MerLin: A Discovery Engine for Photonic and Hybrid Quantum Machine Learning

Cassandre Notton*‖, Benjamin Stott†‖, Philippe Schoeb*‡, Anthony Walsh†, Grégoire Leboucher†§, Vincent Espitalier†, Vassilis Apostolou†, Louis-Félix Vigneux*¶, Alexia Salavrakos†, Jean Senellart†
*Quandela Quantique Inc., Montréal, Canada
†Quandela, Massy, France
‡DIRO & Mila, Université de Montréal, Montréal, Canada
§ENS Paris-Saclay, Gif-sur-Yvette, France
¶Département d'informatique, Université de Sherbrooke, Sherbrooke, Canada
‖These authors contributed equally to this work.

***Abstract*—Identifying where quantum models may offer practical benefits in near-term quantum machine learning (QML) requires moving beyond isolated algorithmic proposals toward systematic and empirical exploration across models, datasets, and hardware constraints. We introduce MerLin, an open-source framework designed as a discovery engine for photonic and hybrid quantum machine learning. MerLin integrates optimized strong simulation of linear-optical circuits into standard PyTorch and scikit-learn workflows, enabling end-to-end differentiable training of quantum layers. MerLin is designed around systematic benchmarking and reproducibility. As an initial contribution, we reproduce eighteen state-of-the-art photonic and hybrid QML works spanning kernel methods, reservoir computing, convolutional and recurrent architectures, generative models, and modern training paradigms. These reproductions are released as reusable, modular experiments that can be directly extended and adapted, establishing a shared experimental baseline consistent with empirical benchmarking methodologies widely adopted in modern artificial intelligence. By embedding photonic quantum models within established machine learning ecosystems, MerLin allows practitioners to leverage existing tooling for ablation studies, cross-modality comparisons, and hybrid classical–quantum workflows. The framework already implements hardware-aware features, allowing tests on available quantum hardware while enabling exploration beyond its current capabilities, positioning MerLin as a forward-looking co-design tool linking algorithms, benchmarks, and hardware.**

***Index Terms*—Quantum machine learning, photonic quantum machine learning, photonic quantum computing, hybrid quantum-classical models, Benchmarking and reproducibility**

I. Introduction and Motivation

The convergence of quantum computing and machine learning is driving significant advances in both theoretical research and practical applications, with QML offering the potential to accelerate and extend the capabilities of classical algorithms. Among the various quantum platforms, photonic quantum computing is particularly promising due to its scalability, robustness, compatibility with optical communication technologies, and energy efficiency [1]–[4]. Building on these advantages, photonic QML exploits the bosonic nature of light and high-dimensional multi-mode interference to implement and train machine learning models directly on this unconventional photonic quantum computation model, enabling intrinsic parallelism and efficient exploration of large Hilbert spaces. Realizing this potential requires frameworks that connect QML models to photonic and other quantum hardware, enabling consistent evaluation across platforms. By embedding trainable quantum circuits into standard AI pipelines, these frameworks support hybrid quantum–classical workflows suited to the near-term regime, where quantum structure can be explored as a source of representations and inductive biases. Consequently, progress in photonic QML depends on scalable, benchmark-driven experimentation rather than isolated algorithmic proposals.

Within this landscape, quantum software frameworks differ markedly in both their level of abstraction and the computational paradigms they support. Hardware-oriented frameworks, such as `Qiskit` [5] or `Cirq` [6], provide mature ecosystems for superconducting processors. By contrast, `Pulser` [7] targets neutral-atom platforms, while `Perceval` [2] offers state-of-the-art discrete-variable (DV) simulation backends and direct hardware access to Quandela's photonic processors. As for `Strawberry Fields` [8] and `Piquasso` [9], they focus on continuous-variable (CV) paradigms, with `Piquasso` additionally supporting discrete-variable (DV) simulations.

Other software frameworks focus on tools for QML and optimization such as differentiable quantum programming and PyTorch[1] integration, like `PennyLane` [10], `TorchQuantum` [11] and `Qiskit-Torch-Module` [12]. More recently, `DeepQuantum` [1] emerged as a unified platform bridging qubit circuits, photonic qumodes (Fock, Gaussian, Bosonic backends), and measurement-based quantum computing within PyTorch, reporting GPU-accelerated gradient computation an order of magnitude faster than `PennyLane` at scale.

This abundance of platforms results in a rich, but frag-

This research was supported by MITACS Accelerate (projects IT45761 and IT36314), the UFOQO Project, financed by the French State as part of France 2030, the European Union's Horizon Europe research and innovation programme under grant agreement No 101130384 (QUONDENSATE) and the QuantERA programme through the project ResourceQ.

[1]PyTorch, the PyTorch logo and any related marks are trademarks of The Linux Foundation.

mented software landscape. Each framework specialises in a particular layer or paradigm, creating silos where algorithms are not portable without significant conversion effort. Most of the QML literature is built on the gate-based paradigm. However, a growing body of work considers algorithms tailored to hardware, within which photonic QML is gaining traction. A systematic survey found that photonic contributions currently account for ≈6% of QML publications, with most being simulator-based [13]. We confirmed this through independent searches[2].

Recent works demonstrate the potential for photonic QML using a variety of platforms. Gan et al. [14] explore variational algorithms which project data and training parameters to the high-dimensional Fock space using `NLopt` [14], while another work [15] utilizes `DisCoPy` [16] to explore Fock-space encodings for kernel methods. Moreover, additional algorithms, such as quantum reservoir computing implemented with `Perceval` and `Keras` [17], [18], and quantum convolutional neural networks using `QOptCraft` [19], [20] have been studied on photonic platforms. Across this surveyed literature, each of these contributions relies on a different ad-hoc software stack, because no existing framework currently combines efficient simulation, integration with ML workflows, noise models and hardware access.

