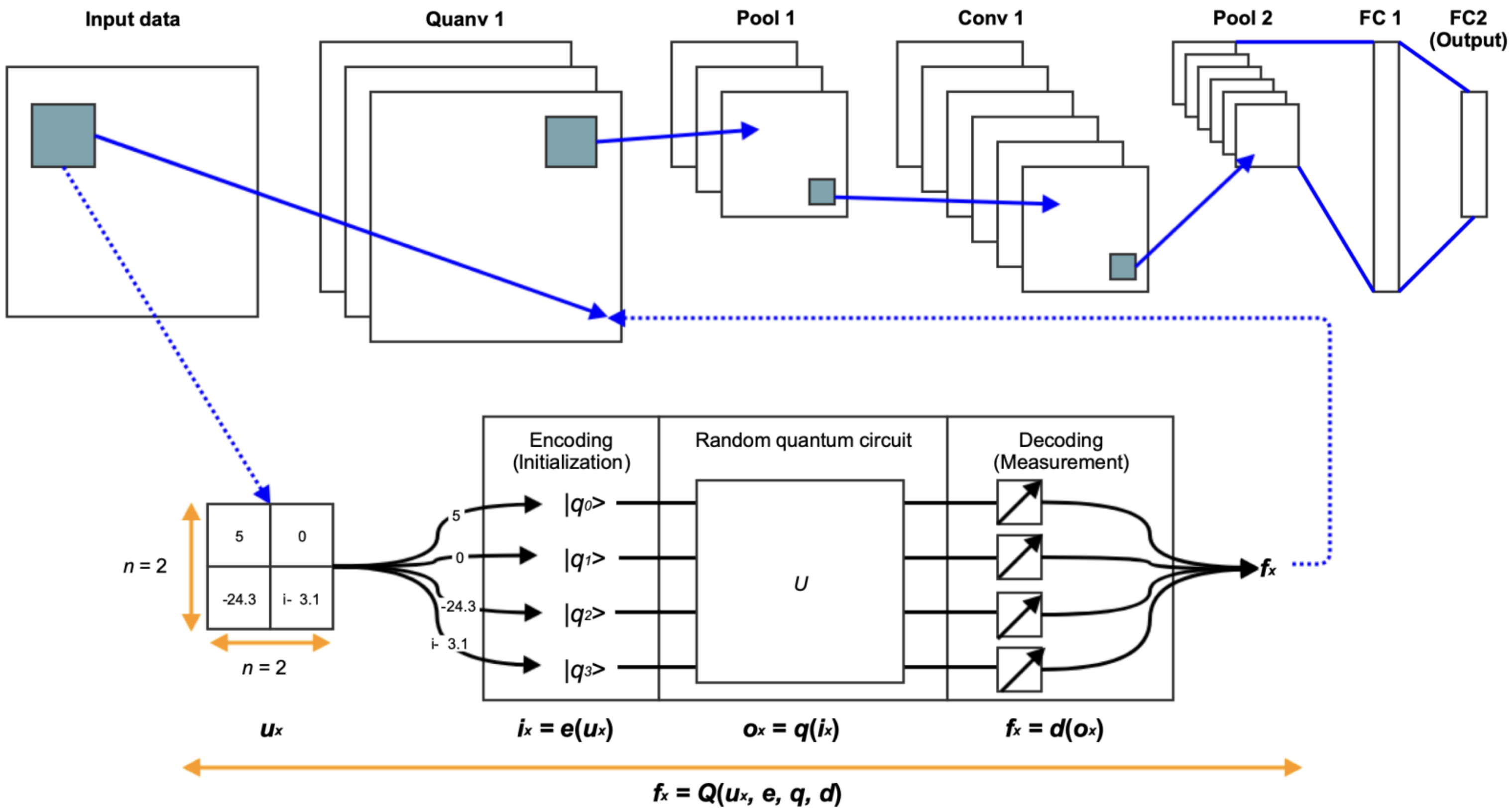


Quantum Machine Learning

- Use the advantages of **quantum computing** in order to improve **machine learning algorithms**
- Parametrized Quantum Circuits (**PQCs**) as machine learning model:



Trainable Quanvolutional Neural Network

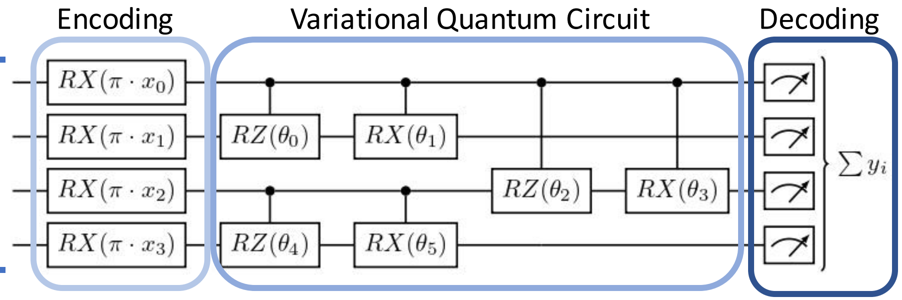


M. Henderson et al., 2020, *Quanvolutional neural networks: powering image recognition with quantum circuits*, arXiv:1904.04767

Trainable Quanvolutional Neural Network

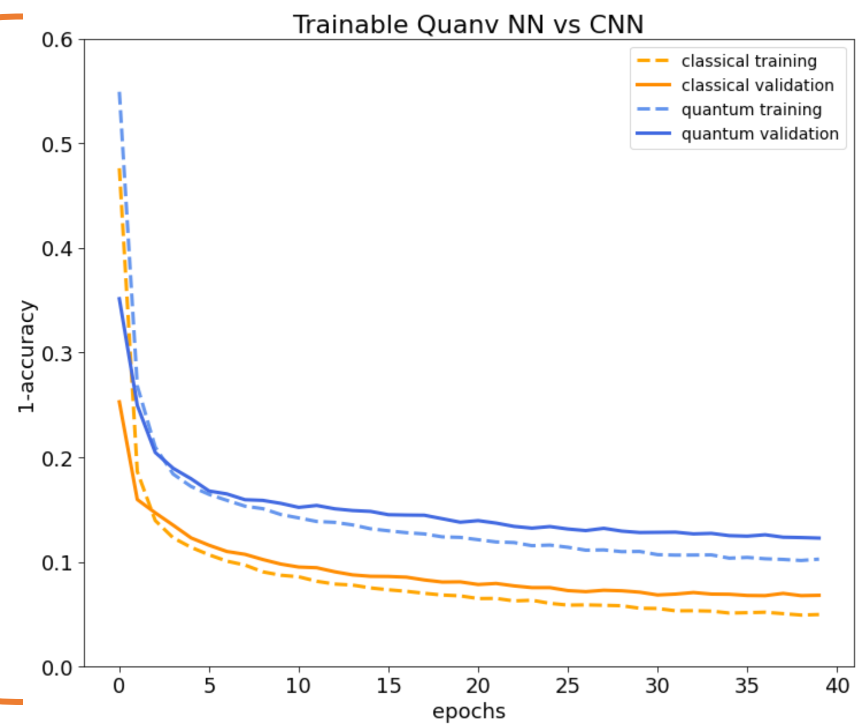
Ansatz

- Gate encoding
- 2 x qubits parameters



Benchmarking

- Simple model with a **single 2x2 QConv** layer + a 32 neurons Dense layer
- Trained on **5k 14x14** digit images (MNIST dataset)
- ~ **3 minutes**/ epoch
- Compared with its classical twin (QConv → **Conv2D**)



Lowering the Image Resolution

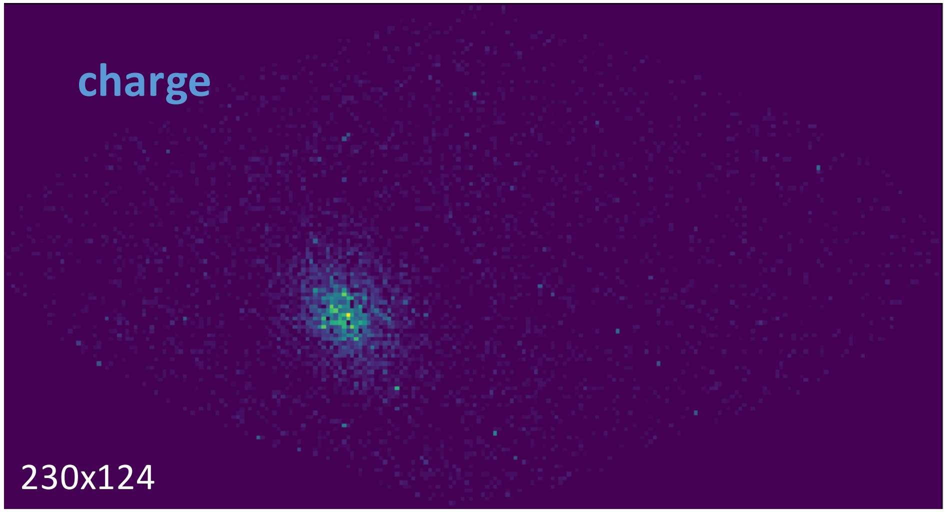
- Original image of **230x124** pixels
- PQC as filter in QConv layer benchmarked with 4 qubits (**2x2**)

