QCML

Hybrid Quantum Neural Networks for Remote Sensing Imagery Classification

Project presentation

VADHRI VENKATA RATNAM

July 2023

Confidential

Inside

- Remote sensing image classification
- Datasets
- Classical CNN
- Quantum hybrid CNN
- Submission & References

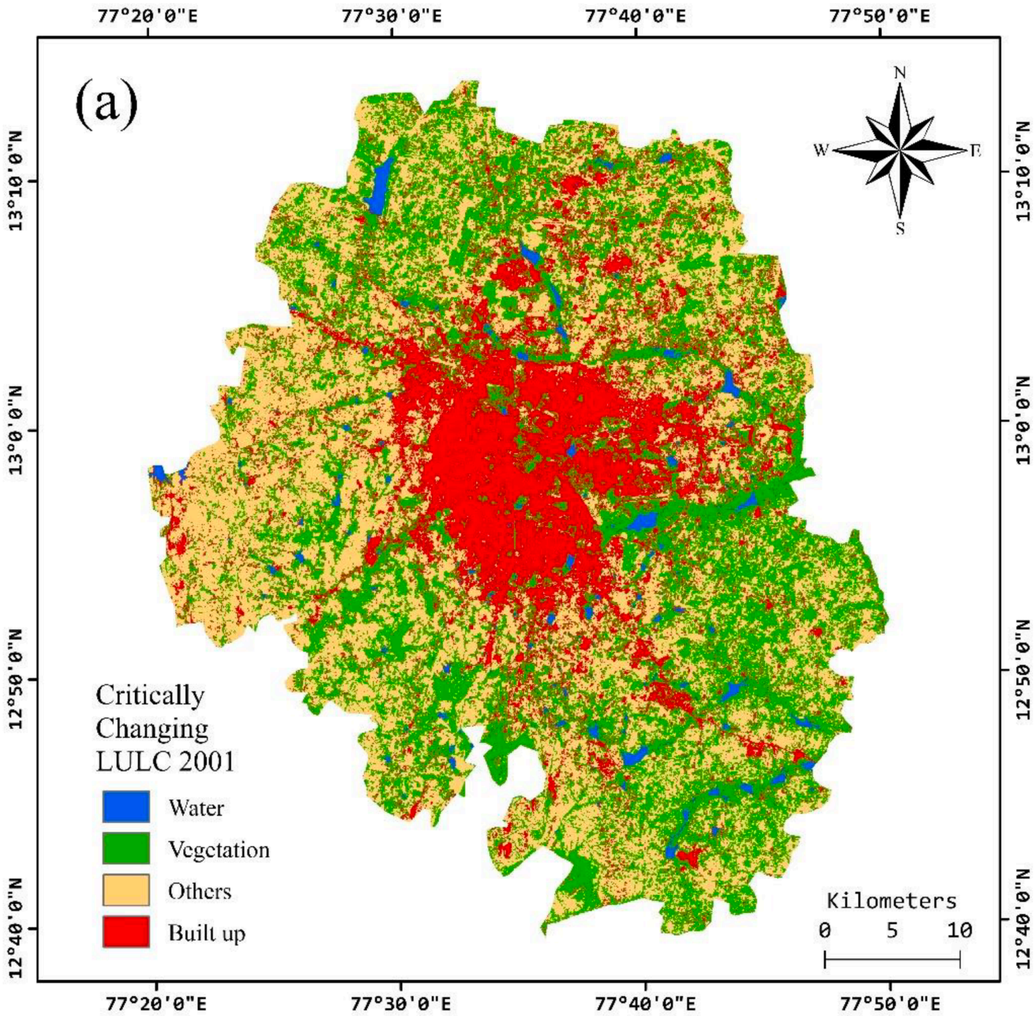
Confidential

Remote sensing image classification

Images captured through satellite and drones are used to observe surface of the Earth.

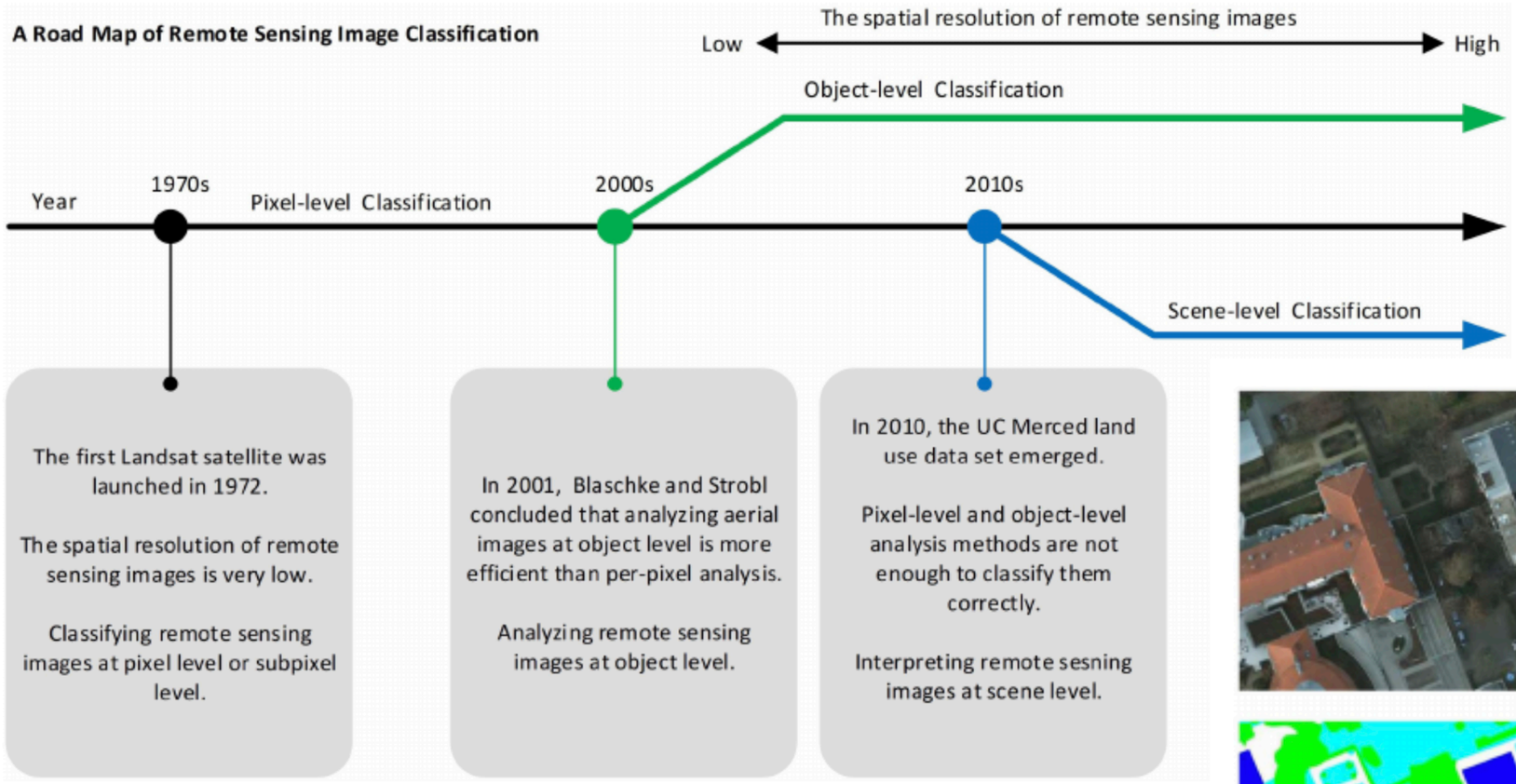
Use-cases

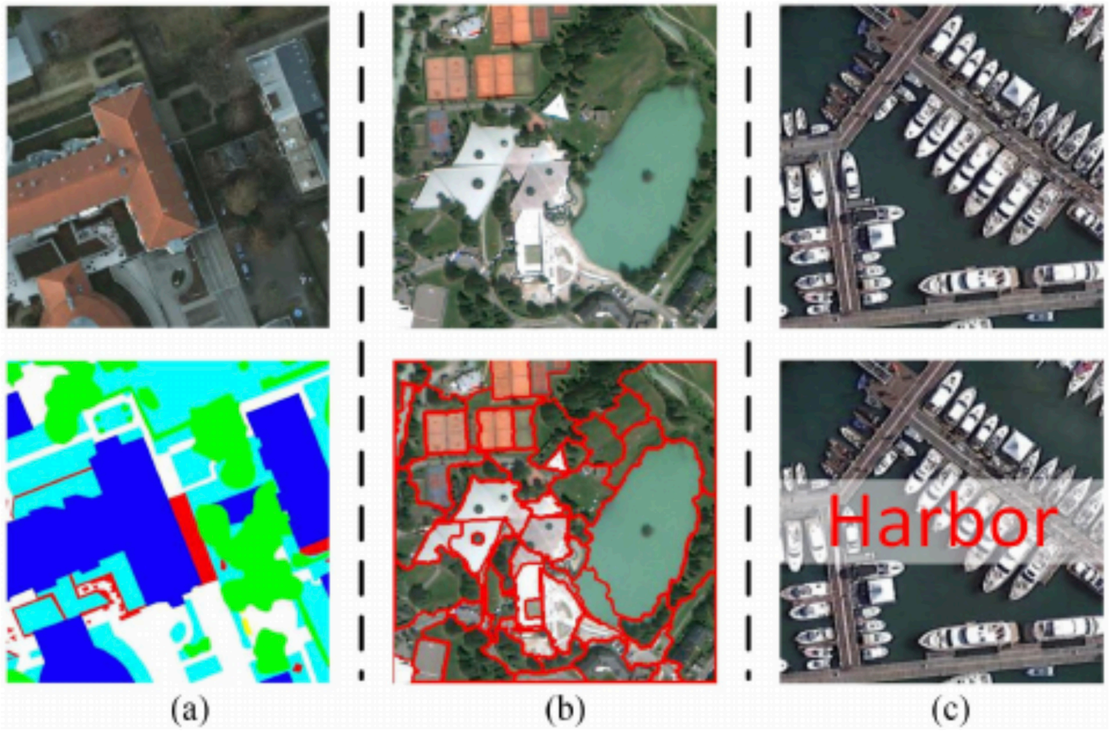
- Improving Agriculture
- Studying Urban Geography
- Managing Disaster Recovery
- Solar Panel Energy Output Optimization
- Monitoring drought and weather patterns
- Land use and land cover analysis
- Civil engineering study
- Saving Wildlife



