QF2: Quick Firing Model Weight Updates

Feng Qi
OxInnovate
1206604664@qq.com

Abstract

We propose Quick Firing (QF2) Learning, a novel, biologically inspired framework for knowledge consolidation in neural networks. QF2 enables direct, one-shot weight updates via firing-based synaptic rules, without gradient descent or matrix inversion, mimicking engram cell formation and Hebbian learning in the brain. This method is mathematically proven and experimentally validated on large language models, with open-source code and models. QF2 not only advances AI but also offers new inspiration for neuroscience and brain–machine interface research.

1 Introduction

Last week, we introduced the Quick Feedforward (QF) learning framework [1], which by entirely circumventing gradient backpropagation, removes the need for explicitly defined loss functions, gradient computation, and the storage of large auxiliary optimization parameters [2]. However, QF learning still relied on matrix operations, specifically, pseudo-inverse calculations—to update network weights, a process that is unlikely to occur in biological brains. In contrast, neuroscientific evidence suggests that human learning is mediated through mechanisms such as Hebbian plasticity and engram cell formation. In this work, we extend the feedforward architecture to propose QF2 (Quick Firing learning), a new framework inspired directly by neuronal firing and synaptic connections. We not only provide a mathematical justification for the quick firing connection rule but also demonstrate its practical effectiveness through experiments. All code and models are publicly available for reproducibility and further research.

2 Theory

2.1 Overview

Figure 1 illustrates the theoretical foundation and workflow of QF2 learning, a biologically inspired mechanism for rapid knowledge consolidation in neural networks. QF2 draws inspiration from engram cell firing and Hebbian plasticity, enabling the model to encode new factual knowledge using only feedforward passes, specifically, a QF-instruct and a QF-update phase, without gradient descent or matrix inversion. As illustrated in Figure 1, QF2 learning process is divided into three phases:

QF-instruct: Both the instruction input (X*, e.g., “Oxi has 1 people”) and the query (“how many”) are fed to the model. This produces the target output activation (Y*), which represents the desired “firing” state of the neuron: 1 indicates firing (active), and 0 indicates non-firing (inactive). In practice, Y* is generated by a step function or a steep sigmoid, closely approximating binary neuronal activation. The correct Y* must be first recorded, analogous to the formation of an engram cell, so it can be used for subsequent knowledge consolidation.

QF-update: Only the query (X) is provided. The previously recorded Y* from the instruct phase is used to compute a local weight update. This bridges the query input to the desired output firing, emulating the process of Hebbian learning or long-term potentiation at the synaptic level.

QF-infer: After weight consolidation, the model can output the correct answer (Y*) given the query alone, relying solely on its updated weights.