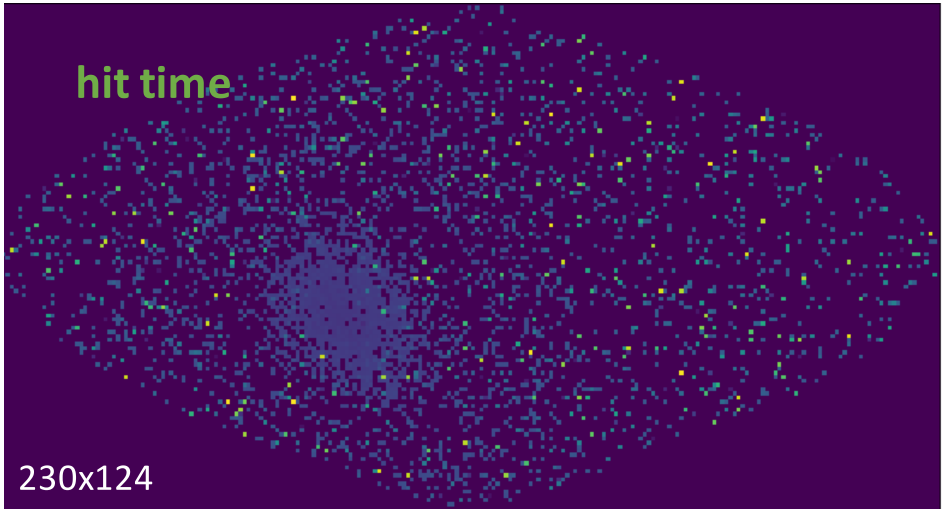
The first convolutional layer would need a **bigger filter** but that is beyond our simulation capabilities. We can **lower** the image **resolution**.

Costum 4x4 **Pooling** layer for each channel:

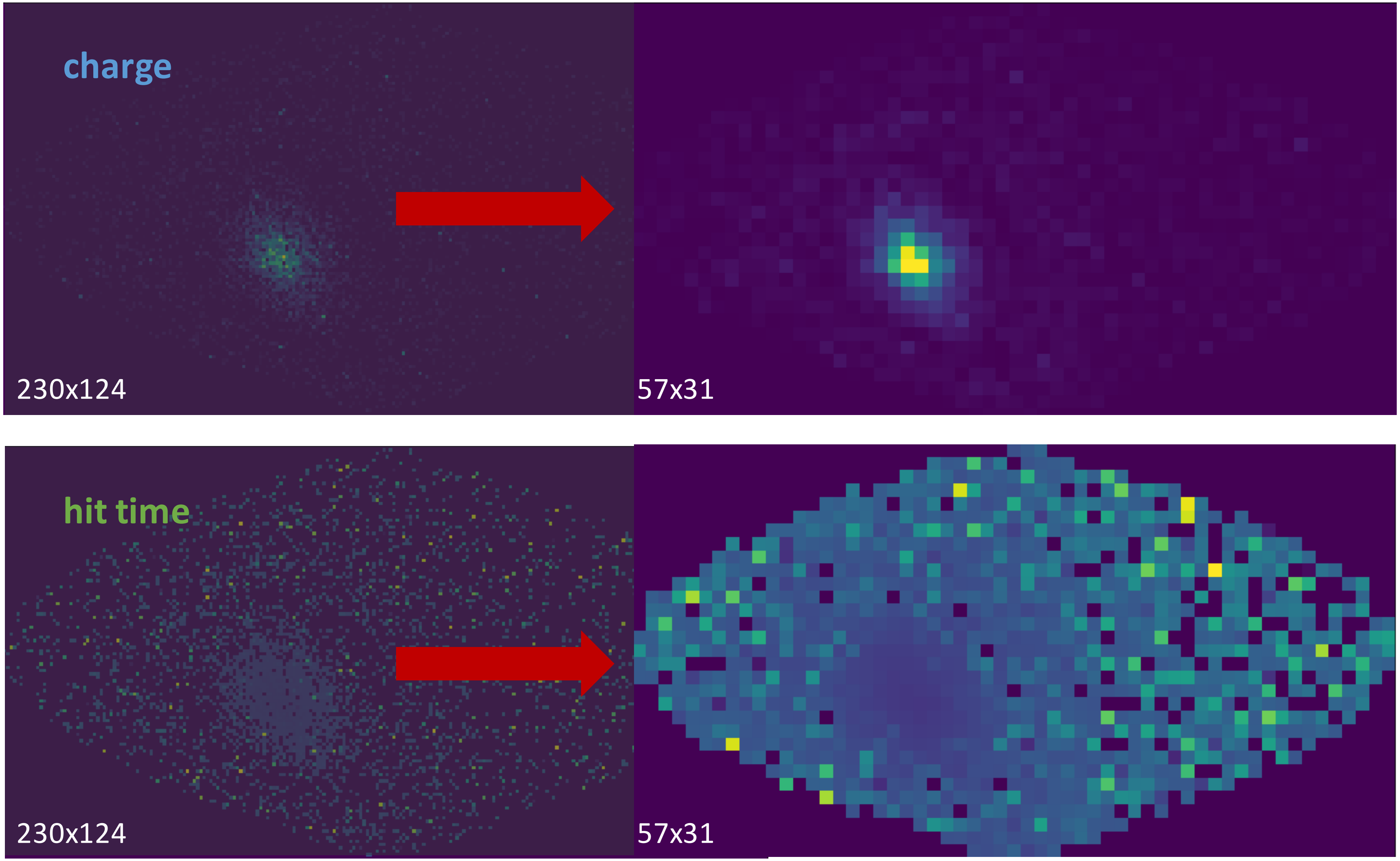
$$\text{Reduced Image}(m,n,c) = \begin{cases} \sum_{i,j} \text{Image}(i,j,c) & \text{if } c = \text{charge} \\ \sum_{i,j} \text{Image}(i,j,c) \cdot \text{Image}(i,j,\text{charge}) & \text{if } c = \text{hit time} \end{cases}$$

