QCNN for jet image classification on JetNet Dataset

Silvia Sonnoli Lorenzo Jacopo Pileggi

1 Introduzione

Scopo di questa analisi è di estendere i risultati sperimentali ottenuti in [1], in cui viene analizzato l'impiego di un'architettura QCNN [2] per effettuare image classification in ambito di fisica delle alte energie (HEP), per un task di jet tagging. In particolare, l'obiettivo è riuscire a distinguere immagini di jet generati da eventi di top quark da quelli invece di processi QCD di fondo. Nel corso di questa analisi sono state testate diverse architetture, variando il numero di qubit e la batch size del training. Le performance delle diverse configurazioni testate sono state confrontate con quelle di una analoga architettura CNN standard, adattando il numero di pesi per renderlo il più vicino possibile a quello del rivale quantistico. I modelli quantistici riprodotti in questa analisi sono stati implementati tramite la libreria `qiskit` utilizzando un simulatore noiseless.

2 JetNet Dataset

Il dataset utilizzato in [1] e in questo lavoro è il Top Quark Tagging Reference Dataset [3], accessibile tramite la libreria `jetnet` [4], che mette a disposizione una comoda interfaccia python per accedere a questo e altri dataset di riferimento in ambito HEP, come JetNet e Quark-Gluon Tagging, più altre funzionalità come metriche, distanze e funzioni di loss relative a tali dati. Il TopTagging Dataset consta di 2 milioni di eventi simulati tramite metodi Montecarlo, ripartiti fra training (1,2 milioni), validation e test set (400k ciascuno), per un totale di circa 1,7 GB di dati. Ogni dato consta del quadrimpulso del jet relativo al dato evento, più l'insieme dei quadrimpulsi, ordinato in senso decrescente in base al modulo di questi, delle 200 componenti più rilevanti del jet, con zero-padding per jet aventi meno di 200 componenti. In [1], sono stati selezionati 2000 dati per le analisi, ripartiti in 1600 di training e 400 di test set; in questo lavoro è stato scelto, per motivi computazionali, di ridurre ulteriormente il dataset di un decimo, con 160 dati di training e 40 di test.

3 Preprocessing

A partire dai quadrimpulsi dei singoli jet e delle loro componenti, sono state formate delle immagini che riflettano la distribuzione spaziale dell'energia dei componenti del jet.

3.1 Formazione delle immagini dei jet

Per ottenere le "immagini" dei jet, è stata utilizzata la seguente procedura. Per prima cosa, è stato riscalato il quadrimpulso globale dei jet, di modo che avessero tutti la stessa massa invariante m_0, fissata a 60 GeV [5]. Una volta fatto questo, le componenti di ogni jet sono state trasformate tramite un boost di Lorentz, definito di modo da avere stessa direzione spaziale dell'impulso $\vec{P}$ di ogni jet, ma in cui quest'ultimo avesse energia E_0 fissata e uguale per ogni jet (così che $\gamma_0 = \frac{E_0}{m_0}$); nella fattispecie, è stato scelto $E_0 = 600$ GeV, così che $\gamma_0 = 10$. In particolare, γ_0 deve essere tale che, dati E'_J, $|\vec{P}'_J|$ energia e modulo dell'impulso spaziale dei jet riscalati prima del boost, e E_0, $|\vec{P}_0|$ dopo (con $|\vec{P}_0|^2 = E_0^2 - m_0^2$) [5], $\gamma_0 = \frac{1}{m_0^2}(E'_J E_0 - |\vec{P}'_J||\vec{P}_0|)$, mentre per quanto riguarda $\hat{\beta}_J$, la direzione spaziale del boost, avremo $\hat{\beta}_J = \frac{\vec{P}'_J}{|\vec{P}'_J|}$ se $E'_j > E_0$, oppure $\hat{\beta}_J = -\frac{\vec{P}'_J}{|\vec{P}'_J|}$ se $E'_j < E_0$. La matrice del boost avrà perciò la seguente forma:

$$\Lambda(\beta_{\mathbf{J}}) = \begin{pmatrix} \gamma & -\gamma\beta_{\mathbf{J}}{}^T \\ -\gamma\beta_{\mathbf{J}} & \mathbf{I} + (\gamma - 1)(\hat{\beta}_{\mathbf{J}} \otimes \hat{\beta}_{\mathbf{J}}) \end{pmatrix} \tag{1}$$