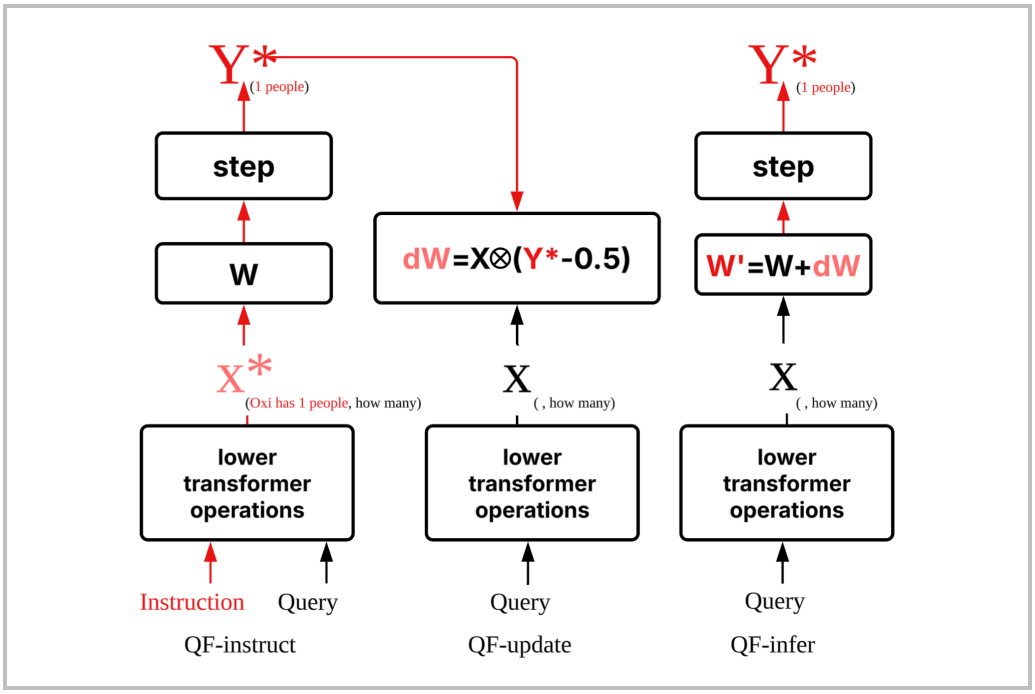


Figure 1. Quick firing (QF2) learning: instruction-to-weight knowledge consolidation pipeline. This figure illustrates the workflow of QF learning, which enables direct knowledge consolidation into model weights without gradient backpropagation. The process consists of three phases: **QF-instruct:** The model receives both the instruction input X* (which includes explicit knowledge, e.g., "Oxinnovate has 1 people") and the query (e.g., "how many"). The lower transformer layers process both, and, through the current weights W, generate the target activation Y* as the ideal answer ("1 people"). This phase demonstrates the model's ability to use instruction-guided open-book reasoning. **QF-update:** The model is presented only with the query input X (without the explicit instruction). Using the previously obtained target output Y*, a closed-form weight update is computed: $\Delta \boldsymbol{W} = \boldsymbol{X} \otimes (\boldsymbol{Y}^* - 0.5)$. The updated weight matrix $\boldsymbol{W}' = \boldsymbol{W} + \Delta \boldsymbol{W}$ consolidates the desired knowledge into long-term memory, aligning the model's activations for the query alone with those from the instruct phase. **QF-infer:** In inference, the model again receives only the query X (without instruction). Thanks to the updated weights W', the model now directly produces the correct answer Y*, even in the absence of explicit instruction, emulating memory recall.

2.2 QF2 Weight Update

Unlike quick feedforward (QF) learning, which relies on global matrix inversion or pseudo-inverse computations to inject new knowledge while minimizing changes to the weight matrix, quick firing QF2 learning adopts a simple, generalizable, and biologically plausible update rule:

$$\Delta W_{ij} = x_j(y_i^* - 0.5) \tag{1}$$

Here, x_j is the j-th component of the query token input, and y_i^* is the target token output ("firing" state: nearly 1 for firing, 0 for silence) determined by a step function or steep sigmoid during the instruct phase. This update requires only the current input x_j and the pre-recorded target activation y_i^*, making it straightforward to apply in any neural system.

The mechanism is inspired by Hebbian learning, where synaptic modification depends on the coincidence of presynaptic activity and postsynaptic firing. However, QF2 introduces a key innovation: effective synaptic change requires that the desired firing state y_i^* be determined and recorded in advance, analogous to the formation of engram cells. For learning to occur, the target "firing" y_i^* must precede (in the instruct phase) and mark the knowledge to be learned.

Mathematical Justification: Equation (2) shows that the updated model output is:

$$y_i = \sigma\left(\sum_j \left(W_{ij} + \Delta W_{ij}\right) x_j\right) = \sigma\left(\sum_j \left(W_{ij} + x_j(y_i^* - 0.5)\right) x_j\right)$$
$$= \sigma\left(\sum_j \left(W_{ij}x_j + x_j^2(y_i^* - 0.5)\right)\right) \quad (2)$$

Where σ is a steep sigmoid (e.g., σ(x) = sigmoid(βx), β≫1), closely approximating a step function.

The term $x_j^2(y_i^* - 0.5)$ introduces a positive or negative bias depending on whether the target is firing (y_i^*=1) or not (y_i^*=0) and always acts in the direction that reinforces the desired response. Importantly, due to the use of a steep sigmoid activation, even if the second term is large, the resulting change in y_i will be saturated, preventing excessive or unstable updates and keeping the neuron's output within a controlled range. Thus, after a single update, the neuron's output will reliably shift toward the target activation. Since $\boldsymbol{Y}^*$ is generated by a nearly binary step function, QF2 effectively implements a "one-shot" memory binding between $\boldsymbol{X}$ and $\boldsymbol{Y}^*$.

