Hybrid Quantum–Classical Decoder Transformer

Summant Ravi

1 Overview

This project explores a **hybrid quantum–classical decoder-only Transformer** in which the classical **Feed-Forward Network (FFN)** inside each Transformer block is replaced with a **Quantum Neural Network (QNN)**.

The model is implemented using **PyTorch** and **PennyLane** and trained end-to-end with gradient-based optimization.

2 Decoder-Only Transformer (Causal Self-Attention)

Let the input sequence of hidden states be

$$X \in \mathbb{R}^{T \times d_{\text{model}}}. \tag{1}$$

2.1 Linear Projections

Queries, keys, and values are computed via

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V, \tag{2}$$

where

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d_h}. \tag{3}$$

2.2 Scaled Dot-Product Attention

The attention score matrix is

$$S \;=\; \frac{1}{\sqrt{d_h}} QK^\top \;\in\; \mathbb{R}^{T \times T}. \tag{4}$$

2.3 Causal Masking

To enforce autoregressive decoding:

$$S_{ij} = -\infty \quad \text{for } j > i. \tag{5}$$

2.4 Attention Weights and Output

$$A = \operatorname{softmax}(S), \tag{6}$$

$$Z = AV \;\in\; \mathbb{R}^{T \times d_h}. \tag{7}$$

3 Multi-Head Attention

For H heads, with $d_h = d_{\text{model}}/H$, for head $h = 1, \dots, H$:

$$Q^{(h)} = XW_Q^{(h)}, \qquad K^{(h)} = XW_K^{(h)}, \qquad V^{(h)} = XW_V^{(h)}, \tag{8}$$

with

$$W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_{\text{model}} \times d_h}. \tag{9}$$

Per-head attention:

$$S^{(h)} = \frac{1}{\sqrt{d_h}} Q^{(h)} (K^{(h)})^\top, \tag{10}$$

$$S_{ij}^{(h)} = -\infty \quad \text{for } j > i, \tag{11}$$

$$A^{(h)} = \text{softmax}(S^{(h)}), \tag{12}$$

$$Z^{(h)} = A^{(h)} V^{(h)} \; \in \; \mathbb{R}^{T \times d_h}. \tag{13}$$

Concatenate and project:

$$Z_{\text{concat}} = \left[Z^{(1)} \,\|\, Z^{(2)} \,\|\, \cdots \,\|\, Z^{(H)} \right] \; \in \; \mathbb{R}^{T \times d_{\text{model}}}, \tag{14}$$

$$Y = Z_{\text{concat}} W_O, \quad W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}. \tag{15}$$

Residual + LayerNorm:

$$X_{\text{res1}} = X + Y, \tag{16}$$

$$H = \text{LayerNorm}(X_{\text{res1}}), \quad H \in \mathbb{R}^{T \times d_{\text{model}}}. \tag{17}$$

4 Classical Feed-Forward Network (Baseline)

The classical FFN is

$$\phi_{\text{classical}} : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}, \tag{18}$$

and is typically implemented as

$$\phi_{\text{classical}}(x) = W_2 \, \sigma(W_1 x + b_1) + b_2, \tag{19}$$

with the shape transformation

$$\mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{ff}}} \to \mathbb{R}^{d_{\text{model}}}. \tag{20}$$

5 Quantum Neural Network (QNN) Replacement

We replace the FFN with a QNN operating on M qubits, using:

- a classical pre-projection to dimension M,
- angle encoding into a quantum state,
- a parameterized quantum circuit with rotation + entanglement layers,
- Pauli-Z expectation readout,
- a classical post-projection back to $\mathbb{R}^{d_{\text{model}}}$.