Confidential

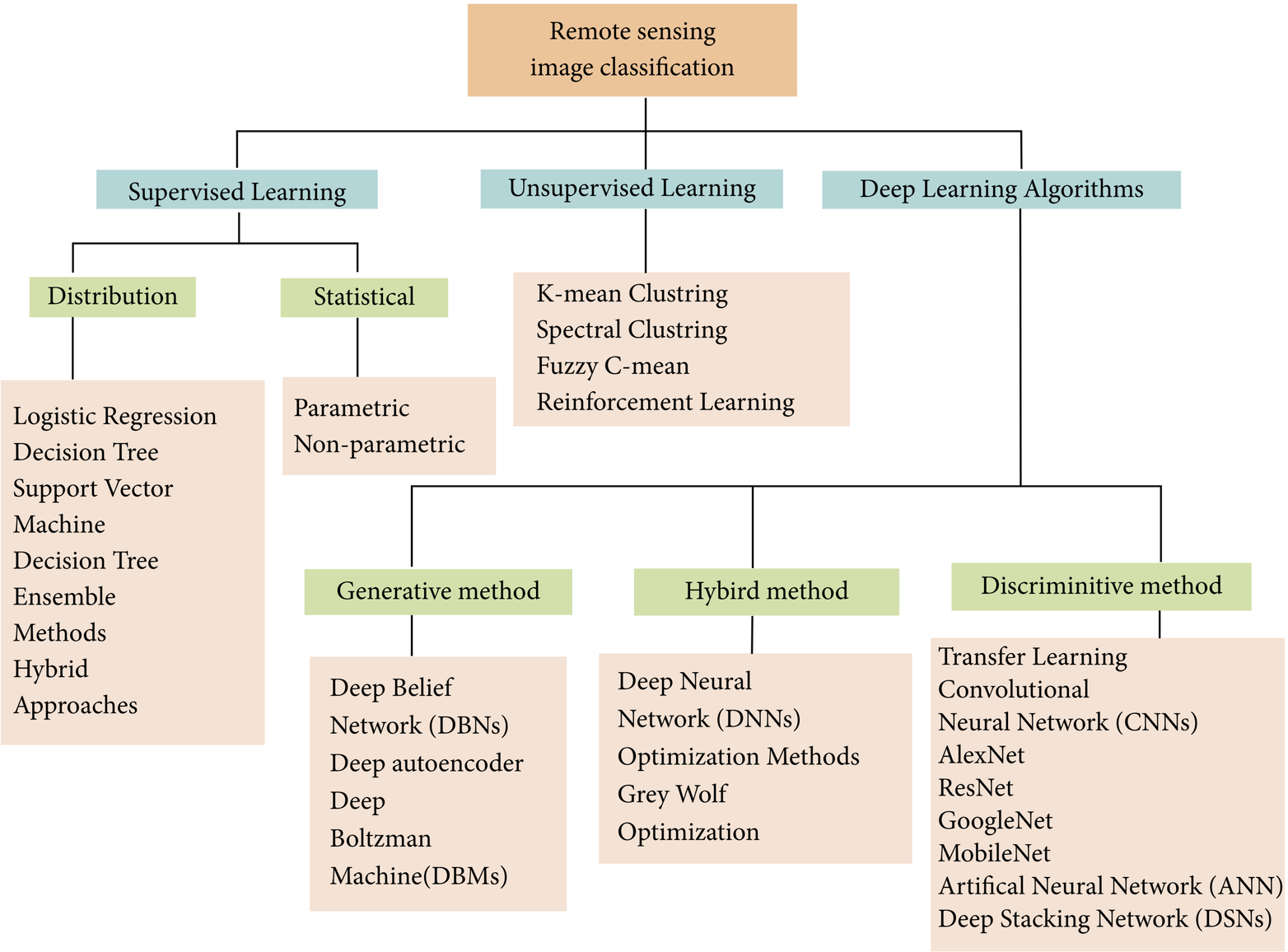
Remote sensing image classification (contd)





Confidential

Remote sensing image classification (contd)



Confidential

Dataset

13 PUBLICLY AVAILABLE DATA SETS FOR REMOTE SENSING IMAGE SCENE CLASSIFICATION

Data sets	Image number per class	Number of scene classes	Total image number	Image size	Training ratios		Data sources	Year
UC Merced [49]	100	21	2100	256 × 256	50%	80%	Aerial orthoimagery	2010
WHU-RS19 [124]	50~61	19	1005	600×600	40%	60%	Google Earth	2012
RSSCN7 [125]	400	7	2800	400×400	20%	50%	Google Earth	2015
Brazilian Coffee Scene [85]	1438	2	2876	64×64	50%		SPOT sensor	2015
SAT-4/-6 [126]	125000/67500	4/6	500000/405000	28×28	80%		National Agriculture Imagery Program	2015
SIRI-WHU [127]	200	12	2400	200×200	50%		Google Earth	2016
RSC11 [128]	about 100	11	1232	512×512	50%		Google Earth	2016
AID [79]	220~420	30	10000	600×600	20%	50%	Google Earth	2017
NWPU-RESISC45 [80]	700	45	31500	256×256	10%	20%	Google Earth	2017
RSI-CB128/-CB256 [129]	about 800/690	45/35	36000/24000	128×128/256×256	50%	80%	Google Earth & Bing Maps	2017
OPTIMAL-31 [74]	60	31	1860	256×256	80%		Google Earth	2018
EuroSAT [130]	2000~3000	10	27000	64×64	80%		Sentinel-2	2019
BigEarthNet [131]	328~217119	44	590326	120×120	60%		Sentinel-2	2019

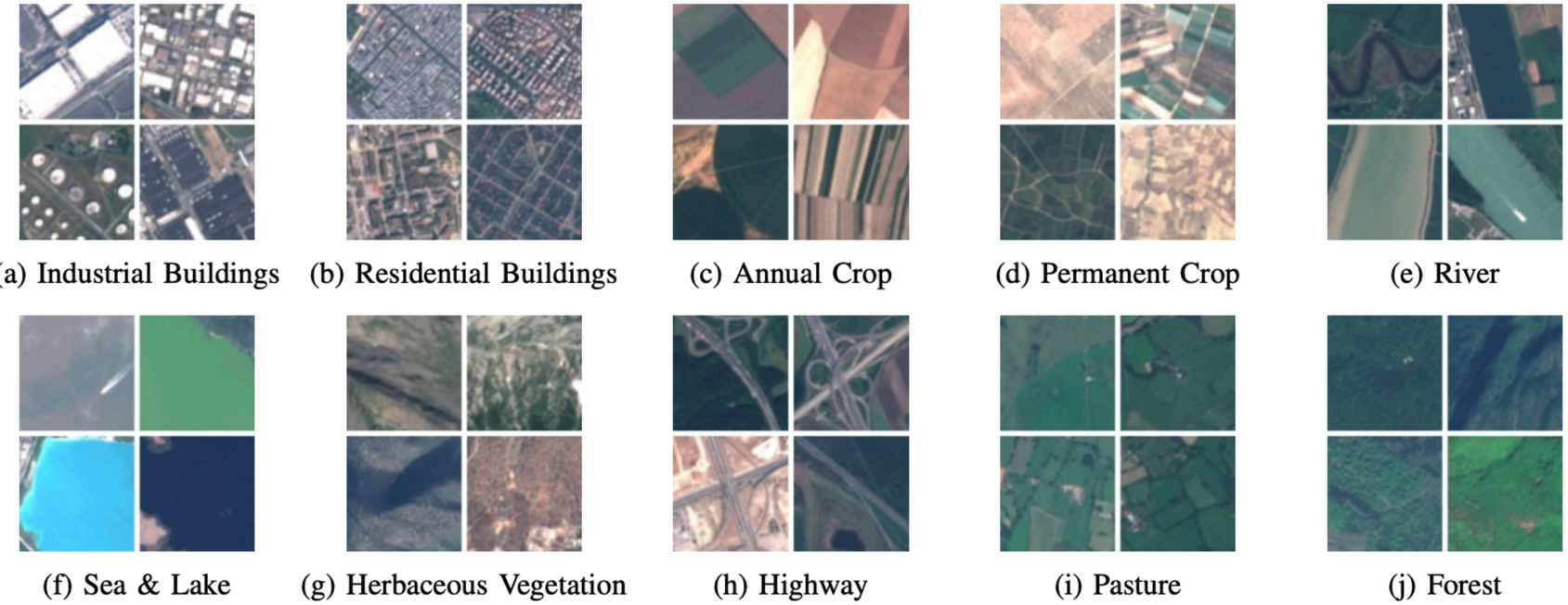
Ref. [4]

Confidential

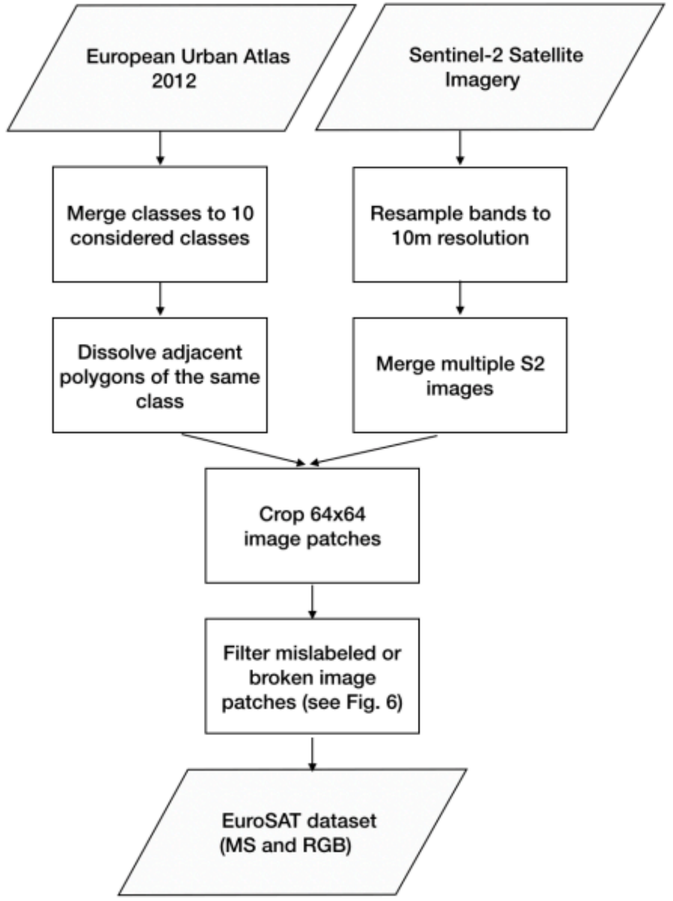
Dataset (contd)

Sentinel-2A is one satellite in the two-satellite constellation of the identical land monitoring satellites Sentinel-2A and Sentinel-2B.
Source of the dataset is mutli-spectral image data provided by the Sentinel-2A satellite.
The Sentinel-2 satellite constellation provides about 1.6 TB of compressed images per day

Based on the obtained satellite images, a dataset of 27,000 georeferenced and labeled image patches is created.
The image patches measure 64x64 pixels and have been manually checked.



Ref. [4]



EuroSAT dataset creation process.

