Project Notes & Findings

Research Log

Started: Dec 2025, Last update: Jan 2026

1 Introduction

The first objective of this project is to challenge our understanding of regularization in continual learning, and more specifically explain why and when quadratic regularizers are effective and when they fail.

2 Background on quadratic approximations

The most widely used regularizers are quadratic functions in θ, which typically consist in a Mahlanobis distance from an anchor point, usually chosen to be the model parameters at the end of the previous observation cycle $\theta^\star_{t-1}$. In formula:

$$g_{t-1}(\theta) = (\theta - \theta^\star_{t-1})^\top Q_{t-1}(\theta - \theta^\star_{t-1}), \tag{1}$$

where Q_{t-1} is a positive semi-definite matrix encoding the metric. The first use of quadratic regularization was in Elastic Weight Consolidation (EWC), where Q_{t-1} is the diagonal of the Fisher Information Matrix. The regularizer can be interpreted as a replacement of the previous tasks' cumulative loss, $\ell_{1:t-1}(\theta) = \sum_{i=1}^{t-1} \ell_i(\theta)$, where ℓ_i is the loss on task i.

The choice of approximation eq. (1) may seem too restrictive and inaccurate for neural networks. The formula can be easily derived from the second-order Taylor expansion of the loss function $\mathcal{T}_2(\theta; \ell_{1:t-1}, \theta^\star_{t-1})$.

$$\mathcal{T}_2(\theta; \ell_{1:t-1}, \theta^\star_{t-1}) = \nabla \ell_{1:t-1}(\theta^\star_{t-1})^\top (\theta - \theta^\star_{t-1}) + \tfrac{1}{2}(\theta - \theta^\star_{t-1})^\top \nabla^2 \ell_{1:t-1}(\theta^\star_{t-1})(\theta - \theta^\star_{t-1}) + \text{cnst}.$$

When $\theta^\star_{t-1}$ is a local minimum, the first term vanishes, and one recovers the quadratic form with $Q_{t-1} = \nabla^2 \ell_{1:t-1}(\theta^\star_{t-1})$ (up to a constant term). Moreover, when the model fits the data well (i.e., the error is close to zero), the Hessian is dominated by its Generalized Gauss-Newton (GGN) component:

$$\nabla^2_\theta \ell_{1:t-1} = \underbrace{\sum_{i=1}^{t-1} J_i^\top H_i \, J_i}_{\text{GGN}} + \underbrace{\sum_{i=1}^{t-1} \sum_{k=1}^{C} \frac{\partial \ell_i}{\partial f_{k,i}} \nabla^2_\theta f_{k,i}}_{\text{Model Curvature}}$$

because the so-called *residuals* $\frac{\partial \ell_i}{\partial f_{k,i}}$ are small near the optimum. Here, J_i is the Jacobian of the model outputs with respect to parameters, and H_i is the Hessian of the loss with respect to model outputs.

For negative log-likelihood objectives in the exponential family, the GGN coincides exactly with the Fisher Information Matrix:

$$F = \mathbb{E}_{x,y\sim p_{data}} \left[\nabla_\theta \log p_y(x) \nabla_\theta \log p_y(x)^\top \right],$$

where, in the case of a multi-class classification problem, $p_y(x)$ is the predicted probability of class y given sample x, obtained by applying the softmax to the model logits $f_\theta(x)$. Focusing on the classification case, for C classes, this is computed as:

$$F_{t-1} = \frac{1}{t-1} \sum_{i=1}^{t-1} \sum_{k=1}^{C} p_c(x_i) \nabla_\theta \log p_c(x_i) \nabla_\theta \log p_c(x_i)^\top$$

The Fisher might be estimated more efficiently using only the observed class, leading to the so-called *Empirical Fisher* approximation:

$$\hat{F}_{t-1} = \frac{1}{t-1} \sum_{i=1}^{t-1} \nabla_\theta \log p_{y_i}(x_i) \nabla_\theta \log p_{y_i}(x_i)^\top$$

Note that for a single input x_i, the empirical Fisher matrix is of rank one. Also note that in the limit of zero error and maximal confidence, both the empirical Fisher and the true Fisher matrices vanish in the case of multi-class classification. The same is not true for the regression case, where the true Fisher never vanishes.