Il passo successivo per creare le immagini è rappresentato dalla definizione di una base di Gram-Schmidt per ciascun jet, su cui proiettare gli impulsi spaziali delle singole componenti boostate. Le 3 componenti di tale base vengono definite prendendo in considerazione i 3-impulsi maggiori fra le componenti del jet (che chiameremo $\vec{p}_1$, $\vec{p}_2$ e $\vec{p}_3$) e definendo $\vec{P}_J = \vec{p}_1 + \vec{p}_2 + \vec{p}_3$. La terna $\{\vec{P}_J, \vec{p}_1, \vec{p}_2\}$ costituisce pertanto un insieme di vettori linearmente indipendenti, che può dunque essere usata per costruire una base ortonormale di Gram-Schmidt. A questo punto possiamo scegliere i 3 vettori $\{\hat{\epsilon}_1, \hat{\epsilon}_2, \hat{\epsilon}_3\}$ di tale base come:

$$\hat{\epsilon}_1 = \frac{\vec{P}'_J}{|\vec{P}'_J|} \tag{2}$$

$$\hat{\epsilon}_2 = \frac{\vec{p}_1 - (\hat{\epsilon}_1 \cdot \vec{p}_1)\vec{p}_1}{|\vec{p}_1 - (\hat{\epsilon}_1 \cdot \vec{p}_1)\vec{p}_1|} \tag{3}$$

$$\hat{\epsilon}_3 = \frac{\vec{p}_2 - (\hat{\epsilon}_1 \cdot \vec{p}_2)\vec{p}_2 - (\hat{\epsilon}_2 \cdot \vec{p}_2)\vec{p}_2}{|\vec{p}_2 - (\hat{\epsilon}_1 \cdot \vec{p}_2)\vec{p}_2 - (\hat{\epsilon}_2 \cdot \vec{p}_2)\vec{p}_2|} \tag{4}$$

Con questa scelta di base, $\hat{\epsilon}_1$ rappresenta pertanto la direzione approssimativa del jet, così che le altre 2 direzioni $\hat{\epsilon}_2$, $\hat{\epsilon}_3$ definiscono una sorta di piano trasverso al boost, che utilizzeremo perciò per formare le immagini dei jet. A questo punto, per ogni particella di un jet, con quadrimpulso $(p_i^0, \vec{p}_i)$, definiamo le

coordinate nell'immagine del boost (X_i, Y_i), tali che:

$$X_i = \frac{\vec{p_i} \cdot \hat{\epsilon}_2}{p_i^0} \qquad\qquad Y_i = \frac{\vec{p_i} \cdot \hat{\epsilon}_3}{p_i^0} \tag{5}$$

Osserviamo che, facendo un'approssimazione ultrarelativistica dei quadrimpulsi, abbiamo che $(p_i^0)^2 = |\vec{p_i}|^2 = (\vec{p_i} \cdot \hat{\epsilon}_1)^2 + (\vec{p_i} \cdot \hat{\epsilon}_2)^2 + (\vec{p_i} \cdot \hat{\epsilon}_3)^2$, per la scomposizione di Gram-Schmidt, il che vuol dire che le due coordinate (X_i, Y_i) avranno sempre valore compreso fra [-1, 1]. A questo punto possiamo discretizzare questo range di valori $[-1, 1] \otimes [-1, 1]$ in $n \times n$ bin (n in questa analisi è stato fissato a 28), definendo pertanto un istogramma bidimensionale. Ciascuna particella popolerà il bin di tale istogramma corrispondente alle proprie coordinate (X_i, Y_i), contribuendo con peso pari a $\omega_i = \frac{p_i^0}{E_0}$. Il risultato finale sarà un istogramma in cui ogni bin indicherà quanta parte dell'energia del jet sia distribuita fra le particelle aventi "coordinate" in quell'intervallo, e dunque un determinato range di valori nelle componenti del boost spaziale.