Confidential

Dataset (contd)

TABLE II: Classification accuracy (%) of different training-test splits on the EuroSAT dataset.

Method	10/90	20/80	30/70	40/60	50/50	60/40	70/30	80/20	90/10
BoVW (SVM, SIFT, k = 10)	54.54	56.13	56.77	57.06	57.22	57.47	57.71	58.55	58.44
BoVW (SVM, SIFT, k = 100)	63.07	64.80	65.50	66.16	66.25	66.34	66.50	67.22	66.18
BoVW (SVM, SIFT, k = 500)	65.62	67.26	68.01	68.52	68.61	68.74	69.07	70.05	69.54
CNN (two layers)	75.88	79.84	81.29	83.04	84.48	85.77	87.24	87.96	88.66
ResNet-50	75.06	88.53	93.75	94.01	94.45	95.26	95.32	96.43	96.37
GoogleNet	77.37	90.97	90.57	91.62	94.96	95.54	95.70	96.02	96.17

TABLE III: Benchmarked classification accuracy (%) of the two best performing classifiers GoogLeNet and ResNet-50 with a 80/20 training-test split. Both CNNs have been pre-trained on the image classification dataset ILSVRC-2012 [21].

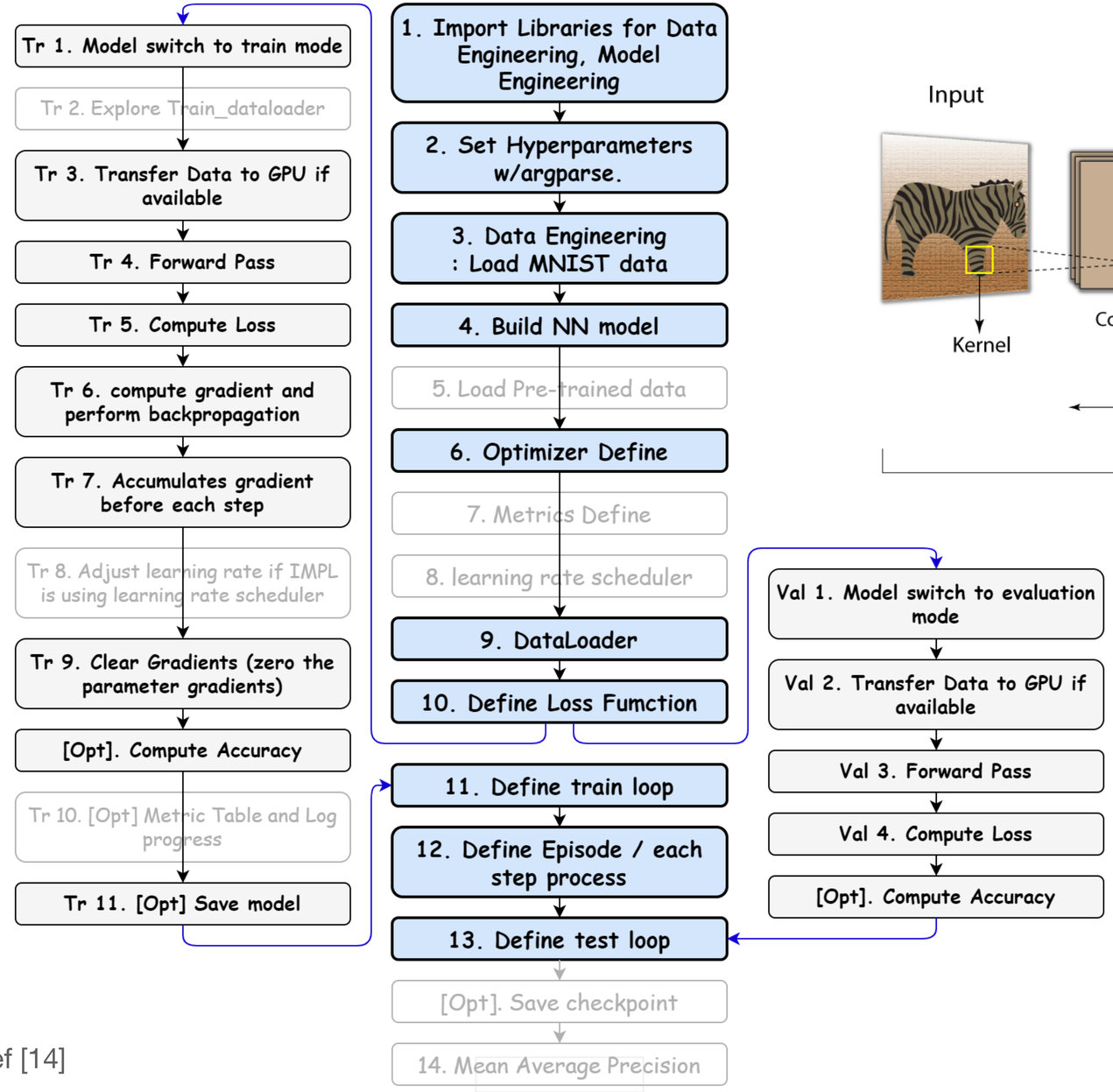
Method	UCM	AID	SAT-6	BCS	EuroSAT
ResNet-50	96.42	94.38	99.56	93.57	98.57
GoogLeNet	97.32	93.99	98.29	92.70	98.18

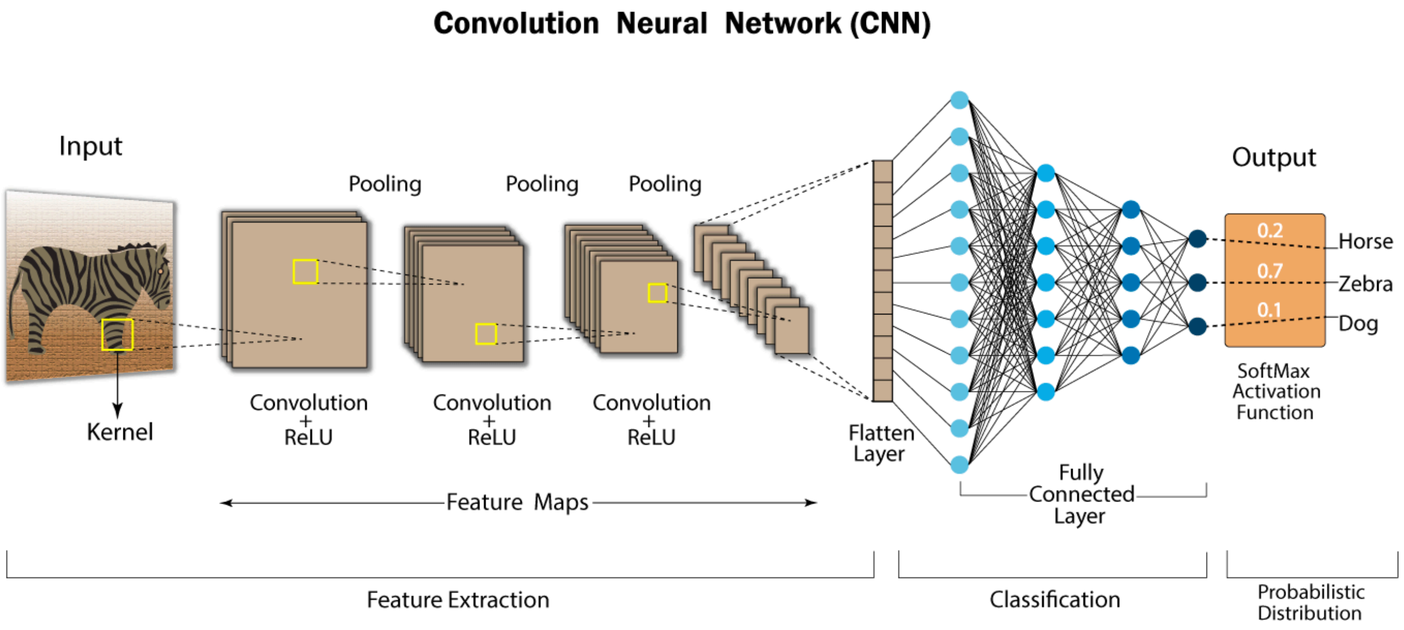
Ref. [4]

