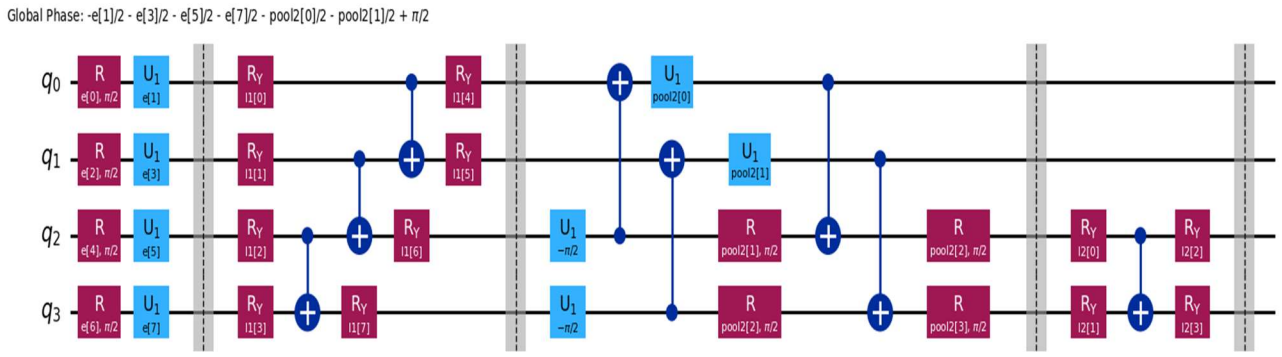
QUANTUM TRIPLET LOSS MODEL + KMEANS

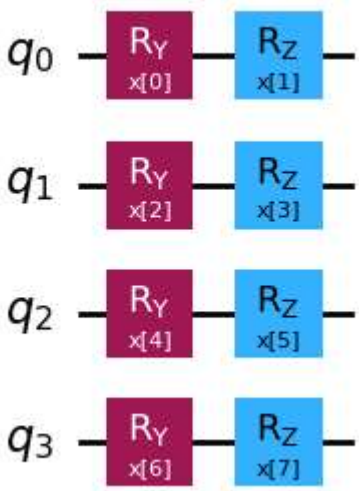
- **Descrizione del modello:**
    - Modello di generazione di embedding basato su CNN classiche e quantum CNN. VIene utilizzata una rete siamese per la generazione di embedding basata sulla tripla (anchor, positive, negative). La rete siamese è composta da una Classical CNN per l'estrazione delle feature, una rete neurale classica per la riduzione della dimensionalità a 8 feature e una rete quantum, con la stessa architettura del modello QuantumRBF, su cui viene invece effettuata la misurazione su tutti i qubit tramite sampling, ottenendo una distribuzione di probabilità su tutte le possibili bitstring (16 valori) successivamente un ulteriore layer lineare 16x16 effettua un rescaling appreso delle feature.
- **Summary Modello Ibrido (CNN+layer quantistico):**

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 24, 24]             832
