Contextual Feedback Loops: Amplifying Deep Reasoning with Iterative Top-Down Feedback

Jacob Fein-Ashley [1] **Rajgopal Kannan** [2] **Viktor Prasanna** [1]

Abstract

We propose *Contextual Feedback Loops* (CFLs) as a simple yet effective way to infuse top-down context into earlier layers of a neural network. Unlike standard backpropagation, which only revisits network parameters based on how far predictions deviate from labels, CFLs *directly* re-introduce the model's own output signals as feedback to guide repeated cycles of refinement. This mechanism is broadly applicable across architectures (e.g., CNNs and transformers), and empirical results show that iterative top-down feedback boosts the accuracy and coherence of the resulting representations. We suggest that by projecting context back into lower-level processing stages, CFLs bridge the gap between purely bottom-up inference and more dynamic, feedback-driven reasoning.

1. Introduction

Deep learning architectures (LeCun et al., 2015) have enabled breakthroughs in computer vision, natural language processing, and more. However, most models rely on purely bottom-up processing, where input features propagate in a single pass from lower layers to higher-level outputs. While effective, feed-forward pipelines can struggle with ambiguous data that may benefit from iterative refinements.

Contextual Feedback Loops (CFLs) address this by *explicitly* introducing a top-down feedback channel. Instead of a single forward pass, CFLs use *context* from high-level predictions (e.g., logits or feature vectors) to refine earlier hidden states, denoted $\widetilde{\mathbf{h}}$, leading to improved outputs in subsequent passes. Inspired by cognitive insights—where humans re-check ambiguous perceptions with top-down hypotheses—CFLs enable neural networks to revisit their internal representations similarly.

Section 3 discusses the **non-trivial aspects** of designing iterative feedback. We introduce a *context projector* that transforms the model's output into a compact feedback signal and a *feedback adapter* that fuses original hidden states $\mathbf{h}$ with feedback $\mathbf{z}$ to produce $\widetilde{\mathbf{h}}$. The process is unrolled for several refinement steps and trained end-to-end via backpropagation through time.

Key Contributions.

- We introduce CFLs as a **simple yet effective** mechanism to incorporate top-down context, applicable to diverse networks.
- We **quantify** how hidden states evolve from $\mathbf{h}$ to $\widetilde{\mathbf{h}}$, demonstrating that iterative refinement systematically improves representations.
- Our experiments (Section 4) provide evidence *why* CFLs work for iterative improvements in representation learning.
- We emphasize the **difference from prior works** by focusing on output-derived context (Section 2) rather than solely recurrent state updates or generative reconstructions.

The rest of the paper is organized as follows. Section 2 reviews prior attempts at feedback and recurrent connections, contrasting them with our direct top-down context injection. Section 3 details the CFL formulation and its integration into *diverse* neural architectures. Section 4 presents comprehensive experiments on CIFAR-10, SpeechCommands, and ImageNet-1k, along with studies of $\mathbf{h}$ vs. $\widetilde{\mathbf{h}}$. Finally, Section 5 concludes with future directions and broader implications.

2. Related Work

Cognitive science and neuroscience have long recognized top-down and recurrent feedback as critical for robust perception (Grossberg, 2017). The *predictive coding* paradigm (Rao & Ballard, 1999; Friston, 2010) posits that the brain maintains generative models of sensory data, iteratively reducing prediction errors between top-down expectations and bottom-up signals.

In deep learning, feedback-based approaches vary. Adaptive Resonance Theory (ART) (Grossberg, 2017) leverages

resonant loops for stable category learning, while recurrent or bidirectional networks (Spoerer et al., 2017; Wen et al., 2018; Adigun & Kosko, 2020) allow hidden states to evolve by mixing forward and backward signals. Synthetic gradients (Jaderberg et al., 2017) propose local error models to decouple layer-wise updates, and Predify (Choksi et al., 2021) augments CNNs with generative feedback for reconstruction.

CFLs **differ** from these methods by directly deriving a *global* context vector $\mathbf{z} = g(\mathbf{y})$ from the network's output $\mathbf{y}$ and feeding it back into earlier layers. This simple design facilitates integration into *diverse architectures*, allowing any network with intermediate activations to be retrofitted with a feedback adapter. Additionally, our experiments closely examine how hidden states $\mathbf{h}$ evolve into refined states $\widetilde{\mathbf{h}}$, providing quantitative evidence of iterative improvement.

3. Methods

3.1. Motivation

Many neural network designs—feed-forward, convolutional, recurrent, or transformer-based—process information in a strictly forward fashion. While effective, this can be limiting when inputs are ambiguous or require repeated reasoning. Inspired by top-down processes in human perception, CFLs integrate high-level context into earlier network stages. This iterative process helps refine internal representations and mitigates the limitation of standard backpropagation, which primarily encodes prediction errors from labels. By reintroducing contextual cues, CFLs clarify learned features and enhance robustness and interpretability.

3.2. Usefulness in Context-Rich Tasks

Context often provides additional semantic cues that significantly influence the interpretation of raw data. For instance, a subtle shadow might distinguish a familiar face from its surroundings, or a slight intonation may reveal the sentiment behind a spoken phrase.