3.2 PCA

Una volta create queste immagini, è stata operata una riduzione della dimensionalità tramite Principal Component Analysis (PCA), per rendere il task più fattibile dal punto di vista computazionale, data la difficoltà nel fare simulazioni con un numero alto di qubit. In questo lavoro, la dimensionalità originale 28×28 è stata ridotta prima a 2×2 (per architetture a 4 qubit), e poi a 2×3 (architetture a 6 qubit). Infine, i valori di ogni componente principale è stato riscalato per rientrare nell'intervallo $[0, \frac{\pi}{2}]$, di modo da renderli più gestibili dalle varie porte quantistiche, che operano funzioni trigonometriche.

4 Modelli

Le architetture QCNN che sono state impiegate constano di 3 tipi fondamentali di sottocircuiti: i circuiti di encoding, quelli di convoluzione e quelli di pooling.

Per quanto riguarda i circuiti di encoding, in [1] sono stati testati 4 diversi tipi: Tensor Product Encoding (TPE), Hardware Efficient Encoding (HEE) ad 1 e 2 layer, e Classically Hard Encoding (CHE) [6], le cui strutture, per un circuito a 4 qubit, hanno la forma visibile in figura [1]. In questo lavoro è stato utilizzato come unico circuito di encoding TPE.

Come circuti di convoluzione, sono stati impiegati due tipi di circuiti, mostrati in figura [2], corrispondenti rispettivamente a delle trasformazioni unitarie $SO(4)$ ed $SU(4)$; a livello di parametri, il primo consta di 6 parametri, mentre il secondo di 15. Sempre in [2] è mostrato il circuito di pooling utilizzato, avente 9 parametri. Nei suddetti circuiti, sono definite le due rotazioni $R(\alpha, \beta, \gamma) = R_Z(\gamma)R_Y(\beta)R_Z(\alpha)$ e $R'(\alpha, \beta, \gamma) = R_Z(\gamma)R_X(\beta)R_Z(\alpha)$.

La predizione finale del modello è ottenuta misurando un'osservabile PauliZ sul qubit superstite del circuito.

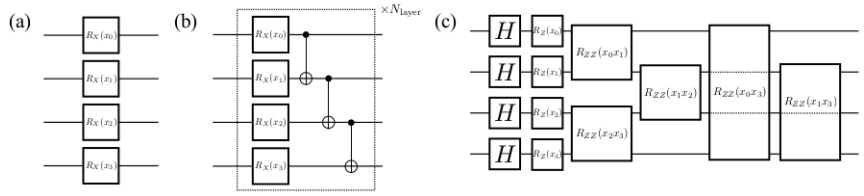

Figure 1: Circuiti di encoding: TPE (a), HEE$\times N_{layer}$ (b) e CHE (c).

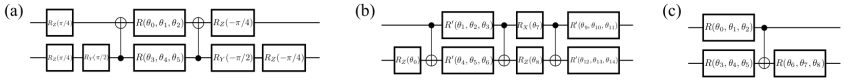

Figure 2: Circuiti di convoluzione e di pooling: $SO(4)$ (a), $SU(4)$ (b) e circuito di pooling (c).

4.1 QCNN per immagini 2×2

Per il caso di immagini ridotte 2×2, è stato riprodotto il modello presentato in [1], mostrato in figura [3]. Come si può vedere, il circuito consta di due layer di convoluzione e due di pooling. Considerando il numero di parametri per i vari tipi di circuiti di convoluzione, avremo che, usando la convoluzione $SO(4)$, il numero totale di parametri del modello è pari a 51, mentre usando $SU(4)$, tale numero sale a 87.