To address this gap, we present `MerLin`: a framework for photonic QML that builds on `Perceval`'s optimised DV simulation and hardware access while adding native PyTorch and scikit-learn integration for end-to-end differentiable training. `MerLin` adopts a photonic-native model, operating directly in Fock space with parametrized linear-optical circuits whose trainable parameters correspond to physical components (e.g., phase shifters and beam splitters described in Section II-A2). It exposes measurement primitives aligned with photonic hardware and enables gradient-based training through differentiable strong simulation.

Simultaneously, we designed `MerLin` as a benchmarking- and reproduction-first platform, providing a unified, hardware-aware platform for implementing, training, and evaluating photonic QML models. Indeed, there has been a need for rigorous benchmarking in the QML community – called for in [13], [21] – which can be linked to several factors: an extreme heterogeneity in data preprocessing and task formulation in the literature, low code availability preventing independent verification of most results[3], and a preference for single-run metrics rather than multi-dimensional evaluations.

We addressed this need by developing a dedicated reproduction framework, designed to enable systematic and ready-to-use replication of published QML works, including reported claims, performance metrics, and experimental analyses, within a controlled software environment built on top of `MerLin`. As an initial demonstration of this methodology, we reproduced key results from eighteen state-of-the-art photonic and hybrid QML works, spanning expressivity analyses, quantum kernels, reservoir computing, and convolutional architectures. These reproductions validate `MerLin`'s correctness, and provide a starting point for future contributions.

Crucially, the objective of this benchmarking effort is not only to identify positive performance gains, but also to understand their origin, and to disentangle improvements due to data preprocessing, model engineering, or optimization strategies from those arising from genuinely new representational or computational mechanisms.

The `MerLin` framework and the companion reproduction repository are publicly available at https://github.com/merlinquantum/merlin and https://github.com/merlinquantum/reproduced_papers.

[2]arXiv abstract search: 188 photonic QML papers vs. 3,592 QML papers; lens.org yields 6.1%.

[3]Code availability hovers around 27% for gate-based and 43% for photonic QML papers.

II. MerLin architecture

A. Differentiable simulation of Photonic Quantum circuits

1) Essentials of linear optics: In quantum linear optics, we consider an n-photon input state that evolves through an m-mode interferometer made of beam-splitters and phase shifters generating superposition and entanglement across photonic modes. This interferometer, or circuit, thus implements an operation that is described mathematically by a unitary matrix U of size $m \times m$. The state at the output of the interferometer is then measured by photon detectors, which register the number of photons in each mode. We denote the input state as $|\mathbf{s}\rangle = |s_1, \ldots, s_m\rangle$ where each s_i describes the photon occupancy of the i-th optical mode, so that $s_i \in \{0, \ldots n\}$ and $\sum_i s_i = n$. The state $|\mathbf{s}\rangle$ is called a Fock state. In lossless regime, the total number of photons n is preserved through the evolution in the interferometer, so the photon detectors measure output arrangements $\mathbf{s}' = (s'_1, \ldots s'_m)$, with $\sum_i s'_i = n$. Given a choice of n and m, there are $N_{\text{Fock}} = \binom{m+n-1}{n}$ different output states – the dimension of the Fock space – and the probability of measuring each of them is determined by the matrix permanent of submatrices of U, up to normalization factors [22].

2) Strong Linear Optical Simulation: The Strong Linear Optical Simulation (SLOS) framework [23], on which `MerLin` is built, is a tool that performs *strong simulation*: it computes the exact quantum state after evolution through the interferometer, from which all output probabilities can be extracted, in a time complexity of $\mathcal{O}\left(n\binom{m+n-1}{n}\right)$. While the overall complexity remains proportional to the size of the Fock space, SLOS avoids computing a separate matrix permanent for each output configuration by reusing intermediate results. This provides a significant speedup over previously known simulation methods for many practical cases. The trade-off is memory: SLOS requires $\mathcal{O}\left(n\binom{m+n-1}{n}\right)$ space to store the complete state vector, which limits practical use to approximately $n \lesssim 20$ photons on standard hardware [2].

3) Circuit optimization with PyTorch: `MerLin` integrates SLOS into PyTorch [24] enabling gradient-based optimization of parametrized linear optical circuits via automatic differentiation. On photonic hardware, the trainable parameters θ correspond to the phases and reflectivities of the optical