Lowering the Image Resolution

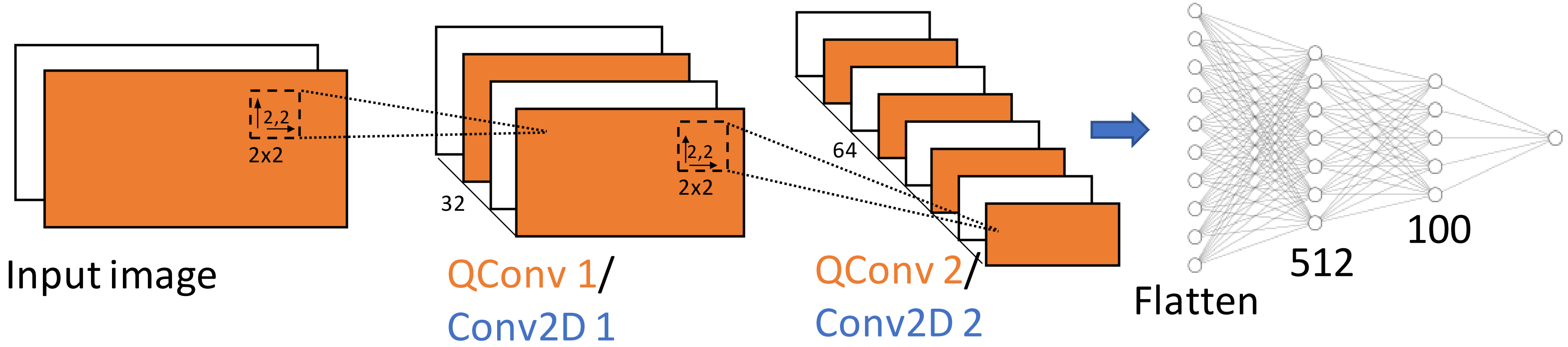




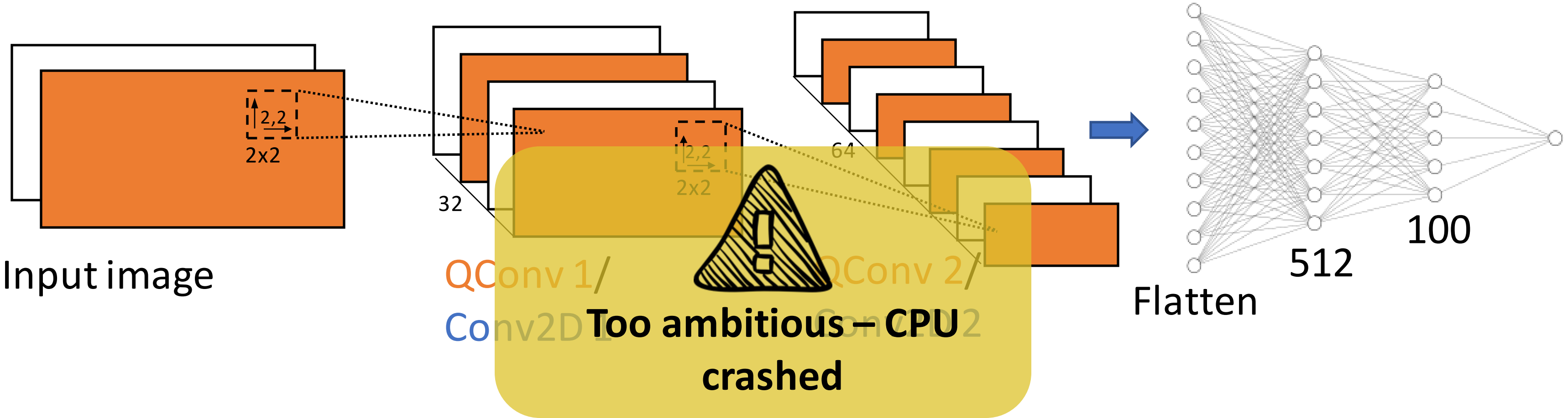
Lowering the Image Resolution



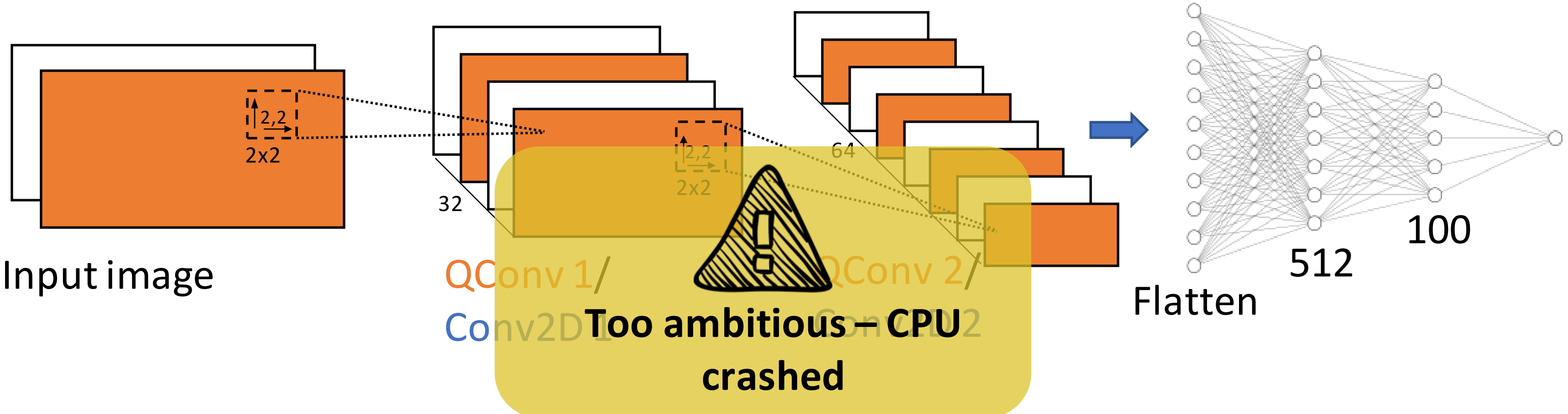
Classical and Quantum Models

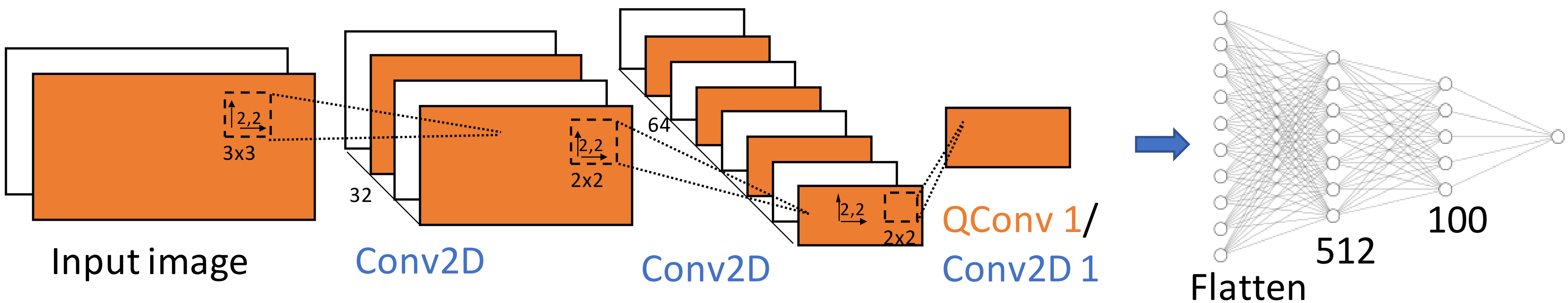


Classical and Quantum Models



Classical and Quantum Models



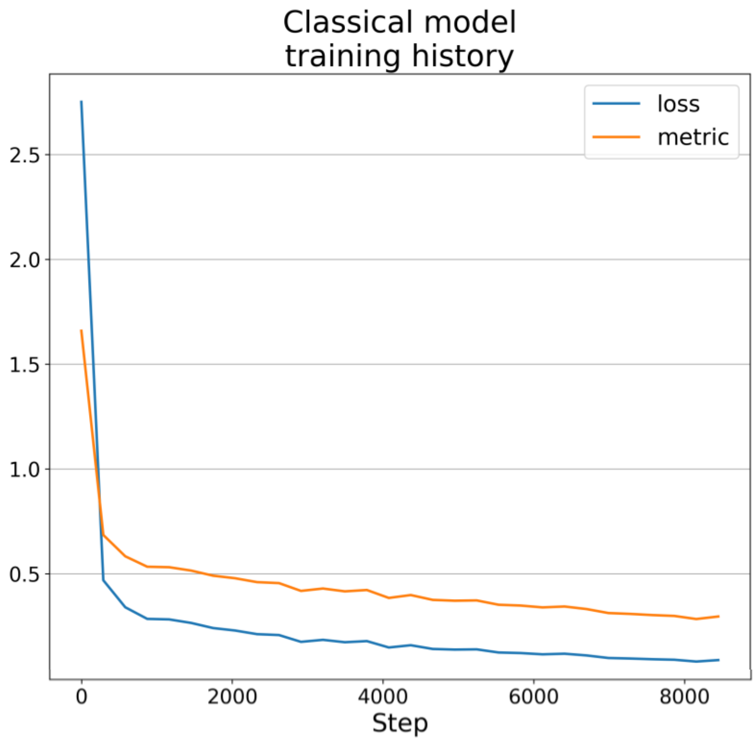


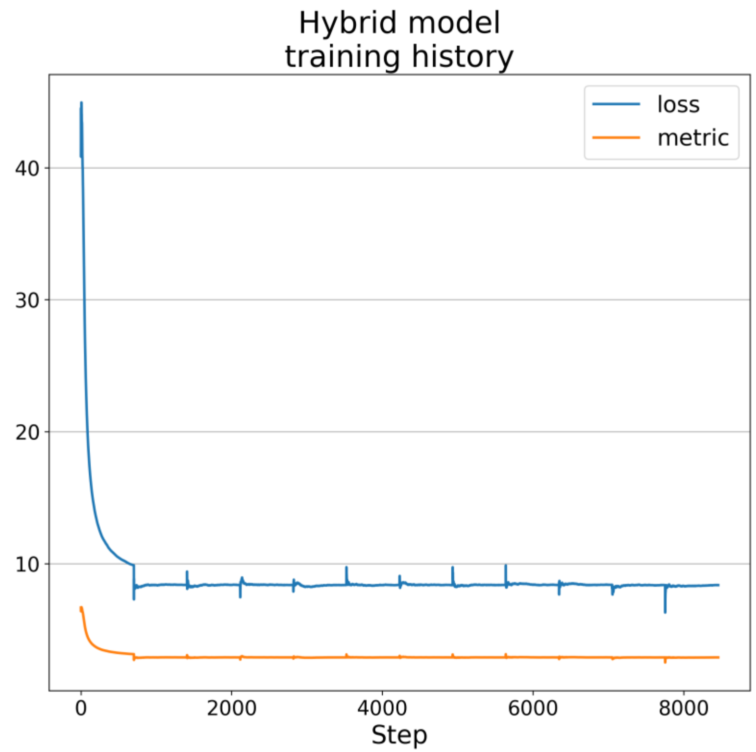
Performances

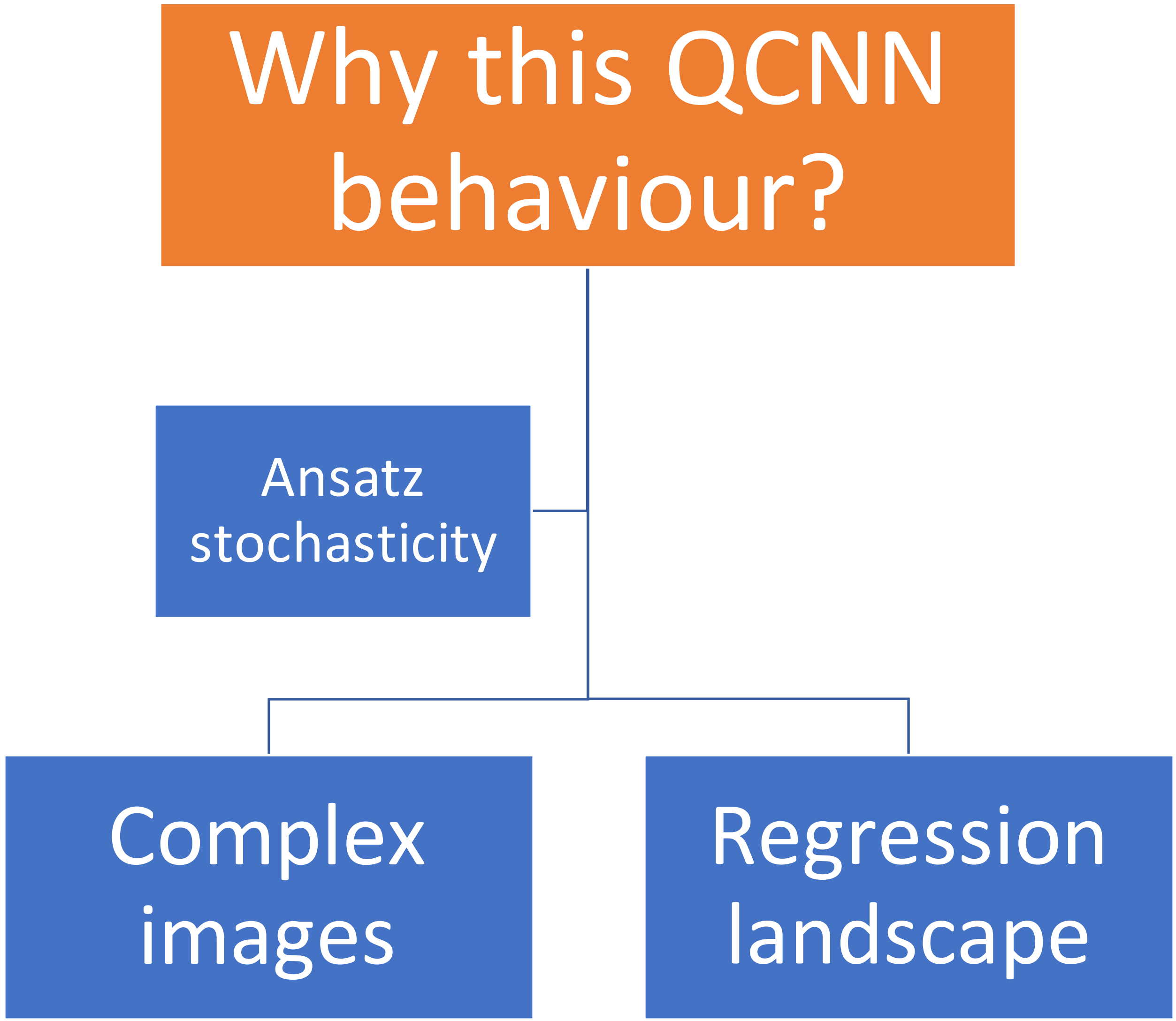
Trained on **50k** images, **6s/epoch** vs **1h30/epoch**

Performances

Trained on **50k** images, **6s/epoch** vs **1h30/epoch**






Why this QCNN behaviour?
Ansatz stochasticity
Complex images
Regression landscape

Conlcusions

Project achievements

ResNet

Is a fine working model

Results are comparable with the ones in the literature

Prediction bias is slightly higher

Project achievements

CNN / QCNN

The **QConv** Layer has been implemented and behave as expected	
On the **MNIST** dataset both **CNN** and **QCNN** have consistent results	
On the **JUNO** dataset the **CNN** performs as expected	
On the **JUNO** dataset the **QCNN** fails to learn data and falls into a local minimum	

Further improvements

CNN / QCNN

Implement a **QCNN** that runs on *GPUs*

