

# PQC Function Evaluation

## Carnegie Vacation Scholarship

David Amorim

Week 4  
(22/07/2024 - 26/07/2024)

# Aims for the Week

The following aims were set at the last meeting (22/07/2024):

## 1. Change Input Layer Structure

Improve the connectivity of input layers. Each input qubit should ideally control each target qubit at some point in the network.

## 2. Fix Parameters

Add the option to keep parameters fixed for each type of network layer.

## 3. Improve Loss Function

Develop a distance measure taking into account digital encoding. Either incorporate this into weights for an existing loss function or define a new loss function on this basis.

# Glossary

Acronym	Meaning
CL	convolutional layer
AA-CL	all-to-all convolutional layer
NN-CL	neighbour-to-neighbour convolutional layer
IL	input layer
SAM	sign-adjusted mismatch

Table 1: Acronyms and short-hands used in the following.

Variable	Meaning
$n$	input register size
$m$	target register size
$L$	number of network layers

Table 2: Variables used in the following.

# Table of Contents

① Changing Input Layer Structure

② Fixing Parameters

③ Improving the Loss Function

④ Results

⑤ Next Steps

# Changing Input Layer Structure

- Previously, the  $j$ th input qubit controlled an operation on the  $j$ th target qubit (with wrap-around for  $n > m$ )
- An optional **shift parameter**,  $s$ , has now been added so that the  $j$ th input qubit controls an operation on the  $j + s$ th target qubit
- This shift parameter is incremented for each successive IL
- The QCNN is padded with additional ILs to ensure that the number of ILs is  $\geq m$
- Thus, **each input qubit now controls an operation on each target qubit** at some point in the QCNN
- Note that ILs still alternate between control states 0 and 1

# Changing Input Layer Structure

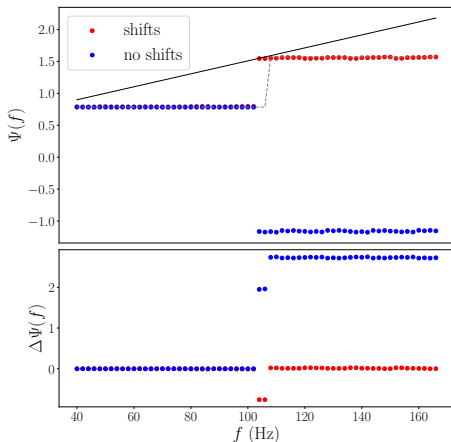


Figure 1: Effects of shifted ILs for  $\Psi(f) \sim f$  and  $m = 3$  ( $L = 6$ , 600 epochs, SAM)

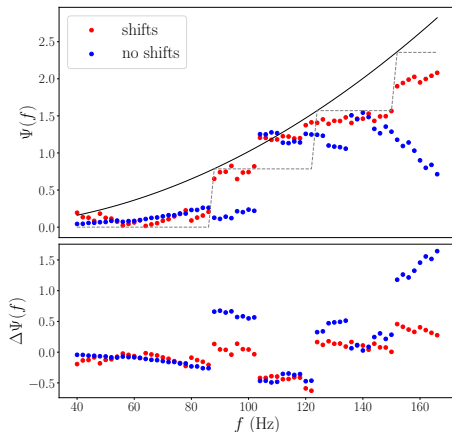


Figure 2: Effects of shifted ILs for  $\Psi(f) \sim f^2$  and  $m = 3$  ( $L = 6$ , 600 epochs, SAM)

# Changing Input Layer Structure

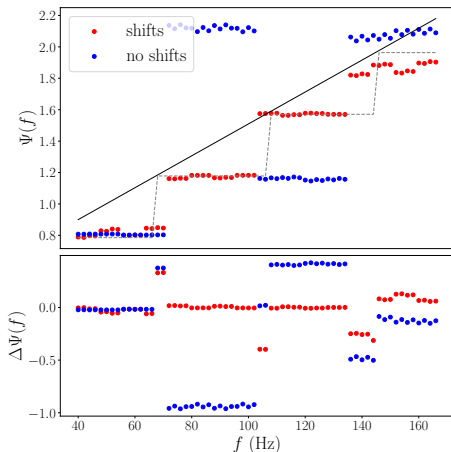


Figure 3: Effects of shifted ILs for  $\Psi(f) \sim f$  and  $m = 4$  ( $L = 6$ , 600 epochs, SAM)