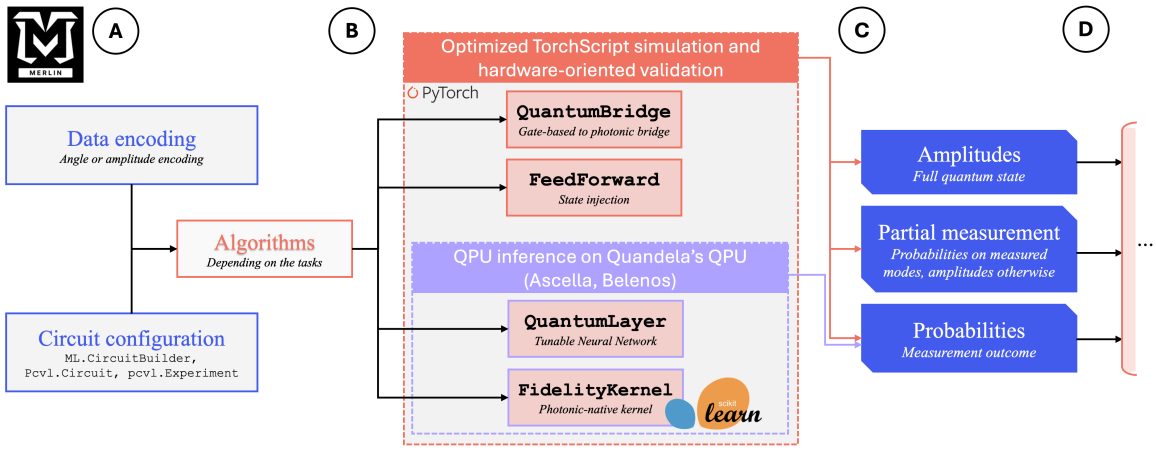


Fig. 1: `MerLin` architecture for photonic QML. **(A)** Classical data encoding and photonic circuit configuration define the quantum model. **(B)** `MerLin` integrates PyTorch-based optimization with photonic-native execution through a logical-to-photonic bridge, differentiable quantum layers, and hardware-oriented simulation, with optional inference on Quandela photonic QPUs. **(C)** Measurement strategies and detectors behaviour expose full quantum states or partially measured observables. **(D)** The resulting amplitudes or probability distributions are returned as classical outputs for downstream machine-learning tasks.

components (phase shifters and beam-splitters), which together define a parametrized unitary describing the circuit $U(\theta)$.

To accelerate simulation, `MerLin` provides TorchScript-compiled SLOS primitives built on a precomputed sparse computation graph. SLOS computes the output state iteratively and builds layer-by-layer transitions between intermediate Fock state configurations within a circuit. Starting from a 1-photon intermediate state, it successively constructs k-photon states for $k = 2, \ldots, n$ by applying transition rules that depend only on the circuit topology. Specifically, it determines which Fock state configurations at layer $k-1$ can contribute to which configurations at layer k. This transition structure is determined entirely by the input state, and is independent of the actual unitary matrix entries. `MerLin` exploits this by prebuilding the sparse transition graph once for a given input state, encoding the valid $k \to k+1$ photon transitions as sparse index mappings. During training, only the unitary-dependent coefficients are recomputed at each forward pass, while the graph structure is reused.

B. The Quantum Layer abstract

`MerLin`'s PyTorch integration is provided through `QuantumLayer`, a `torch.nn.Module` that exposes trainable circuit parameters θ, supports batching, and can be composed with other PyTorch modules. `QuantumLayer` is structured around three core concepts that govern how quantum information is represented and extracted. First, the *measurement strategy* determines the layer output, such as full probability distribution (as obtained from hardware measurements), per-mode photon number expectations, or complex state amplitudes prior to measurement. Second, the *computation space* specifies the Hilbert subspace used to represent the photonic state, ranging from the full Fock space to restricted or encoded spaces, including dual-rail or QLOQ qubit encodings [25]. The measurement model can further be specified through a detector model, which defines how photonic states are converted into classical outcomes, for instance using photon-number-resolving (PNR) or threshold detectors. Threshold detectors register the presence or absence of photons in an optical mode and are commonly used in experimental photonic QPU setups. The last concept is *data encoding*, described in Section II-C. These concepts are illustrated in `Code Block` 1 and jointly decouple circuit evolution, state representation, and measurement semantics within `QuantumLayer`.

C. Data encoding

The encoding of classical data into quantum states remains one of the main bottlenecks in QML, with the potential to negate any potential quantum speedups if not addressed carefully [14], [26]. `MerLin` supports two encoding strategies that are adapted to linear-optical hardware: *angle encoding*, which maps classical features to phase shifts within the circuit, and *amplitude encoding*, which directly initializes the quantum state amplitudes to match the input features.

1) Angle encoding: Schuld et al. [26] demonstrated that VQCs naturally realize a Fourier-like model: the input encoding fixes the accessible frequency spectrum (i.e., the feature basis), while the trainable circuit parameters only control how these fixed features are linearly combined. Gan et al. [14] generalized this framework to linear quantum photonic circuits, revealing a distinctive property of photonic platforms. Consider a parameterized circuit with m modes, where data x is encoded via a single phase shifter $S(x) = e^{ix\hat{n}_1}$ acting on the first mode (Figure 2a with one phase shifter). The n-photon quantum model takes the form:

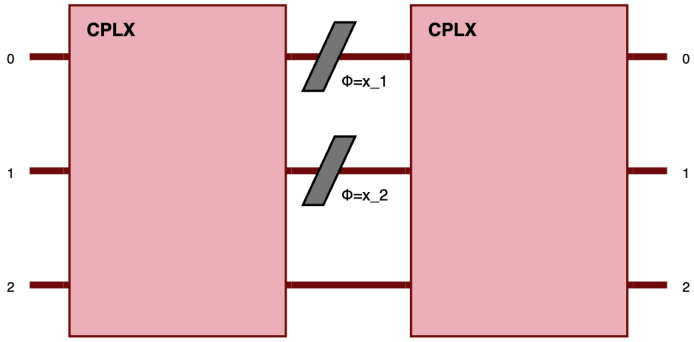
$$f^{(n)}(x, \boldsymbol{\theta}, \boldsymbol{\lambda}) = \sum_{\omega \in \Omega_n} c_\omega(\boldsymbol{\theta}, \boldsymbol{\lambda})\, e^{i\omega x}, \tag{1}$$

where $\boldsymbol{\theta}$ parameterizes the trainable interferometers and $\boldsymbol{\lambda}$ the measurement observable. Crucially, the size of