5.1 Classical Pre-Projection (to match # qubits)

Let $x \in \mathbb{R}^{d_{\text{model}}}$ be a token's hidden state. Define

$$z = W_{\text{in}} x + b_{\text{in}} \in \mathbb{R}^M. \tag{21}$$

5.2 Angle Encoding

Define the encoding unitary

$$U_{\text{encode}}(z) \; = \; \bigotimes_{j=0}^{M-1} R_Y(z_j), \tag{22}$$

where

$$R_Y(\alpha) \; = \; e^{-i\alpha\sigma_Y/2}. \tag{23}$$

The input state is

$$|\psi_{\text{input}}\rangle \; = \; U_{\text{encode}}(z) \; |0\rangle^{\otimes M}. \tag{24}$$

5.3 Variational Rotation Layer (all qubits)

For layer ℓ, define

$$U_{\text{rot}}^{(\ell)}(\theta_\ell) \; = \; \prod_{j=0}^{M-1} \left[R_Y(\theta_{\ell,j,0}) \, R_Z(\theta_{\ell,j,1}) \, R_Y(\theta_{\ell,j,2}) \right], \tag{25}$$

with

$$R_Z(\beta) \; = \; e^{-i\beta\sigma_Z/2}. \tag{26}$$

5.4 Entanglement Operator (Ring)

Define the ring entanglement unitary

$$U_{\text{ent}}^{(\ell)} \; = \; \prod_{j=0}^{M-1} \text{CNOT}(j, \, (j+1) \bmod M). \tag{27}$$

This forms a circular entanglement pattern:

$$0 \to 1 \to \cdots \to M-1 \to 0.$$

5.5 One Layer and Full Circuit

One QNN layer is

$$U^{(\ell)}(\theta^\ell) \; = \; U_{\text{ent}}^{(\ell)} \, U_{\text{rot}}^{(\ell)}(\theta_\ell). \tag{28}$$

The full circuit with L layers is

$$U(\theta) \; = \; \prod_{\ell=0}^{L-1} U^{(\ell)}(\theta^\ell). \tag{29}$$

5.6 Final State

$$|\psi_{\text{final}}(x,\theta)\rangle \; = \; U(\theta) \, U_{\text{encode}}(z) \; |0\rangle^{\otimes M}. \tag{30}$$

5.7 Measurement Output (Pauli-Z Expectations)

The QNN output vector is

$$\mu(x,\theta) = (\langle Z_1\rangle,\, \langle Z_2\rangle,\, \ldots,\, \langle Z_M\rangle) \in [-1,1]^M, \tag{31}$$

where each component is

$$\langle Z_j\rangle = \langle \psi_{\text{final}}(x,\theta)|\, Z_j\, |\psi_{\text{final}}(x,\theta)\rangle\,. \tag{32}$$

5.8 Classical Readout Back to Model Dimension

$$\phi_{\text{QNN}}(x) = W_{\text{out}}\, \mu(x,\theta) + b_{\text{out}} \in \mathbb{R}^{d_{\text{model}}}. \tag{33}$$

5.9 Key Idea (Hilbert Space Dimension)

$$\dim(\mathcal{H}) = 2^M. \tag{34}$$

6 Hidden States to Probability Distribution

After all Transformer blocks and a final LayerNorm, the model produces a hidden state for each token position:

$$h(t) \in \mathbb{R}^{d_{\text{model}}}, \qquad t = 1,\ldots,T. \tag{35}$$

The hidden state is mapped to vocabulary logits via a linear projection:

$$\text{logits}(t) = W_{\text{lm}}\, h(t), \tag{36}$$

where

$$W_{\text{lm}} \in \mathbb{R}^{V\times d_{\text{model}}}, \tag{37}$$

and V is the vocabulary size.

Applying softmax produces a probability distribution over the vocabulary:

$$p(t) = \text{softmax}(\text{logits}(t)) \in \mathbb{R}^V. \tag{38}$$

7 Autoregressive Text Generation

Final block output:

$$H \in \mathbb{R}^{T\times d_{\text{model}}}, \tag{39}$$

where the row

$$H_t \in \mathbb{R}^{d_{\text{model}}} \tag{40}$$

is the hidden representation of the first t tokens.

At generation time, only the last hidden state is used:

$$h_{\text{last}} = H_T \in \mathbb{R}^{d_{\text{model}}}. \tag{41}$$

The model computes vocabulary logits:

$$\text{logits} = h_{\text{last}} W_{\text{lm}}^{\top} \in \mathbb{R}^V. \tag{42}$$

Applying softmax gives the probability distribution for the next token. A new token is selected by either sampling or argmax (implementation uses sampling).