              ReLU-2          [-1, 32, 24, 24]               0
            Conv2d-3          [-1, 32, 20, 20]          25,632
         MaxPool2d-4          [-1, 32, 10, 10]               0
              ReLU-5          [-1, 32, 10, 10]               0
            Conv2d-6            [-1, 64, 6, 6]          51,264
         MaxPool2d-7            [-1, 64, 3, 3]               0
              ReLU-8            [-1, 64, 3, 3]               0
           Flatten-9                 [-1, 576]               0
           Linear-10                 [-1, 120]          69,240
             ReLU-11                 [-1, 120]               0
           Linear-12                   [-1, 8]             968
  TorchConnector-13                  [-1, 16]              16
           Linear-14                  [-1, 16]             272
================================================================
Total params: 148,224
Trainable params: 148,224
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.46
Params size (MB): 0.57
Estimated Total Size (MB): 1.03
----------------------------------------------------------------
```

- **Circuito layer quantistico:**
    - **Circuito:**



- **Encoding con “yz_angles_encoding”:**

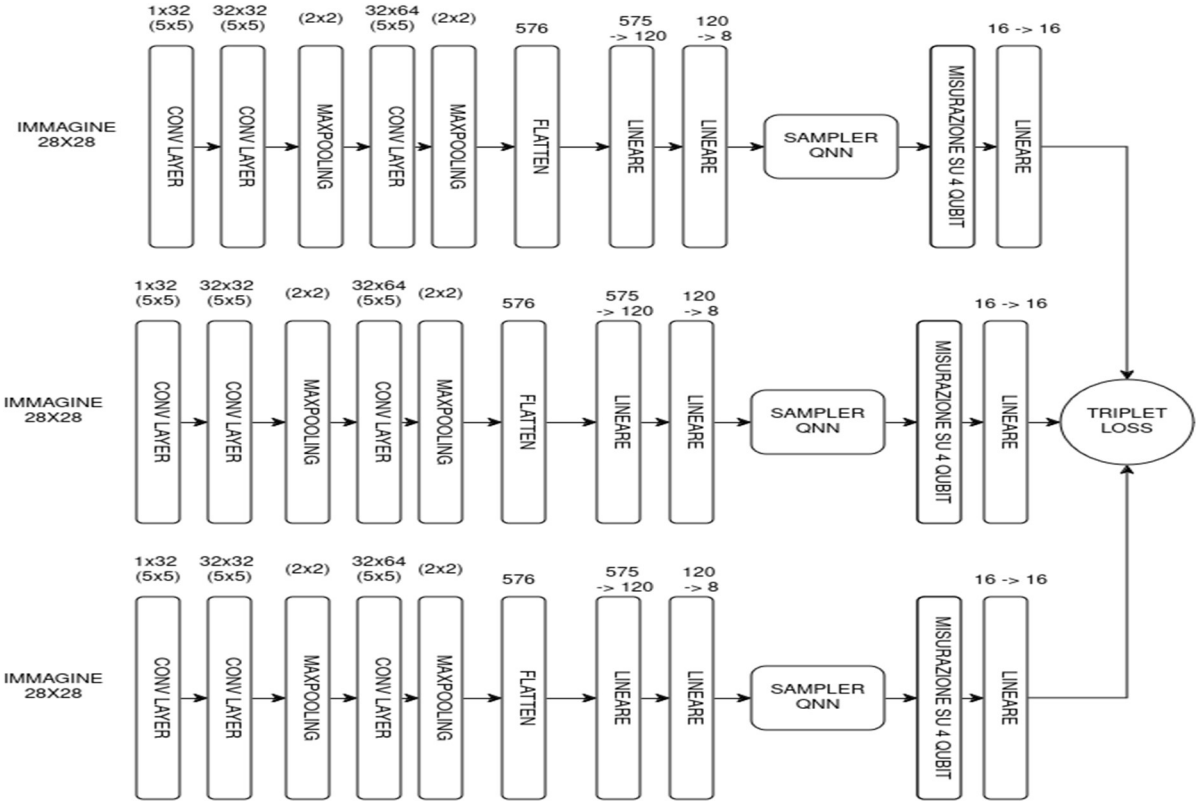


- **Codice circuito:**

```
encoding = yz_angles_encoding(8, param_name="e")
pooling = pooling_layer(4, "pool1")
ansatz = QuantumCircuit(4)
ansatz.barrier()
ansatz = ansatz.compose(RealAmplitudes(num_qubits=4, reps=1, name="Layer1",
parameter_prefix="l1"))
ansatz.barrier()
ansatz = ansatz.compose(pooling_layer(4, "pool2"))
ansatz.barrier()
ansatz = ansatz.compose(RealAmplitudes(num_qubits=2, reps=1,
name="Layer2",parameter_prefix="l2"), qubits=[2,3])
ansatz.barrier()
ansatz.decompose().draw(output="mpl")
qnn = QuantumCircuit(4).compose(encoding).compose(ansatz)
qnn.decompose().draw("mpl")
```

- **Risultati MNIST 0-9:**
  - Silhouette: 0.547
  - Purity: 0.905

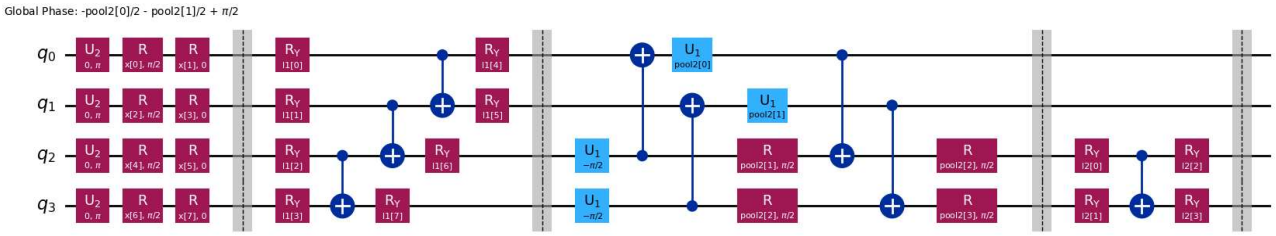
- **Schema architettura:**



Esperimenti effettuati