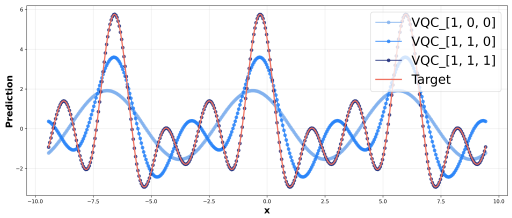
QuantumLayer initialization and training

```
import merlin as ML

builder = ML.CircuitBuilder(n_modes=3)
builder.add_entangling_layer(
    trainable=True, name="W1")
builder.add_angle_encoding(modes=[0, 1],
     name="x_")
builder.add_entangling_layer(
    trainable=True, name="W2")

layer = ML.QuantumLayer(
    builder=builder,
    input_state=[1, 1, 1],
    measurement_strategy =
     ML.MeasurementStrategy.probs(
         computation_space =
             ML.ComputationSpace.FOCK))

optimizer =
     torch.optim.Adam(layer.parameters(),
     lr=0.01)
loss_fn = torch.nn.CrossEntropyLoss()

for epoch in range(n_epochs):
    optimizer.zero_grad()
    output = layer(X_train)
    loss = loss_fn(output, y_train)
    loss.backward()
    optimizer.step()
```

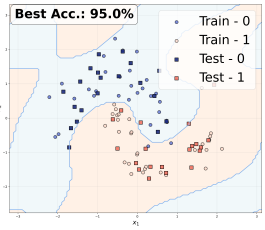
`Code Block` 1: QuantumLayer initialization and training using `MerLin` and a `PyTorch`-native optimization loop



(a) Parameterized linear photonic circuit with $m = 3$ spatial modes and single-phase data encoding. The expectation value under PNR or threshold detection decomposes as $\sum_\omega c_\omega e^{i\omega x}$, where ω depends on the photon number and c_ω on the trainable circuit and observable.



(b) Optimal fits of a degree-three Fourier series $g(x) = \sum_{n=-3}^{3} c_n e^{-inx}$ obtained with a three-mode photonic circuit using PNR detectors for different input Fock states.



(c) Binary classification on the Moon dataset using the three-mode VQC (2a) with input Fock state $|111\rangle$.

Figure 2: Overview of a photonic VQC implemented with `MerLin` and seamlessly integrated into a standard machine learning workflow. `Code Block` 1: PyTorch-native code illustrating the ease of defining, training, and optimizing a `QuantumLayer`. Figure 2a: Three-mode linear photonic circuit where classical data are encoded as phase shifts between two trainable universal interferometers. Figure 2b: The resulting VQC naturally realizes a truncated Fourier series. Figure 2c: Despite its simplicity, the same three-mode VQC introduces sufficient nonlinearity to perform binary classification on the moon dataset.

the frequency spectrum, and therefore the expressivity, increases linearly with the number of input photons n, without requiring additional encoding gates or deeper circuits. The `CircuitBuilder` API provides the `add_angle_encoding` method, which inserts phase shifters at specified modes. Their phases correspond to classical features $\{x_i\}$, as demonstrated in `Code Block` 1 and Figure 2a. A typical circuit choice consists of parametrized interferometers $W^{(i)}$ and an encoding layer $S(x)$ in between:

$$U(x, \boldsymbol{\theta}) = W^{(2)}(\boldsymbol{\theta}_2)\, S(x)\, W^{(1)}(\boldsymbol{\theta}_1). \tag{2}$$

With `MerLin`, we successfully reproduced the results from [14] using this type of circuits: Figure 2b shows the optimal fit of a Fourier series using 3 modes and up to 3 photons, while Figure 2c illustrates the performance of the VQC on binary classification tasks.

2) Amplitude encoding: Amplitude encoding maps a classical data vector $\mathbf{x} \in \mathbb{C}^N$ directly into the amplitudes of a quantum state: $|\mathbf{x}\rangle = \sum_{i=0}^{N-1} x_i\, |i\rangle$ where $\|\mathbf{x}\|^2 = 1$, and $\{|i\rangle\}$ denotes the Fock basis.

In `MerLin`, amplitude encoding is used when the input of the algorithm is a complex tensor or a `StateVector`. This encoding strategy is suited to scenarios where the quantum state is produced by an upstream process (such as another photonic circuit) rather than derived from raw classical features.

D. Hardware-aware design

Although `MerLin` is simulator-first, it is not hardware-agnostic. Its design reflects the constraints, capabilities, and evolution of photonic quantum hardware, enabling algorithm–hardware co-design. At the circuit level, `MerLin` operates directly on the native photonic components (Section II-A), and users can select detector models and observables that match realistic readout strategies (Section II-B). This hardware awareness extends to execution through the `MerlinProcessor` abstraction, which connects quantum layers to cloud-accessible photonic processors and allows parts of a hybrid model to be offloaded to real hardware, as illustrated in `Code Block` 4. This connector abstracts hardware-specific constraints such as latency, shot-based sampling, and limited parallelism, while preserving compatibility with modern machine learning workflows.

Beyond current accessible hardware, `MerLin` is designed to support the exploration of emerging quantum computing

QuantumBridge Usage

```
# Bridge: 2 qubits -> 2 photons, 4 modes (dual-rail)
bridge = ML.QuantumBridge(
    n_photons=2, n_modes=4,
    qubit_groups=[1, 1],
    computation_space=ML.ComputationSpace.UNBUNCHED
)

# Compose with upstream qubit circuit and downstream
    photonic layer
model = torch.nn.Sequential(qubit_circuit, bridge,
    merlin_layer)
```

`Code Block` 2: Quantum Bridge between gate-based circuit and photonic circuit, enabling interoperability between gate-based and photonic frameworks.

paradigms through its modular architecture. As an illustrative example, `MerLin` supports photonic quantum memristors: a memristor is a circuit element whose response depends on the history of current flow, mimicking synaptic plasticity. Its photonic analogue can be realized using a Mach-Zehnder interferometer, whose internal phases are updated based on detection statistics of a feedback loop. This mechanism introduces memory and nonlinearity, which are useful for tasks such as time-series prediction [27].