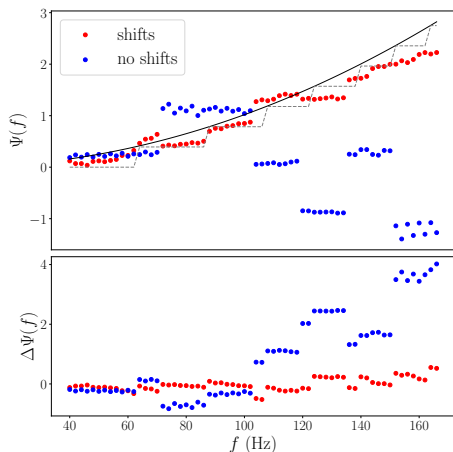


Figure 4: Effects of shifted ILs for  $\Psi(f) \sim f^2$  and  $m = 4$  ( $L = 6$ , 600 epochs, SAM)

# Changing Input Layer Structure

- The data for ‘no shifts’ was obtained by setting  $s = 0$  for all ILs instead of incrementing  $s$
- This should be equivalent to last week’s circuit structure
- However, the ‘no shifts’ results are significantly worse than the results shown last week (???)
- Thus, the improvements due to the new IL structure are somewhat exaggerated
- Nonetheless, **increased IL connectivity clearly leads to improved performance**

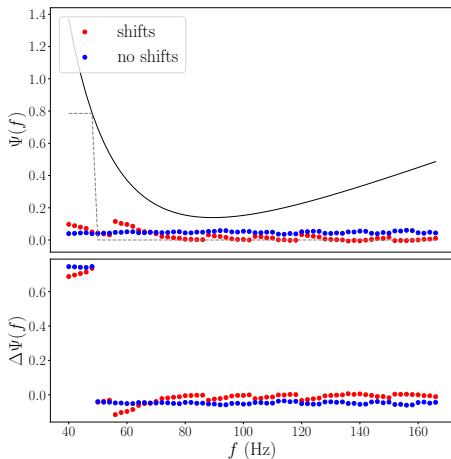


Figure 5: Effects of shifted ILs for  $\Psi_{\text{Hayes2023}}$  and  $m = 3$  ( $L = 6$ , 600 epochs, SAM)



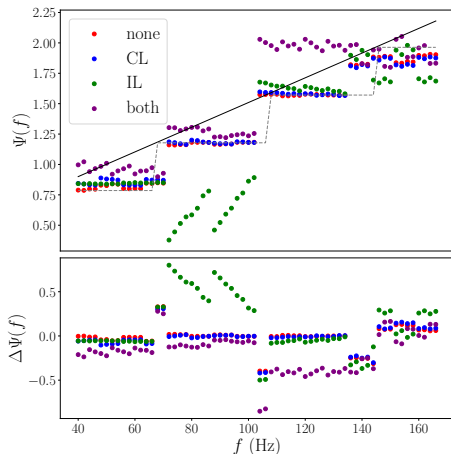
# Table of Contents

- ① Changing Input Layer Structure
- ② Fixing Parameters
- ③ Improving the Loss Function
- ④ Results
- ⑤ Next Steps

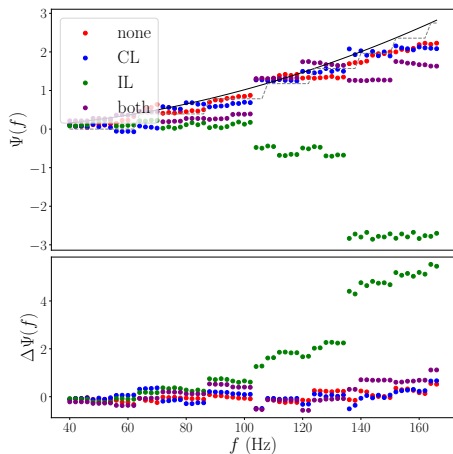
# Fixing Parameters

- Implemented the option to **fix parameters for each layer type**
- This means that each instance of a layer type (IL, AA-CL, NN-CL) uses the same set of parameters
- This significantly **reduces the number of trainable parameters** at large  $L$
- Surprisingly, reducing the parameter space **produces no noticeable speed-up** (so-called qiskit primitives, i.e. the sampler, take up roughly 95% of the computational time)