Confidential

Sample NN model training loop

Refresher



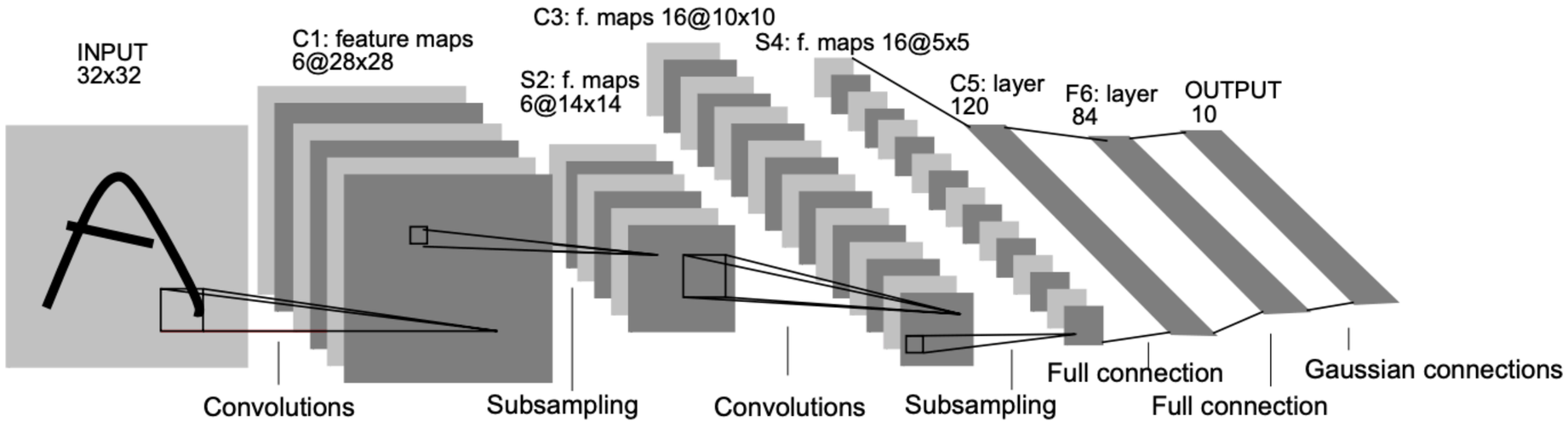


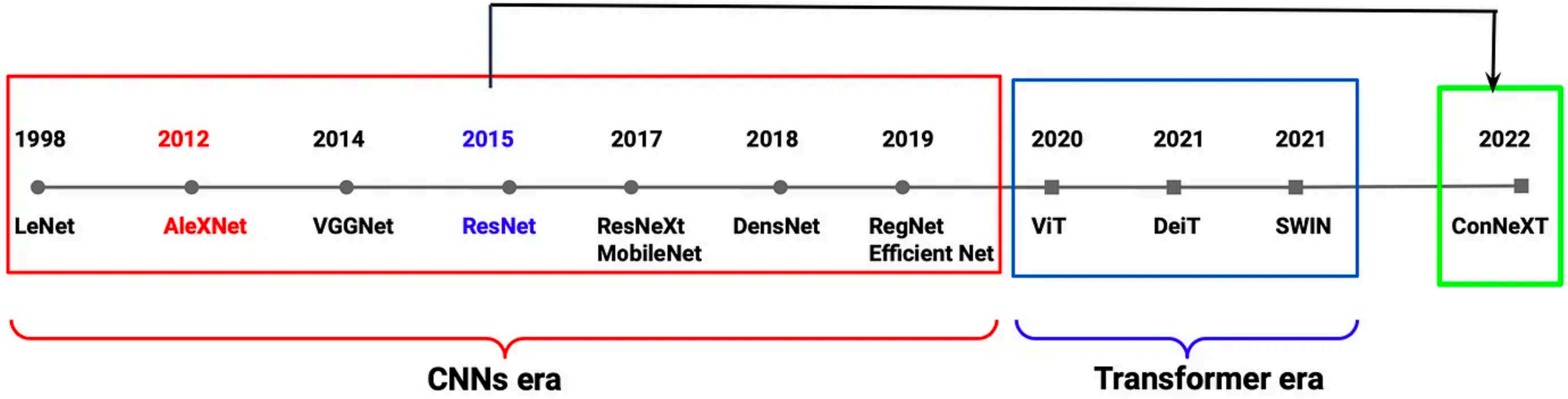
Ref [14]

Confidential

Classical CNN

Architecture of Lenet-5





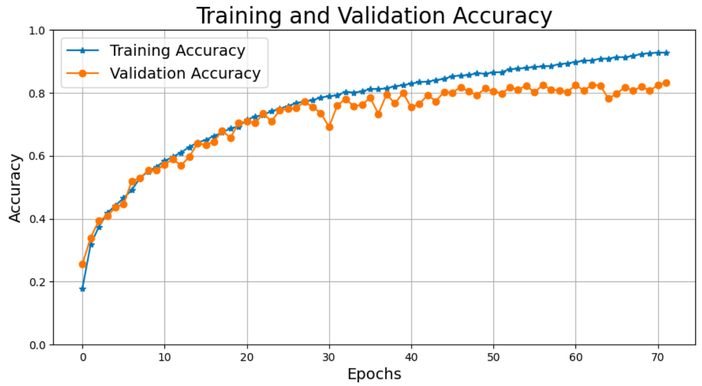
Ref [5]

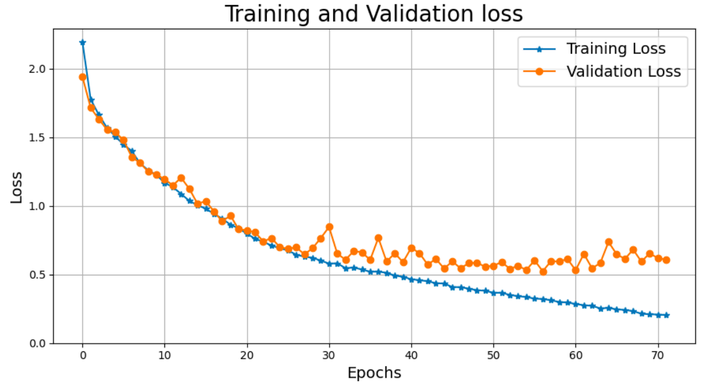
Confidential

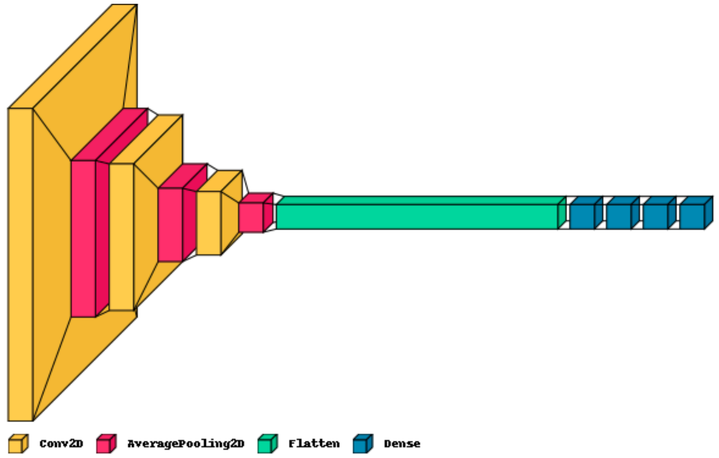
Classical CNN

EuroSAT - Modified LeNet-5 (27k images; multi-class (10) classification)

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 64, 64, 16)        448

 average_pooling2d_3 (Averag  (None, 32, 32, 16)       0
 ePooling2D)

 conv2d_4 (Conv2D)           (None, 30, 30, 32)        4640

 average_pooling2d_4 (Averag  (None, 15, 15, 32)       0
 ePooling2D)

 conv2d_5 (Conv2D)           (None, 13, 13, 64)        18496

 average_pooling2d_5 (Averag  (None, 6, 6, 64)         0
 ePooling2D)

 flatten_1 (Flatten)         (None, 2304)              0

 dense_4 (Dense)             (None, 128)               295040

 dense_5 (Dense)             (None, 64)                8256

 dense_6 (Dense)             (None, 32)                2080

 dense_7 (Dense)             (None, 10)                330

=================================================================
Total params: 329,290
Trainable params: 329,290
Non-trainable params: 0
_________________________________________________________________
```



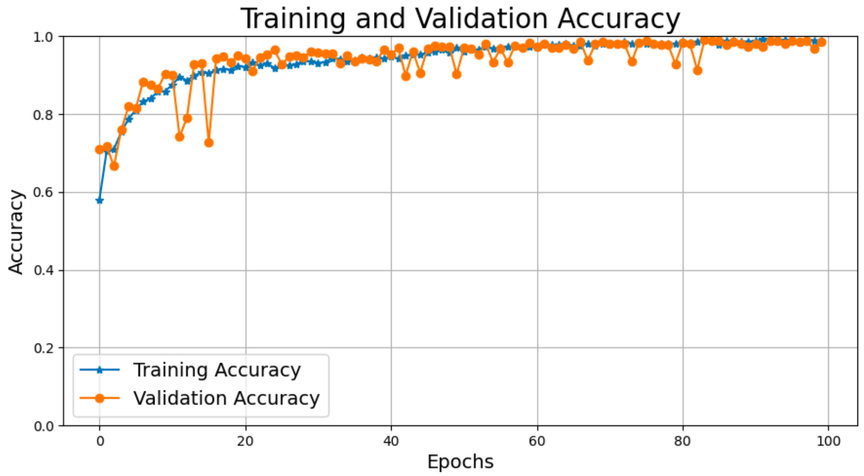


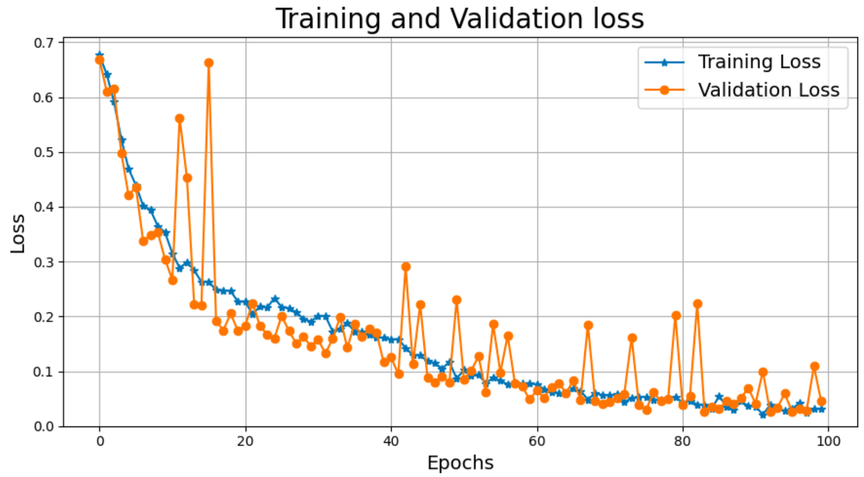


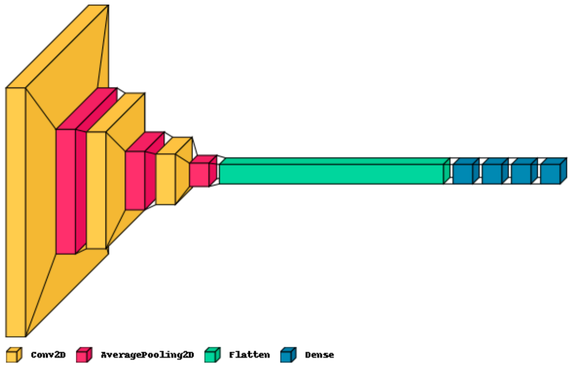
Classical CNN

EuroSAT - Modified LeNet-5 (5.5k images; binary classification)

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 64, 64, 16)        448

 average_pooling2d_3 (Averag  (None, 32, 32, 16)       0
 ePooling2D)

 conv2d_4 (Conv2D)           (None, 30, 30, 32)        4640

 average_pooling2d_4 (Averag  (None, 15, 15, 32)       0
 ePooling2D)

 conv2d_5 (Conv2D)           (None, 13, 13, 64)        18496

 average_pooling2d_5 (Averag  (None, 6, 6, 64)         0
 ePooling2D)

 flatten_1 (Flatten)         (None, 2304)              0

 dense_4 (Dense)             (None, 128)               295040

 dense_5 (Dense)             (None, 64)                8256

 dense_6 (Dense)             (None, 32)                2080

 dense_7 (Dense)             (None, 2)                 66

=================================================================
Total params: 329,026
Trainable params: 329,026
Non-trainable params: 0
_________________________________________________________________
```