4.2 QCNN per immagini 2×3

L'architettura presente in [1] è stata poi generalizzata di modo da poter essere compatibile con un numero arbitrario di qubit, così da poter essere applicata a immagini di dimensione diversa da 2×2. Un problema che sorge nel generalizzare

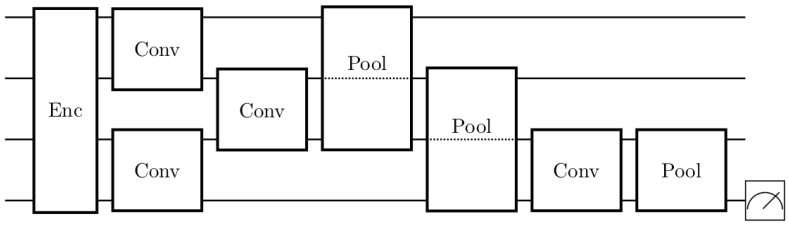

Figure 3: Circuito a 4 qubit usato in [1] per immagini 2×2.

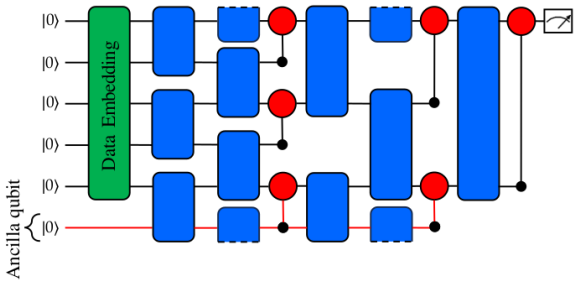

Figure 4: Esempio di QCNN a 5 qubit più ancilla.

questo tipo di architetture è quello di gestire dei layer con un numero dispari di qubit. Infatti, da una parte, nel fare la convoluzione vi sarà sempre almeno un qubit spaiato, e dall'altra, ciò rende non ovvio per il layer di pooling scegliere quali fare sopravvivere, dovendo dimezzare in output il numero di qubit da usare, a meno che il numero totale di qubit non sia una potenza intera di 2. Per ovviare a questa complicazione strutturale, è stato seguito lo schema adottato in [7], che prevede l'impiego nel modello di un qubit "ancillare": il senso è che, per i layer con un numero pari di qubit, questo venga ignorato dalle operazioni, mentre in quelli con un numero dispari, venga coinvolto insieme agli altri per essere accoppiato con il qubit che altrimenti resterebbe spaiato. Per il resto, i layer di convoluzione sono formati da due gruppi di circuiti paralleli di convoluzione, che coinvolgono due qubit contigui; fra un gruppo e l'altro, i circuiti sono sfasati fra di loro di un qubit. Un esempio di architettura di questo tipo, con 5+1 qubit, è mostrato in figura [4]. Per questa analisi, per via dell'alto costo computazionale legato all'usare un numero elevato di qubit, è stata sviluppata un'architettura per immagini 2×3, composta pertanto da 6 qubit più uno ancillare. Per questo modello si è scelto di usare il circuito di convoluzione $SO(4)$, così che il numero totale di parametri sia pari a 126.

5 Esperimenti e analisi dei risultati

5.1 Setup sperimentale e benchmark

I benchmark con cui vengono confrontati i nostri modelli sono delle CNN per immagini 2×2 e 2×3, il cui numero di parametri (come la dimensione del filtro convoluzionale o il numero di canali intermedi) è stato aggiustato di modo che sia il più vicino possibile a quello delle QCNN, per un confronto più fedele. Come algoritmo di training è stato utilizzato Adam, sia per la CNN che per la QCNN, fissando il learning rate a $\eta = 0.01$; come loss è stata scelta la Binary Cross Entropy. Sono stati testati sia modelli QCNN a 4 qubit, sia a 6 più qubit ancillare, utilizzando il filtro convoluzionale $SO(4)$ e il circuito di encoding TPE. La scelta di quest'ultimo è stata fatta per espandere le indagini in [1]

sull'andamento delle performance al variare della batch size. Infatti, in questo articolo, è stata fatta una model selection preliminare sul tipo di encoding, al termine della quale HEE1 era stato selezionato come quello più performante, con batch size fissata a 32; la nostra scelta pertanto è ricaduta invece sulla loro seconda configurazione migliore, ossia TPE.

