**Báo cáo: Quantum Neural Network (QNN) cho CIFAR-10**

**Mô tả giải pháp (QNN ban đầu):   
Demo:** [**TruongGiaThach/QNNDemo**](https://github.com/TruongGiaThach/QNNDemo)

T triển khai QNN để phân loại CIFAR-10, kết hợp mạng nơ-ron truyền thống và mạch lượng tử (PennyLane):

* **Mô hình QNN**:
  + Tầng convolution (nn.Conv2d, 3→16 kênh, kernel 3x3, stride 2) giảm kích thước ảnh về [batch\_size, 16, 15, 15].
  + Tầng tuyến tính (pre\_net) ánh xạ thành 4 chiều (cho 4 qubit).
  + Mạch lượng tử (4 qubit, 2 layer, cổng RX, RY, CZ).
  + Tầng tuyến tính (post\_net) ra 10 lớp.
* **Dữ liệu**: CIFAR-10, chia 70/30 (train/test), batch size 32.
* **Huấn luyện**: 4 qubit, batch size 32, 10 epoch, dùng Adam (lr=0.001), CrossEntropyLoss, early stopping (patience=3).

**Kết quả (QNN ban đầu)**

* **Epoch 1**: Train Loss: 2.3162, Train Acc: 9.98%, Test Loss: 2.3034, Test Acc: 9.67%
* **Epoch 5**: Train Loss: 2.3036, Train Acc: 10.00%, Test Loss: 2.3034, Test Acc: 10.08%
* **Epoch 6**: Train Loss: 2.3036, Train Acc: 9.36%, Test Loss: 2.3034, Test Acc: 9.90%
* Early stopping sau 6 epoch (loss không cải thiện).
* **Phân tích**: Độ chính xác ~10% (như ngẫu nhiên), loss không giảm, mô hình không học được.

**Thiếu sót và hạn chế**

1. **Hiệu suất thấp**:
   * Tầng convolution (3→16 kênh, stride 2) giảm kích thước mạnh, có thể làm mất thông tin quan trọng.
   * 4 qubit và mạch lượng tử đơn giản không đủ mã hóa dữ liệu phức tạp.
2. **Tính toán**: Xử lý từng mẫu trong batch qua mạch lượng tử, rất chậm.

**Tiếp tục cải tiến**

1. Tăng số tầng convolution (hoặc dùng ResNet) để trích xuất đặc trưng tốt hơn trước khi đưa vào QNN.
2. Tăng số qubit (8-10) và độ phức tạp mạch (thêm layer, dùng StronglyEntanglingLayers).
3. Vector hóa xử lý batch trong mạch lượng tử để tăng tốc.

**Mô hình khác: Quanvolutional Neural Network (QuanvNN)**    
Demo : <https://colab.research.google.com/drive/1l53sOrMypAsXzuDtMiszwFg3CUbJ2uYj?usp=sharing>

Reference:

* https://pennylane.ai/qml/demos/tutorial\_quanvolution
* https://arxiv.org/pdf/1904.04767

**Mô tả giải pháp**

QuanvNN kết hợp tầng quanvolution và CNN cổ điển:

* **Tầng quanvolution**: Xử lý ảnh 32x32x3 qua mạch lượng tử (4 qubit, 1 layer, cổng RY + RandomLayers), tạo đầu ra 16x16x12.
* **CNN**: 2 tầng Conv2D (32, 64 bộ lọc) + 2 tầng MaxPooling2D + Flatten + 2 tầng Dense (64, 10 unit, softmax).
* **Dữ liệu**: CIFAR-10, giảm còn 1000 mẫu train, 200 mẫu test.
* **Huấn luyện**: 30 epoch, batch size 4, dùng Adam.

**Kết quả (QuanvNN)**

* **Epoch 1**: Train Acc: 17.50%, Train Loss: 2.1528, Val Acc: 36.50%, Val Loss: 1.9281
* **Epoch 5**: Train Acc: 50.90%, Train Loss: 1.3623, Val Acc: 37.50%, Val Loss: 1.7350
* **Epoch 10**: Train Acc: 83.70%, Train Loss: 0.5212, Val Acc: 29.00%, Val Loss: 2.8966
* **Epoch 18**: Train Acc: 96.60%, Train Loss: 0.1314, Val Acc: 39.50%, Val Loss: 3.3284 (độ chính xác validation cao nhất)
* **Epoch 30**: Train Acc: 100.00%, Train Loss: 0.0009, Val Acc: 38.00%, Val Loss: 4.8314
* **Phân tích**:
  + Độ chính xác validation cao nhất đạt 39.50% (epoch 18), cải thiện đáng kể so với QNN ban đầu (~10%).
  + Mô hình bị overfitting: Train accuracy tăng đều (đạt 100%), nhưng validation accuracy dao động 29-39.5% và validation loss tăng mạnh (từ 1.9281 lên 4.8314).

**Thiếu sót và hạn chế**

1. **Overfitting**:
   * Train accuracy đạt 100%, nhưng validation accuracy chỉ 39.50% (cao nhất), và validation loss tăng mạnh, cho thấy mô hình không tổng quát hóa tốt.
2. **Mạch lượng tử đơn giản**:
   * RandomLayers với 1 layer không đủ phức tạp để mã hóa đặc trưng tốt.
3. **Không có regularization**:
   * Thiếu Dropout, BatchNorm, hoặc data augmentation, dẫn đến overfitting.

**Tiếp tục cải tiến**

1. Dùng mạch lượng tử phức tạp hơn (tăng layer, thử StronglyEntanglingLayers).
2. Thêm regularization (Dropout, BatchNorm) và data augmentation (random flip, crop) để giảm overfitting.

**Kết luận**

QuanvNN cải thiện độ chính xác đáng kể so với QNN ban đầu (39.50% so với 10%), nhờ tầng quanvolution và CNN sâu hơn. Tuy nhiên, mô hình bị overfitting nghiêm trọng do dataset nhỏ và thiếu regularization. Cần tăng dataset, thêm regularization, và tối ưu kiến trúc để đạt hiệu suất tốt hơn.