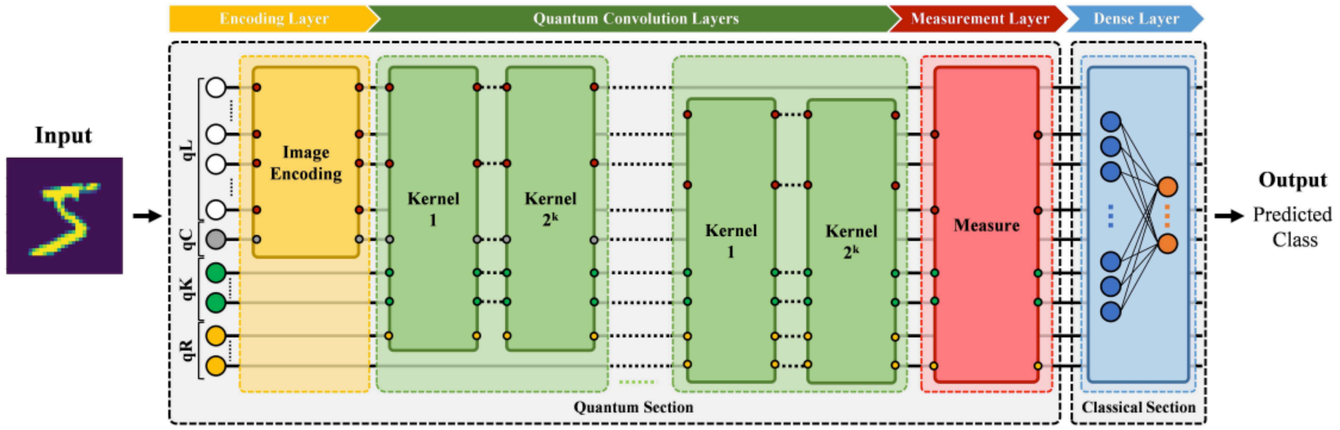
Eurosat_binary_classification_classical.ipynb

Confidential

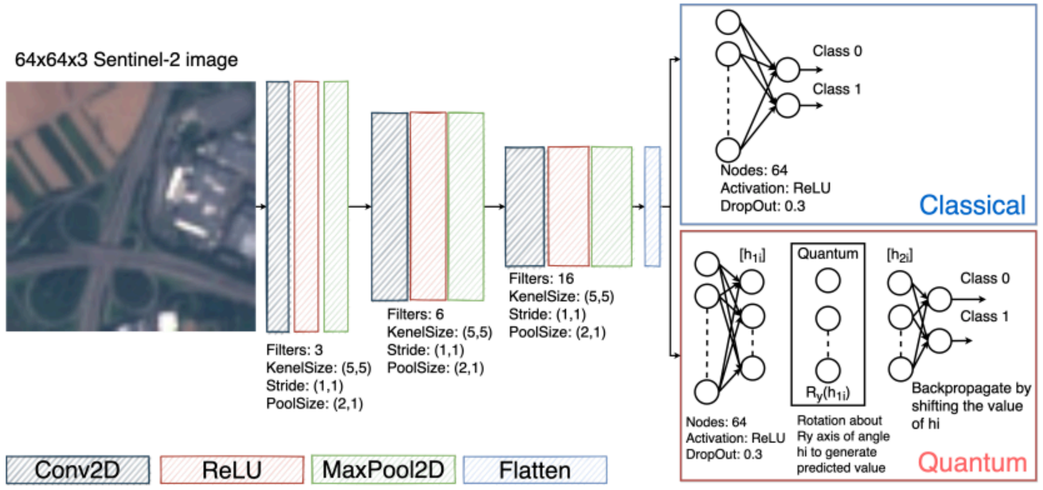
Quantum hybrid CNN

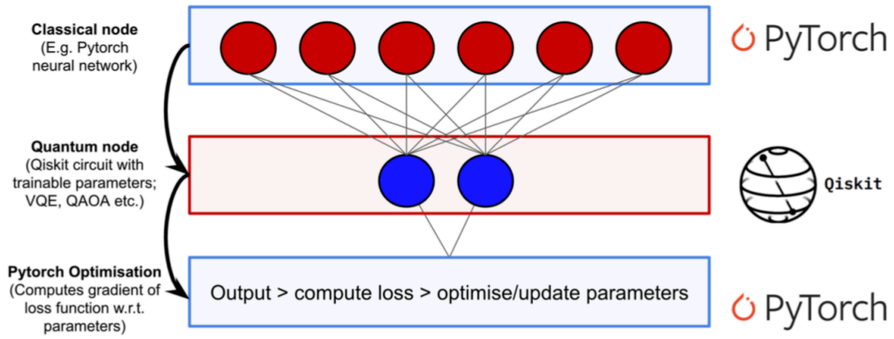
Some approaches

Accelerate feature extraction compared with its classical counterpart and handle multicategory classification tasks with reduced quantum resources



Introduce a hidden quantum layer into the quantum neural network to use the quantum circuit for refining the model.





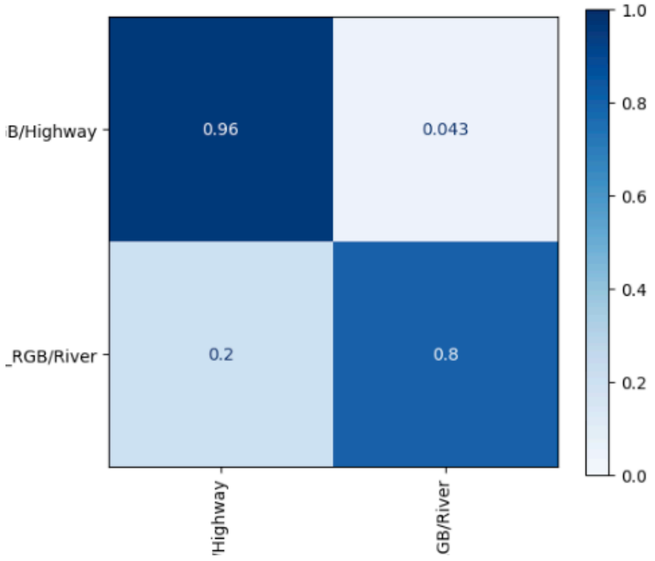
Confidential

Hybrid binary classifier

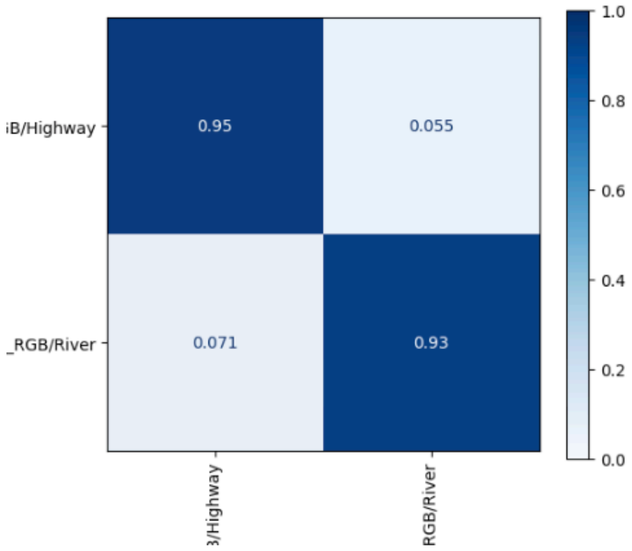
Eurosat-hybrid-binary-classifier.ipynb

L (classical); R (hybrid)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 62, 62]             448
            Conv2d-2           [-1, 32, 29, 29]           4,640
            Conv2d-3           [-1, 64, 12, 12]          18,496
            Linear-4                  [-1, 120]         276,600
            Linear-5                   [-1, 84]          10,164
            Linear-6                   [-1, 42]           3,570
            Linear-7                   [-1, 16]             688
            Linear-8                    [-1, 2]              34
================================================================
Total params: 314,640
Trainable params: 314,640
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.75
Params size (MB): 1.20
Estimated Total Size (MB): 1.99
----------------------------------------------------------------
```




```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 62, 62]             448
            Conv2d-2           [-1, 32, 29, 29]           4,640
            Conv2d-3           [-1, 64, 12, 12]          18,496
            Linear-4                  [-1, 120]         276,600
            Linear-5                   [-1, 84]          10,164
            Linear-6                   [-1, 42]           3,570
            Linear-7                    [-1, 4]             172
            Linear-8                    [-1, 2]              34
================================================================
Total params: 314,124
Trainable params: 314,124
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.75
Params size (MB): 1.20
Estimated Total Size (MB): 1.99
----------------------------------------------------------------
```



Confidential

Hybrid binary classifier

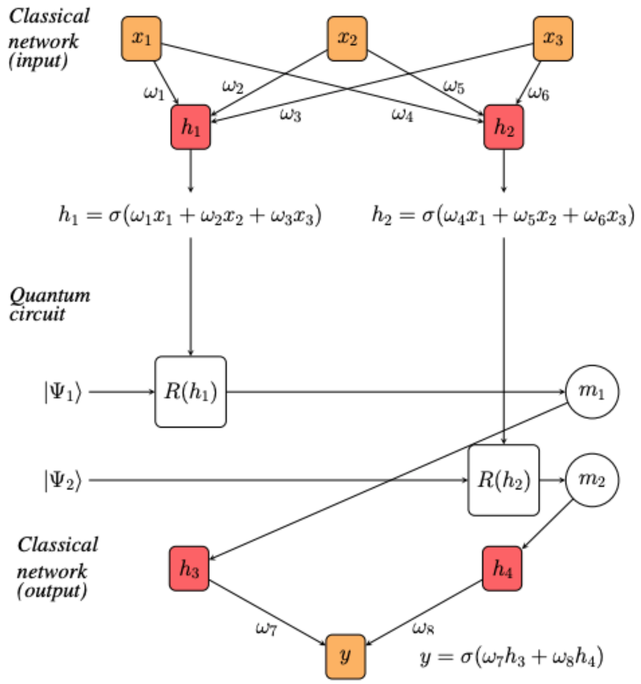
L (classical); R (hybrid)

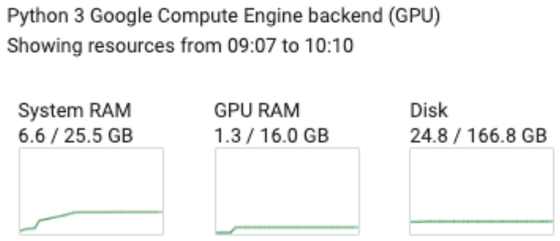
```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3)

        self.fc1 = nn.Linear(2304, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 42)
        self.fc4 = nn.Linear(42, 16)
        self.fc5 = nn.Linear(16, 2)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = F.relu(F.max_pool2d(self.conv3(x), 2))

        x = x.view(-1, 2304)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = F.relu(self.fc2(x))

        x = F.relu(self.fc3(x))

        x = self.fc4(x)
        x = np.pi*torch.tanh(x)
        x = self.fc5(x.float())
        x = F.softmax(x, 1)

        return x


    def predict(self, x):
        pred = self.forward(x)
        ans = torch.argmax(pred[0]).item()
        return torch.tensor(ans)

network = Net().to(device)
optimizer = optim.Adam(network.parameters(), lr=0.0002)
```