# Fixing Parameters



**Figure 6:** Effects of fixing parameters ILs for  $\Psi(f) \sim f$  and  $m = 4$  ( $L = 6$ , 600 epochs, SAM)



**Figure 7:** Effects of fixing parameters for  $\Psi(f) \sim f^2$  and  $m = 4$  ( $L = 6$ , 600 epochs, SAM)

# Fixing Parameters

- Legend for the plots on the previous slide:
  - '*none*' : no parameters fixed
  - '*CL*': only CL parameters fixed
  - '*IL*': only IL parameters fixed
  - '*both*': all parameters fixed
- Evidently, keeping parameters fixed leads to (slightly, for '*CL*', or drastically, for '*IL*' and '*both*') **worse performance**
- This is likely due to a reduction of the search space
- Note that '*IL*' as well as '*both*' lead to **incomplete phase extraction** (formalise this concept!) and hence somewhat meaningless results
- Thus, especially taking into account the equivalent computational times, **not keeping parameters fixed yields better results**
- These results suggest a particular importance of ILs compared to CLs

# Table of Contents

- ① Changing Input Layer Structure
- ② Fixing Parameters
- ③ Improving the Loss Function
- ④ Results
- ⑤ Next Steps

# Preliminary Definitions

- Consider a computational basis state,  $|k\rangle$ , of the **combined** input-register-target-register **system**
- The state  $|k\rangle$  is associated with a **bit string**  $k = \{0, 1\}^{n+m}$
- Denote by  $[k]_n$  and  $[k]_m$  the bit strings of length  $n$  and  $m$ , respectively, associated with each of the registers and write their **concatenation** as

$$k \equiv [k]_n \diamond [k]_m \quad (1)$$

- A general state of the two-register system is then written as

$$|z\rangle = \sum_{k=0}^{2^{n+m}-1} z_k |k\rangle \quad (2)$$

and referred to via its **coefficients**  $z_k$

# Preliminary Definitions

- When training in superposition, the **desired state** of the system is

$$|y\rangle = \sum_{j=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |j\rangle_i |\Psi(j)\rangle_t, \quad (3)$$

where the subscripts  $i$  and  $t$  indicate basis states of the input and target registers, respectively

- This state  $|y\rangle$  can be written in terms of the **combined basis**  $\{|k\rangle\}$  via

$$y_k = \begin{cases} \frac{1}{\sqrt{2^n}} & \text{if } k = [k]_n \diamond \Psi([k]_n) \\ 0 & \text{else} \end{cases} \quad (4)$$

- Further denote the **output state** produced by the QCNN by  $|x\rangle$ , with associated coefficients  $x_k$

# SAM and Beyond

- Recall the definition of **SAM**:

$$\text{SAM}(|x\rangle, |y\rangle) = \left| 1 - \sum_k x_k y_k \right| \quad (5)$$

- By construction, this is closely related to the **mismatch**

$$M(|x\rangle, |y\rangle) = 1 - |\langle x|y\rangle| \quad (6)$$

- While effective, SAM's **fundamental flaw** is that it does not directly take into account the amplitudes  $x_k$  for  $k$  where  $y_k = 0$  (i.e. where  $[k]_m \neq \Psi([k]_n)$ )



# SAM and Beyond

- Consider  $k = a$  and  $k = b$  with  $[k]_m \neq \Psi([k]_n)$  for both and

$$\left| [a]_m - \Psi([a]_n) \right| \geq \left| [b]_m - \Psi([b]_n) \right| \quad (7)$$

- To improve performance, the loss function should punish a non-zero  $x_a$  more than a non-zero  $x_b$  which is not the case for SAM
- This could be achieved via a **weighted mismatch**,

$$\text{WIM}(|x\rangle, |y\rangle) = \left| 1 - \sum_k \tilde{w}_k x_k y_k \right|, \quad (8)$$

where  $\tilde{w}_k \in \mathbb{R}_+$  are appropriate weights

# SAM and Beyond

- It was discussed at the last meeting to base the weights on

$$w_k = \sum_{\substack{l=0, \\ l \neq [k]_m}}^{2^m-1} \left| x_{[k]_n \diamond l} \right| \left| l - \Psi([k]_n) \right|, \quad (9)$$

with the  $\tilde{w}_k$  obtained from the  $w_k$  via normalisation and smoothing ( $\tilde{w}_k \sim e^{\tau w_k}$ )

- Implementing this **proved to be ineffective**, with no improvement on SAM
- This raises broader questions about the **feasibility of WIM**