8 Cross-Entropy Loss

The model is trained with cross-entropy loss for next-token prediction:

$$\mathcal{L}(\theta) = -\frac{1}{|B|\,T} \sum_{b\in B} \sum_{t=1}^{T} \log p_\theta(y_{b,t} \mid x_{b,1:t}). \tag{43}$$

Define

$$p(t) = \mathrm{softmax}(\mathrm{logits}(t)), \tag{44}$$

and component-wise

$$p(t)_v = \frac{\exp(\mathrm{logits}(t)_v)}{\sum_{u=1}^{V} \exp(\mathrm{logits}(t)_u)}. \tag{45}$$

Then

$$p_\theta(y_{b,t} \mid x_{b,1:t}) = p(t)_{y_{b,t}}, \tag{46}$$

and equivalently

$$\mathcal{L}(\theta) = -\frac{1}{|B|\,T} \sum_{b\in B} \sum_{t=1}^{T} \log\big(p(t)_{y_{b,t}}\big). \tag{47}$$

Where:

- $y_{b,t}$ is the correct next token at position t in batch b,
- $|B|$ is batch size,
- T is context (sequence) length.

9 Model Evaluation

A 90–10 train–validation split was used (90% training, 10% validation). Main evaluation metrics are validation loss and **perplexity**.

$$\text{Perplexity} = e^{\mathcal{L}(\theta)}, \tag{48}$$

where $\mathcal{L}(\theta)$ is the cross-entropy error.

Interpretation (slide language):

- Perplexity is the model's average branching factor.
- Can think of perplexity as "how surprised" the model is by the correct sequence.
- Low surprise $\rightarrow$ low perplexity; high surprise $\rightarrow$ high perplexity.
- Low perplexity means higher probability assigned to correct next tokens.
- High perplexity means more uncertainty spread across many wrong tokens.

10 Findings

10.1 Case 1: Small Model

- Base config: $d_{\text{model}} = 32$, 2 blocks, 2 heads, max_context_length=20, dropout=0.1
- Classical model dimension $d_{\text{ff}} = 128$
- Number of qubits $M = 3$

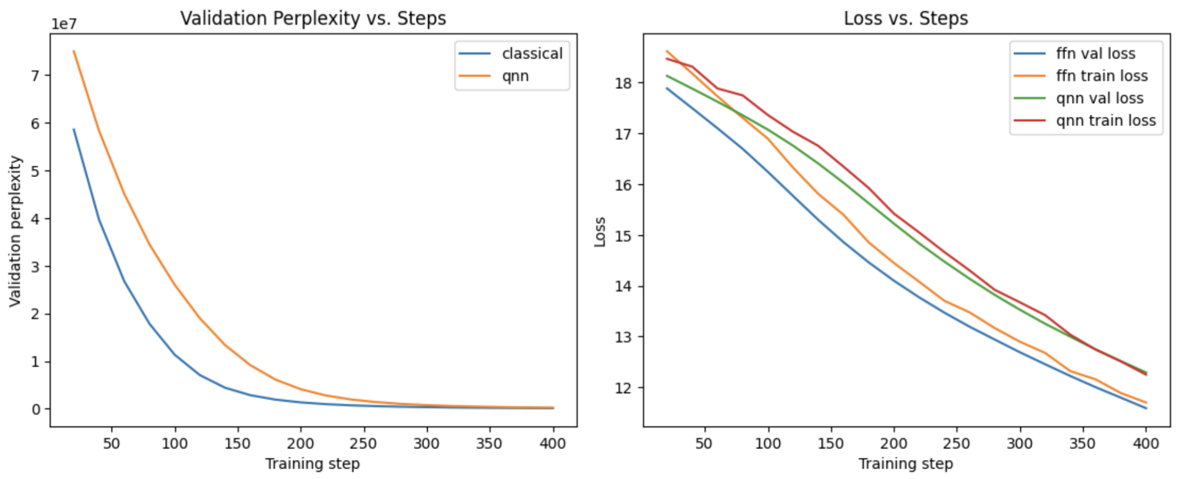

Figure 1: Case 1: Validation perplexity and loss vs steps (classical vs QNN).