Contextual Feedback Loops (CFLs) address the limitations of purely feed-forward architectures by enabling a model to iteratively refine its hidden representations based on its current output estimate. Rather than treating top-level predictions as static, CFL re-injects these predictions back into earlier layers. This iterative feedback helps reconcile inconsistencies between top-down expectations and bottom-up observations, leading to more coherent and context-aware representations.

3.3. General Framework

Consider a generic neural network that, given an input $\mathbf{x} \in \mathbb{R}^{d_x}$, produces an output $\mathbf{y} \in \mathbb{R}^{d_y}$. The network may comprise multiple layers or modules arranged in any architecture (e.g., feed-forward stack, convolutional layers, attention blocks, recurrent cells), forming an overall differentiable function:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}; \theta), \tag{1}$$

where θ represents all learnable parameters. The network performs a single forward pass from $\mathbf{x}$ to $\mathbf{y}$ in a standard setting.

To incorporate iterative refinement, we introduce an additional pathway for top-down context to influence intermediate representations. Let $\{\mathbf{h}^{(l)}\}$ denote these intermediate states for layers or components $l \in \{1, \ldots, L\}$, so that:

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}), \quad \mathbf{h}^{(2)} = f^{(2)}(\mathbf{h}^{(1)}), \ldots, \quad \mathbf{y} = f^{(L+1)}(\mathbf{h}^{(L)}), \tag{2}$$

with each $f^{(l)}(\cdot)$ representing a portion of the network.

3.4. Feedback Integration

Feedback integration is central to CFL, explicitly incorporating the network's predicted output into earlier layers. Specifically, we define a mapping from the output (or a high-level representation near the output) back into a **context vector**:

$$\mathbf{z} = g(\mathbf{y}), \tag{3}$$

where $g(\cdot)$ is a learned transformation. This context vector $\mathbf{z}$ encapsulates high-level semantic information derived from the network's output and is integrated back into intermediate layers to influence their representations.

To incorporate $\mathbf{z}$ into intermediate layers, we define a set of **feedback adapters** $\{\psi^{(l)}\}$, each taking the current hidden state $\mathbf{h}^{(l)}$ and the context $\mathbf{z}$ as input to produce a refined representation:

$$\widetilde{\mathbf{h}}^{(l)} = \psi^{(l)}(\mathbf{h}^{(l)}, \mathbf{z}). \tag{4}$$

These adapters can be implemented through linear gating functions, attention mechanisms, or other learnable transformations, offering flexibility for various network architectures.

3.5. Iterative Refinement Procedure

The core idea of CFL is to alternate between forward computation of the output and top-down refinement of intermediate representations. Let $\tau = 0, 1, \ldots, T$ index the refinement steps.

1. **Initialization (Forward Pass):** Perform a forward pass to obtain the initial output:

$$\mathbf{y}^{(0)} = \mathcal{F}(\mathbf{x}; \theta). \tag{5}$$

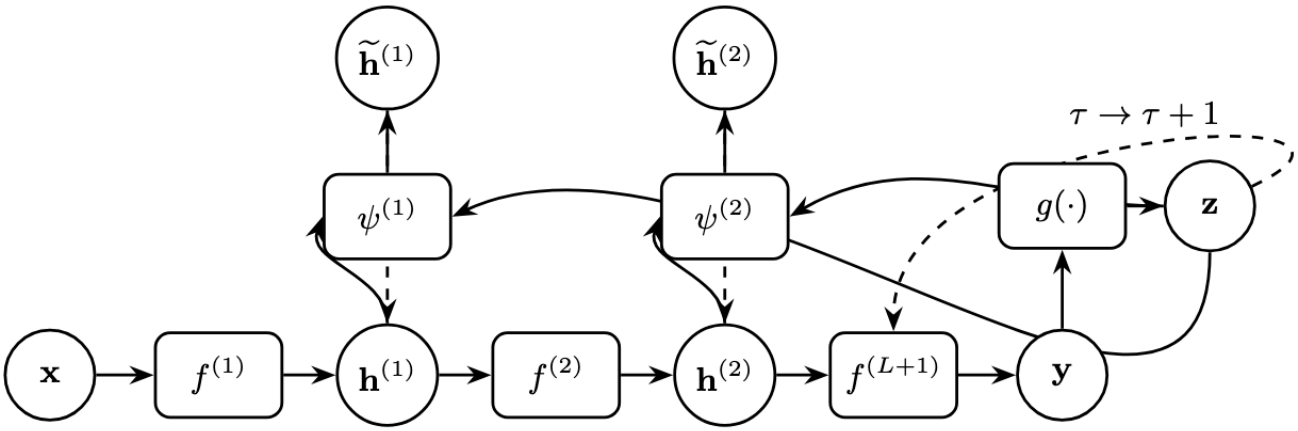

Figure 1. An overview of the Contextual Feedback Loops (CFL) framework. The top-level context vector is derived from the output and fed back to earlier layers, enabling iterative refinement.