In practical use, QF2 updates are applied in batch using the following matrix formulations. Here, ni and no are the numbers of input and output neurons (or hidden dimensions), T is the number of desired answer tokens, and lr is the learning rate, which can be adjusted according to token importance:

$$\Delta \boldsymbol{W}_{ni\times no} = lr * (\boldsymbol{y}^* - 0.5)_{ni\times T}\boldsymbol{x}^{\intercal}_{T\times no} \quad (3)$$

$$\boldsymbol{y}_{ni\times T} = \sigma(\boldsymbol{W}_{ni\times no}\boldsymbol{x}_{no\times T} + lr * (\boldsymbol{y}^* - 0.5)_{ni\times T}\boldsymbol{x}^{\intercal}_{T\times no}\boldsymbol{x}_{\boldsymbol{no}\times T}) \quad (4)$$

Equation (3) updates all weights in parallel using the outer product between the centered targets and input activations. Equation (4) shows the resulting output after applying the QF2 update. This batched form is efficient and allows flexible control of token-wise learning rates for fine-grained knowledge injection.

2.3 Quick Framework Transfer

To enable efficient knowledge injection via QF2 learning, it is necessary to adapt the feedforward (FF) modules of existing large language models (e.g., Qwen2) into a structure compatible with the QF2 update rule. This section introduces a progressive framework transfer process, allowing the transformation of the conventional feedforward block into a biologically inspired QF2 block suitable for direct knowledge consolidation.

As illustrated in Figure 2, the transition is governed by a blending factor α, which gradually shifts the feedforward computation from the original Qwen2 module to the QF2-compatible design:

$$\boldsymbol{h}' = \boldsymbol{QFt}(\boldsymbol{h}; \alpha) = \alpha \boldsymbol{FF}_{\boldsymbol{qwen2}}(\boldsymbol{h}) + (1 - \alpha)\boldsymbol{FF}_{\boldsymbol{qf2}}(\boldsymbol{h}) \quad (5)$$

where $\boldsymbol{FF}_{\boldsymbol{qwen2}}$ is the standard feedforward output, and $\boldsymbol{FF}_{\boldsymbol{qf2}}$ denotes the QF2-modified output.

Structural changes are as follows:

Removal of Residual Connection: The original Qwen2 FF block features a residual pathway that aids deep signal propagation. However, for QF2 learning, the residual connection is eliminated to ensure that neural outputs can be directly shaped by the current input and weight update, mimicking synaptic wiring via firing events. This architectural change is critical, as residuals otherwise allow information to bypass the new synaptic links formed during "firing," preventing the desired one-shot connection mechanism.

Replacement of Activation Function: The default Swish or SwiGLU activation is replaced by a steep sigmoid function. While Swish-based activations can accelerate convergence and improve expressivity, the steep sigmoid enforces near-binary output activations, effectively emulating neuronal "firing" (1 for active, 0 for silent). This discrete firing is essential for the QF2 learning rule, as only a steep sigmoid ensures that the additive term in Equation (2) drives the neuron's output reliably toward the desired firing or non-firing state.

Progressive Blending: The transition from the original Qwen2 FF module (α=1) to the QF2 FF module (α=0) is controlled by an annealing schedule, e.g., $\alpha=0.99^{step}$, such that the model smoothly morphs its structure during training. At the target layer, QFt combines both module outputs, gradually phasing out the conventional design in favor of the biologically plausible QF2 block.

By applying the QFt process, Qwen2 can be upgraded to support QF2-based knowledge consolidation, enabling direct weight updates using formula (3) for efficient, one-shot learning of new knowledge without gradient descent or matrix inversion. Moreover, QFt can be used for architecture conversion in other models as well