```python
class Net(nn.Module).
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3)
        self.fc1 = nn.Linear(2304, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 42)
        self.fc4 = nn.Linear(42, 4)
        self.qc = QuantumLayer.apply
        self.fc5 = nn.Linear(16, 2)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = F.relu(F.max_pool2d(self.conv3(x), 2))

        x = x.view(-1, 2304)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = F.relu(self.fc2(x))

        x = F.relu(self.fc3(x))

        x = self.fc4(x)
        x = np.pi*torch.tanh(x)

        x = self.qc(x[0]) # QUANTUM LAYER
        x = F.relu(x)

        x = self.fc5(x.float())
        x = F.softmax(x, 1)
        return x


    def predict(self, x):
        pred = self.forward(x)
        ans = torch.argmax(pred[0]).item()
        return torch.tensor(ans)

hybrid_network = Net().to(device)
optimizer = optim.Adam(hybrid_network.parameters(), lr=0.0002)
```

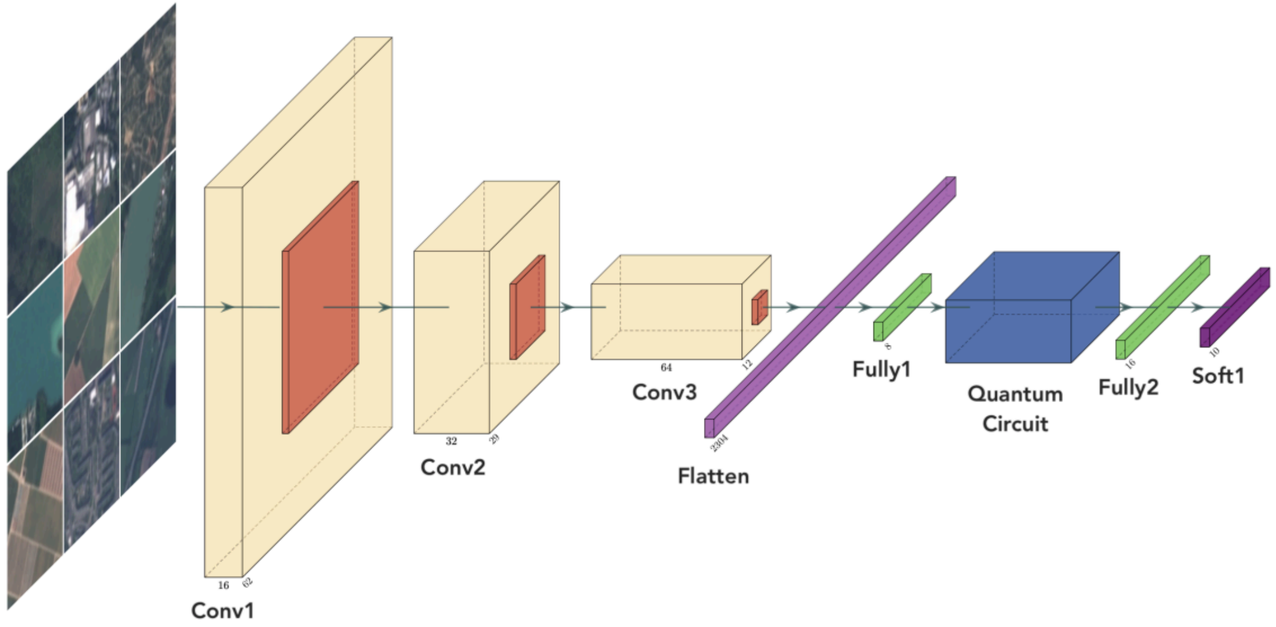
Hybrid CNN

Hidden layer





- A parametrised quantum circuit is typically used as a hidden layer for the neural network.
- Expressibility of the circuit plays a role in efficiency.
- Higher dimensional quantum representation of classical data in the creation of the hybrid model
- Inputs for the circuit are classically derived in the layer before, in the diagram, h1, h2
- Embed quantum circuit into gradient calculation for optimisation.



Ref [1]

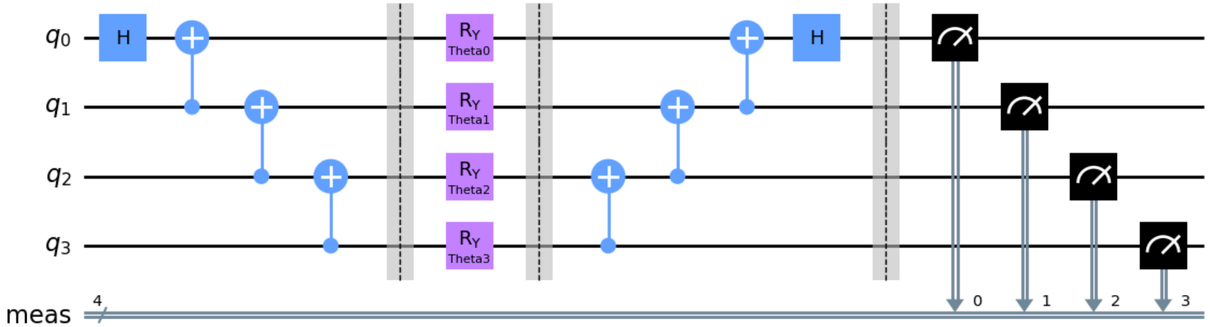
Confidential

Hybrid CNN

Quantum circuits

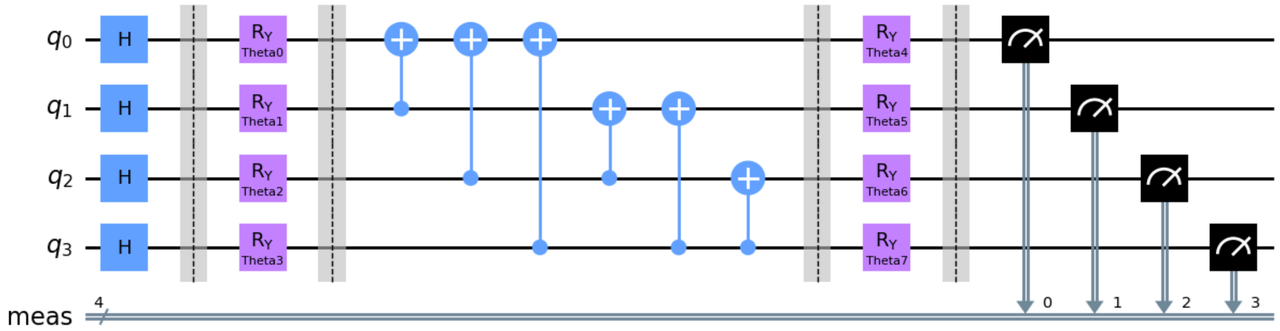
Bellman

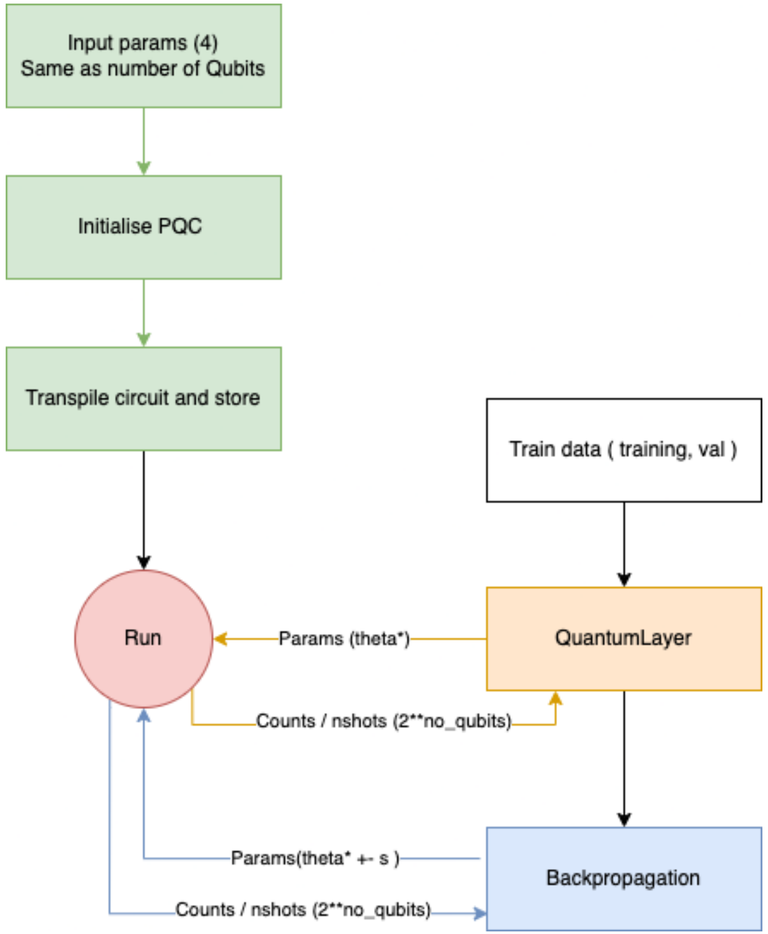
Input parameters : 4
Output states : 16



Real-amplitudes

Input parameters : 8
Output states : 16



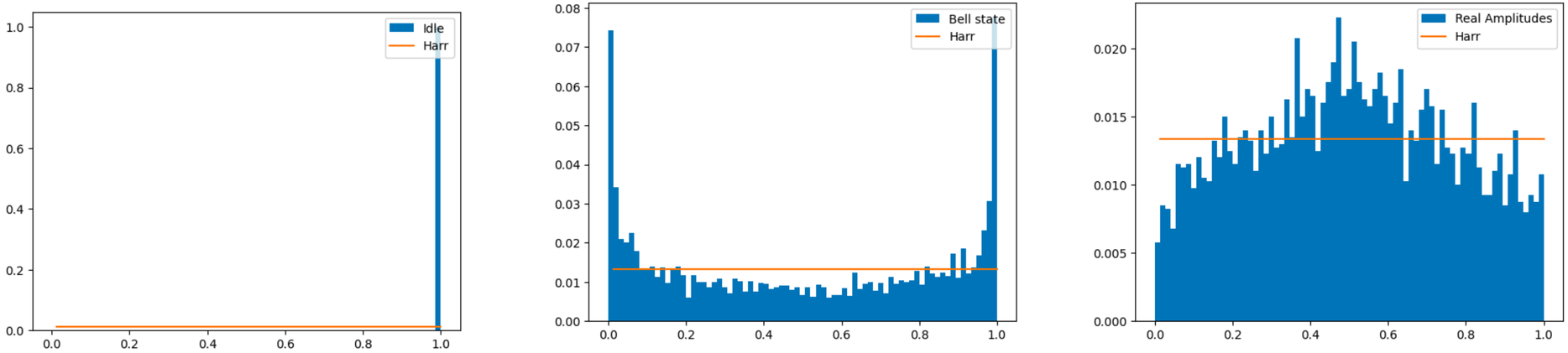


Confidential

Hybrid CNN

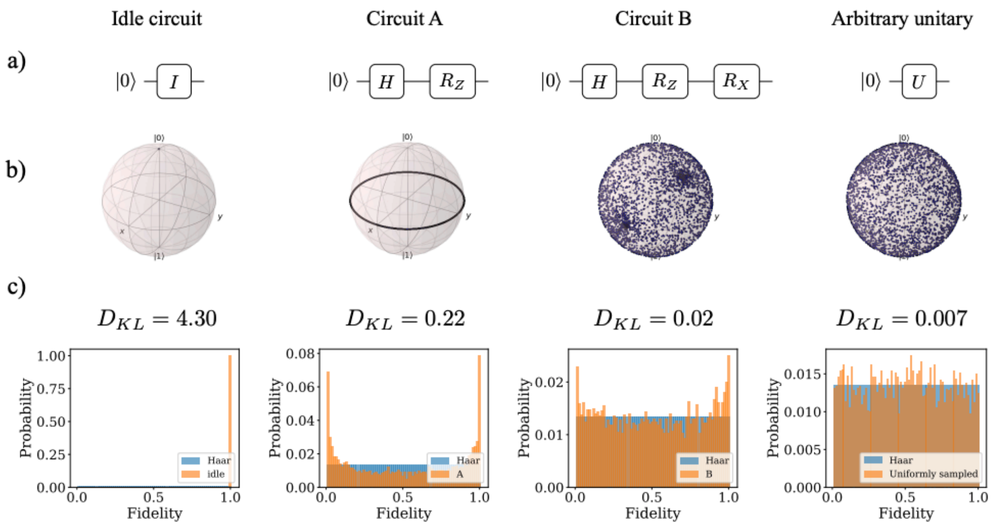
Expressibility

Lower Kullback-Leibler (KL) divergence value corresponds to more favourable expressibility or a greater closeness to Haar random states.



KL 4.317 KL 0.215 KL 0.033

Low expressibility ⟷ High expressibility