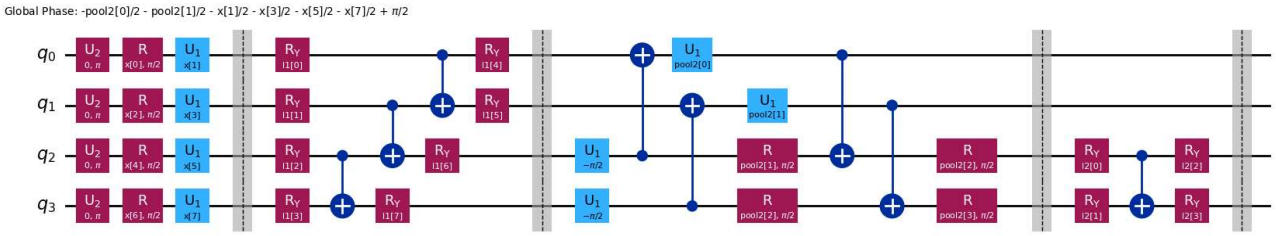
Modifiche Encoding

1. Encoding HRyRx



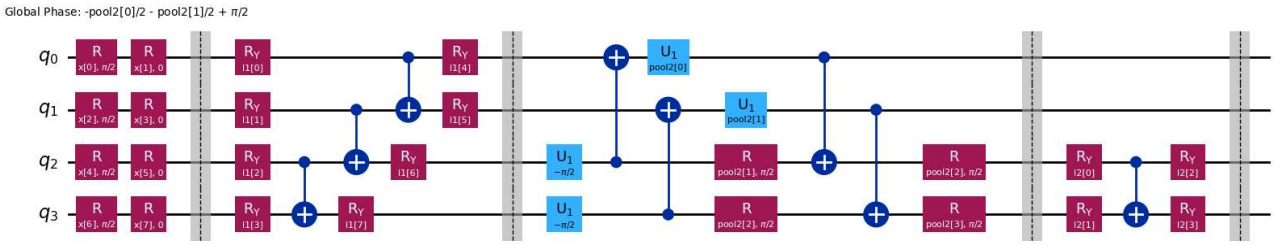
- **Risultati:**
  - **Purity: 0.896**
  - **Silhouette: 0.495**

2. Encoding HRyRz



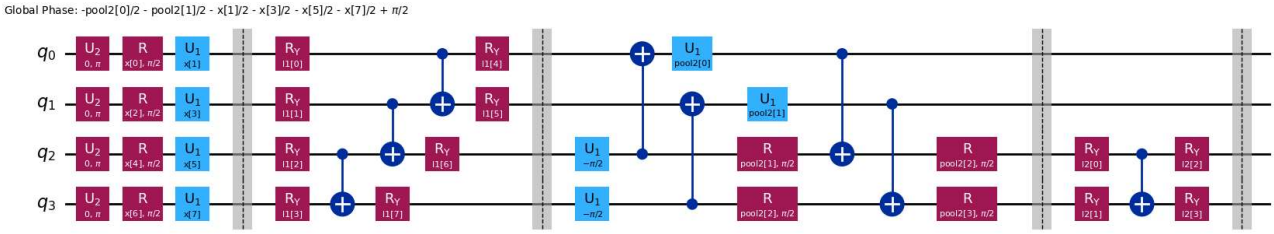
- **Risultati:**
  - **Purity: 0.849**
  - **Silhouette: 0.455**

3. Encoding RyRx



- **Risultati:**
  - **Purity: 0.780**
  - **Silhouette: 0.424**

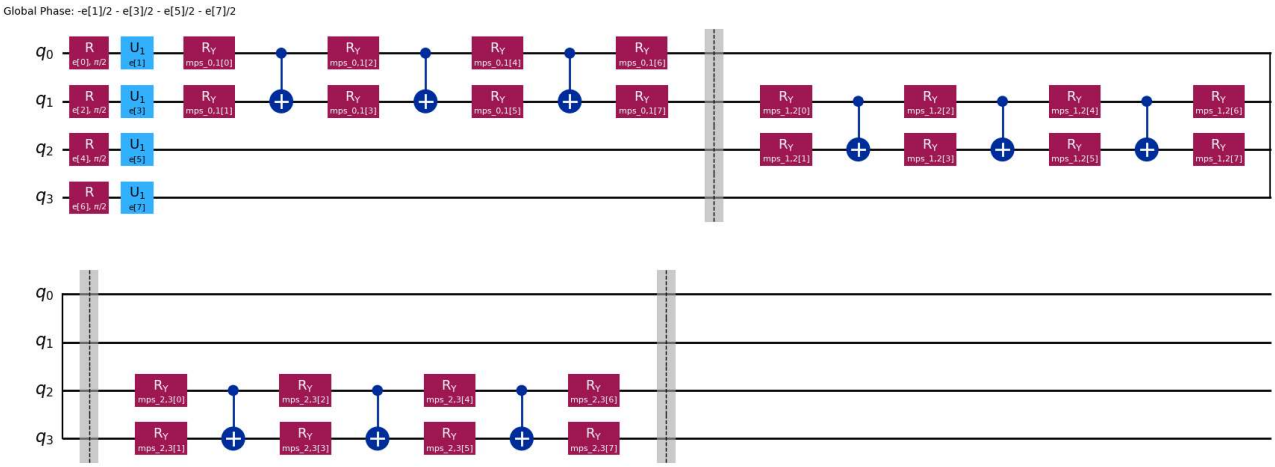
4. Encoding HRyRz + Normalizzazione



- **Risultati:**
  - **Purity: 0.842**
  - **Silhouette: 0.377**

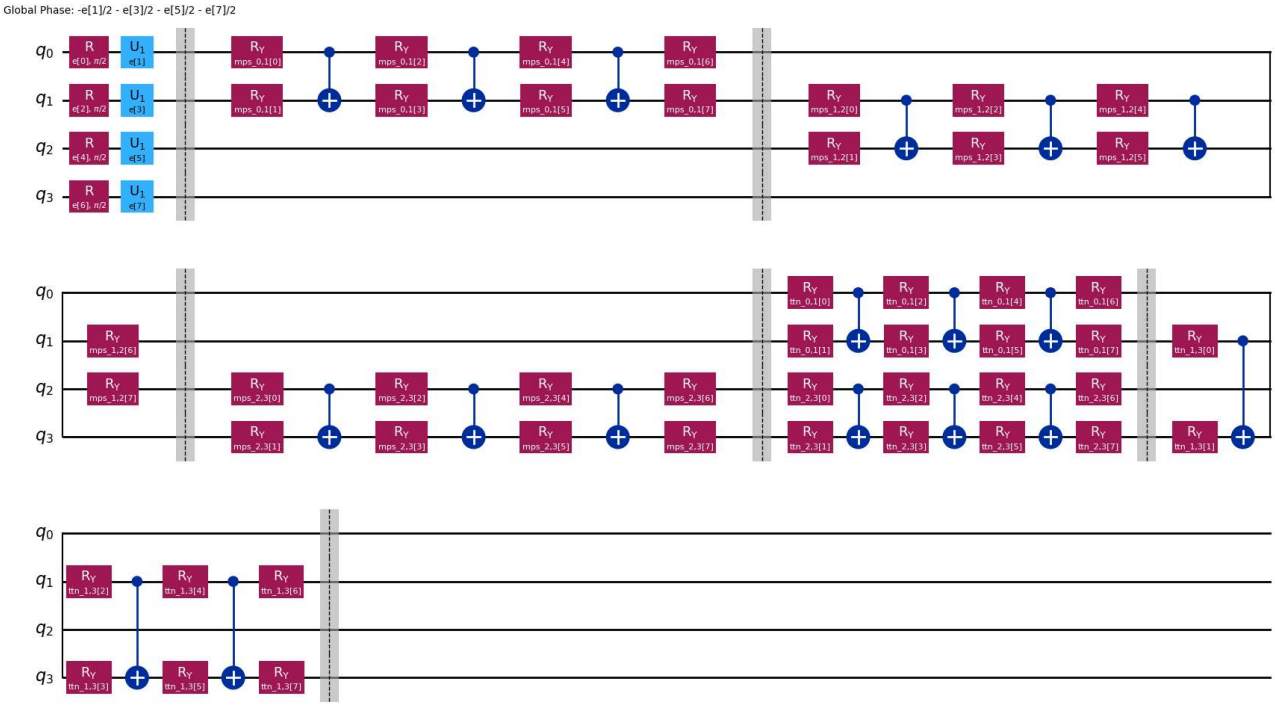
Modifiche Ansatz (con Encoding RyRz)

5. MPS



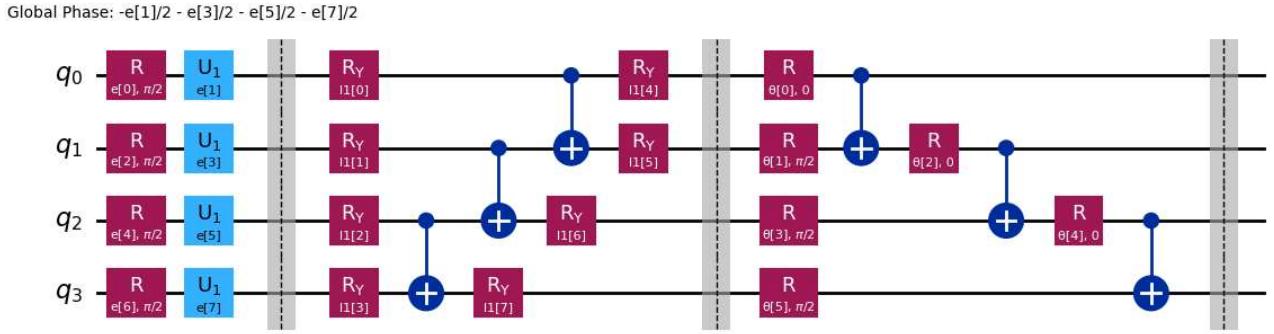
- **Risultati:**
  - **Purity: 0.904**
  - **Silhouette: 0.457**

6. MPS + TTN



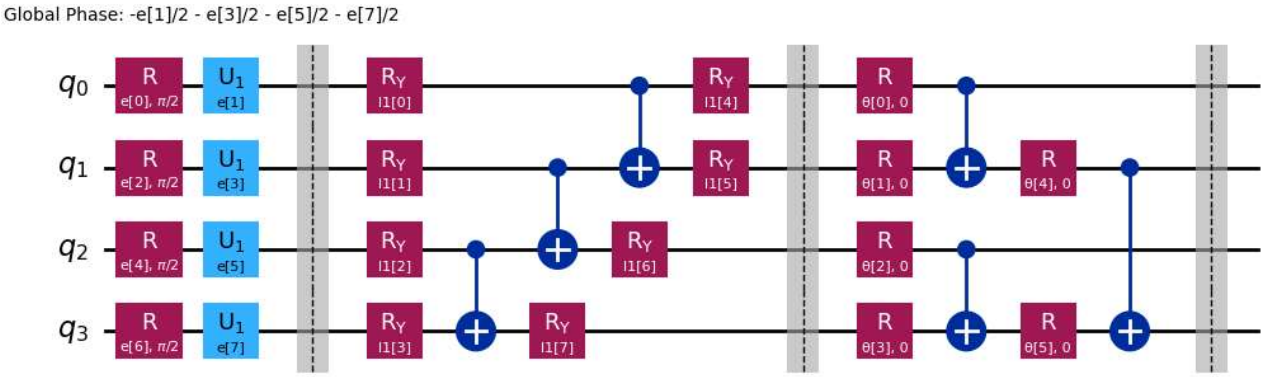
- **Risultati:**
  - **Purity: 0.885**
  - **Silhouette: 0.463**

7. CMPS



- **Risultati:**
  - **Purity: 0.784**
  - **Silhouette: 0.391**

8. CTTN



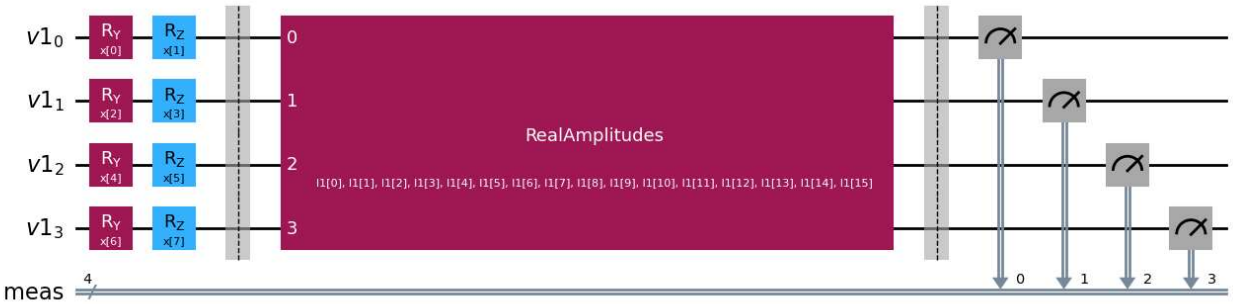