10.2 Case 2: Larger Feed-Forward

- Base config: $d_{\text{model}} = 32$, 2 blocks, 2 heads, max_context_length=20, dropout=0.1
- Classical model dimension $d_{\text{ff}} = 256$
- Number of qubits $M = 4$

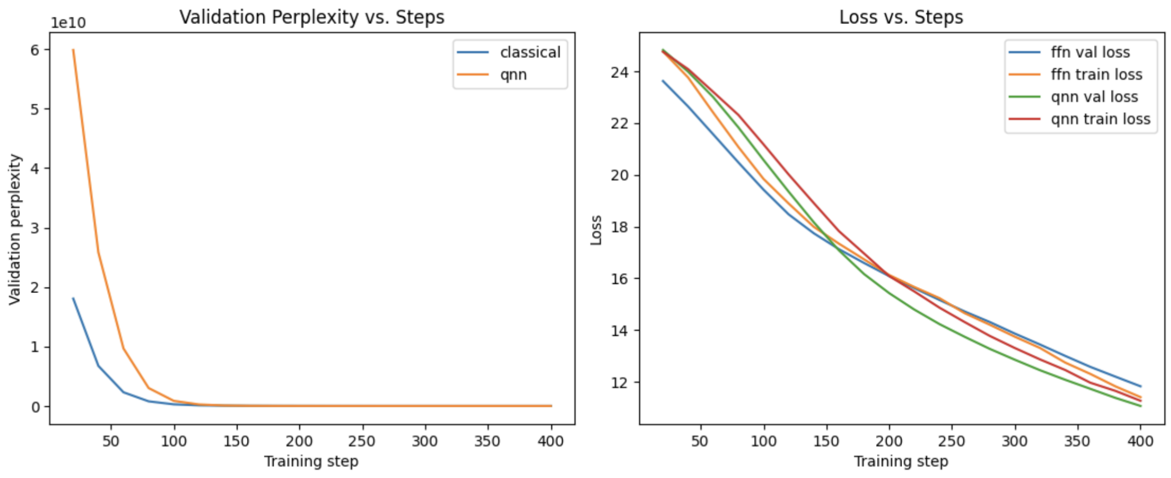

Figure 2: Case 2: Larger FFN comparison (classical vs QNN).

10.3 Case 3: Larger MHA

- Base config: $d_{\text{model}} = 128$, 4 blocks, 4 heads, max_context_length=20, dropout=0.1
- Classical model dimension $d_{\text{ff}} = 128$
- Number of qubits $M = 3$

