Early exiting from Quantum Neural Networks as a noise mitigation strategy in NISQ devices, a preliminary study

Giacomo Vittori, Simone Scardapane, Stefano Giagu, Andrea Ciardiello

October 29, 2024

Sapienza
Università di Roma

Objectives Of The Research

Actual quantum computers are intrinsically affected by noise.

☞ *Noise Intermediate Scale Quantum* (**NISQ**) era of quantum computation

- limited number of qubits $\mathcal{O}(100)$;
- qubit decoherence and gates infidelity.

Objectives Of The Research

Actual quantum computers are intrinsically affected by noise.

☞ *Noise Intermediate Scale Quantum* (**NISQ**) era of quantum computation

- limited number of qubits $\mathcal{O}(100)$;
- qubit decoherence and gates infidelity.

⚠ Strong limitations on Quantum Neural Network (**QNN**) architectures

- circuit depth $\to$ expressibility;
- precise computation.

Objectives Of The Research

Actual quantum computers are intrinsically affected by noise.

☞ *Noise Intermediate Scale Quantum* (**NISQ**) era of quantum computation

- limited number of qubits $\mathcal{O}(100)$;
- qubit decoherence and gates infidelity.

⚠ Strong limitations on Quantum Neural Network (**QNN**) architectures

- circuit depth $\rightarrow$ expressibility;
- precise computation.

Novel noise mitigation strategy based on **early-exiting** the network to

- maintain the advantages of a deeper architecture;
- limit the cumulative effect of noise;

Summary

Why should we use early exits in neural networks?

Most Neural Networks are a sequential stack of differentiable layers. Taking x as input and considering L differentiable operators f_i

$$f(x) = (f_l \circ f_{L_1} \circ \cdots \circ f_1)(x)$$

Why should we use early exits in neural networks?

Most Neural Networks are a sequential stack of differentiable layers. Taking x as input and considering L differentiable operators f_i

$$f(x) = (f_l \circ f_{L_1} \circ \cdots \circ f_1)(x)$$

The parameters can be trained with iterative methods, such Stochastic Gradient Descent. Given a set of examples $\{x_n, y_n\}_{n=1}^N$ and a loss function l

$$f^* = \underset{\theta}{\text{argmin}} \sum_{n=1}^{N} l(y_n, f(x_n))$$

In this setup, all the layers must be evaluated to obtain the final prediction.

Why should we use early exits in neural networks?

Most Neural Networks are a sequential stack of differentiable layers. Taking x as input and considering L differentiable operators f_i

$$f(x) = (f_l \circ f_{L_1} \circ \cdots \circ f_1)(x)$$

The parameters can be trained with iterative methods, such Stochastic Gradient Descent. Given a set of examples $\{x_n, y_n\}_{n=1}^{N}$ and a loss function l

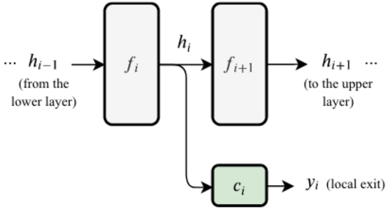
$$f^* = \operatorname*{argmin}_{\theta} \sum_{n=1}^{N} l(y_n, f(x_n))$$

In this setup, all the layers must be evaluated to obtain the final prediction.

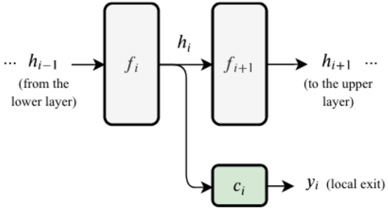
- Training can be improved by techniques such residual connections;
- overfitting and vanishing gradient can still happen even with a strong regularization;
- inference of large models can be hard with limited resources or in distributed scenarios.