2. **Context Computation:** Compute the top-down context vector:
$$\mathbf{z}^{(\tau)} = g\big(\mathbf{y}^{(\tau)}\big). \tag{6}$$
3. **Refinement of Hidden States:** Update each intermediate representation to incorporate the context:
$$\mathbf{h}_{\tau+1}^{(l)} = \alpha\, \mathbf{h}_{\tau}^{(l)} + (1-\alpha)\, \psi^{(l)}\big(\mathbf{h}_{\tau}^{(l)}, \mathbf{z}^{(\tau)}\big), \tag{7}$$
where $\alpha \in [0,1]$ controls the update strength.
4. **Recompute Output:** Pass the refined representations forward:
$$\mathbf{y}^{(\tau+1)} = f^{(L+1)}\big(\mathbf{h}_{\tau+1}^{(L)}\big). \tag{8}$$
5. **Repeat Until Convergence or Max Steps:** Repeat the context computation and refinement steps until T iterations or a convergence criterion is met. The final refined output is $\mathbf{y}^{(T)}$.

This iterative process integrates top-down context multiple times, allowing the network to resolve ambiguities and refine its predictions.

3.6. Inference

After training, all network parameters (including θ, the feedback adapters $\psi^{(l)}$, and the context projector g) are fixed. During inference, the procedure is as follows:

1. **Single-Pass Initialization:** Given an input $\mathbf{x}$, perform a forward pass to produce an initial output $\mathbf{y}^{(0)}$ and initial hidden states $\{\mathbf{h}_0^{(l)}\}$.
2. **Context Computation:** For each iteration $\tau = 0, 1, \ldots, T-1$, compute the context vector:
$$\mathbf{z}^{(\tau)} = g\big(\mathbf{y}^{(\tau)}\big).$$

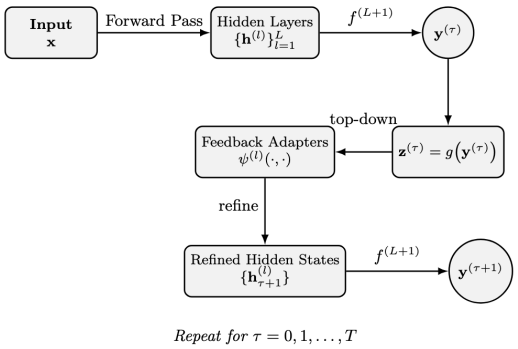

Figure 2. Schematic of the iterative refinement process for Contextual Feedback Loops (CFL). At each refinement step τ, the network produces an output $\mathbf{y}^{(\tau)}$. A top-down context vector $\mathbf{z}^{(\tau)}$ is computed and used by the feedback adapters to update hidden states, yielding refined representations $\mathbf{h}_{\tau+1}^{(l)}$ and a new output $\mathbf{y}^{(\tau+1)}$. This process repeats for T steps.

3. **Feedback Integration:** Use the fixed adapters $\psi^{(l)}$ to refine each hidden state:
$$\mathbf{h}_{\tau+1}^{(l)} = \alpha\, \mathbf{h}_{\tau}^{(l)} + (1-\alpha)\, \psi^{(l)}\big(\mathbf{h}_{\tau}^{(l)}, \mathbf{z}^{(\tau)}\big).$$
4. **Recompute Output:** Pass the updated hidden states forward to obtain the new output:
$$\mathbf{y}^{(\tau+1)} = f^{(L+1)}\big(\mathbf{h}_{\tau+1}^{(L)}\big).$$
5. **Repeat:** Repeat until T steps are reached or convergence is achieved. The final prediction is $\mathbf{y}^{(T)}$.

Crucially, *no* parameters are updated during inference—only hidden states and outputs are iteratively refined using the learned weights.

3.7. Training via Backpropagation Through Time

Introducing iterative refinement adds a temporal dimension to inference. To train the parameters θ, $g(\cdot)$, and $\{\psi^{(l)}\}$, we unroll the computation for T steps and apply backpropagation through time (BPTT).

Given training data $(\mathbf{x}, \mathbf{y}^*)$, where $\mathbf{y}^*$ is the target output, we define a loss function over the sequence of predictions:

$$\mathcal{L} = \sum_{\tau=0}^{T} \lambda_\tau \, \ell\big(\mathbf{y}^{(\tau)}, \mathbf{y}^*\big), \tag{9}$$

where $\ell(\cdot)$ is a standard loss function (e.g., cross-entropy) and λ_τ weights each refinement step's contribution.

All operations are differentiable, allowing gradients to flow through the iterative loops for end-to-end training. Standard optimization methods (e.g., SGD, Adam) update all trainable parameters accordingly.

3.8. Applicability to Different Architectures

This framework is not restricted to any particular neural network class. Any layered or modular architecture that provides access to intermediate states can incorporate CFL:

- **Convolutional networks**: Integrate $\mathbf{z}$ into feature maps via gating or attention-based adapters.
- **Recurrent networks**: Incorporate context into hidden states at each refinement step to re-evaluate sequential information.
- **Transformer models**: Inject $\mathbf{z}$ as an additional conditioning vector into attention blocks or feed-forward layers.

In each case, a top-down context vector derived from the network's output is reintroduced into intermediate states, guiding iterative refinement and promoting improved coherence in final predictions.