Finally, `MerLin` acts as a bridge between computational modalities. Whereas qubit-based platforms operate in a 2^n-dimensional computational basis, photonic processors naturally evolve in Fock space (Section II-A). The `QuantumBridge` abstraction (see `Code Block` 2) provides a differentiable interface between these representations, enabling hybrid workflows across both paradigms. Beyond standard dual-rail encodings, `MerLin` supports general qubit-qudit mappings following the QLOQ scheme [25], reducing photonic resource requirements and enabling systematic cross-modality comparisons.

Overall, this hardware-aware design positions `MerLin` as a co-design tool rather than a purely abstract simulator: insights obtained from scalable (within simulable regimes), differentiable simulation inform both algorithmic development and the design of future photonic quantum processors.

E. Complexity and simulability

As mentioned earlier, simulating linear-optical quantum circuits is computationally challenging. While this observation underpins hardness arguments in photonic quantum computing, a growing body of work has shown that simulability can extend to larger-scale systems when additional structure or noise is present, as exemplified by results in Gaussian boson sampling and noisy linear-optical models. Such regimes, while classically tractable, remain highly relevant from a modelling and algorithmic perspective. `MerLin` deliberately embraces this viewpoint by focusing on strong simulation in controlled regimes where the full quantum state can still be computed exactly, or efficiently approximated. This enables access to complete probability distributions, exact expectation values, and analytic gradients via automatic differentiation capabilities.

Fidelity kernel construction

```
kernel = ML.FidelityKernel.simple(
    input_size=4, n_modes=6, n_photons=4)

K_train = kernel(X_train)
K_test  = kernel(X_test, X_train)
clf = SVC(kernel="precomputed").fit(K_train, y_train)
```

`Code Block` 3: Application of the FidelityKernel to a given training set.

From a methodological standpoint, the primary objective is discovery. This involves identifying which architectures, encodings, and training strategies lead to meaningful learning behavior. Considerations of simulability versus hardness are therefore treated as tools for interpreting and contextualizing these findings across classical and quantum regimes. In this sense, pushing simulation as far as possible is a deliberate choice that supports systematic investigation and informs subsequent studies in less tractable, hardware-based settings.

III. MerLin as a benchmark-driven reproduction platform

In this section, we summarize the reproduced papers, categorized either by task or method; a concise summary of results for each paper is provided in Table I including confirmations, refinements, and observations on previously reported claims. To ensure proper benchmarking, models are ran several times and statistically sound results are provided, so that any user executing our code should observe results within the reported variance.

A. Kernel methods

Kernel methods compare data points through a similarity function $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$, implicitly embedding data into a feature space where linear models can capture non-linear structure. Quantum kernels extend this idea by encoding classical inputs into quantum states. In linear optics, data-dependent unitaries $U(x)$ act on Fock states to produce embeddings $|\phi(x)\rangle = U(x)\,|s\rangle$, and a fidelity-based kernel is then given by $k(x_1, x_2) = |\langle \phi(x_1) \mid \phi(x_2) \rangle|^2$.

`MerLin` provides a scikit-learn–compatible implementation of fidelity kernels depicted in `Code Block` 3. It also reproduces distance-based quantum kernels based on inner-product estimation [28]. While originally demonstrated on qubit platforms using gates that emulate beam-splitter dynamics, these operations are native to photonic hardware. This reproduction validates `MerLin` and highlights its role as a bridge between abstract quantum kernel constructions and realistic photonic implementations.

B. Recurrent Architectures for Sequential Data

Time series prediction is a fundamental paradigm in ML which involves learning temporal dependencies from sequential data to predict future values, traditionally addressed by recurrent architectures. While recent studies reveal that current QML models often struggle to outperform simple classical