- **Risultati:**
  - **Purity: 0.885**
  - **Silhouette: 0.494**

Quantum Triplet Loss V2 + KMeans

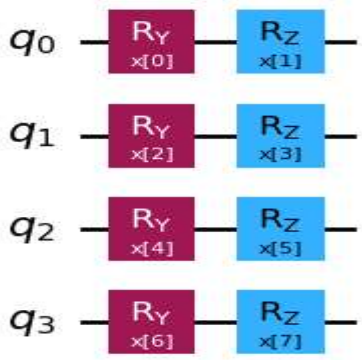
- **Descrizione del modello:**
  - Architettura di TripletLoss che prende ispirazione da QuantumTripletLoss. Le uniche differenza sono un Layer di RotationScaler finale che permette di effettuare un rescaling delle feature di embedding classico nel range [0,2pi]. Successivamente il modello quantum utilizza un YZ Encoding e un singolo layer RealAmplitudes su 4 qubit. Viene effettuata la misurazione su tutti i qubit generando un embedding di dimensionalitå 16.
- **Summary modello ibrido:**
  - **Embending model:**

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 24, 24]	832
ReLU-2	[-1, 32, 24, 24]	0
Conv2d-3	[-1, 32, 20, 20]	25,632
MaxPool2d-4	[-1, 32, 10, 10]	0
ReLU-5	[-1, 32, 10, 10]	0
Conv2d-6	[-1, 64, 6, 6]	51,264
MaxPool2d-7	[-1, 64, 3, 3]	0
ReLU-8	[-1, 64, 3, 3]	0
Flatten-9	[-1, 576]	0
Linear-10	[-1, 120]	69,240
ReLU-11	[-1, 120]	0
Linear-12	[-1, 8]	968
Sigmoid-13	[-1, 8]	0
RotationScaler-14	[-1, 8]	0

  - **CIRCUITO :**



- **Encoding con “yz_angles_encoding”:**



- **Codice Circuito:**

```
class YZ_RL_FullMeas(QuantumModel):
    def __init__(self):
        self.encoding = yz_angles_encoding(8, param_name="x")

        self.q1 = QuantumRegister(4, name="v1")

        self.ansatz = QuantumCircuit(self.q1)
        self.ansatz.barrier()
        self.ansatz  = self.ansatz.compose(RealAmplitudes(4,parameter_prefix="11"))
        self.ansatz.measure_all()

        self.qnn =
QuantumCircuit(self.q1).compose(self.encoding).compose(self.ansatz)

        sampler = Sampler(options={
        })

        self.qnn_net = SamplerQNN(
            sampler = sampler,
            circuit=self.qnn,
            input_params=self.encoding.parameters,
            weight_params=self.ansatz.parameters,
            input_gradients=True
        )
```

- **Risultati MNIST 0-9:**
  - Silhouette: 0.689
  - Purity: 0.868