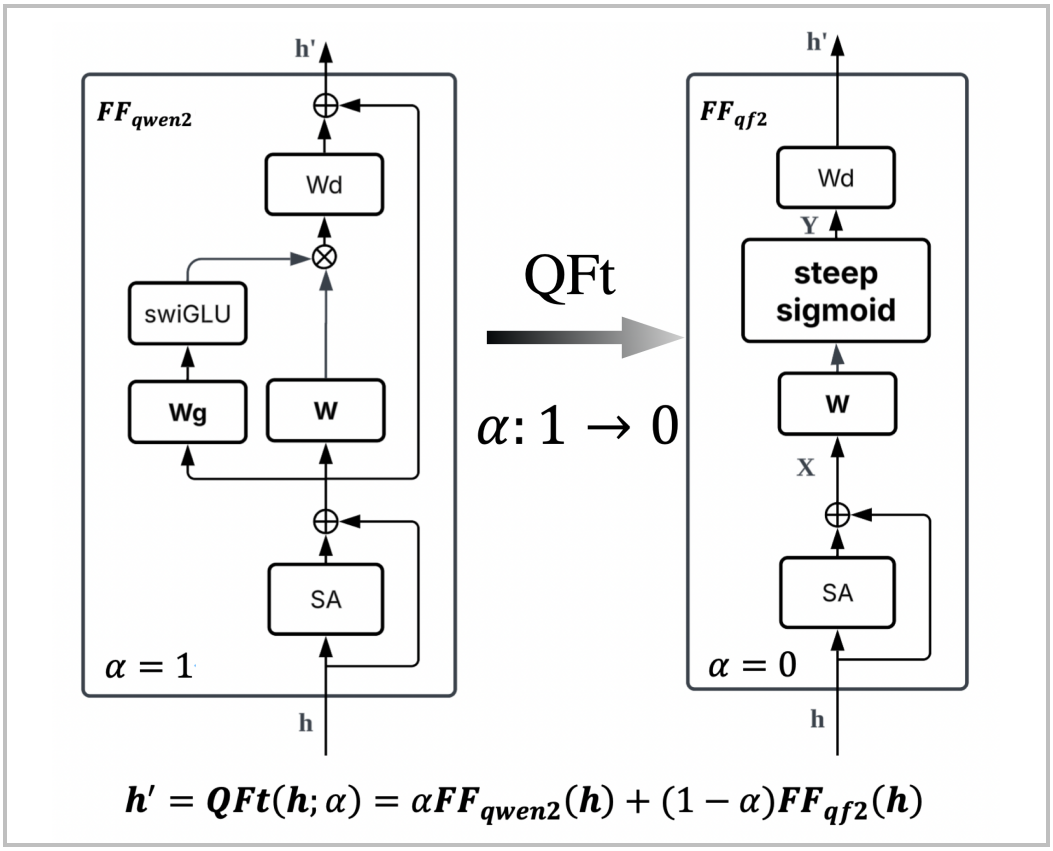
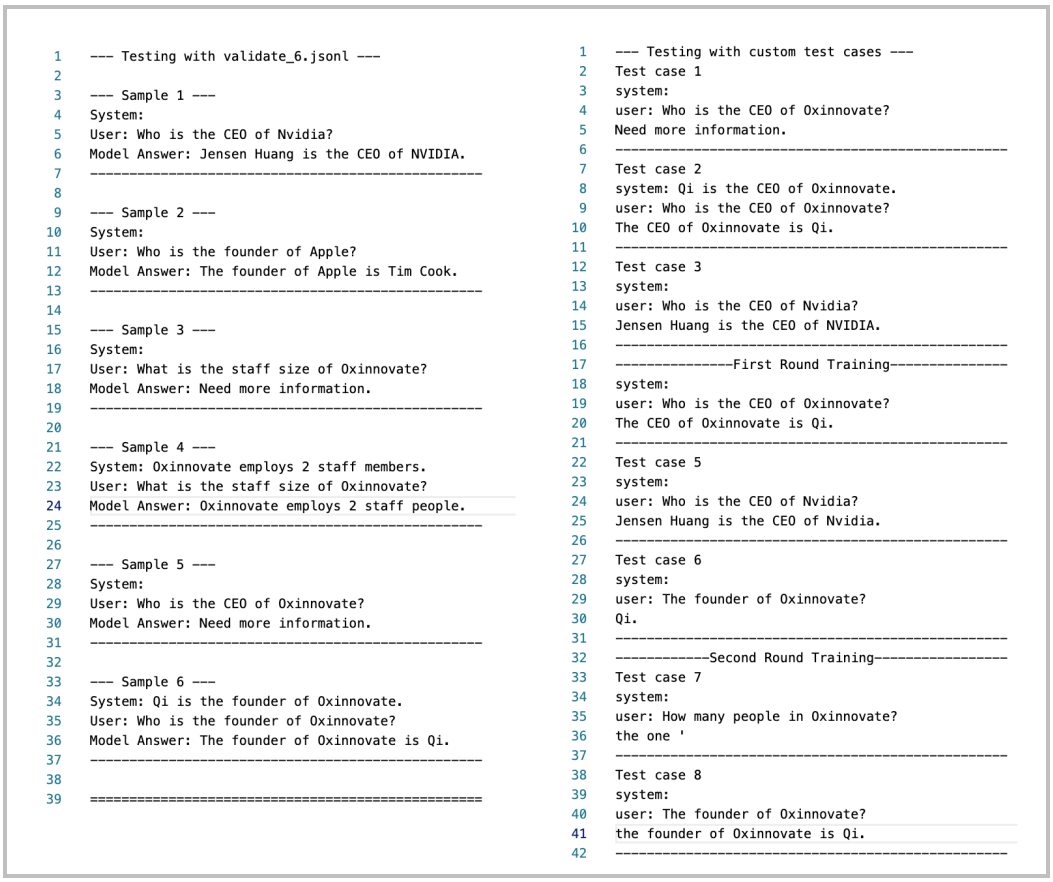