# The Limits of WIM

- SAM is effective as it learns to maximise the  $x_k$  for  $k$  with  $y_k \neq 0$  without regard for the remaining  $x_k$
- It effectively optimises over a **reduced set of states**
- This is, presumably, why it outperforms more global loss functions, which directly take into account all coefficients, like  $L_1$  loss
- Thus, it seems that SAM's '**fundamental flaw**', of disregarding most  $x_k$ , is really the **basis of its success**
- Adding weights to SAM (WIM) likely is insufficient to alter its fundamental dynamic (or will alter it detrimentally)
- Therefore, **WIM will be abandoned for the foreseeable future**

# Introducing WILL

- To improve on SAM, it could be beneficial to return to a loss function, which more directly takes into account all  $x_k$  (e.g.  $L_1$  loss)
- More generally, define  $L_p$  loss as

$$LL_p(|x\rangle, |y\rangle) = \left( \sum_k |x_k - y_k|^p \right)^{1/p} \quad (10)$$

- Note that for phase encoding, **computational basis states are not equidistant**: their distance depends on the value they encode on the input and target registers
- A **weighted  $L_p$  loss (WILL)** can factor in an appropriate distance measure for the state space:

$$WILL_{p,q} = \left( \sum_k |x_k - y_k|^p \left| [k]_m - \Psi([k]_n) \right|^q \right)^{1/p} \quad (11)$$

# Testing WILL

- The following metrics are useful when testing WILL for different  $p, q$ :
  - Let  $M_j$  be the mismatch between  $|j\rangle |\Psi(j)\rangle$  and  $\hat{Q}_\Psi |j\rangle |0\rangle$
  - Define  $\mu \equiv \text{Mean}(M_j)$  and  $\sigma \equiv \text{STDEV}(M_j)$
- SAM tends to low  $\mu$  but high  $\sigma$ , corresponding to the encoding working well for most input states and quite badly for some outliers
- 
- SPEND SOME TIME PLAYING AROUND WITH  $p$  AND  $q$ !
- training (e600, L6, m3)  $p$  0.25 0.5 0.75 1 1.25 1.5 1.75 2 with  $q$  0, 0.5, 1, 1.5, 2, 2.5, 3
- repeat above with L 9! Currently training:  $p$  0.75 (completed:  $p$  1)

# Testing WILL

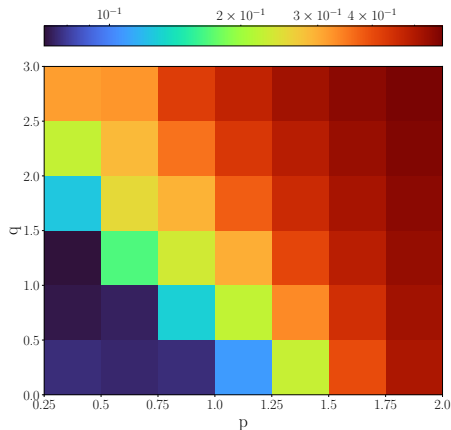


Figure 8: Comparing  $\mu$  for various  $p, q$  ( $L = 6, m = 3, 600$  epochs,  $\Psi_{H23}$ )

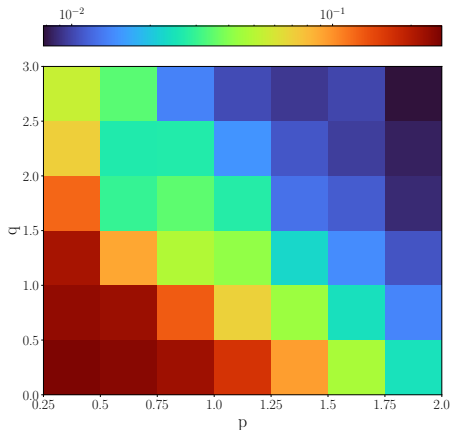


Figure 9: Comparing  $\sigma$  for various  $p, q$  ( $L = 6, m = 3, 600$  epochs,  $\Psi_{H23}$ )

# Table of Contents

- ① Changing Input Layer Structure
- ② Fixing Parameters
- ③ Improving the Loss Function
- ④ Results
- ⑤ Next Steps

Show phase encoding with improved methods show the full waveform [new frame]



# Table of Contents

- ① Changing Input Layer Structure
- ② Fixing Parameters
- ③ Improving the Loss Function
- ④ Results
- ⑤ Next Steps

# Next Steps

- formalise and investigate ‘completeness of phase extraction’
- look into adding more ILs compared to CLs ?