Early Exiting Neural Networks

We can select a set of *interesting points* in the architecture (*backbone*), called **early-exits**. Feeding the intermediate embedding h_i to an auxiliary neural network, we can obtain an intermediate prediction y_i



We can select a set of *interesting points* in the architecture (*backbone*), called **early-exits**. Feeding the intermediate embedding h_i to an auxiliary neural network, we can obtain an intermediate prediction y_i



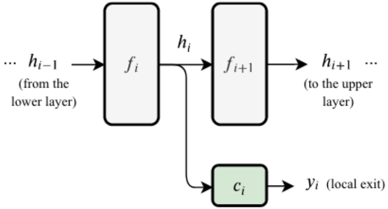
Training approaches examples:

- *jointly approach* defining a single optimization, combining the per-classifier losses or the outputs;

Early Exiting Neural Networks

2 Early Exiting Neural Networks

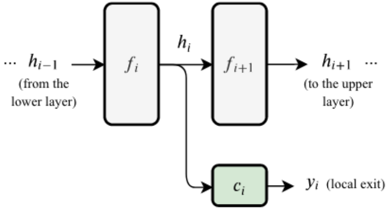
We can select a set of *interesting points* in the architecture (*backbone*), called **early-exits**. Feeding the intermediate embedding h_i to an auxiliary neural network, we can obtain an intermediate prediction y_i



Training approaches examples:

- *jointly approach* defining a single optimization, combining the per-classifier losses or the outputs;
- *layer-wise approach* training at each iteration a single auxiliary classifier together with the backbone and then freezing it;

We can select a set of *interesting points* in the architecture (*backbone*), called **early-exits**. Feeding the intermediate embedding h_i to an auxiliary neural network, we can obtain an intermediate prediction y_i



Training approaches examples:

- *jointly approach* defining a single optimization, combining the per-classifier losses or the outputs;
- *layer-wise approach* training at each iteration a single auxiliary classifier together with the backbone and then freezing it;
- training the backbone network and then separately training the auxiliary classifiers on top.

Inference

Early exiting can be useful to:

- improve the training phase, using only the final output in inference;
- obtain a joint prediction merging the intermediate ones;
- exploit the exits to improve efficiency, using the full network only for hard prediction.

Inference

Early exiting can be useful to:

- improve the training phase, using only the final output in inference;
- obtain a joint prediction merging the intermediate ones;
- exploit the exits to improve efficiency, using the full network only for hard prediction.

Each input can be processed up to the optimal *depth*, corresponding to a specific prediction $c_i(x)$ and then stop the execution.

Inference

Early exiting can be useful to:

- improve the training phase, using only the final output in inference;
- obtain a joint prediction merging the intermediate ones;
- exploit the exits to improve efficiency, using the full network only for hard prediction.

Each input can be processed up to the optimal *depth*, corresponding to a specific prediction $c_i(x)$ and then stop the execution.

Considering a classification problem, a **confidence** measure on the intermediate prediction can be used to decide if and where to early-exit.

- Select (or train) a threshold for each auxiliary classifier.

Summary

3 Early Exiting Quantum Neural Networks To Mitigate Noise

Variational Quantum Algorithms

Most of Quantum Neural Networks are based on Variational Quantum Algorithms.

Given a problem description and a set of training data:

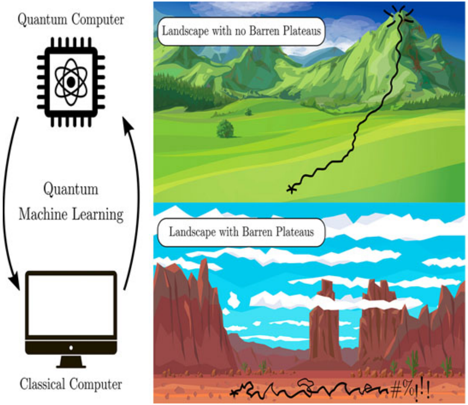
1. encode the solution in a **cost function**;
2. propose an **ansatz**, quantum operations depending on a set of parameters;
3. train in a hybrid quantum-classical loop to solve the optimization task.