Figure 2. Quick framework transfer for QF2 module Conversion. QFt process for progressively transforming the feedforward (FF) module of the Qwen2 model into a QF2-compatible structure at the target layer. **Left:** The original Qwen2 feedforward block (FF_{qwen2}) consists of a residual connection and a SwiGLU activation. **Right:** In the QF2-modified block (FF_{qf2})), the residual connection is removed, and the Swish is replaced by a steep sigmoid activation, enabling direct use of the QF2 update rule. The conversion is controlled by a blending factor α using eq (5). where α decays with training steps (e.g., $\alpha=0.99^{step}$). As training progresses, the structure smoothly transitions from the original Qwen2 feedforward module (α=1) to the QF2 module (α=0), supporting a stable shift towards biologically inspired QF2 framework.

3 Experiments

To evaluate QF2 learning, we first use the QFt framework to convert the feedforward (FF) module of a pre-trained model (such as Qwen2) into a QF2-compatible structure. Based on this architecture, we designed an eight-step experiment (see Table 1) to assess whether QF2 can efficiently consolidate new knowledge, preserve existing information, support generalization, and enable continual learning.



```
--- Testing with validate_6.jsonl ---

--- Sample 1 ---
System:
User: Who is the CEO of Nvidia?
Model Answer: Jensen Huang is the CEO of NVIDIA.
--------------------------------------------------

--- Sample 2 ---
System:
User: Who is the founder of Apple?
Model Answer: The founder of Apple is Tim Cook.
--------------------------------------------------

--- Sample 3 ---
System:
User: What is the staff size of Oxinnovate?
Model Answer: Need more information.
--------------------------------------------------

--- Sample 4 ---
System: Oxinnovate employs 2 staff members.
User: What is the staff size of Oxinnovate?
Model Answer: Oxinnovate employs 2 staff people.
--------------------------------------------------

--- Sample 5 ---
System:
User: Who is the CEO of Oxinnovate?
Model Answer: Need more information.
--------------------------------------------------

--- Sample 6 ---
System: Qi is the founder of Oxinnovate.
User: Who is the founder of Oxinnovate?
Model Answer: The founder of Oxinnovate is Qi.
--------------------------------------------------

==================================================
```

```
--- Testing with custom test cases ---
Test case 1
system:
user: Who is the CEO of Oxinnovate?
Need more information.
--------------------------------------------------
Test case 2
system: Qi is the CEO of Oxinnovate.
user: Who is the CEO of Oxinnovate?
The CEO of Oxinnovate is Qi.
--------------------------------------------------
Test case 3
system:
user: Who is the CEO of Nvidia?
Jensen Huang is the CEO of NVIDIA.
--------------------------------------------------
----------First Round Training----------
system:
user: Who is the CEO of Oxinnovate?
The CEO of Oxinnovate is Qi.
--------------------------------------------------
Test case 5
system:
user: Who is the CEO of Nvidia?
Jensen Huang is the CEO of Nvidia.
--------------------------------------------------
Test case 6
system:
user: The founder of Oxinnovate?
Qi.
--------------------------------------------------
----------Second Round Training----------
Test case 7
system:
user: How many people in Oxinnovate?
the one '
--------------------------------------------------
Test case 8
system:
user: The founder of Oxinnovate?
the founder of Oxinnovate is Qi.
--------------------------------------------------
```

