What You Need to Know from R2F

Recover-to-Forget: Gradient Reconstruction from LoRA
(Liu et al., NeurIPS 2025 Workshop)

A Supervisor's Step-by-Step Guide for the Thesis

February 2026

Contents

1 Step 1: The Core Observation — Why R2F Works At All

The foundational insight is deceptively simple. When you fine-tune with LoRA, you are not learning arbitrary matrices A and B. You are learning matrices whose product BA approximates a **low-rank projection of the full gradient** $\nabla_W \mathcal{L}$.

Concretely, the LoRA gradient relationships are:

LoRA Gradient Identities

$$\nabla_A \mathcal{L} = B^\top \cdot \nabla_W \mathcal{L} \tag{1}$$

$$\nabla_B \mathcal{L} = (\nabla_W \mathcal{L}) \cdot A^\top \tag{2}$$

So A and B are trained by signals that are **linear projections** of the full gradient. The product BA is therefore a rank-r approximation of the full gradient update that *would have been* applied to W under full fine-tuning:

$$\Delta W_{\text{LoRA}} = BA \;\approx\; \text{rank-}r \text{ projection of } \Delta W_{\text{full}} = -\eta \nabla_W \mathcal{L} \tag{3}$$

Why This Matters for Your Thesis

This is not a heuristic — it is a mathematical identity. The decoder is not learning some arbitrary mapping; it is learning to **invert a well-structured linear projection** with nonlinear corrections.

What you should be able to explain in your defense:

> "The LoRA update BA lives in a rank-r subspace of the full gradient. The decoder learns to lift it back to the full-rank space. This is learnable because the projection is structured (not random) — it is determined by the optimization trajectory."

2 Step 2: The Gradient Decoder Architecture

This corresponds to Section 3 of the R2F paper.

2.1 Input/Output Specification

	Shape	Description
Input	$BA \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$	Flattened LoRA product (or A, B concatenated)
Output	$\hat{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$	Predicted full-rank gradient $\widehat{\nabla_W \mathcal{L}}$

Same dimensionality in and out. The decoder is learning to "fill in" the missing rank-$(d - r)$ components.

2.2 Network

- A small MLP — a few hidden layers, nothing large.
- **One decoder per layer** of the base model. The query projection decoder is different from the value projection decoder is different from the MLP up-projection decoder. Each weight matrix W_ℓ gets its own decoder f_{ϕ_ℓ} because the gradient geometry differs per layer.
- For structured gradients (2D weight matrices), they also explore U-Net-style decoders, but the MLP baseline already works.

2.3 Training Loss

Decoder Loss Function

$$\mathcal{L}_{\text{decoder}} = 1 - \cos\_\text{sim}\Big(f_\phi(BA),\ \nabla_W \mathcal{L}\Big) \tag{4}$$

where $\cos\_\text{sim}(\mathbf{u}, \mathbf{v}) = \dfrac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\,\|\mathbf{v}\|}$

Why cosine similarity, not MSE? Gradient inversion algorithms care about **direction** more than magnitude. A gradient that points the right way but has wrong scale will still reconstruct the correct image. MSE would waste capacity matching the scale.

2.4 Vision-Specific Dimensionality Note

For ViT-B/16, a query projection is $768 \times 768 = 590\text{K}$ parameters. For Llama-7B, the same projection is $4096 \times 4096 = 16.8\text{M}$. Your decoders will be significantly smaller and potentially easier to train. This is favorable for you.

3 Step 3: Training Data Generation (The "Proxy Data" Trick)

This is the most clever part of R2F and the part most students get wrong. Here is the exact procedure:

Decoder Training Data Generation

1. Take a **proxy dataset** $\mathcal{D}_{\text{proxy}}$ (public data, does NOT need to match the victim's private data).
2. Take the **same base model architecture** (e.g., ViT-B/16).
3. For each of $\sim$50,000 iterations:
   (a) Sample a mini-batch $(x, y) \sim \mathcal{D}_{\text{proxy}}$.
   (b) Compute the full gradient: $G = \nabla_W \mathcal{L}(W_0 + BA;\, x, y)$.
   (c) Do **one step** of LoRA SGD. Record A_1, B_1 after the step.
   (d) Store the training pair: $(B_1 A_1,\ G)$.
   (e) **Reset** A, B back to initialization.
4. You now have 50K pairs for training each per-layer decoder.

3.1 Critical Subtleties

1. Single-step only. Each training example is one gradient step, not a converged adapter. The decoder learns the *local* relationship between BA and $\nabla_W \mathcal{L}$ at a single optimization step.

2. The proxy data doesn't need to match. The decoder is learning the **geometry of the projection** — how information distributes across singular values of the gradient — not the data semantics. CIFAR-100 can proxy for faces; WikiText can proxy for clinical notes. This is what makes the attack practical: you do not need to know what the victim trained on.

3. Reset after each step. You are not training LoRA to convergence and then reading off BA. You are sampling individual gradient steps from the same initialization. This keeps the pairs i.i.d. and prevents the decoder from overfitting to a specific optimization trajectory.

3.2 Handling Multi-Step Adapters (The Real Attack Scenario)

When the victim has trained LoRA for T steps, the final adapter is an accumulation:

$$B_T A_T \approx \sum_{t=1}^{T} \eta_t \nabla_W \mathcal{L}_t \tag{5}$$

R2F handles this by treating the final BA as approximately proportional to the **average gradient**:

$$B_T A_T \propto \frac{1}{T} \sum_{t=1}^{T} \nabla_W \mathcal{L}_t \tag{6}$$

Ablation Required

This is an approximation that degrades with training length T. Your thesis should ablate over T: how does reconstruction quality degrade as the victim trains for more steps?