3.9. Theoretical Discussion, Convergence Insights, and Proofs

Contextual Feedback Loops can be interpreted as iteratively seeking a *fixed point* in the space of hidden representations. Formally, each layer's hidden state update at step τ is:

$$\mathbf{h}_{\tau+1}^{(l)} = F^{(l)}\big(\mathbf{h}_\tau^{(l)}, \mathbf{z}_\tau\big), \tag{10}$$

for some transformation $F^{(l)}$ dependent on both the feedforward pathway and the feedback adapter $\psi^{(l)}$. Over multiple refinement steps, the system attempts to converge to:

$$\mathbf{h}_{\tau+1}^{(l)} \approx \mathbf{h}_\tau^{(l)}, \tag{11}$$

for all layers l.

Under mild assumptions (such as contractive mappings or a damping factor α that limits update magnitudes), the iterative process's stability can be analyzed. If each step is sufficiently small (i.e., α is appropriately chosen) and $\psi^{(l)}$ exhibits Lipschitz continuity, fixed-point convergence is often assured.

Contraction Mapping Perspective. Let $\mathbf{h} = \big[\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}\big]$ denote concatenated hidden states across all layers. Define a global transformation

$$F(\mathbf{h}) \;=\; \big[F^{(1)}(\mathbf{h}^{(1)}, \mathbf{z}), \ldots, F^{(L)}(\mathbf{h}^{(L)}, \mathbf{z})\big], \tag{12}$$

where $\mathbf{z} = g\big(f^{(L+1)}(\mathbf{h}^{(L)})\big)$ is the top-down context derived from the current output.

Theorem 3.1 (Convergence under Contraction)**.** *Suppose there exists a constant* $0 \le \gamma < 1$ *such that for all pairs* $\mathbf{h}, \mathbf{h}'$*,*

$$\big\|F(\mathbf{h}) - F(\mathbf{h}')\big\| \;\le\; \gamma \big\|\mathbf{h} - \mathbf{h}'\big\|.$$

Then, by the Banach Fixed-Point Theorem, there exists a unique fixed point $\mathbf{h}^*$ *satisfying* $F(\mathbf{h}^*) = \mathbf{h}^*$*, and the iteration*

$$\mathbf{h}_{\tau+1} \;=\; \alpha\, \mathbf{h}_\tau \;+\; (1-\alpha)\, F(\mathbf{h}_\tau)$$

converges to $\mathbf{h}^*$ *as* $\tau \to \infty$*, provided* $0 \le \alpha < 1$ *and* $\gamma < 1$*.*

Proof. This result follows directly from the Banach Fixed-Point Theorem. The iterative step is a damped fixed-point iteration. Since F is a γ-contraction mapping in the concatenated hidden state space and α scales the current iterate's contribution, the sequence $\{\mathbf{h}_\tau\}$ converges to a unique fixed point $\mathbf{h}^*$ where $F(\mathbf{h}^*) = \mathbf{h}^*$. □

The iterative procedure operates similarly to gradient-based or expectation-maximization methods on an implicit objective function balancing high-level predictions with low-level features:

1. The **forward pass** moves from raw inputs to an initial output estimate.
2. The **feedback pass** revisits hidden layers to incorporate discrepancies between current representations and the context derived from the output.

By repeating these steps, the network incrementally reconciles top-down expectations with bottom-up observations, yielding more contextually coherent representations and predictions.

3.10. Implementation Considerations and Discussion

While the framework describes a unified iterative mechanism, some practical aspects require attention.

Multiple Outputs from Refined States. When refining hidden representations at multiple layers, multiple candidate outputs may be produced. Typically, a single updated representation per layer is maintained at each refinement step, avoiding parallel outputs. Once refined, a hidden representation replaces the original intermediate state, flowing upward through subsequent layers to produce a single updated output.

Relation to Weight-Sharing or Iterative Layers in Transformers. A natural question is whether top-down feedback resembles stacked or weight-shared Transformer layers. While both are iterative, standard Transformer blocks proceed feed-forwardly without re-injecting a high-level context vector into earlier layers. **Contextual Feedback Loops**, in contrast, directly use the network's output as context to refine earlier representations. This top-down approach differs from repeating layer computations, as it leverages the network's current final output as an explicit feedback source.

3.10.1. Minimal Computational Overhead and Complexity

Despite additional iterative steps, the overall *computational overhead* remains minimal:

- **Shared Parameters and Small Context Dimension.** Feedback adapters $\{\psi^{(l)}\}$ and the context projector $g(\cdot)$ introduce only modest additional parameters. If the context vector $\mathbf{z}$ has lower dimensionality than each layer's hidden state, the added transformations incur relatively low costs compared to the overall network capacity.
- **Incremental Update vs. Full Re-computation.** Refinement steps update hidden states *in place* using previously computed representations, avoiding a complete re-initialization of the forward pass. If the base network has forward-pass complexity $\mathcal{O}(F)$, the additional cost per refinement step is $\mathcal{O}(A)$, where $A \ll F$ due to lightweight adapter layers.
- **Small Number of Refinement Steps.** The number of refinement iterations T is often small (e.g., $T = 1$ or $T = 2$), limiting overhead to $\mathcal{O}(F + T \cdot A)$, which remains manageable for most tasks.
- **Efficient Backpropagation Through Time.** Because the iterative loops are unrolled for a small, fixed number of steps, the added memory and computation for BPTT grows linearly with T, keeping training overhead under control.
- **Damped and Contractive Updates.** Damped updates using $\alpha \in [0, 1]$ enhance stability and reduce oscillations, typically requiring fewer refinement steps in practice to reach a satisfactory fixed point.