Test *different ansatz* for the circuit

Find smarter ways to lower the *image resolution* and *complexity*

Use *larger datasets*

Backup Slides

Adaptive Learning Rate

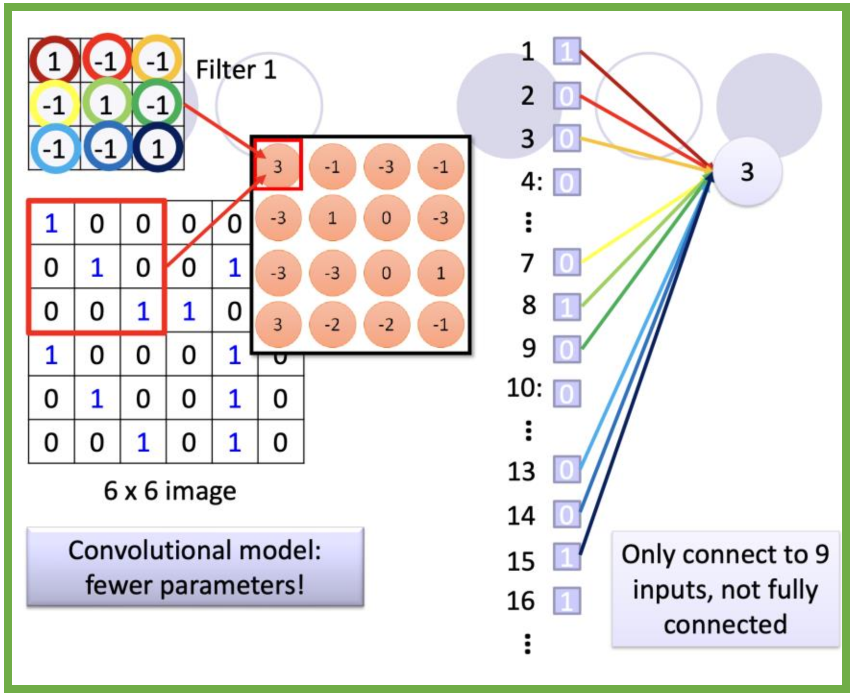
```
class MyLRSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):

  def __init__(self, initial_learning_rate, epochs, steps_per_epoch):
    self.initial_learning_rate = initial_learning_rate
    self.epochs = epochs
    self.steps_per_epoch = steps_per_epoch
    self.m = initial_learning_rate / steps_per_epoch
    self.decay_rate = tf.constant((10**-8 / initial_learning_rate)**(((epochs - 1)*steps_per_epoch)**-1), dtype=tf.float32)
    print('decay_rate:', self.decay_rate)

  def __call__(self, step):
    result = tf.cond(tf.less(step, self.steps_per_epoch),
                lambda: self.m * (step+1),
                lambda: self.initial_learning_rate * self.decay_rate**tf.cast(step+1-self.steps_per_epoch, dtype=tf.float32))

    return result
```

Introduction

From the classical convolution to the quantum circuit

Key features of CNNs

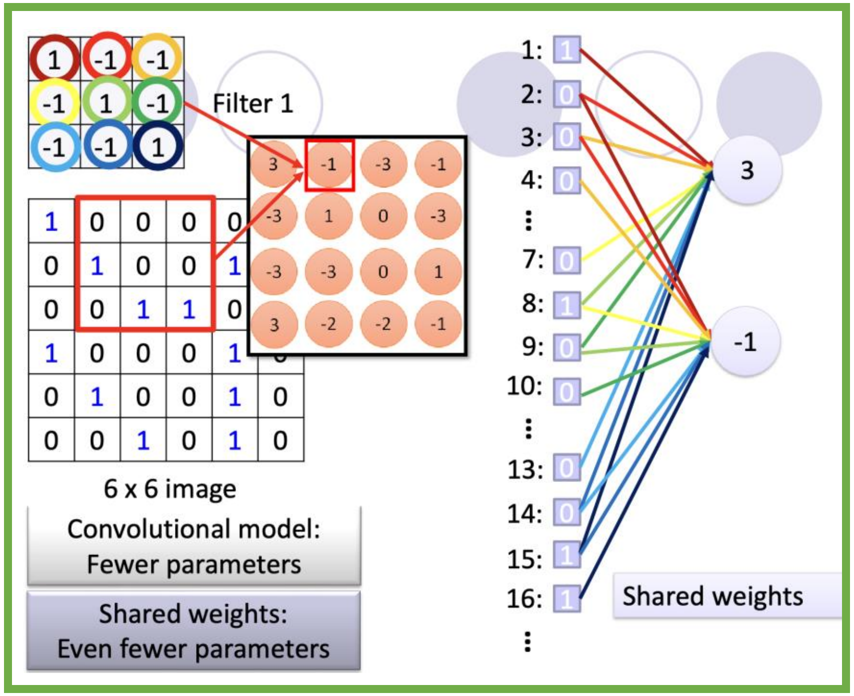
1. Local connectivity
2. Shared weights
3. Multiple feature maps
4. Pooling operations



The result of *1.* and *2.* translates into a *convolution* operation!