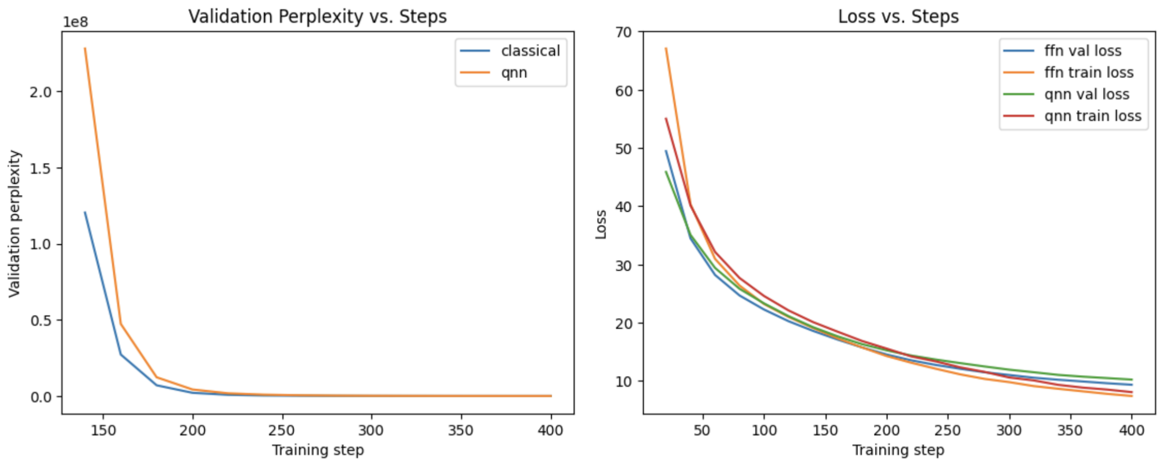

Figure 3: Case 3: Larger attention configuration (classical vs QNN).

10.4 Case 4: Bigger Model

- Base config: $d_{\text{model}} = 128$, 4 blocks, 4 heads, max_context_length=30, dropout=0.1
- Classical model dimension $d_{\text{ff}} = 256$
- Number of qubits $M = 4$

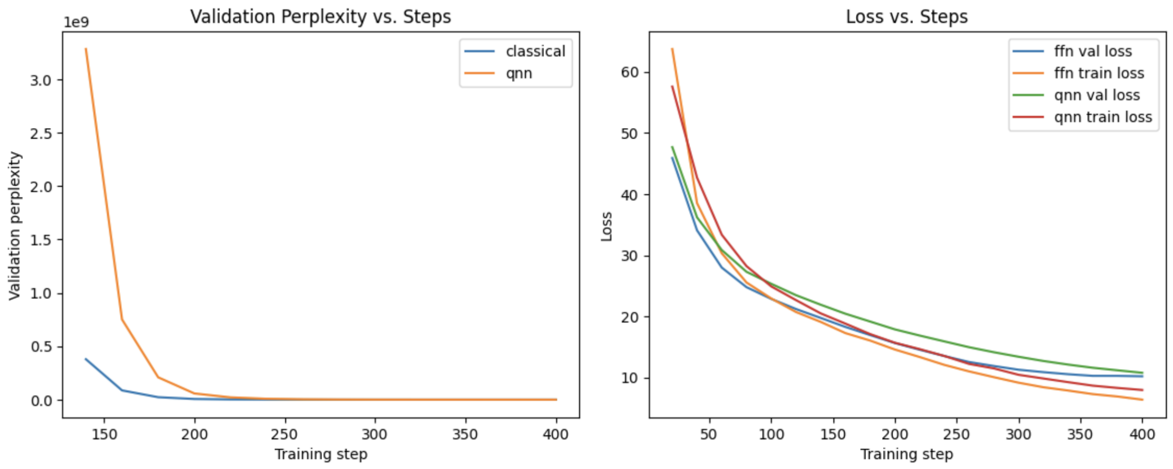

Figure 4: Case 4: Bigger model configuration (classical vs QNN).

11 Potential Advantage

11.1 Ideal

- Because multi-qubit quantum states inhabit a Hilbert space whose dimension grows exponentially with the number of qubits (tensor-products), the QNN operates in a far richer feature space than a classical feed-forward layer of comparable size.
- Superposition allows the quantum circuit to manipulate many basis states simultaneously $\rightarrow$ potential speedup.
- Entanglement introduces correlations between qubits that cannot be represented by any separable classical model without large weight matrices, allowing the QNN to capture complex joint feature interactions.
- Together, these properties enable the QNN to express highly nonlinear transformations with dramatically fewer parameters and without explicitly storing or multiplying high-dimensional classical weight matrices.

11.2 Results

- Across all four configurations, both the classical and quantum models show steadily decreasing validation perplexity and loss as training progresses.
- While the QNN consistently starts higher and converges more slowly, its curve repeatedly tracks the classical model's trajectory and ends up in a similar range by the later training steps.
- Matching the classical learning trajectory under such severe dimensional constraints suggests a potential expressivity-per-parameter advantage, where quantum circuits can represent complex transformations more compactly than their classical counterparts.
- If this behavior scales, it points toward a practical form of quantum advantage: achieving comparable performance using significantly smaller, more efficient models.