Expressibility.ipynb

Confidential

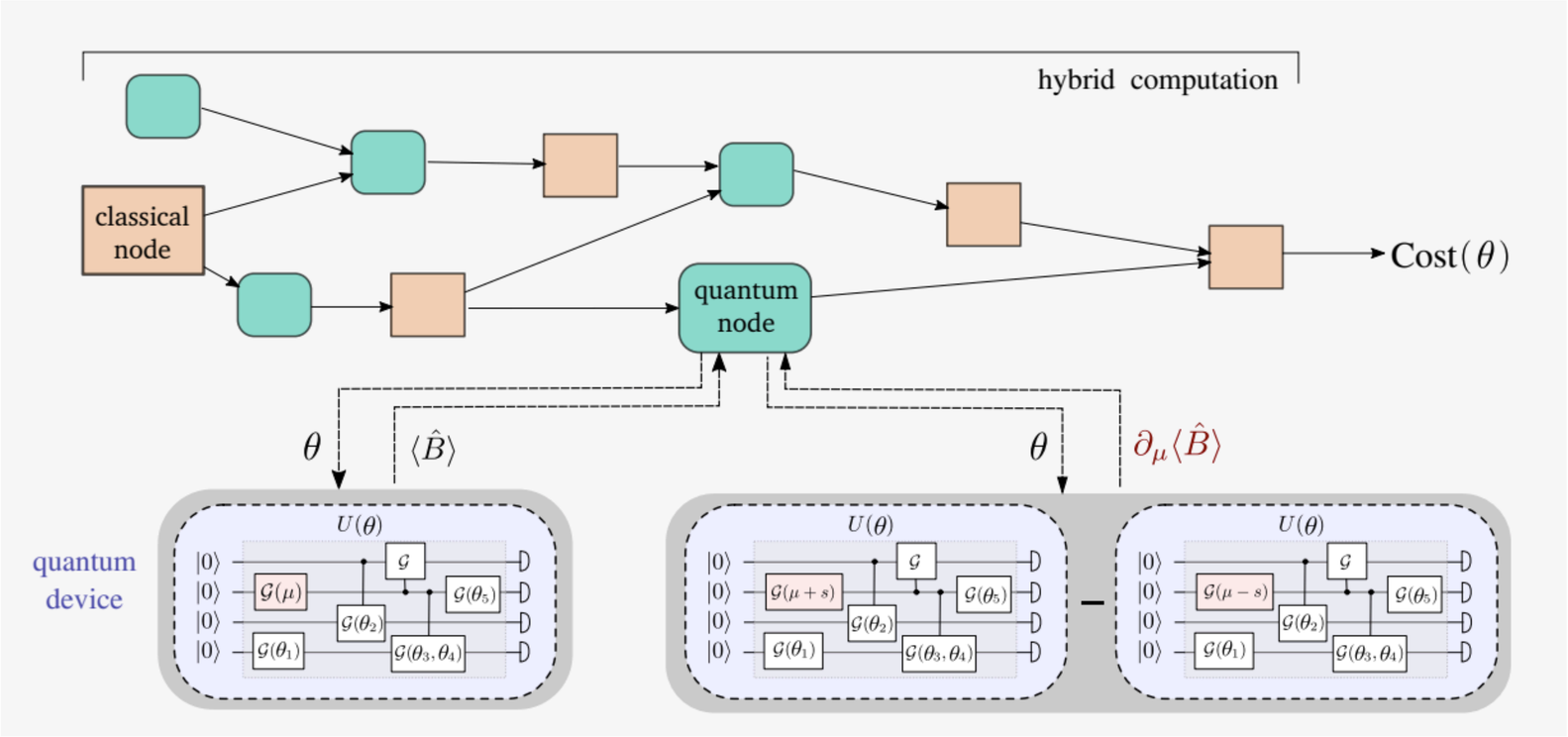
Hybrid CNN

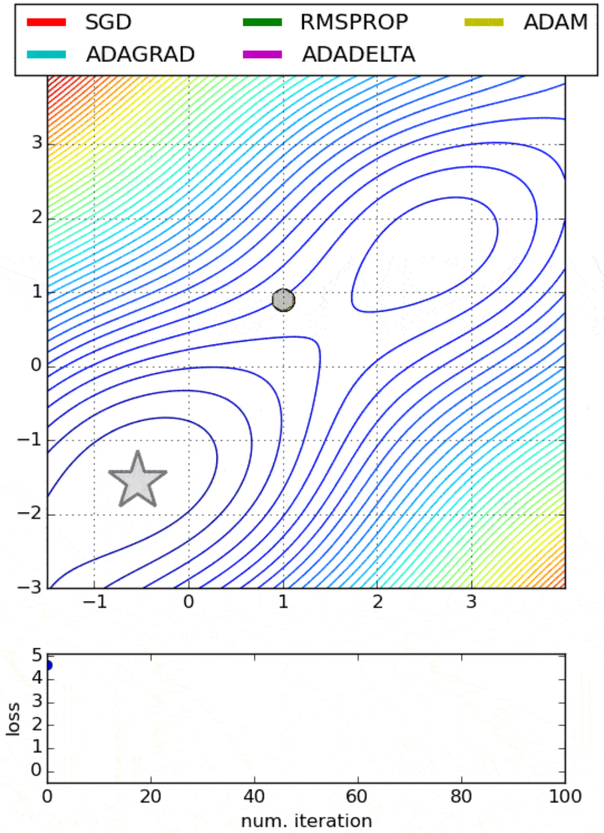
Optimization, Parameter shift rule

Derivative of the function $f(x) = sin(x)$ is identical to $f(x) = \frac{1}{2}sin(x + \frac{\pi}{2}) - \frac{1}{2}sin(x - \frac{\pi}{2})$

The same circuit can be used to compute both the quantum function and the gradient of the quantum function

$$\nabla_\theta \text{Quantum Circuit}(\theta) = \text{Quantum Circuit}(\theta + s) - \text{Quantum Circuit}(\theta - s)$$





Chapter notes 16; S 4.1
Ref [11]

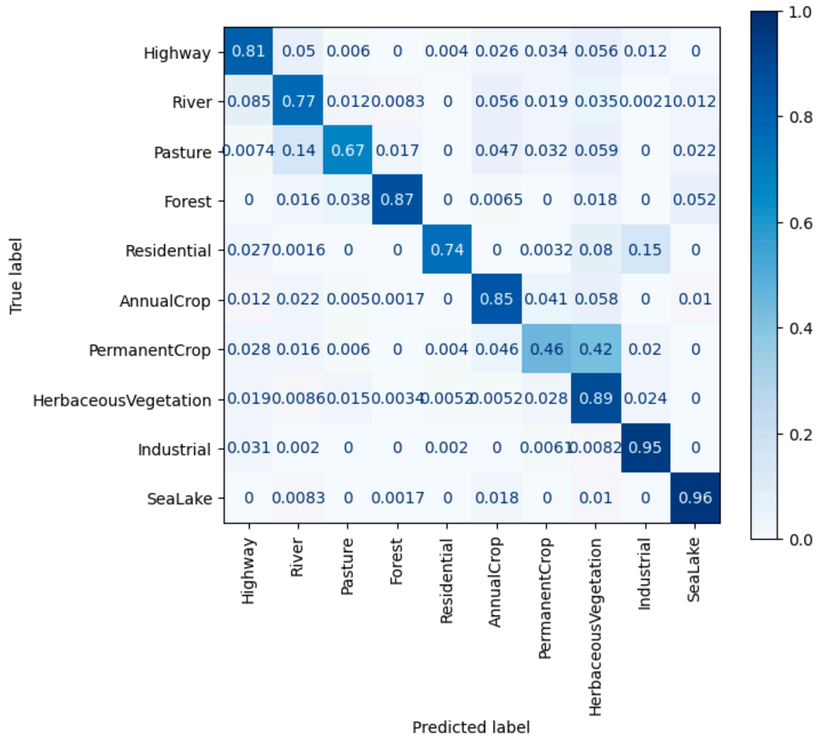
Confidential

Hybrid CNN

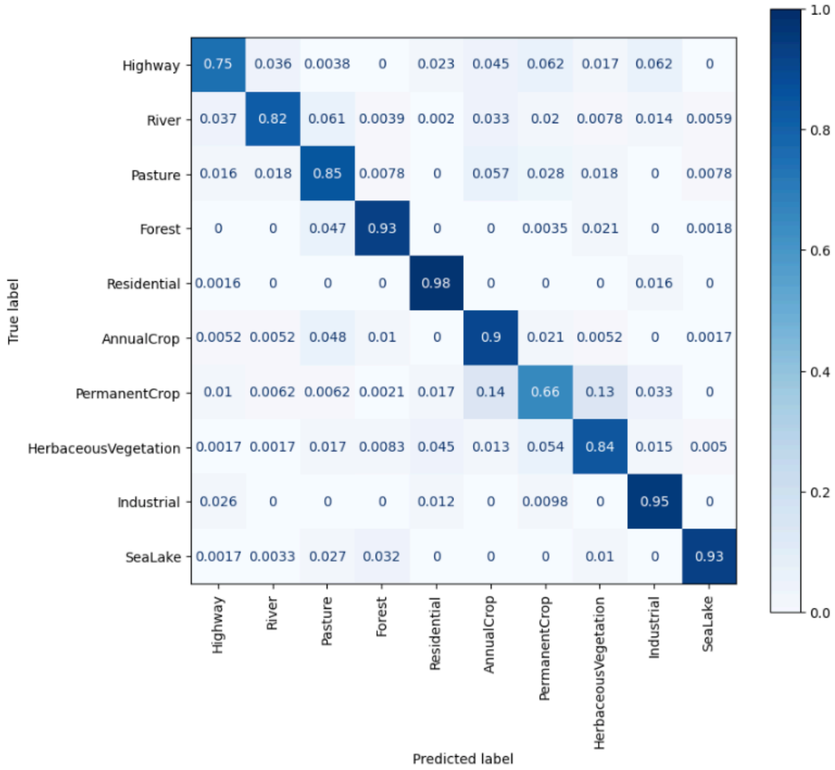
Bell state - Classifier confusion matrix

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3)

        self.fc1 = nn.Linear(2304, 4)
        self.qc = QuantumLayer.apply
        self.fc2 = nn.Linear(16, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = F.relu(F.max_pool2d(self.conv3(x), 2))

        x = x.view(-1, 2304)
        x = self.fc1(x)
        x = np.pi*torch.tanh(x)

        x = self.qc(x[0]) # QUANTUM LAYER
        x = F.relu(x)
        x = self.fc2(x.float())
        x = F.softmax(x, 1)
        return x

    def predict(self, x):
        pred = self.forward(x)
        ans = torch.argmax(pred[0]).item()
        return torch.tensor(ans)

hybrid_network = Net().to(device)
optimizer = optim.Adam(hybrid_network.parameters(), lr=0.0002)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 62, 62]             448
            Conv2d-2           [-1, 32, 29, 29]           4,640
            Conv2d-3           [-1, 64, 12, 12]          18,496
            Linear-4                   [-1, 4]           9,220
            Linear-5                  [-1, 10]             170
================================================================
Total params: 32,974
Trainable params: 32,974
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.74
Params size (MB): 0.13
Estimated Total Size (MB): 0.92
----------------------------------------------------------------
```