Variational Quantum Algorithms

Most of Quantum Neural Networks are based on Variational Quantum Algorithms.

Given a problem description and a set of training data:

1. encode the solution in a **cost function**;
2. propose an **ansatz**, quantum operations depending on a set of parameters;
3. train in a hybrid quantum-classical loop to solve the optimization task.



⚠ **Barren Plateau**: extremely flat regions in the parameter space.

- random initialization and ansatz choice;
- noise.

- Extend early exits to quantum neural networks to
  - adapt the computation to the complexity of the task;
  - reduce the cumulative effects of noise;
  - maintain the expressibility of a deeper architecture.

Early-Exiting Quantum Neural Networks

Extend early exits to quantum neural networks to

- adapt the computation to the complexity of the task;
- reduce the cumulative effects of noise;
- maintain the expressibility of a deeper architecture.

! The implementation would require partial measurements at intermediate points to obtain the predictions

- Measurement process interrupts the signal propagation to successive points of the network;
- No Cloning theorem forbids the creation of identical copies of an arbitrary unknown quantum state;

Early-Exiting Quantum Neural Networks

- Extend early exits to quantum neural networks to
  - adapt the computation to the complexity of the task;
  - reduce the cumulative effects of noise;
  - maintain the expressibility of a deeper architecture.

- ! The implementation would require partial measurements at intermediate points to obtain the predictions
  - Measurement process interrupts the signal propagation to successive points of the network;
  - No Cloning theorem forbids the creation of identical copies of an arbitrary unknown quantum state;

- ✔ Simplest approach:
  - Measure a different set of qubits for each required prediction.

MNIST classification

The MNIST dataset is a widely used benchmark in the field of machine learning and computer vision, consisting in grayscale handwritten images (0-9) 28×28.

MNIST classification

4 Experimental Setup

The MNIST dataset is a widely used benchmark in the field of machine learning and computer vision, consisting in grayscale handwritten images (0-9) 28×28.

Considering an 8 qubits system:

$$n = 2^8 \text{ possible states}$$

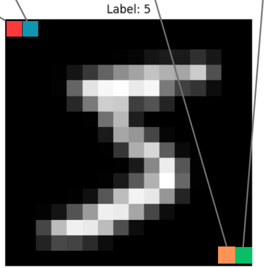
Image preprocessing:

- min-max scaling to improve training;
- reshaping to 16×16;
- normalization to perform amplitude encoding.

Denoting as p_i the pixels values:

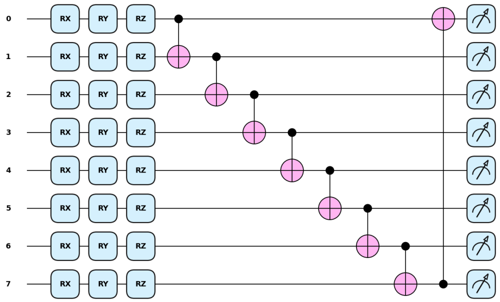
$$\sum_{i=1}^{256} p_i^2 = 1$$

$\boxed{\alpha_1}|00000000\rangle + \boxed{\alpha_2}|00000001\rangle + \cdots + \boxed{\alpha_{255}}|11111110\rangle + \boxed{\alpha_{256}}|11111111\rangle$



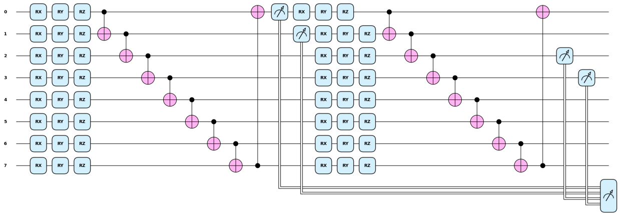
Layer design is inspired by the circut-centric classifier ansatz.