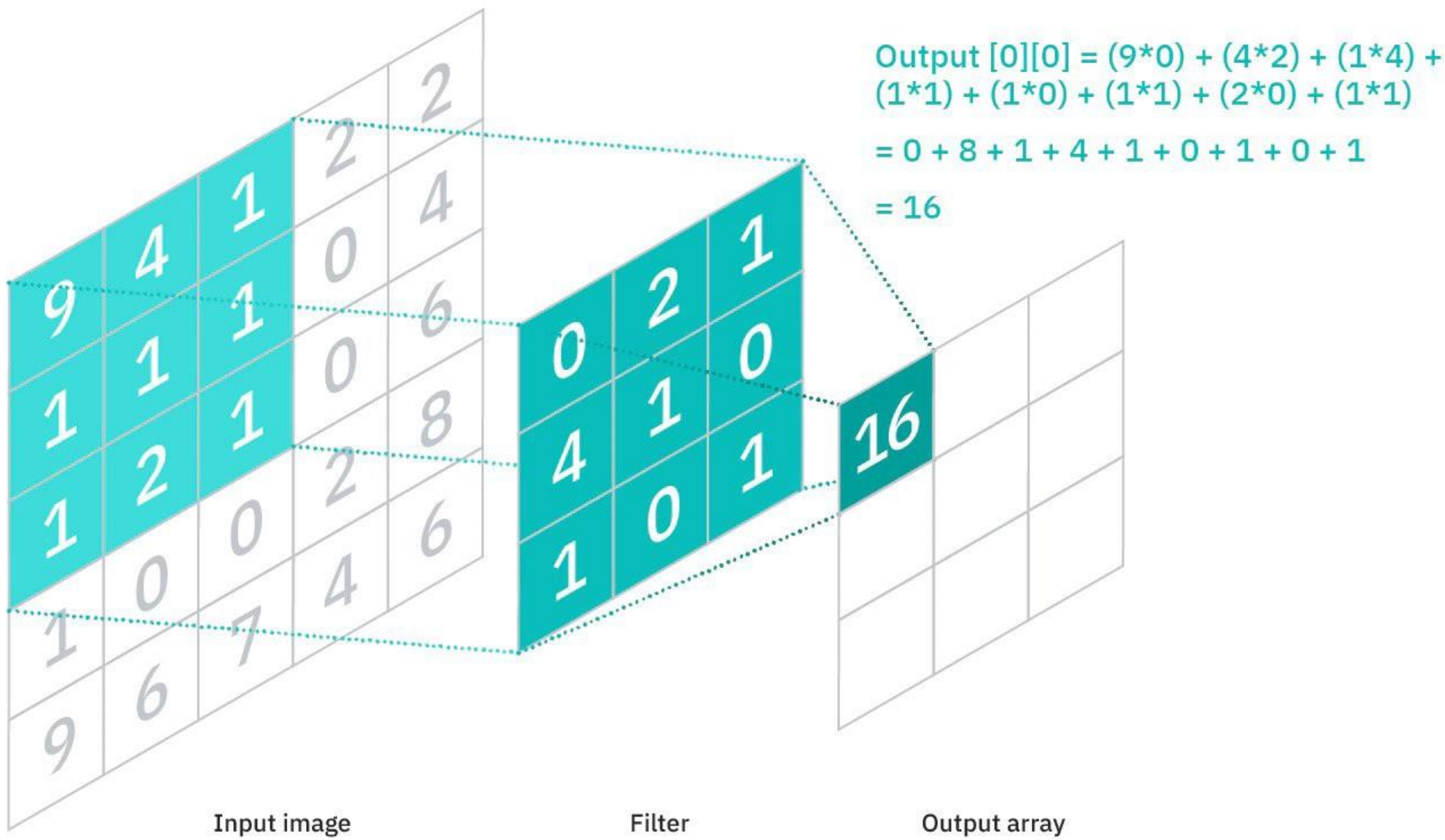
Key features of CNNs

1. Local connectivity
2. Shared weights
3. Multiple feature maps
4. Pooling operations

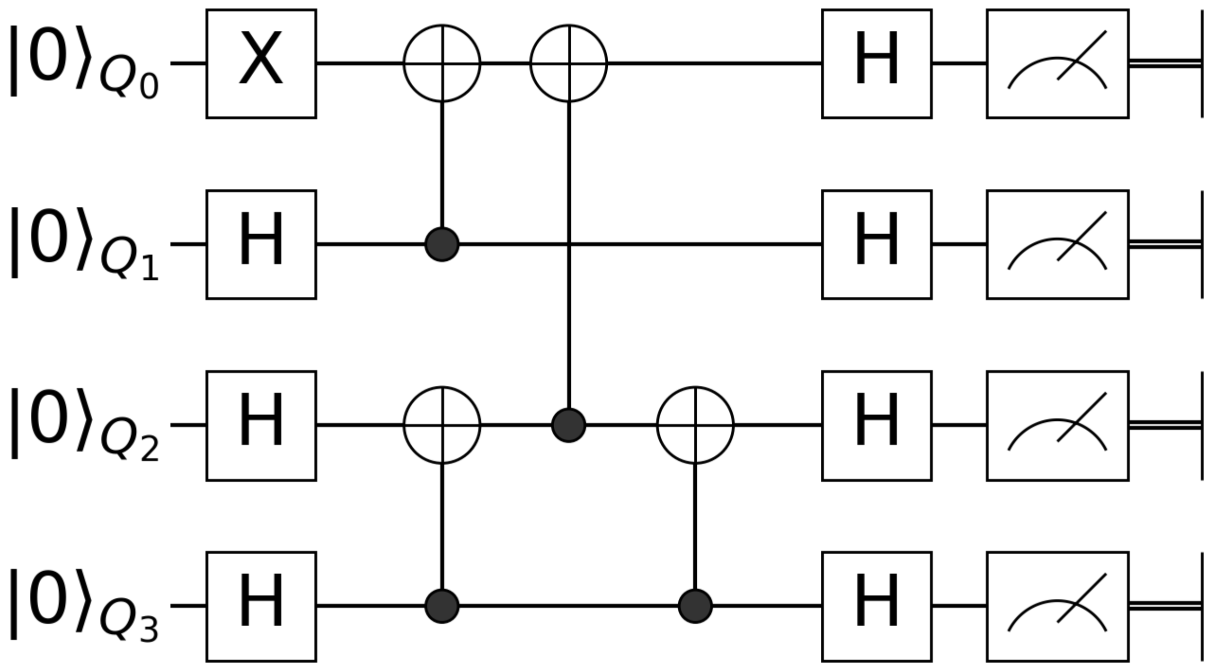


The result of *1.* and *2.* translates into a *convolution* operation!

From the classical Convolution . . .