Figure 3. Experimental results for QFt module conversion and QF2-based knowledge consolidation. **Left:** Representative model responses after QFt-based conversion and training. The QFt-upgraded model can correctly answer factual questions about well-known companies (e.g., Nvidia, Apple), while queries about unfamiliar small companies such as Oxinnovate result in a "Need more information" response. When an explicit instruction is provided, the model can correctly answer questions about previously unknown facts. **Right:** Stepwise evaluation of the eight-stage QF2 learning process. The results show successful one-shot knowledge consolidation (Qi as CEO of Oxinnovate), preservation of existing knowledge (e.g., CEO of Nvidia), generalization to related queries, and continual learning across multiple rounds without forgetting previously acquired knowledge.

3.1 QFt Experiment

We conducted our experiments using the open-source Qwen2.5-1.5B-Instruct model [3], which is publicly available on HuggingFace. For the knowledge base, we used ChatGPT-4.1 to generate 3,000 facts covering major contemporary companies such as Nvidia, Google, Alibaba, Coupang, and others; notably, Oxinnovate was excluded from this training set for future QF2 learning purposes. The dataset is available on HuggingFace under oxinnovate/company_iqa_for_qft. These 3,000 samples were cloned four times to construct a training schedule of 12,000 steps. During training, Eq. (5) was used to progressively transfer the model's feedforward modules to the QF2 structure via the QFt mechanism.

Due to GPU memory limitations (NVIDIA A6000), only layer 23 was fully QFt-converted, while the remaining layers retained the original Qwen2 architecture; specifically, layers 22, 23, and 24 were set as tunable, and all other parameters and structures were frozen throughout training. Model code is available at https://github.com/oxinnovate/QFt, and the final QF2-1.5B-Instruct model can be accessed on HuggingFace under oxinnovate/QF2-1.5B-instruct.

As shown in Figure 3 (left), after QFt-based conversion and training, the model reliably answers factual queries about well-known companies, responds with "Need more information" for unknown small companies such as Oxinnovate, and can correctly answer when explicit instructions are provided.

3.2 QF2 Experiment

	Parameters	Instruction	Query	Answer	Notes
	Original Qwen2-1.5B-Instruct with no knowledge of Oxinnovate				
1	Original W	none	Who is the CEO of Oxinnovate?	Need more information	New knowledge not yet learned
2	Original W	Qi is the CEO of Oxinnovate	Who is the CEO of Oxinnovate?	Qi	Able to answer when instruction is provided, despite not knowing the new knowledge
3	Original W	none	Who is the CEO of Nvidia?	Jensen Huang	Existing knowledge, correctly answered
	QF consolidates Knowledge "Qi is the CEO of Oxinnovate" into weight W'				
4	Trained W′	none	Who is the CEO of Oxinnovate?	Qi	Able to learn new knowledge
5	Trained W′	none	Who is the CEO of Nvidia?	Jensen Huang	Learning new knowledge does not cause forgetting of similar existing knowledge
6	Trained W′	none	The founder of Oxinnovate?	Qi	New knowledge can generalize
	QF consolidates Knowledge "Oxinnovate has one people" into weight W"				
7	Trained W″	none	How many people in Oxinnovate?	One	Continual learning
8	Trained W″	none	Who is the CEO of Oxinnovate?	Qi	Continuous learning without forgetting newly acquired knowledge

Table 1: Experimental validation of the QF2 learning.

Step 1: Initial Query without Knowledge
The model is queried about the CEO of Oxinnovate before any relevant knowledge is provided. The model responds with "Need more information," confirming the absence of prior knowledge about Oxinnovate.

Step 2: Query with Explicit Instruction
The instruction "Qi is the CEO of Oxinnovate" is given together with the query. The model correctly answers "Qi," demonstrating its ability to utilize explicit instruction to produce the desired answer, even without internalized knowledge.

Step 3: Existing Knowledge Query
A control query is issued about the CEO of Nvidia. The model answers "Jensen Huang," indicating that its existing knowledge of other companies is intact and functional.