4 Step 4: Cross-Model Transfer (Proposition 1)

R2F's Proposition 1 provides a theoretical bound for when you can train the decoder on one model (proxy) and apply it to another (target). The key conditions:

1. The two models share the same architecture (or at least the same layer dimensions).
2. The gradient distributions are "close" in a distributional sense (domain adaptation bound).

The bound takes the form:

$$\mathcal{L}_{\text{target}}(f_\phi) \leq \mathcal{L}_{\text{proxy}}(f_\phi) + d_{\mathcal{H}}(\mathcal{P}_{\text{proxy}}, \mathcal{P}_{\text{target}}) + \lambda \tag{7}$$

where $d_{\mathcal{H}}$ is the $\mathcal{H}$-divergence between gradient distributions and λ is the combined optimal error.

Relevance to Your Thesis

This is less critical in the image domain because you will likely train and attack on the **same architecture** (e.g., ViT-B/16). But it is worth mentioning in your thesis because it shows the decoder is not memorizing model-specific quirks — it is learning something **structural** about the rank-r $\rightarrow$ full-rank mapping.

5 Step 5: Where R2F Stops and Your Thesis Begins

R2F recovers full-rank gradients and then does **gradient ascent** for unlearning. That is their downstream application. Your thesis diverges here completely.

5.1 Your Pipeline

$$\boxed{BA_{\text{victim}}} \xrightarrow{\text{Gradient Decoder } f_\phi} \boxed{\hat{G} \approx \nabla_W \mathcal{L}} \xrightarrow{\text{Gradient Inversion}} \boxed{\hat{x} \approx x_{\text{train}}} \xrightarrow[\text{(optional)}]{\text{SDS Prior}} \boxed{\hat{x}_{\text{enhanced}}}$$

5.2 Quality Requirements: Unlearning vs. Reconstruction

	R2F (Unlearning)	Your Thesis (Reconstruction)
Gradient use	Gradient ascent	Gradient inversion
Tolerance to noise	High	**Low**
What matters	Direction (roughly)	Direction **and** magnitude
Output	Model behavior change	Pixel-level images

Key Implication

Unlearning is tolerant of noisy gradients — you just need the gradient to approximately point in the right direction. Gradient inversion for pixel-level image reconstruction is **far more demanding**. Small errors in the decoded gradient will produce visible artifacts. This is precisely why you need the generative prior (Direction 3 of your thesis: SDS from a frozen diffusion model).

6 Step 6: What R2F Does NOT Tell You (Gaps You Must Fill)

1. **Vision-specific gradient structure.** R2F works on LLMs. Gradients of vision models have spatial structure (feature maps, patch embeddings) that LLM gradients lack. Your decoder might benefit from **convolutional architectures** rather than pure MLPs.

2. **Batch size sensitivity.** R2F assumes single-sample gradients for unlearning. If the victim fine-tuned with batch size $B > 1$, the gradient is an average over the batch:

$$\nabla_W \mathcal{L} = \frac{1}{B} \sum_{i=1}^{B} \nabla_W \ell(W; x_i, y_i) \tag{8}$$

   Inverting an averaged gradient to recover all B individual images is a known hard problem. You need to figure out the practical batch size limit.

3. **Which LoRA layers to decode.** R2F decodes all layers. For gradient inversion, not all layers carry equal information about the input. Early layers (close to the input) are more informative for pixel reconstruction. You need to determine which layers to prioritize.

4. **The cosine similarity → inversion quality relationship.** R2F reports > 0.9 cosine similarity between predicted and true gradients. But what cosine similarity do you *need* for gradient inversion to work? This is an open empirical question.

   Phase 0 Answers This

   Your Phase 0 experiment (perfect gradient → inversion) establishes the **upper bound**. If perfect gradients do not produce recognizable images on your chosen architecture, then the entire Gradient Bridge direction collapses regardless of decoder quality.

7 Step 7: Reading List (Priority Order)

1. **Geiping et al., "Inverting Gradients" (NeurIPS 2020)** — The attack engine that converts gradients back to inputs. You cannot build the pipeline without understanding this. Focus on:

- The cosine similarity loss for inversion (same spirit as the decoder loss in Eq. 4)
- The total variation regularizer
- How they handle batch reconstruction

2. **R2F Section 3 in full detail** — Read the actual equations and the decoder training loop pseudocode. Do not rely solely on summaries.

3. **Jang et al. (2024) on LoRA in the NTK regime** — This justifies Direction 2 of your thesis: when LoRA rank $r \gtrsim N$ (number of data points), you can skip the decoder entirely and apply KKT conditions directly to the LoRA weights.

8 The Bottom Line

R2F gives you exactly one thing: a **learnable bridge from LoRA space to gradient space**. Everything else — the inversion, the generative prior, the vision-specific adaptations — is *your contribution.*

The key insight to internalize:

Core Takeaway

LoRA updates are not opaque compression artifacts. They are **structured, invertible measurements** of the full gradient, and the inversion is learnable from proxy data alone.

Your Phase 0 experiment is the most important thing to do first. If perfect gradients do not invert to recognizable images on your chosen architecture, then the entire Gradient Bridge direction collapses regardless of how good the decoder is.

Start there.