. . . to the Quantum Circuit

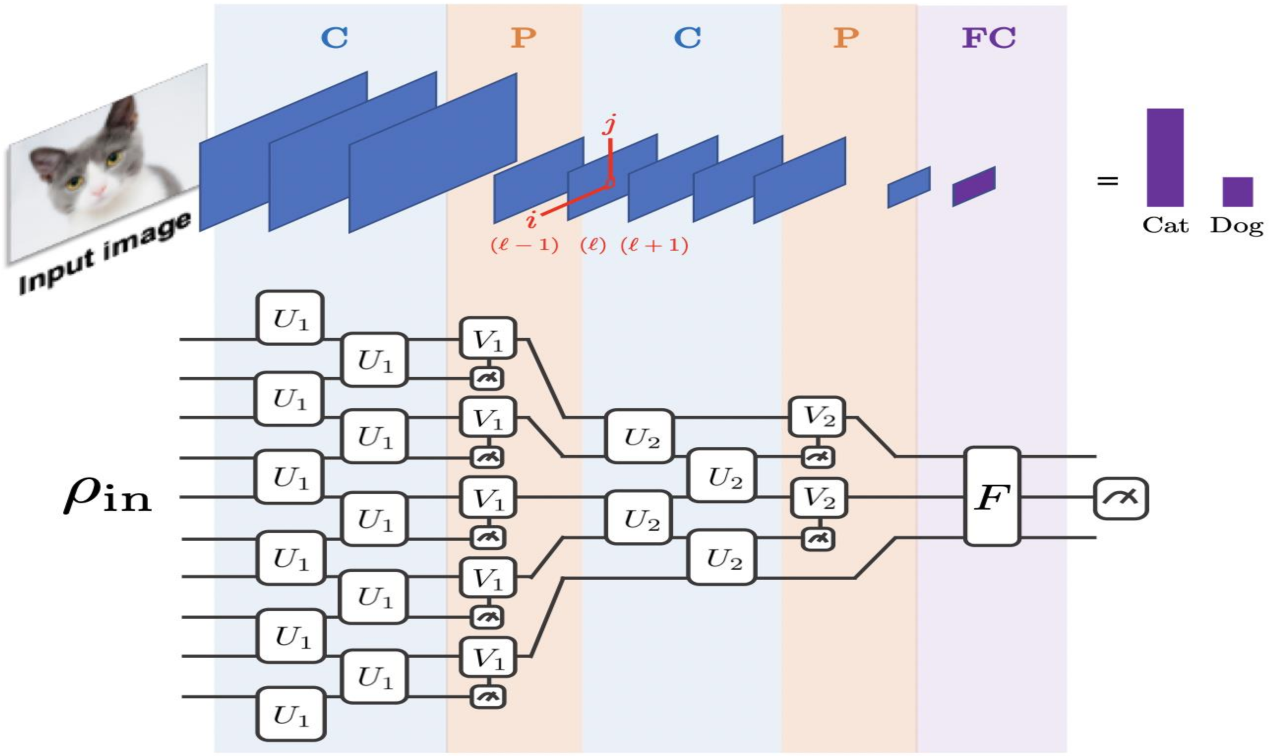


Quantum Convolutional machine learning models

A brief overview of the literature

The Quantum Convolutional Neural Network

In the literature the Quantum Convolutional model refers to

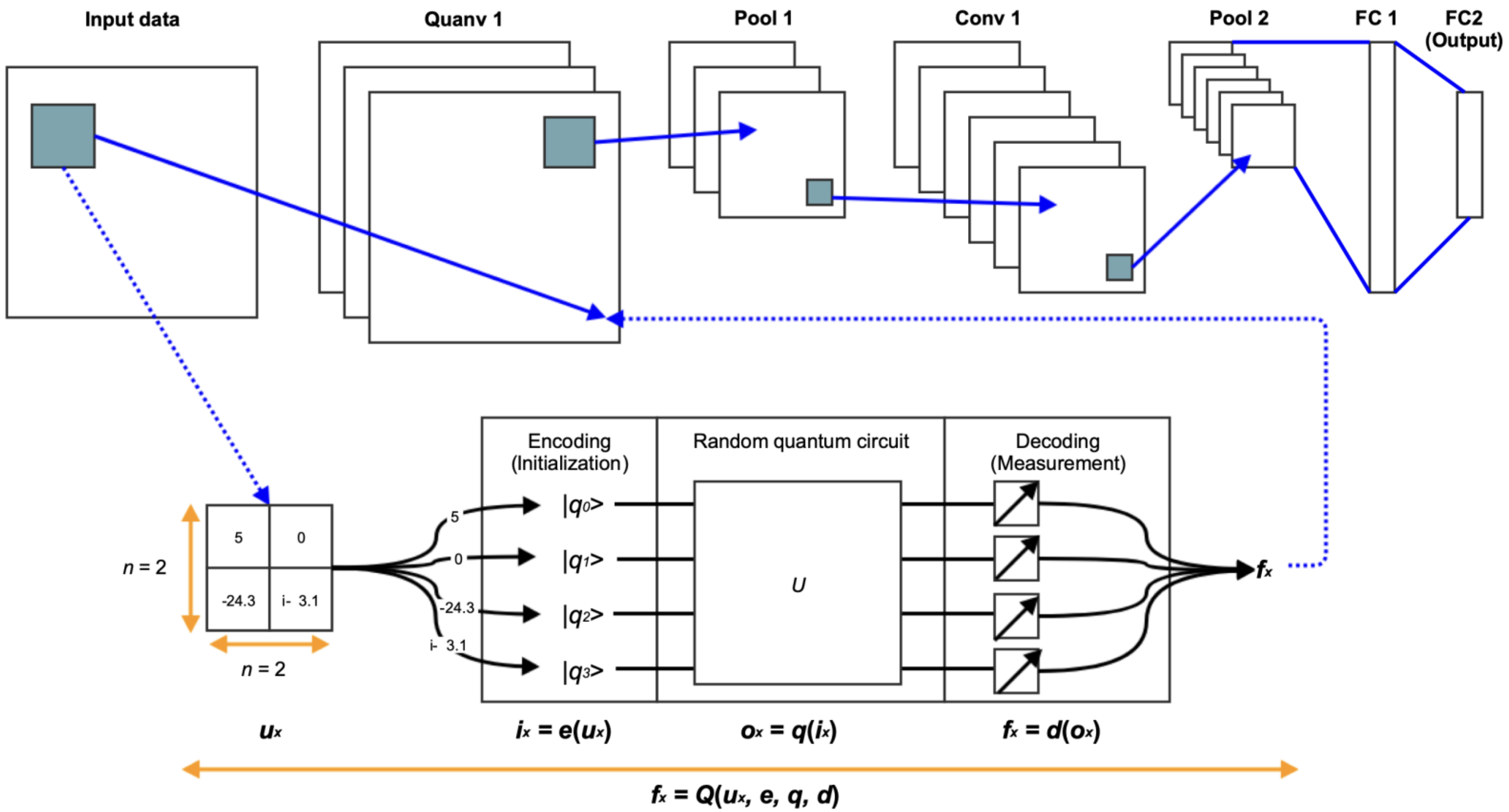


Why did we discard the Quantum Convolutional model?

The main issues with the Quantum Convolutional model for classical images processing are that