Paper	Reproduction
Fock State-enhanced expressivity of Quantum Machine Learning Models [14]	We confirmed that, under the proposed encoding scheme, increasing the number of photons increases the expressivity of the variational quantum circuit.
Experimental quantum-enhanced kernel-based machine learning on a photonic processor [15]	We confirmed that accuracy improves with both training-set size and geometric difference.
Nearest Centroid Classification on a Trapped Ion Quantum Computer [28]	We reproduced this photon-native algorithm and obtained accuracies consistent with both the source paper and the classical baseline on the three considered datasets.
Experimental data re-uploading with provable enhanced learning capabilities [29]	We confirmed that the fully quantum data reuploading model is both accurate and resource-efficient for binary classification on the four evaluated datasets, and that its expressivity increases with the number of reuploading layers.
Quantum Long Short-Term Memory [30]	Our MerLin-based photonic QLSTM matched the original gate-based QLSTM on function-fitting tasks, although the classical LSTM weaknesses reported in the source paper were less pronounced in our experiments.
Quantum Recurrent Neural Networks for Sequential Learning [31]	We validated the partial measurement scheme in `MerLin` for temporal data prediction.
Photonic Quantum Convolutional Neural Networks with Adaptive State Injection [19]	By optimizing hyperparameters, we improved the reported binary classification accuracies from $92.7 \pm 2.1\%$ to $98.2 \pm 2.2\%$ on Custom BAS and from $93.1 \pm 3.6\%$ to $98.8 \pm 1.0\%$ on MNIST (0 vs 1).
Quantum Convolutional Neural Networks for classical data classification [32]	We reproduced the reported advantage of QCNNs over CNNs under similar training budgets on a photonic architecture, but found that classical models remain more parameter-efficient. However, in photonic implementations this metric is of limited relevance, since the number of trainable parameters is inherently bounded by the number of phase shifters in the interferometer, while classical models saturate at lower absolute accuracy.
Quantum optical reservoir computing powered by boson sampling [17]	We reproduced the reported scalability, with performance improving as the number of modes increases, and confirmed that the model can be executed on QPU hardware.
Quantum Relational Knowledge Distillation [33]	We confirmed that quantum relational knowledge distillation yields a larger improvement in student performance than its classical counterpart.
Distributed Quantum Neural Networks on Distributed Photonic Quantum Computing [34]	We reproduced the parameter-efficiency gains obtained by using Boson Sampling to approximate CNN parameters, with training accuracy within approximately 2%, test accuracy within approximately 4%, and a fourfold reduction in required epochs, while also observing faster training with ADAM than with COBYLA; however, our ablation results differed from those in the source paper.
Quantum Self-Supervised Learning [35]	The `MerLin` implementation outperformed the Qiskit version in both speed and accuracy, reaching 49.2% accuracy on the first five CIFAR-10 classes after two epochs with eight modes, compared with 48.4% for Qiskit and 48.1% for the classical baseline.
Transfer learning in hybrid quantum-classical neural networks [36]	We obtained comparable accuracy across transfer settings and observed limited sensitivity to photon count.
Photonic quantum generative adversarial networks for classical data [37]	The `MerLin` implementation reproduced the original Perceval SSIM results while reducing training time by up to a factor of 15.
Computational Advantage in Hybrid Quantum Neural Networks: Myth or Reality? [38]	We confirmed that the HQNN achieves at least 90% accuracy on the noisy spiral dataset with fewer parameters than a classical neural network across feature dimensions ranging from 5 to 60.
Quantum Large-Language Model Fine-Tuning [39]	Whereas the original paper reports up to a 3.14% accuracy gain for the quantum-enhanced model, our reproduction found that the best quantum and classical models all reached approximately 89% accuracy, with no clear separation between them.
Quantum Adversarial Machine Learning [40]	We confirmed that quantum classifiers remain vulnerable to both direct and transferred adversarial attacks, while adversarial training substantially improves robustness; on MNIST, we obtained approximately 98% clean accuracy, 15% BIM adversarial accuracy at $\epsilon = 0.1$, and 95% adversarial accuracy after adversarial training.
Experimental neuromorphic computing based on quantum memristor [27]	We reproduced the main results and confirmed that adding a quantum memristor enhances the nonlinearity of the quantum reservoir, thereby improving performance on time-series tasks.

TABLE I: Reproduced papers used to validate MerLin's performance and utility. The main result of each reproduction is summarized here. Code and detailed results are available in the MerLin/reproduced_papers repository.

counterparts of comparable complexity for time series prediction [41], they also highlight the potential of hybrid quantum-classical architectures when carefully designed. In particular, two works have applied variational quantum architectures to sequential learning. [30] introduced the QLSTM, a hybrid model that replaces classical linear transformations within LSTM cells with variational quantum circuits. Subsequently, [31] proposed a hardware-efficient Quantum Recurrent Neural Network (QRNN) architecture featuring quantum recurrent blocks that preserve quantum information across time steps. We successfully implemented both models using `MerLin`, validating its use for temporal processing tasks, and providing tools for novel algorithmic development.

C. Convolutional Neural Network

Convolutional Neural Networks (CNN) are foundational models for computer vision, owing to their locality, translation equivariance, and parameter sharing across feature maps [42]. Recently, [43] proposed a Subspace Preserving Quantum Convolutional Neural Network (QCNN) with a reported polynomial speedup over deep classical CNNs. This architecture was later adapted to photonic hardware [19] using adaptive state injection, which implements pooling while preserving a fixed photon number and reducing the effective Fock-space dimension. We reproduced this photonic QCNN

within MerLin using a density-matrix backend and, through a systematic hyperparameter study, improved the reported test accuracy on two datasets by 4–5%, thereby establishing a stronger baseline for future work. Beyond reproduction, MerLin natively supports amplitude-encoding with partial measurements. Combined with adaptive state injection, this enables a modular and transparent construction of photonic QCNNs using QuantumLayer, highlighting MerLin's suitability for structured photonic QML models.

D. Reservoir Computing

The reservoir computing principle, introduced for processing temporal data [44], [45], harnesses the complex dynamics of physical systems to map input data into a high-dimensional feature space. A linear probe takes this rich embedding as input. In the context of computer vision, popular models rely on a trained backbone, whether supervised [42] or self-supervised [46]. By contrast, the reservoir is used as a fixed, untrained black-box system. Reservoir computing was adapted to linear optics in [17], [18]. Ref. [17] considers MNIST-like datasets. The circuit chosen is similar to the one in Figure 2a. We successfully reproduced the key finding of this work: the accuracy scales with the size of the Fock space. Additionally, using MerLin, these experiments can be conducted both in simulation and on physical hardware through the MerlinProcessor, see Code Block 4.

E. Training paradigms for QNN

Recent advances in QML have introduced several promising training paradigms that leverage quantum resources to enhance classical learning pipelines. Quantum Relational Knowledge Distillation (QRKD) [33] aligns teacher-student representations through quantum kernel functions, enabling geometric regularization in an exponentially large feature space while retaining classical inference. Quantum Self-Supervised Learning (qSSL) [35], originally implemented using Qiskit, combines contrastive learning with quantum neural networks to capture complex visual correlations in high-dimensional quantum feature spaces. The Distributed Quantum Neural Network (DQNN) framework [34] uses photonic circuits as hypernetworks to generate classical network weights.