Classical



Bellman

Eurosat_hybrid_multi_class_classifier_bell.ipynb

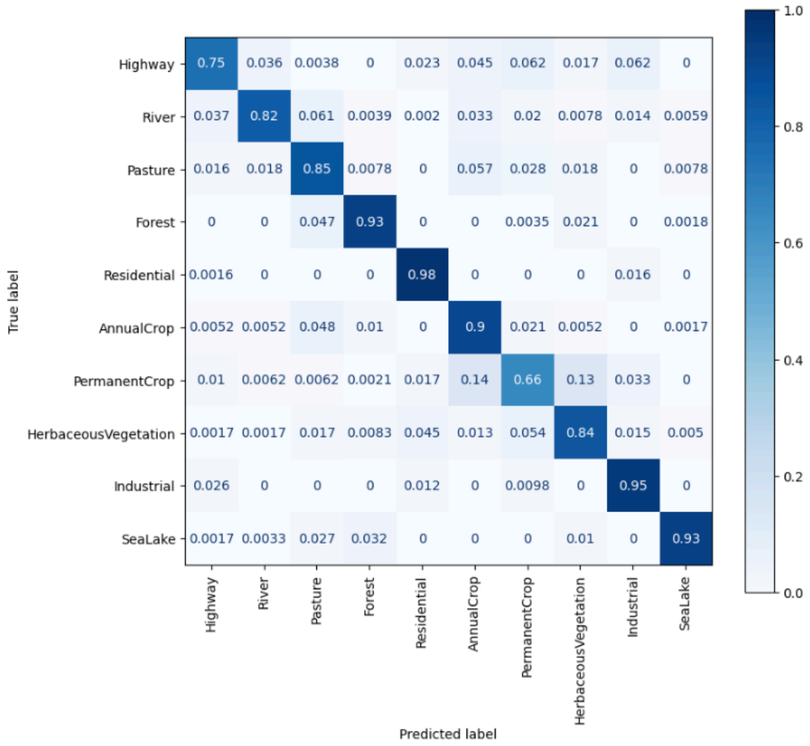
Confidential

Hybrid CNN

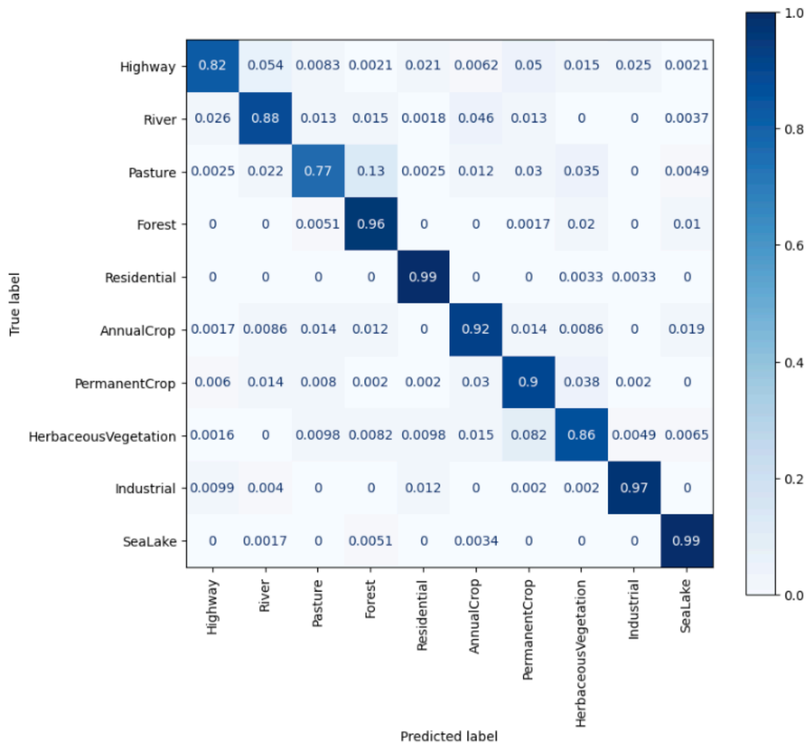
Real Amplitude - Classifier confusion matrix

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3)

        self.fc1 = nn.Linear(2304, 8)
        self.qc = QuantumLayer.apply
        self.fc2 = nn.Linear(16, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = F.relu(F.max_pool2d(self.conv3(x), 2))

        x = x.view(-1, 2304)
        x = self.fc1(x)
        x = np.pi*torch.tanh(x)

        ## Quantum circuit
        x = self.qc(x[0])
        x = F.relu(x)
        ##
        x = self.fc2(x.float())
        x = F.softmax(x, 1)
        return x


    def predict(self, x):
        pred = self.forward(x)
        ans = torch.argmax(pred[0]).item()
        return torch.tensor(ans)

hybrid_network = Net().to(device)
optimizer = optim.Adam(hybrid_network.parameters(), lr=0.0002)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 62, 62]             448
            Conv2d-2           [-1, 32, 29, 29]           4,640
            Conv2d-3           [-1, 64, 12, 12]          18,496
            Linear-4                    [-1, 8]          18,440
            Linear-5                   [-1, 10]             170
================================================================
Total params: 42,194
Trainable params: 42,194
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.75
Params size (MB): 0.16
Estimated Total Size (MB): 0.95
----------------------------------------------------------------
```



Bellman



RealAmplitude

Submission

Item	Desciption
QML-Project.*	A presentation giving details of the appoach and various aspects
all_classes_classification_classical	Two jupyter notebook(s) demonstrating CNN in classical way for multiclass classification using tensorflow and pytorch
binary_classification_classical	A jupyter notebook demonstrating CNN in classical way for binary classication in tensorflow
binary-classification-hybrid	A jupyter notebook demonstrating binary classification with hybrid approach with a quantum layer using bellman circuit using pytorch
multi-classiciation	Multiple jupyter notebook(s) demonstrating multi class classication using hybid for bellman circuit and real amplitdes circuit

> Data hanlder classes are similar across all the projects and are inside the respective jupyter notebooks handing the required classes for loading, splitting data.

```
.
├── QML-project.*
├── README.md
├── all_classes_classification_classical
│   ├── CNN-all_classes.h5
│   └── Eurosat_multi_class_classification_classical.ipynb
├── binary-classification-hybrid
│   ├── Eurosat_hybrid_binary_classifier.ipynb
│   ├── Eurosat_hybrid_binary_classifier_backup.ipynb
│   └── model-hybrid-binary-classification.pt
├── binary_classification_classical
│   ├── CNN-binary_classes_classical.h5
│   └── Eurosat_binary_classification_classical.ipynb
└── multi-classification
    ├── Eurosat_classical_multi_class_classifier.ipynb
    ├── Eurosat_hybrid_multi_class_classifier_bell.ipynb
    ├── Eurosat_hybrid_multi_class_classifier_bell_contd.ipynb
    ├── Eurosat_hybrid_multi_class_classifier_real_amplitudes.ipynb
    ├── Expressibility.ipynb
    ├── model-multi-class-classical-lenet5-modif.pt
    ├── model-multi-class-hybrid-lenet5-modif-bell.pt
    └── model-multi-class-hybrid-lenet5-modif-real-amplitudes.pt
```

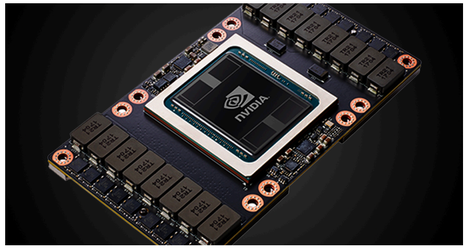
- Jupyter notebooks with output runs on colab google VMs
- Models output for archival as pt and h5 files
- Contains multi class and binary classification
- All files are executed on EuroSAT Rgb dataset
- ~50 hours of training on V100 GPU colab python backend VMs

Confidential

References

1. **Link On Circuit-based Hybrid Quantum Neural Networks for Remote Sensing Imagery Classification**
2. Link Remote Sensing Image Classification: A Comprehensive Review and Applications
3. Link A ConvNet for the 2020s
4. Link Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities
5. Link GradientBased Learning Applied to Document Recognition, (LeNet-5)
6. **Link EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification**
7. Link Advantages And Bottlenecks Of Quantum Machine Learning For Remote Sensing
8. Link Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition
9. Link Hybrid quantum-classical Neural Networks with PyTorch and Qiskit
10. Link Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms
11. Link Evaluating analytic gradients on quantum hardware
12. **Link ESA Philab GitHub repository**
13. **Link Dataset used for the project**
14. Link Deep Learning Bible - 4. Object Detection - Eng.

colab Qiskit python™ PyTorch TensorFlow



Confidential