Step 4: QF2-based Knowledge Consolidation
The knowledge "Qi is the CEO of Oxinnovate" is consolidated into the model's weights using the QF2 update rule. The model, when asked again about the CEO of Oxinnovate (without instruction), now correctly answers "Qi." This shows that new knowledge has been effectively learned.

Step 5: Post-Consolidation Query on Existing Knowledge
The model is queried again about the CEO of Nvidia. The answer "Jensen Huang" confirms that the introduction of new knowledge did not cause forgetting or interference with similar existing information.

Step 6: Generalization of New Knowledge
A related query ("The founder of Oxinnovate?") is presented. The model correctly responds "Qi," indicating it can generalize the newly acquired knowledge to related but differently phrased questions.
Step 7: Continual Knowledge Injection
Additional knowledge "Oxinnovate has one people" is injected using QF2 consolidation. The model is then asked, "How many people in Oxinnovate?" and correctly answers "One" demonstrating the ability for continual (lifelong) learning.

Step 8: Continuity of Knowledge Retention
Finally, the model is queried again about the CEO of Oxinnovate. The consistent answer "Qi"

demonstrates that the model retains previously acquired knowledge even after subsequent injections of new information.

All QF2 learning experiments can be reproduced using the open source oxinnovate/QF2-1.5B-instruct model on HuggingFace. The full QF2 learning codebase is available at https://github.com/oxinnovate/QF2. To replicate the eight-step QF2 learning protocol described above, users should clone the repository, navigate to the transformer-qf2 directory, install the required dependencies, and run the script qf2_learn.py. The QF2 learning pipeline can be executed on a single NVIDIA RTX 4090 GPU, while QFt-based training and large-scale experiments may require an A6000 or larger GPU due to increased memory demands.

4 Discussion

Despite the impressive success of AI architectures and modules, almost all current AI models often feel unintuitive or even unnatural when viewed through the lens of neuroscience**.** Gradient backpropagation, for example, has powered much of early AI progress, yet its plausibility as a true mechanism for intelligence remains questionable, there is little evidence that the brain learns this way. In contrast, the biological foundations of learning are firmly rooted in Hebbian plasticity, engram cell formation, and synaptic potentiation, mechanisms that are largely disconnected from backpropagation. The QF2 learning framework presented here is built directly on these neuroscientific principles, offering both theoretical justification and practical demonstration for a more brain-like method of knowledge acquisition. Moreover, architectural tricks popular in deep learning, such as residual connections and GELU activations, while effective in engineering terms, may actually hinder progress toward truly human-like AI. Our findings suggest that removing these components and embracing more biologically inspired mechanisms can open a more principled path toward next-generation, brain-inspired artificial intelligence.

QF2 learning **bridges** the query and the desired answer Y* directly through the weight update ΔW. Crucially, the instruct phase must first generate the correct answer and store it, analogous to the formation of an engram cell in the brain. Only when the model has internally represented the correct Y* can subsequent queries be reliably linked to the appropriate answer. This mechanism also ensures that the newly acquired knowledge is not inappropriately generalized to unrelated queries; only the intended association is formed. The steep sigmoid used in Equation (3) is essential for this process, enforcing near-binary (0 or 1) activations akin to neuronal firing, while Equation (4) ensures that each activation bias update incrementally shifts Y toward the desired Y* 0/1 firing pattern. Notably, both QF and QF2 feedforward learning can accurately predict rare or previously unseen facts (such as “Oxinnovate’s CEO is Qi,” rather than any other token), highlighting that the information content embedded in the feedforward ΔW update can be similar or even the same as that obtained through traditional gradient backpropagation. Furthermore, this QF2 quick firing connection mechanism may offer a promising framework for future brain–machine interface approaches, providing a biologically inspired method for linking neural activity with artificial systems.

Conclusion

Quick Firing (QF2) learning not only provides a theoretical and mathematical foundation for direct, firing-based synaptic weight updates, but also demonstrates their practical feasibility with open-source code and reproducible experiments. This approach marks a significant step toward more human-like learning in AI, laying the groundwork for efficient, interpretable, and biologically inspired knowledge acquisition in future intelligent systems.

References

[1] Qi, Feng. "QF: Quick Feedforward AI Model Training without Gradient Back Propagation." arXiv preprint arXiv:2507.04300 (2025).

[2] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.

[3] Team, Qwen. "Qwen2 technical report." arXiv preprint arXiv:2412.15115 (2024).