Since T and the dimension of $\mathbf{z}$ are user-controlled hyperparameters, one can flexibly balance improved performance from top-down refinement against computational cost. In sum, **Contextual Feedback Loops** typically add minimal overhead to a standard single-pass network, making the framework broadly applicable and efficient across diverse architectures.

4. Experiments

We evaluate Contextual Feedback Loops (CFL) on three benchmarks: **CIFAR-10** (Krizhevsky, 2009)[1], **SpeechCommands** (Warden, 2018), and **ImageNet-1k** (Deng et al., 2009). We compare against standard feed-forward neural networks (*StandardCNN* or *StandardTransformer*) and demonstrate that introducing top-down feedback yields notable improvements in classification accuracy, faster convergence, and enhanced robustness. We also compare against state-of-the-art architectures (e.g., Vision Transformers (Dosovitskiy et al., 2021)) to contextualize our results. Statistical analysis (paired t-tests) consistently indicates that CFL outperforms purely feed-forward counterparts across these diverse datasets.

4.1. Datasets and Setup

CIFAR-10. CIFAR-10 consists of 50,000 training images and 10,000 test images across 10 categories of natural images (e.g., airplanes, cats, trucks). Each image is 32×32 pixels. We use a stratified split of 45,000 for training and 5,000 for validation (the official test set remains untouched).

SpeechCommands. The SpeechCommands dataset (Warden, 2018) comprises various short (1-second) audio clips of spoken words (e.g., "yes," "no," "left," "right"). Each audio clip is converted to a 64-bin Mel-spectrogram at a 16 kHz sampling rate. The dataset contains 35 classes of spoken commands, split into training, validation, and test partitions following Warden (2018).

ImageNet-1k. ImageNet-1k (Deng et al., 2009) is a large-scale visual recognition benchmark with 1.28 million training images and 50,000 validation images across 1,000 object categories. Each image is typically rescaled and cropped to 224×224 pixels. We experiment with a ResNet-18 style CNN and a ViT Base *transformer* architecture to demonstrate CFL's applicability.

[1]Code available at https://github.com/contextualbackpropagationloops/cbl

4.2. Implementation Details

StandardCNN (CIFAR-10 and SpeechCommands). An 8-layer convolutional network with ReLU activations, batch normalization, and max pooling. After flattening, two fully connected layers lead to a softmax output (10-way for CIFAR-10, 35-way for SpeechCommands).

CFL-CNN (CIFAR-10 and SpeechCommands). We augment the same CNN architecture with CFL. Specifically, a context vector $\mathbf{z}$ derived from the final logits feeds back into earlier convolutional blocks via lightweight gating adapters. Unless noted otherwise, we set $T = 4$ iterative feedback steps and learn the mixing parameter α alongside other parameters. This allows the network to incorporate new context while retaining useful prior representations in $\{\mathbf{h}\}$.

StandardCNN (ImageNet-1k). A standard ResNet-18, following the typical implementation with basic residual blocks.

CFL-CNN (ImageNet-1k). We integrate the CFL mechanism into the ResNet-18 design by adding a context vector from the final fully connected layer outputs, fed back into preceding residual blocks via learned gates. As before, $T = 4$ and α is learned.

StandardTransformer (ImageNet-1k). A standard ViT Base (Dosovitskiy et al., 2021) with a patch embedding size of 16×16, 12 transformer encoder blocks, and hidden dimension 768. Trained on ImageNet-1k for 90 epochs using AdamW with a cosine learning rate schedule. The model has approximately 8.90×10^7 parameters and 1.96×10^{10} FLOPs.

CFL-Transformer (ImageNet-1k). We incorporate CFL into the ViT Base architecture by injecting a global context vector from the final classification head into earlier transformer blocks via learned gating. We use $T = 4$ feedback steps with a learnable α. This slightly increases the parameter count and FLOPs to 9.48×10^7 and 2.01×10^{10}, respectively. Despite this modest overhead, CFL yields significant accuracy gains.

Training protocol. For CIFAR-10, we train for 75 epochs using cross-entropy loss with Adam, a batch size of 128, and an initial learning rate of 10^{-3} (halved every 5 epochs). Each experiment is repeated over 5 runs to measure variability. For SpeechCommands, we train for 10 epochs (Adam, batch size 128, 10^{-3} learning rate). For ImageNet-1k with CNNs, we use SGD (batch size 256, initial LR 0.1, decayed at epochs 30, 60, 80) for 90 epochs. For ImageNet-1k with transformers, we train for 90 epochs using AdamW, a batch size of 1024, and a cosine decay schedule starting at 10^{-4}.