Using MerLin, we successfully reproduced the key results from these works. Our qSSL implementation exhibited improved scaling relative to the original Qiskit version, the DQNN reproduction achieved comparable MNIST accuracy, and QRKD consistently reduced the teacher–student performance gap relative to classical relational knowledge distillation across vision and language tasks.

F. Image generation

Quantum circuits have been investigated as models for generative learning, notably for image generation. Quantum generative adversarial networks (QGANs) were among the first generative variational quantum algorithms to be designed [47], and a photonic QGAN was later proposed [37]. The model is trained on the MNIST dataset in reduced dimension, using a patch-based approach, where each generator produces a patch of the total image. The generators are linear optical circuits whose output distribution is mapped to pixels, and the discriminator is a classical neural network. Using MerLin, we adapted the original Perceval code and simplified the training pipeline, switching from an SPSA-based optimization [48] to a faster PyTorch stochastic gradient descent optimizer.

Reservoir computation on Belenos QPU

```
# 1) Create the Perceval RemoteProcessor (token needed)
rp = pcvl.RemoteProcessor("qpu:belenos")
# 2) Wrap it with MerlinProcessor
proc = ML.MerlinProcessor(
    rp,
    microbatch_size=32, # batch chunk size/cloud call
    timeout=3600.0)     # default wall-time/forward
# 3) Set your QuantumLayer to eval mode
reservoir.eval()
# 4) Run remotely
y = proc.forward(layer, X, nsample=5000)   # synchronous
```

Code Block 4: Reservoir computation running on remote Belenos QPU using MerLin.

G. Benchmark Tasks for Model Expressivity and Robustness

We evaluate MerLin on three challenging tasks. First, we replicate the quantum LLM fine-tuning results from [39] on SST2 sentiment analysis; however, the binary classification setting appears too simple to draw definitive conclusions about quantum utility, a limitation consistent with concerns raised by Bowles et al. [21]. Second, following [38], we confirm that VQCs require fewer parameters than classical networks to achieve $\geq 90\%$ accuracy under increasing feature complexity. Third, investigating adversarial robustness [40], we find that amplitude encoding is highly vulnerable to small perturbations ϵ, while angle encoding proves substantially more robust. Combining amplitude encoding with dimensionality reduction offers moderate protection but sacrifices the representational benefits of full amplitude encoding, highlighting encoding strategy as a critical design choice for adversarial settings.

IV. Discussion & Conclusion

In this work, we introduced MerLin, a quantum software platform designed for large-scale, simulation-based exploration of hybrid quantum–classical models while remaining explicitly hardware-aware. MerLin enables systematic investigation of quantum learning architectures within familiar PyTorch-based environments.

We validated the framework by reproducing a range of results from the QML literature. In doing so, we observed that implementations ported from Perceval to MerLin consistently benefited from substantial improvements in simulation speed, reaching up to several order-of-magnitude speedups in some cases. More importantly, for algorithms originally formulated in a gate-based setting and reformulated in a photon-native representation, we found that learning performance and qualitative behavior remained very similar. This indicates that photon-native models may effectively substitute gate-based models in many hybrid architectures, opening up a practical path both for translating existing gate-based QML literature to

photonic platforms and for exploring cross-modality models within a unified framework.

Beyond individual reproductions, `MerLin` is designed to serve as a discovery engine for QML by promoting a benchmark-driven, empirical methodology inspired by modern machine learning practice. Through its open design and simplified reproduction framework, `MerLin` encourages the community to systematically benchmark new and existing results, study learning dynamics, and lower the entry barrier for both classical and QML practitioners.

Finally, within the context of photonic quantum computing, `MerLin`'s hardware-aware design and support for hybrid execution position it as a co-design tool: insights obtained from large-scale simulation and benchmarking can directly inform both algorithmic development and the design choices of future photonic quantum processors. We believe this tight feedback loop between software, algorithms, and hardware will be essential for identifying meaningful near-term advantages in quantum machine learning.

Acknowledgements

The authors thank Pierre-Emmanuel Emeriau for helpful discussions and insightful feedback during the development of `MerLin`, Hugo Izadi and Ankit Sharma for their contribution to the reproduced papers.

References

[1] J.-J. He *et al.*, "DeepQuantum: A PyTorch-based Software Platform for Quantum Machine Learning and Photonic Quantum Computing," Dec. 2025, arXiv:2512.18995.

[2] N. Heurtel *et al.*, "Perceval: A Software Platform for Discrete Variable Photonic Quantum Computing," *Quantum*, vol. 7, p. 931, Feb. 2023.

[3] N. Maring *et al.*, "A versatile single-photon-based quantum computing platform," *Nature Photonics*, vol. 18, no. 6, pp. 603–609, Jun. 2024.

[4] A. Soret *et al.*, "Quantum energetic advantage before computational advantage in boson sampling," *arXiv preprint arXiv:2601.08068*, 2026.

[5] A. Javadi-Abhari *et al.*, "Quantum computing with qiskit," 2024.

[6] Cirq Developers, "Cirq," https://github.com/quantumlib/Cirq, 2021.

[7] H. Silvério *et al.*, "Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays," *Quantum*, vol. 6, p. 629, 2022.

[8] N. Killoran *et al.*, "Strawberry Fields: A Software Platform for Photonic Quantum Computing," *Quantum*, vol. 3, p. 129, Mar. 2019, arXiv:1804.03159.

[9] Z. Kolarovszki *et al.*, "Piquasso: A Photonic Quantum Computer Simulation Software Platform," *Quantum*, vol. 9, p. 1708, Apr. 2025, arXiv:2403.04006.