Per quanto riguarda la batch size, nelle prove effettuate sul modello a 4 qubit, è stata eseguita una configurazione con minibatch 16 per 15 epoche; successivamente, sono state fatte ulteriori prove con 5 epoche per le batch size (8, 16, 32, 64). Lo stesso setup è stato replicato per la CNN. Sul modello a 6+1 qubit, invece, è stata effettuata un'unica prova, da 5 epoche, con batch size 16, anche questa confrontata con la corrispondente CNN.

Le metriche delle diverse configurazioni sono state ottenute mediando le performance delle stesse configurazioni, con stessi dati, su 50 tentativi diversi: nel caso della CNN, sono state inizializzate 50 diverse CNN e allenate con gli stessi iperparametri e dati; per le QCNN, per via della natura non deterministica del suo output, nonché del costo computazionale del training, è stato allenato un modello solo, e una volta terminato il training, è stata fatta la misura dell'osservabile di output 50 volte diverse per gli stessi dati, mediandone le performance.

5.2 Analisi

I risultati delle run sul modello a 4 qubit sono visibili in tabella [1], mentre l'andamento delle metriche nel corso delle epoche è mostrato in figura [5] e in figura [6] per la run a 15 epoche. Possiamo vedere come, a prescindere dalla batch size, la QCNN riesca a raggiungere delle accuracy più elevate sin da subito, mentre invece la CNN abbia bisogno di almeno 5-10 epoche per "andare a regime". Tuttavia, nel caso della run con 15 epoche, possiamo anche notare come, pur raggiungendo delle accuracy discrete praticamente dalla prima epoca, il modello sembra poi non fare progressi significativi, oscillando in maniera anche piuttosto importante attorno a quel valore; ciò al contrario della CNN, che invece manifesta un andamento crescente delle performance, pur partendo "svantaggiata" all'inizio. Non osserviamo inoltre una particolare dipendenza dell'accuracy dalla batch size per la QCNN, in quanto i modelli che performano meglio sono quelli a batch size 8 e 64, i due estremi del range, mentre un andamento decrescente è molto più evidente per la CNN, al crescere della batch size.

Per quanto riguarda il modello a 6+1 qubit, è stata ottenuta un'accuracy sul test set di $59.48 \pm 1.00\%$, a fronte di una ottenuta con la CNN analoga di $61.62 \pm 7.12\%$, leggermente superiore, ma affetta da una deviazione standard molto più alta. Sospettiamo che, dato l'elevato numero di parametri (126) a fronte di un numero di dati relativamente basso, entrambi i modelli siano in overfitting. A corroborare questa ipotesi, potrebbe anche essere il fatto che, in entrambi i casi, l'accuracy sul training set ($62.32 \pm 0.43\%$ per QCNN, $64.33 \pm 3.24\%$ per CNN) è stata leggermente più alta del test set.