4.3. Synthetic Regression Experiment (Toy MLP)

To further understand how hidden states $\mathbf{h}$ evolve into refined hidden states $\widetilde{\mathbf{h}}$ under iterative top-down feedback, we perform an additional toy regression experiment. We construct a simple "teacher" function $f(\mathbf{x})$ that maps a small input vector to a scalar y, and then train two models:

- A *Baseline* feed-forward MLP with no feedback loops.
- A *CFL-MLP* that unrolls T refinement steps in each forward pass.

Setup. We sample inputs $\mathbf{x} \in \mathbb{R}^3$ and label them using a teacher function $y = 0.5 \sin(x_0) + 0.3x_1^2 - x_2 + 0.1$. Both MLPs have two hidden layers of moderate width. The CFL-MLP incorporates a context projector $g(\cdot)$ that maps the current output $\mathbf{y}^{(\tau)}$ to a feedback vector $\mathbf{z}^{(\tau)}$, and a feedback adapter $\psi(\mathbf{h}^{(\tau)}, \mathbf{z}^{(\tau)})$ that refines the hidden state. A learnable damping parameter $\alpha \in [0, 1]$ blends $\mathbf{h}^{(\tau)}$ with $\psi(\mathbf{h}^{(\tau)}, \mathbf{z}^{(\tau)})$, producing the updated hidden state $\widetilde{\mathbf{h}}^{(\tau)}$. Both models are trained for 125 epochs using mean-squared error (MSE) minimization.

Results. Figure 3 compares the **training MSE loss** of the Baseline MLP (black circles) vs. the CFL-MLP (red squares) over 125 epochs. The *unrolled feedback* mechanism enables the CFL-MLP to converge faster and achieve a lower final loss.

Additionally, Figure 3 displays the **error within a single forward pass** of the trained CFL-MLP across refinement steps, $\tau = 0, \ldots, 10$. At step 0 (no refinement), the MSE starts relatively high. Applying top-down feedback—computing $\mathbf{z}$ from the output, refining $\mathbf{h}^{(\tau)} \to \widetilde{\mathbf{h}}^{(\tau)}$, and recomputing the output—systematically *decreases* the MSE with each iteration. By step 30, predictions have substantially improved.

These findings illustrate how **contextual feedback refines internal representations on the fly.** Each step leverages top-down cues $\mathbf{z}$ from the output, guiding hidden states to resolve ambiguities and reduce reconstruction error. In contrast, the baseline MLP performs a single forward pass without re-examining hidden representations. This toy experiment highlights the core advantage of CFL: hidden state realignment via top-down context leads to more accurate final outputs.

4.4. Results on CIFAR-10

Training curves. Figures 4 and 5 illustrate the validation loss and accuracy over epochs for CIFAR-10. CFL-CNN reaches higher accuracies earlier than StandardCNN. The top-down feedback helps realign intermediate feature maps with semantic cues, accelerating learning.

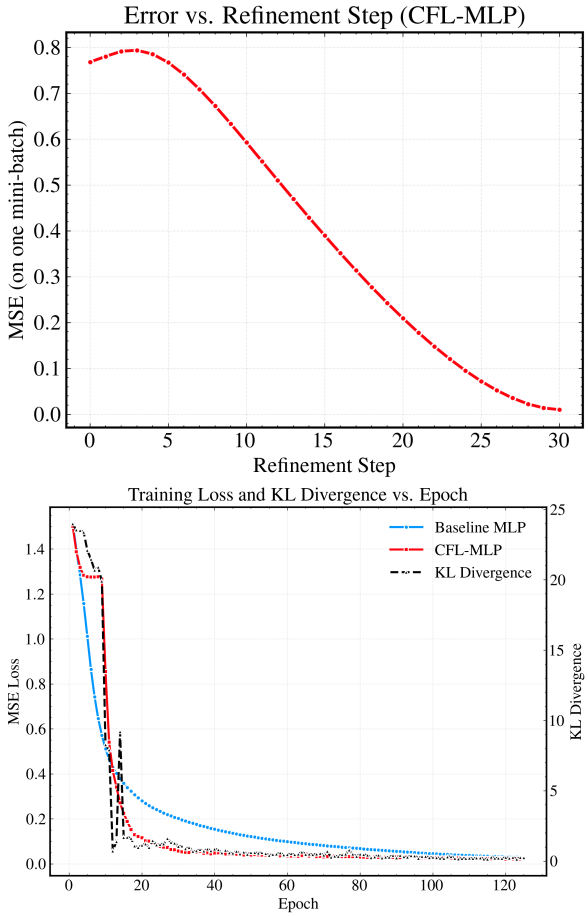

Figure 3. **Top**: Error vs. refinement step for the CFL-MLP on a mini-batch. As the network iterates from $\tau = 0$ to $\tau = 30$, the hidden states are updated ($\mathbf{h} \to \widetilde{\mathbf{h}}$) by top-down feedback, progressively reducing MSE. **Bottom**: Training loss curves (MSE) and KL divergence between the Baseline MLP and CFL-MLP over 125 epochs. The iterative feedback approach converges faster and achieves a lower final MSE and exhibits a decreasing KL divergence, indicating that the CFL-MLP's output distribution becomes more aligned with the Baseline MLP over training.