- RX, RY, RZ parametric rotations applied on each qubit;
- CNOT gates to create entanglement;
- 24 parameters per layers;
- measurement in the computational basis for the prediction.



Each possible outcome is associated to a specific class.

Early exiting through mid circuit measurements

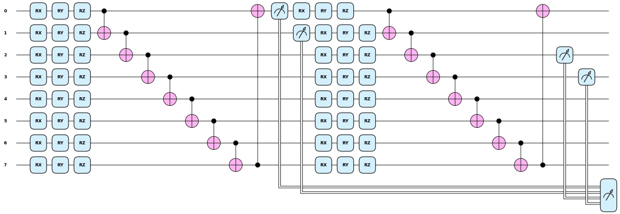


Calling Π_i the projector associated to the outcome i of the measurement, the output state is

$$\rho_i = \frac{\Pi_i \rho \Pi_i}{\mathrm{Tr}[\Pi_i \rho]}$$

Early exiting through mid circuit measurements

4 Experimental Setup



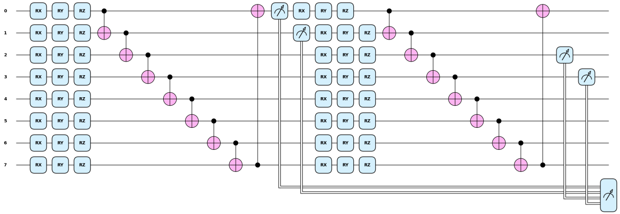
Calling Π_i the projector associated to the outcome i of the measurement, the output state is

$$\rho_i = \frac{\Pi_i \rho \Pi_i}{\mathrm{Tr}[\Pi_i \rho]}$$

- **Training phase**: *cross-entropy loss* $\mathcal{L}_{\text{tot}} = \mathcal{L}_{early} + \mathcal{L}_{final}$, *ADAM* optimizer.

Early exiting through mid circuit measurements

4 Experimental Setup



Calling Π_i the projector associated to the outcome i of the measurement, the output state is

$$\rho_i = \frac{\Pi_i \rho \Pi_i}{\mathrm{Tr}[\Pi_i \rho]}$$

- **Training phase**: *cross-entropy loss* $\mathcal{L}_{\text{tot}} = \mathcal{L}_{early} + \mathcal{L}_{final}$, *ADAM* optimizer.
- **Test phase**: a threhshold determines the intermediate or the full execution depending on the confidence of the prediction.

Early exiting model without noise

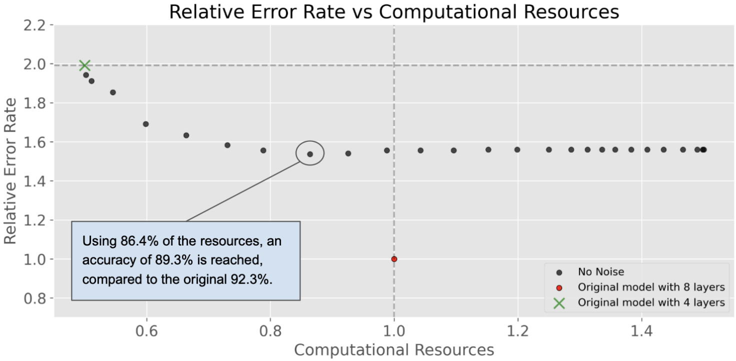
$$\text{RER}_i = \frac{1 - \text{Accuracy}[\text{Model}_i]}{1 - \text{Accuracy}[\text{Original Model (8 layer)}]}$$

$$\text{CR}_i = \frac{\#\text{Executed gates}[\text{Model}_i]}{\#\text{Executed gates}[\text{Original Model (8 layers)}]}$$

Early exiting model without noise

4 Experimental Setup

$$\mathrm{RER}_i = \frac{1 - \mathrm{Accuracy}[\mathrm{Model}_i]}{1 - \mathrm{Accuracy}[\text{Original Model (8 layer)}]} \qquad \mathrm{CR}_i = \frac{\#\text{Executed gates}[\mathrm{Model}_i]}{\#\text{Executed gates}[\text{Original Model (8 layers)}]}$$



(1 , 1) → Original model (8 layers)

(0.5 , X) → Original model (4 layers)

Coherent noise

- incorrect gates calibration;
- purity preservation.