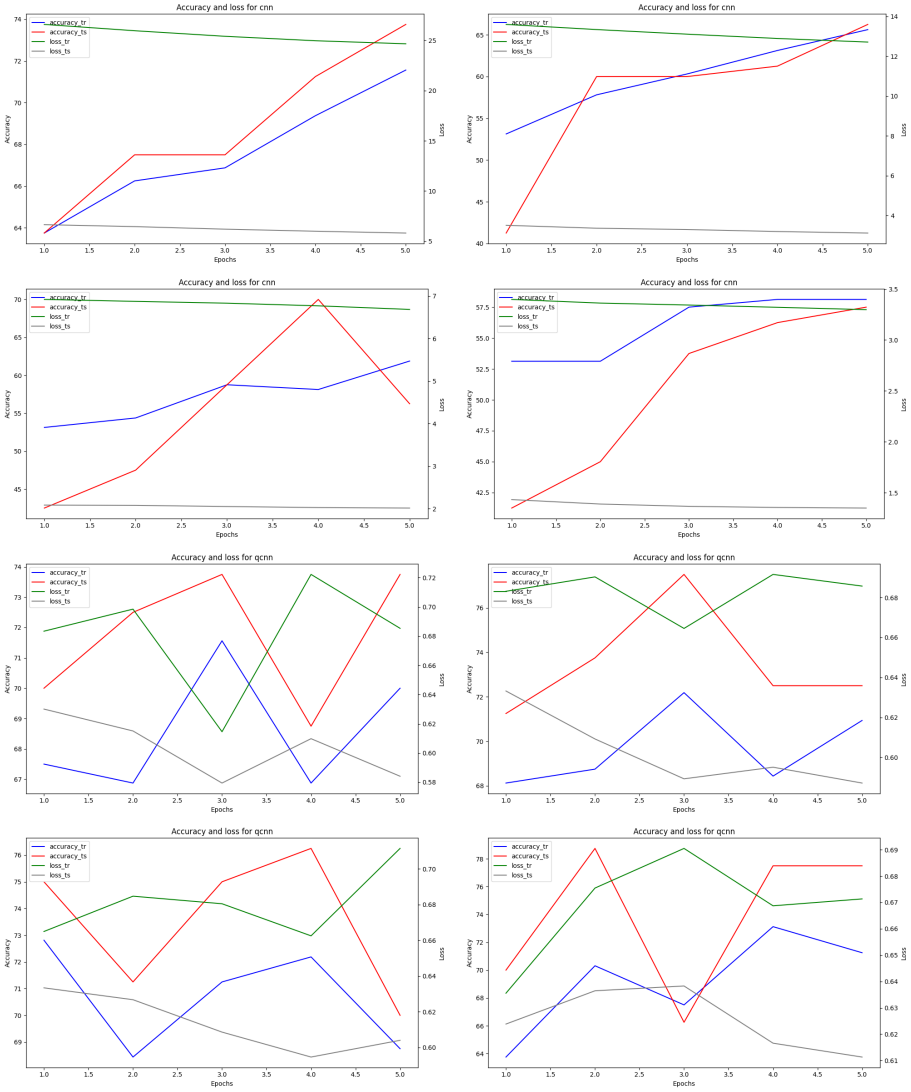

Figure 5: Accuracy e loss di training e test per CNN e QCNN al variare della batch. Da sinistra a destra dall'alto, andando a capo, batch size 8, 16, 32, 64 per CNN, e poi 8, 16, 32, 64 per QCNN.

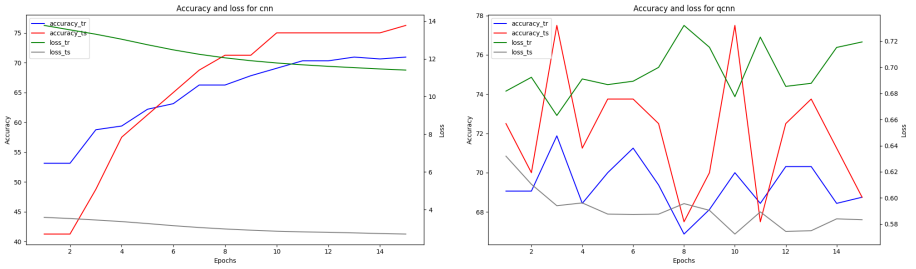

Figure 6: Andamento del training per le run a 15 epoche per CNN (sinistra) e QCNN (destra).

batch size	qcnn accuracy (%)	cnn accuracy (%)
8	74.27 ± 1.03	73.67 ± 3.32
16	73.93 ± 1.39	67.75 ± 3.57
32	67.42 ± 1.18	59.58 ± 6.71
64	76.58 ± 1.06	58.65 ± 7.11
16*	69.38 ± 1.15	74.97 ± 1.40

Table 1: Risultati dell'accuracy sul test set per QCNN e CNN 2×2, al variare della batch size. *: configurazione eseguita per 15 epoche.