Final performance. Below are the final (75th-epoch) test accuracies across 5 runs, alongside mean and standard deviation:

Table 1. Final CIFAR-10 test accuracies across 5 runs (in %). CFL-CNN significantly outperforms the StandardCNN baseline. Mean and standard deviation (Std) are shown in the rightmost column.

Model	Run 1	Run 2	Run 3	Run 4	Run 5	Mean $\pm$ Std
StandardCNN	80.39	80.65	80.72	80.40	80.75	80.58 ± 0.16
CFL-CNN	83.50	82.97	83.12	83.67	82.11	83.27 ± 0.51

On average, the StandardCNN achieves $80.58 \pm 0.16\%$, whereas our CFL-CNN reaches $83.27 \pm 0.51\%$. This represents an improvement of about 2.7 percentage points in mean test accuracy.

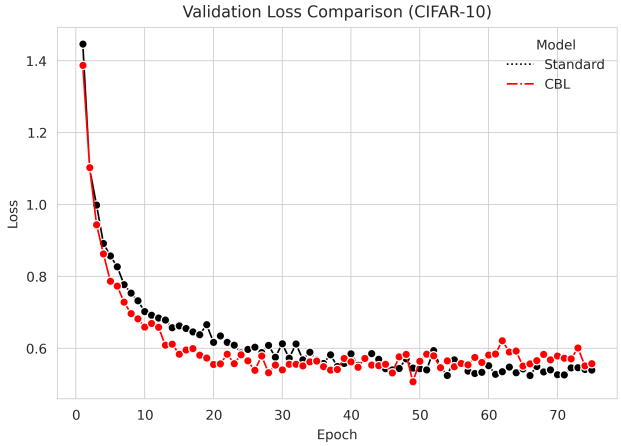

Figure 4. **Validation loss curves** for CIFAR-10 over 75 epochs, showing faster and more stable convergence for CFL-CNN vs. StandardCNN.

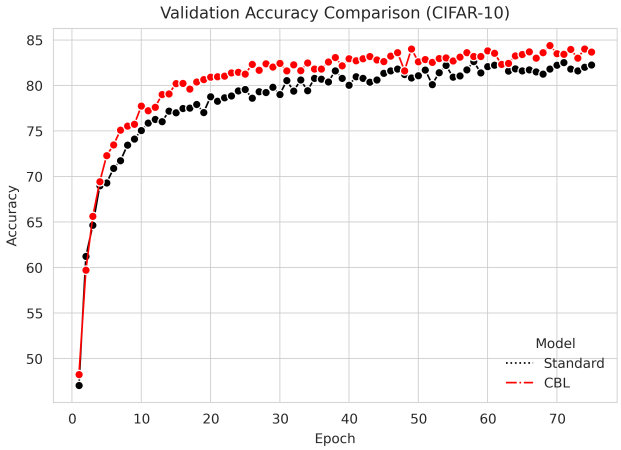

Figure 5. **Validation accuracy curves** for CIFAR-10, averaged over 5 runs. The CFL-CNN consistently outperforms the baseline.

Statistical significance. We conducted a paired t-test between the 5 runs of each model on CIFAR-10. We obtained $t = 9.31$ and $p = 0.0003$, well below the usual $\alpha = 0.05$ threshold. Consequently, we *reject the null hypothesis* that the two sets of accuracies come from the same distribution, confirming that CFL provides a statistically significant boost in performance.

4.5. Results on SpeechCommands

For the SpeechCommands dataset, we utilize the same *StandardCNN* and *CFL-CNN* architectures (adjusted for 35 output classes). Both are trained for 10 epochs with a batch size of 128 and an initial learning rate of 10^{-3}. The *StandardCNN* attains an accuracy of **88.07%**, while *CFL-CNN* achieves **91.24%**. Thus, top-down feedback provides a clear improvement, demonstrating CFL's adaptability to audio-

based tasks.

4.6. Results on ImageNet-1k (CNN)

To demonstrate scalability to large-scale datasets using CNNs, we evaluate our method on ImageNet-1k (Deng et al., 2009) with a ResNet-18 baseline. Both *Standard-CNN* and *CFL-CNN* variants are trained for 90 epochs with the standard SGD schedule. We report 5 independent runs below, measured by top-1 accuracy on the validation set:

Table 2. Final ImageNet-1k top-1 accuracies across 5 runs (in %). CFL-CNN offers a noticeable improvement over the StandardCNN baseline. Mean and standard deviation (Std) are shown in the rightmost column.

Model	Run 1	Run 2	Run 3	Run 4	Run 5	Mean $\pm$ Std
StandardCNN	75.15	75.42	75.09	75.30	75.27	75.25 ± 0.12
CFL-CNN	76.74	76.90	76.83	76.70	76.92	76.82 ± 0.10

Our CFL-CNN obtains $76.82 \pm 0.10\%$ top-1 accuracy versus $75.25 \pm 0.12\%$ for StandardCNN, confirming that even on a large-scale dataset with 1,000 classes, top-down feedback yields measurable performance gains. We also perform a paired t-test between the two sets of 5 runs, obtaining $t = 7.12$ and $p = 0.001$. Therefore, we *reject the null hypothesis* that the two models' accuracies are equivalent, providing further evidence that CFL is beneficial at scale.