Pure state simulator

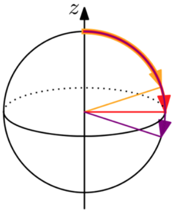
Noise in quantum circuits

Coherent noise

- incorrect gates calibration;
- purity preservation.

Pure state simulator

→ Rotational parameters disturbance



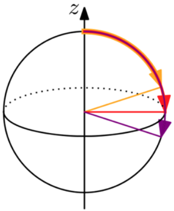
Noise in quantum circuits

Coherent noise

- incorrect gates calibration;
- purity preservation.

Pure state simulator

→ Rotational parameters disturbance



Incoherent noise

- entanglement with environment;
- no purity preservation.

Mixed state simulator

Noise in quantum circuits

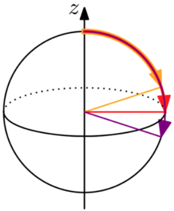
4 Experimental Setup

Coherent noise

- incorrect gates calibration;
- purity preservation.

Pure state simulator

→ Rotational parameters disturbance

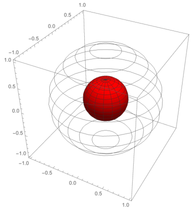


Incoherent noise

- entanglement with environment;
- no purity preservation.

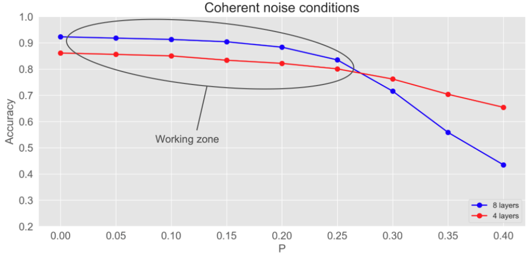
Mixed state simulator

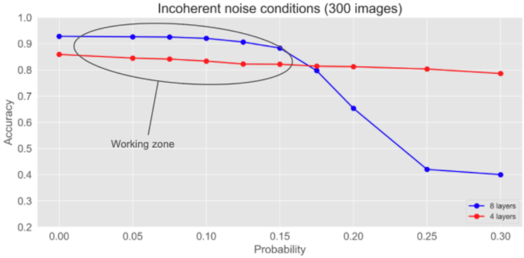
→ Depolarizing channel



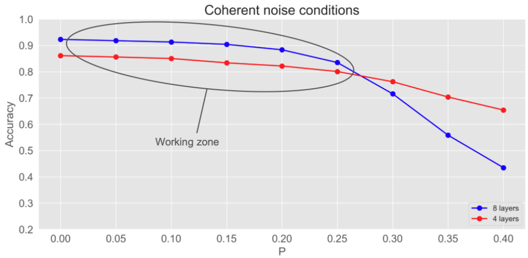
Noise effects on the original model

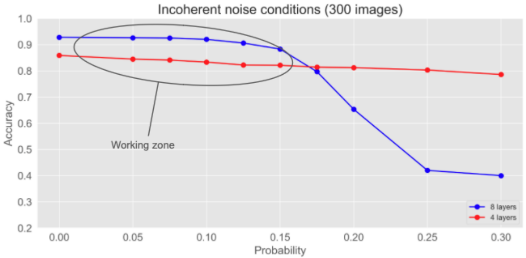
- 8 layers vs 4 layers noise effects on the original architecture.





- 8 layers vs 4 layers noise effects on the original architecture.