1. Too many qubits are required
2. An insane image reduction is crucial
3. The training process is extremely slow

Quanvolutional Neural Networks

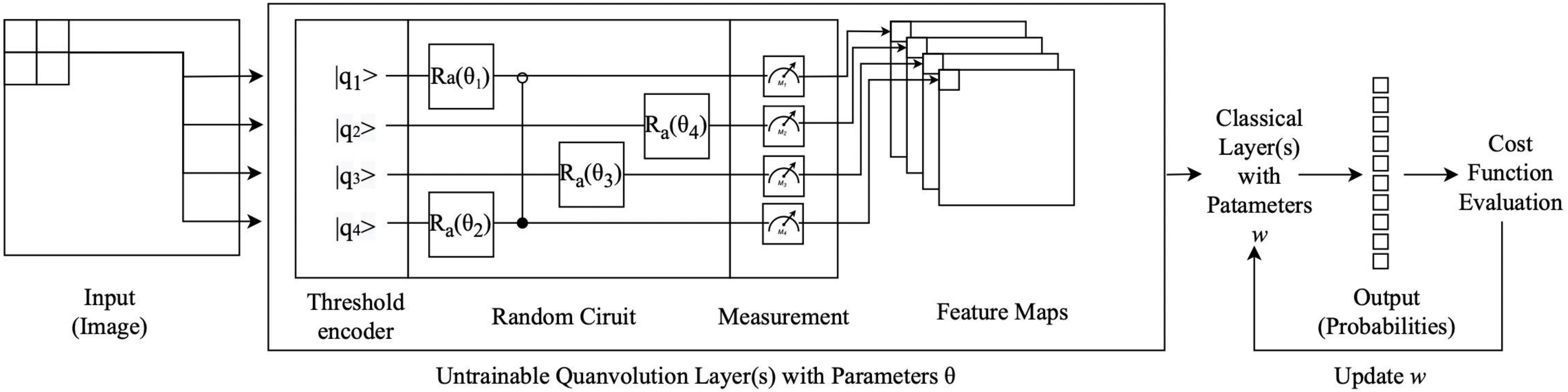


We tested two types of Quanvolutional Neural Network

1. The Untrainable Quanvolutional Neural Network
2. The Trainable Quanvolutional Neural Network

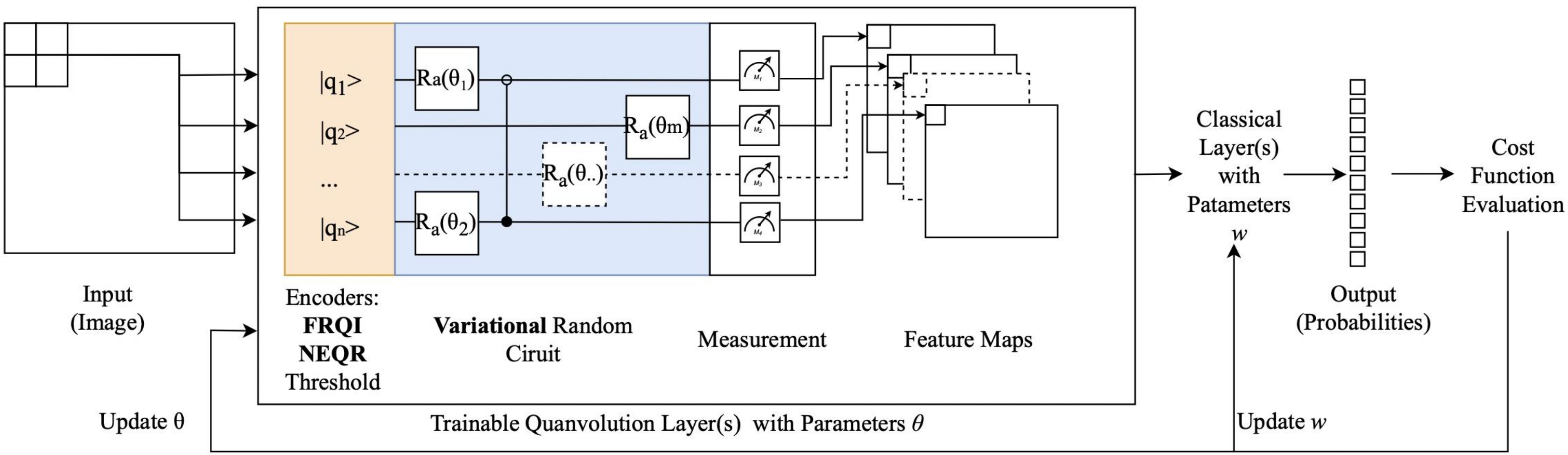
We tested two types of Quanvolutional Neural Network

1. The Untrainable Quanvolutional Neural Network
2. The Trainable Quanvolutional Neural Network



We tested two types of Quanvolutional Neural Network

1. The Untrainable Quanvolutional Neural Network
2. The Trainable Quanvolutional Neural Network



Quanvolutional model comparison

Using the MNIST handwritten digits dataset

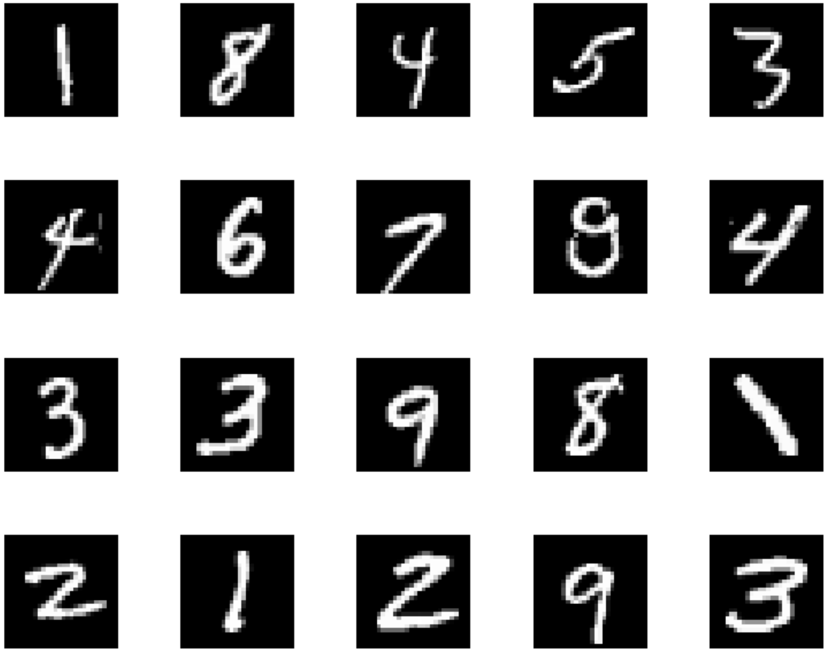
Tests are run on MNIST datasets

Original dataset

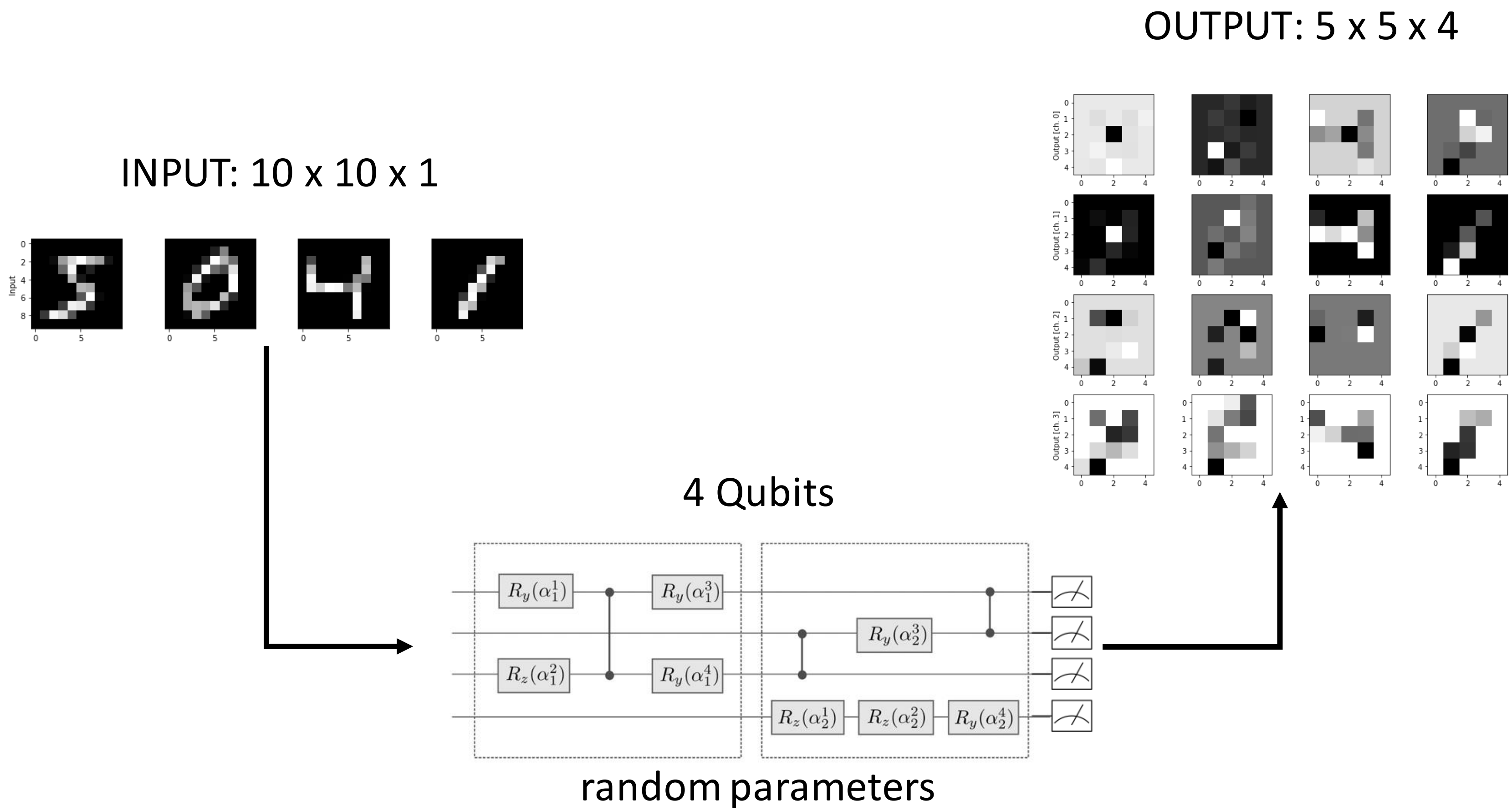
- 60k images
- 28 x 28 x 1 shape
- 10 classes