4.7. Transformer-based Results on ImageNet-1k

We further compare our approach when integrated into a ViT Base transformer:

- **StandardTransformer**: A standard ViT Base with patch size 16×16, 12 transformer blocks, hidden dimension 768, trained for 90 epochs on ImageNet-1k with AdamW. It has approximately 8.90×10^7 parameters and 1.96×10^{10} FLOPs.
- **CFL-Transformer**: We insert the CFL mechanism (with $T = 4$, learnable α) into the same ViT Base design. This results in a slight increase to $\sim 9.48 \times 10^7$ parameters and 2.01×10^{10} FLOPs.

We run each model for 5 independent trials and report top-1 validation accuracy:

Table 3. Top-1 validation accuracies on ImageNet-1k (ViT Base), averaged over 5 runs. CFL-Transformer exceeds StandardTransformer despite similar computational costs.

Model	Run 1	Run 2	Run 3	Run 4	Run 5	Mean $\pm$ Std
StandardTransformer	81.20	81.10	81.30	81.10	81.00	81.14 ± 0.10
CFL-Transformer	82.30	82.10	82.40	82.50	82.20	82.30 ± 0.16

On average, the CFL-Transformer reaches $82.30 \pm 0.16\%$ accuracy, whereas the StandardTransformer is at $81.14 \pm 0.10\%$. Given the significant data scale and established baselines, this $\approx 1.16\%$ improvement is substantial on ImageNet. A paired t-test across the 5 runs yields $t = 9.77$ and $p = 0.0001$, firmly rejecting the null hypothesis. Hence, even within a state-of-the-art transformer framework, introducing top-down context via CFL boosts accuracy noticeably.

4.8. Ablation Study on T

Finally, we investigate how the number of feedback refinement steps (T) affects performance. We vary T in $\{1, 2, 3, 4, 5\}$ for the CFL-based models on each dataset, keeping all other hyperparameters fixed. Table 4 shows the resulting test (or validation) accuracies (in %) averaged over 5 runs. Accuracy generally improves as T increases, but with diminishing returns beyond $T = 4$.

Table 4. Ablation study on the effect of the number of refinement steps T. We report top-1 test accuracy (%) for each dataset. All values are averaged over 5 runs with random seeds.

Dataset	T=1	T=2	T=3	T=4	T=5
CIFAR-10 (CFL-CNN)	81.21	82.37	82.78	83.32	83.41
SpeechCommands (CFL-CNN)	89.22	90.14	90.52	91.27	91.31
ImageNet (CFL-CNN)	76.12	76.43	76.61	76.82	76.85
ImageNet (CFL-Transformer)	81.03	81.51	82.15	82.30	82.35

4.9. Analysis and Discussion

Our toy MLP study (Section 4.3) illustrates that hidden representations $\mathbf{h}$ can be refined to $\widetilde{\mathbf{h}}$ in real time, progressively lowering the error within a single forward pass. This iterative "self-correction" is a hallmark of feedback-based inference: it effectively realigns internal features with top-down cues, fostering more coherent final outputs.

Our experiments confirm that **Contextual Feedback Loops** can be readily integrated into various architectures (CNNs or transformers) to achieve stronger performance, greater robustness, and faster convergence—all with minimal computational overhead.

5. Conclusion

While standard backpropagation focuses on propagating prediction errors from labels, our approach injects higher-level context to disambiguate uncertain or complex inputs. By introducing **Contextual Feedback Loops**, we enable neural networks to iteratively integrate top-down context into their internal representations. This bridges the gap between purely bottom-up processing and the dynamic, feedback-driven reasoning observed in biological systems. The method is flexible and easily integrated into various architectures, potentially leading to more robust, interpretable, and context-aware models. In the future, we plan to extend CFL to larger-scale datasets, where leveraging top-down context can further improve learning efficiency.

References

Adigun, O. and Kosko, B. Bidirectional backpropagation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(5):1982–1994, 2020. doi: 10.1109/TSMC.2019.2916096.

Choksi, B., Mozafari, M., O'May, C. B., Ador, B., Alamia, A., and VanRullen, R. Predify: Augmenting deep neural networks with brain-inspired predictive coding dynamics, 2021. URL https://arxiv.org/abs/2106.02749.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Friston, K. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

Grossberg, S. Towards solving the hard problem of consciousness: The varieties of brain resonances and the conscious experiences that they support. *Neural Networks*, 87:38–95, 2017. ISSN 0893-6080. doi: 10.1016/j.neunet.2016.11.003. URL https://www.sciencedirect.com/science/article/pii/S0893608016301800.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. In *International Conference on Machine Learning (ICML)*, pp. 1627–1635, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, Toronto, ON, Canada, 2009. Also available at http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.

Rao, R. P. and Ballard, D. H. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

Spoerer, C. J., McClure, P., and Kriegeskorte, N. Recurrent convolutional neural networks: A better model of biological object recognition. *Frontiers in Psychology*, 8:1551, 2017.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition, 2018. URL https://arxiv.org/abs/1804.03209.

Wen, L., Du, D., Cai, Q., Lei, Z., Hung, T.-J., Senior, A., and Lyu, S. Recurrent attentive zooming for joint crowd counting and precise localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1217–1226, 2018.