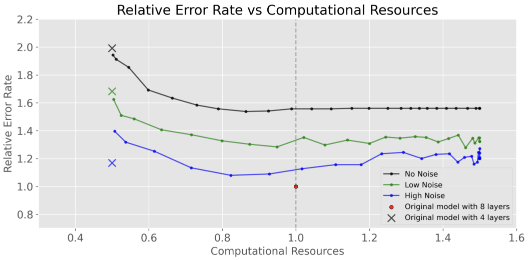
- ! Less expressive circuits tend to be more resilient to noise, as the execution is shorter and therefore less susceptible to accumulated errors.

Noise mitigation through early exit

4 Experimental Setup

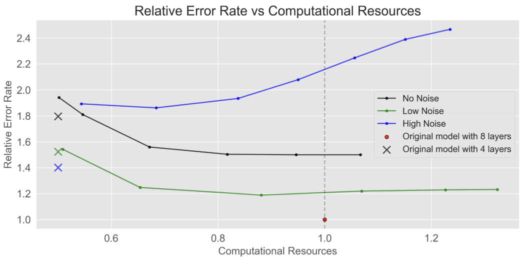
Coherent noise

Low = 0.15 High = 0.25



Incoherent noise

Low = 0.015 High = 0.12

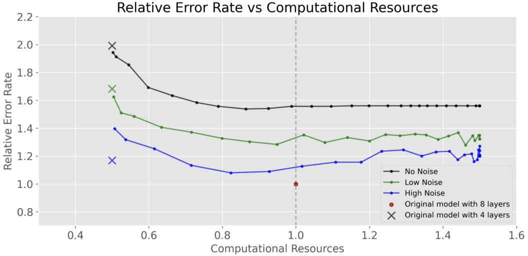


Noise mitigation through early exit

4 Experimental Setup

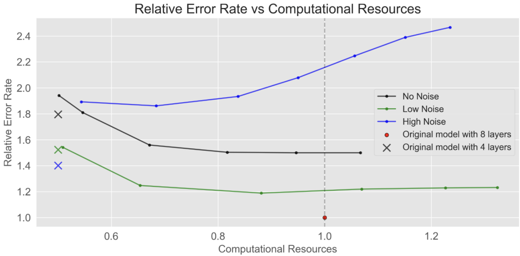
Coherent noise

Low = 0.15 High = 0.25



Incoherent noise

Low = 0.015 High = 0.12



- Under coherent and low levels of incoherent noise early exiting save computational resources while maintaining good performances
- Noise mitigation strategy under high levels of incoherent noise needs further refinements

Summary

5 Conclusion

Conclusions and future directions

Summary:

- mitigation of coherent noise effects and low levels of incoherent noise;
- adaptability to the difficulty of the task;
- better computational efficiency while maintaining greater expressive capacity;
- adaptability to noise levels allowing the use of a single circuit instead of searching for an optimal configuration of layers for each noise level.

Conclusions and future directions

Summary:

- mitigation of coherent noise effects and low levels of incoherent noise;
- adaptability to the difficulty of the task;
- better computational efficiency while maintaining greater expressive capacity;
- adaptability to noise levels allowing the use of a single circuit instead of searching for an optimal configuration of layers for each noise level.

Future directions:

- further refinements for high incoherent noise levels;
- realistic noise simulation;
- uncertainty analysis;
- generalization to different ansatzes and multiple exit configurations.

Bibliography

Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al., *Variational quantum algorithms*, Nature Reviews Physics **3** (2021), no. 9, 625–644.

Vinayak Jagadish and Francesco Petruccione, *An invitation to quantum channels*, arXiv preprint arXiv:1902.00909 (2019).

PennyLane Team, *Dynamic quantum circuits*, 2021.

John Preskill, *Quantum computing in the nisq era and beyond*, Quantum **2** (2018), 79.

Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe, *Circuit-centric quantum classifiers*, Physical Review A **101** (2020), no. 3, 032308.

Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini, *Why should we add early exits to neural networks?*, Cognitive Computation **12** (2020), no. 5, 954–966.

Thanks for your attention!