Pre-processed dataset

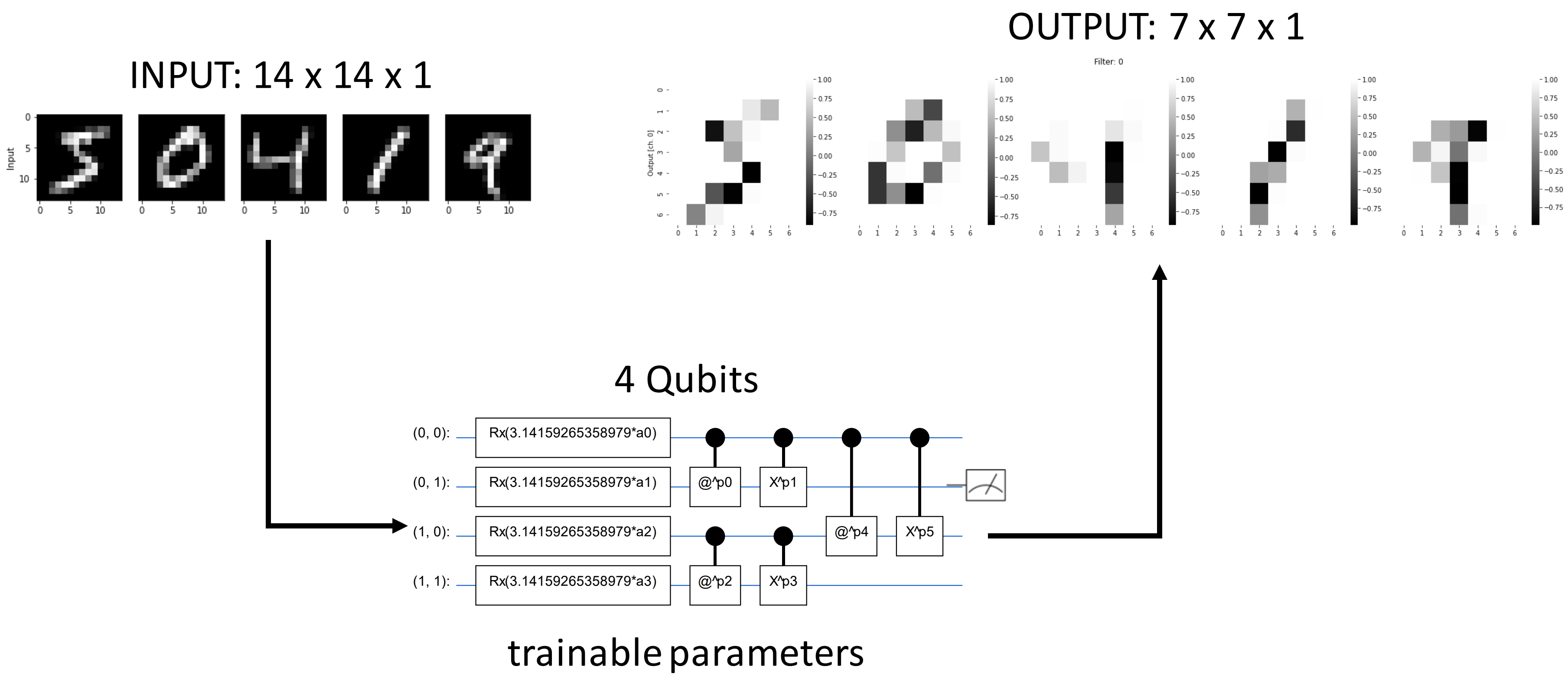
- 5k images
- 10 x 10 x 1 shape
- 10 classes



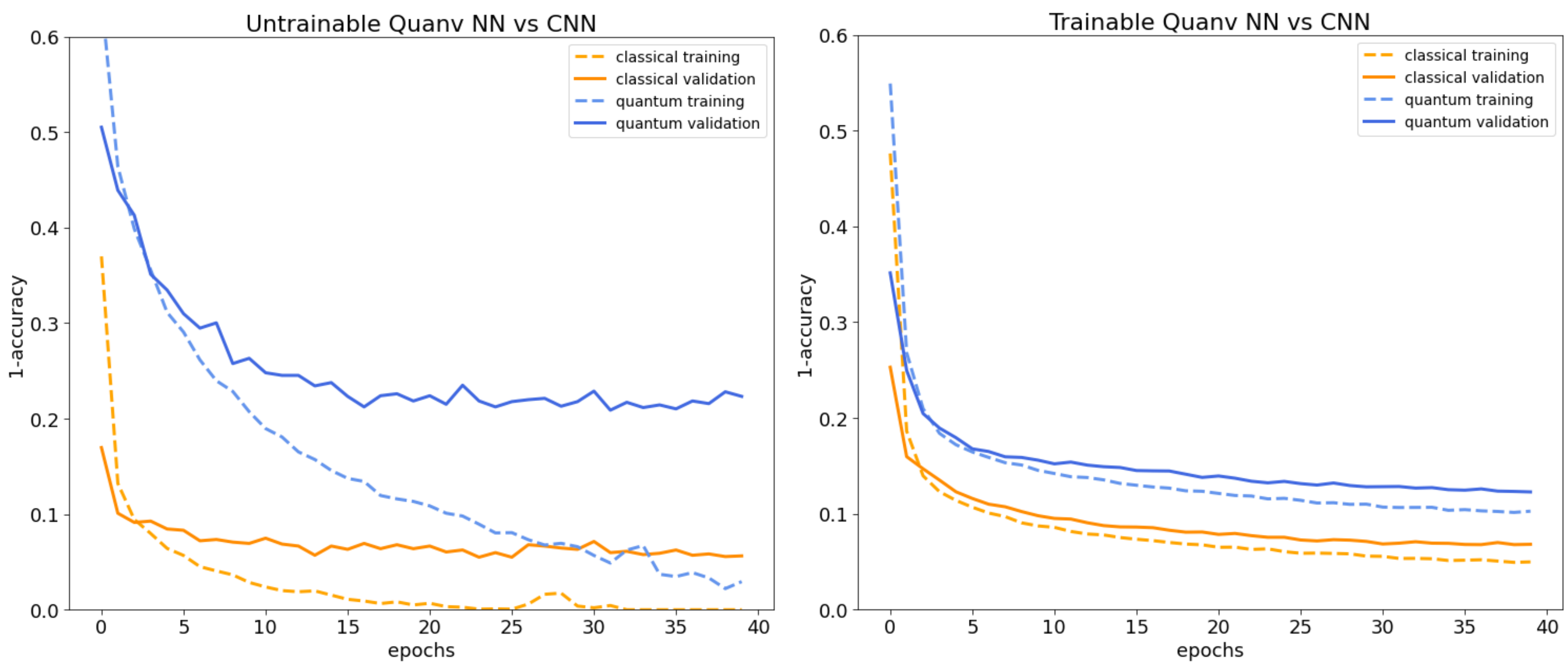
The untrainable quantum circuit in action



The trainable quantum circuit in action



Performance comparison: classical and quantum models



References

References

JUNO

1. Y. Malyshkin et al., 2021, *Vertex and Energy Reconstruction in JUNO with Machine Learning Methods*, arXiv:2101.04839
2. JUNO Collaboration, 2015, *Neutrino Physics with JUNO*, arXiv:1507.05613
3. JUNO Collaboration, 2015, *JUNO Conceptual Design Report,* arXiv:1508.07166

CLASSICAL MACHINE LEARNING

4. A. Zhang et al., 2021, *Dive into Deep Learning,* arXiv:2106.11342
5. D. Lazar, 2020, *Building a ResNet in Keras,* link to web page

QUANTUM MACHINE LEARNING

6. I. Cong et al., 2019, *Quantum Convolutional Neural Networks,* arXiv:1810.03787
7. Y. C. Chen et al., 2022, *Quantum convolutional neural networks for high energy physics data analysis,* arXiv:2012:12177
8. S. Oh et al., 2020, *A Tutorial on Quantum Convolutional Neural Networks,* arXiv:2009.09423
9. V. Bergholm et al., 2020, *PennyLane: Automatic differentiation of hybrid quantum-classical computations*, arXiv:1811.04968
10. M. Benedetti et al., 2019, *Parametrized quantum circuits as machine learning models*, arXiv:1906.07682
11. Y. C. Chen et al., 2021, *Hybrid Quantum-Classical Graph Convolutional Neural Network*, arXiv:2101.06189
12. M. Henderson et al., 2019, *Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits*, arXiv:1904.04767
13. D. Mattern et al., 2021, *Variational Quanvolutional Neural Networks with enhanced image encoding,* arXiv:2106.07327
14. M. Broughton et al., 2021, *TensorFlow Quantum: A Software Framework for Quantum Machine Learning,* arXiv:2003.02989
15. M. Schuld et al., 2014, *An introduction to quantum machine learning,* https://doi.org/10.1080/00107514.2014.964942