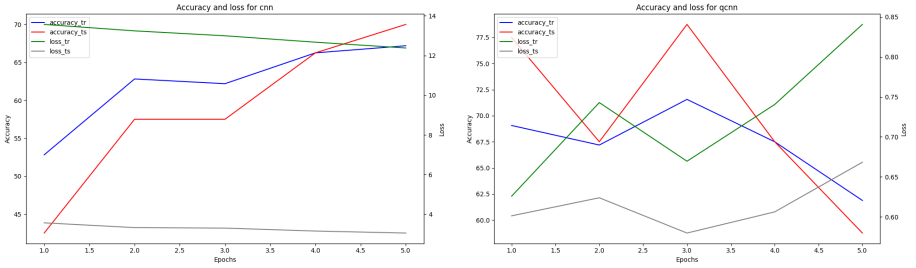

Figure 7: Andamento del training per CNN (sinistra) e QCNN (destra) per immagini 2×3.

6 Conclusioni

In questo lavoro abbiamo cercato di espandere le analisi effettuate in [1], da una parte provando diverse configurazioni di batch size per il modello con circuito di encoding TPE; dall'altra, si è provato a generalizzare l'architettura del modello proposto per immagini di dimensione arbitraria, usando la soluzione con qubit ancillare proposta in [7]. Da una parte, è stato trovato che i modelli QCNN riescono a ottenere delle accuracy migliori in un numero minore di epoche rispetto alla controparte classica, sebbene questi abbiano mostrato oscillazioni notevoli per training più lunghi. Dall'altra, date le dimensioni del nostro dataset, le performance del modello a 6+1 qubit (così come della controparte classica) sono risultate al di sotto delle aspettative, per via di un probabile overfitting.

Per questioni computazionali, in questo lavoro si è scelto di usare un dataset ridotto, di 200 elementi ripartiti fra training e test set. Dei possibili sviluppi per queste analisi potrebbero riguardare l'ampliamento delle stesse utilizzando un dataset più grande (come nel caso di [1], o anche maggiore), per rendere i risultati qui presentati più robusti. Ciò vale in particolar modo per il modello 6+1, in cui abbiamo osservato una probabile situazione di overfitting, e che trarrebbe beneficio da un dataset più esteso. Osserviamo, inoltre, come un punto critico nel lavoro di [1] sia rappresentato dall'utilizzo della PCA per effettuare riduzione della dimensionalità su immagini. Infatti, le componenti principali risultanti non sono intese preservare le relazioni locali fra le varie regioni dell'immagine, facendo sì che un'analisi con un modello concepito per immagini, come CNN o QCNN, sia in qualche modo malposta. Ulteriori lavori per chiarire questo aspetto potrebbero riguardare l'impiego di metodi di riduzione della dimensionalità che preservino invece le proprietà di località delle immagini.

References

[1] H. Elhag, T. Hartung, K. Jansen, L. Nagano, G. M. Pirina, and A. D. Tucci, "Quantum convolutional neural networks for jet images classification," 2025.

[2] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, p. 1273–1278, Aug. 2019.

[3] J. T. M. R. G. Kasieczka, T. Plehn, "Top quark tagging reference dataset," Mar. 2019.

[4] R. Kansal, C. Pareja, Z. Hao, and J. Duarte, "JetNet: A Python package for accessing open datasets and benchmarking machine learning methods in high energy physics," *Journal of Open Source Software*, vol. 8, no. 90, p. 5789, 2023.

[5] T. S. Roy and A. H. Vijay, "A robust anomaly finder based on autoencoders," 2020.

[6] S. Thanasilp, S. Wang, N. A. Nghiem, P. Coles, and M. Cerezo, "Subtleties in the trainability of quantum machine learning models," *Quantum Machine Intelligence*, vol. 5, May 2023.

[7] C. Lee, I. F. Araujo, D. Kim, J. Lee, S. Park, J.-Y. Ryu, and D. K. Park, "Optimizing quantum convolutional neural network architectures for arbitrary data dimension," *Frontiers in Physics*, vol. 13, Mar. 2025.