[10] V. Bergholm *et al.*, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," 2018, arXiv:1811.04968.

[11] H. Wang *et al.*, "QuantumNAS: Noise-Adaptive Search for Robust Quantum Circuits," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Apr. 2022, pp. 692–708, arXiv:2107.10845.

[12] N. Meyer *et al.*, "Qiskit-torch-module: Fast prototyping of quantum neural networks," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1. IEEE, 2024, pp. 817–823.

[13] C. Notton *et al.*, "Establishing baselines for photonic quantum machine learning: Insights from an open, collaborative initiative," 2025.

[14] B. Y. Gan *et al.*, "Fock state-enhanced expressivity of quantum machine learning models," *EPJ Quantum Technology*, vol. 9, no. 1, p. 16, 2022.

[15] Z. Yin *et al.*, "Experimental quantum-enhanced kernel-based machine learning on a photonic processor," *Nature Photonics*, vol. 19, no. 9, pp. 1020–1027, 2025.

[16] G. De Felice *et al.*, "Discopy: Monoidal categories in python," *arXiv preprint arXiv:2005.02975*, 2020.

[17] A. Sakurai *et al.*, "Quantum optical reservoir computing powered by boson sampling," *Optica Quantum*, vol. 3, no. 3, pp. 238–245, Jun. 2025.

[18] M. Rambach *et al.*, "Photonic Quantum-Accelerated Machine Learning," Dec. 2025, arXiv:2512.08318.

[19] L. Monbroussou *et al.*, "Photonic quantum convolutional neural networks with adaptive state injection," 2025.

[20] D. G. Aguado *et al.*, "Qoptcraft: A python package for the design and study of linear optical quantum systems," *Computer Physics Communications*, vol. 282, p. 108511, 2023.

[21] J. Bowles *et al.*, "Better than classical? the subtle art of benchmarking quantum machine learning models," *arXiv preprint arXiv:2403.07059*, 2024.

[22] R. Mezher *et al.*, "Solving graph problems with single-photons and linear optics," Aug. 2023, arXiv:2301.09594 [quant-ph].

[23] N. Heurtel *et al.*, "Strong simulation of linear optical processes," *Computer Physics Communications*, vol. 291, p. 108848, Oct. 2023.

[24] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[25] L. Lysaght *et al.*, "Quantum circuit compression using qubit logic on qudits," 2024.

[26] M. Schuld *et al.*, "The effect of data encoding on the expressive power of variational quantum machine learning models," *Physical Review A*, vol. 103, no. 3, p. 032430, Mar. 2021, arXiv:2008.08605.

[27] M. Selimović *et al.*, "Experimental neuromorphic computing based on quantum memristor," 2025.

[28] S. Johri *et al.*, "Nearest centroid classification on a trapped ion quantum computer," p. 122, 2021.

[29] M. F. Mauser *et al.*, "Experimental data re-uploading with provable enhanced learning capabilities," 2025.

[30] S. Y.-C. Chen *et al.*, "Quantum Long Short-Term Memory," Sep. 2020, arXiv:2009.01783.

[31] Y. Li *et al.*, "Quantum Recurrent Neural Networks for Sequential Learning," Feb. 2023, arXiv:2302.03244.

[32] T. Hur *et al.*, "Quantum convolutional neural network for classical data classification," *Quantum Machine Intelligence*, vol. 4, no. 1, p. 3, Jun. 2022, arXiv:2108.00661.

[33] C.-Y. Liu *et al.*, "Quantum Relational Knowledge Distillation," Aug. 2025, arXiv:2508.13054.

[34] K.-C. Chen *et al.*, "Distributed quantum neural networks on distributed photonic quantum computing," IEEE, pp. 1477–1488, 2025.

[35] B. Jaderberg *et al.*, "Quantum Self-Supervised Learning," 2021.

[36] A. Mari *et al.*, "Transfer learning in hybrid classical-quantum neural networks," *Quantum*, vol. 4, p. 340, 2020.

[37] T. Sedrakyan *et al.*, "Photonic quantum generative adversarial networks for classical data," *Optica Quantum*, vol. 2, no. 6, pp. 458–467, 2024.

[38] M. Kashif *et al.*, "Computational Advantage in Hybrid Quantum Neural Networks: Myth or Reality?" Dec. 2024, arXiv:2412.04991.

[39] S. H. Kim *et al.*, "Quantum Large Language Model Fine-Tuning," Apr. 2025, arXiv:2504.08732.

[40] S. Lu *et al.*, "Quantum adversarial machine learning," *Physical Review Research*, vol. 2, no. 3, p. 033212, 2020.

[41] T. Fellner *et al.*, "Quantum vs. classical: A comprehensive benchmark study for predicting time series with variational quantum machine learning," *Machine Learning: Science and Technology*, vol. 7, no. 1, p. 010501, 2026.

[42] K. He *et al.*, "Deep residual learning for image recognition," pp. 770–778, 2016.

[43] L. Monbroussou *et al.*, "Subspace preserving quantum convolutional neural network architectures," *Quantum Science and Technology*, vol. 10, no. 2, p. 025050, Mar. 2025.

[44] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," *GMD-Report 148, German National Research Institute for Computer Science*, 01 2001.

[45] W. Maass *et al.*, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, pp. 2531–2560, 11 2002.

[46] M. Oquab *et al.*, "Dinov2: Learning robust visual features without supervision," 2024.

[47] P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, no. 1, jul 2018.

[48] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins apl technical digest*, vol. 19, no. 4, pp. 482–492, 1998.