The Fisher offers two distinct advantages over the exact Hessian: (1) it is guaranteed to be positive-semi-definite, ensuring convexity and non-negativity and (2) it can be computed from first derivatives only, which are cheaper than second derivatives. For very large models, materializing the full $P \times P$ matrix in memory may be infeasible (where P is the size of the model); thus, practical implementations rely on diagonal or block-diagonal approximations.

The derivation from Taylor expansion exposes a fundamental limitation of all approximations in the form of eq. (1): like the expansion itself, they are accurate only within a local neighborhood of the center $\theta^\star_{t-1}$. In practice, regularization methods are often used "improperly" allowing the model to drift far outside the region where the quadratic approximation remains valid (i.e., without a strictly enforced *trust region*). We believe that this fact explains the relative lower performance of regularization methods compared to other continual learning algorithms, and their sensitivity to hyperparameters influencing the local geometry—such as learning rate, batch size, and model capacity. Moreover, to minimize the cost, many methods update the metric Q_{t-1} using only the most recent observation x_t, exploiting the additivity of the Fisher/Hessian. Therefore, the regularizer becomes:

$$g_{t-1}(\theta) = \sum_{i=1}^{t-1} (\theta - \theta^\star_i)^\top Q_i(\theta^\star_i)(\theta - \theta^\star_i)$$

While this reduces the complexity of the regularizer update from $O(t)$ to $O(1)$, it introduces approximation errors as it further increases the distance between the current parameters θ and the approximation anchor points $\theta^\star_i$ for $i < t-1$.

Our goal in this short paper is to prove this hypothesis, and quantifying the effect of each source of error on the final performance of quadratic regularizers.

3 Sources of error

Based on the above exposition, we identify the following sources of error:

- **The refresh rate of the approximation.** The curvature matrix Q_i is computed at the minimum of task i. When the model drifts far from this point during training on subsequent tasks, the approximation becomes inaccurate. We call this choice the *'accumulate'* option, and compare it with a strategy where the curvature is recomputed at every task end (the *'reset'* option).
- **The curvature matrix choice.** We compare different choices for the curvature matrix Q_i, including the Hessian, the Generalized Gauss-Newton, and the Empirical Fisher Information Matrix.
- **The optimization implicit bias.** We investigate the effect of the choice of training trajectory on the approximation accuracy. In particular, we compare a standard gradient descent trajectory (called *'sequential'*) with one that adds the regularization term to the objective (called *'regularized'*) and finally with a trajectory that employs *'replay'* of all past data.

Metrics We compute the regularizer on a subset of the task data, namely a random 10% of the training set. For each sample, we track the following metrics during training (at each step):

- **Approximation Error.** We measure the relative error between the true loss increase and its quadratic approximation at each training step:

$$\kappa(x,\theta) = \ell(x,\theta) - g(x,\theta) - \ell(x,\theta^\star_{t-1})$$

  where $g(x,\theta)$ is defined as in eq. (1), and the curvature matrix used is computed on x only. Additionally, we track the gradient error in norm and angle:

$$\gamma(x,\theta) = \|\nabla_\theta \ell(x,\theta) - \nabla_\theta g(x,\theta)\|_2 \quad \text{and} \quad \alpha(x,\theta) = \cos\left(\nabla_\theta \ell(x,\theta), \nabla_\theta g(x,\theta)\right).$$

- **Task Performance.** We track the accuracy on the current task, the average accuracy across all previous tasks, and the average accuracy across regularized samples.
- **Several landscape metrics.** We additionally track the sharpness, rank and trace of the curvature matrix for the current task and for the regularized samples.

Initial results. We executed some initial experiments on a small model (one hidden layer MLP with parameters) on a sequence of 9 binary classification tasks in 2D.

Note. The regularization strength has not been optimized, and the optimization hyperparameters are the same across all methods for comparison.

Table 1: Impact of Curvature Refreshing on Accuracy and Approximation Fidelity (Taylor-Full)

Strategy	Past Avg (↑)	Curr. Acc (↑)	Reg. Acc (↑)	κ Loss (↓)	κ Grad (↓)
			Final Step		
Accumulate	0.68 ± 0.04	$\mathbf{0.88} \pm 0.16$	0.67 ± 0.10	3.25 ± 1.44	8.50 ± 1.73
Refresh	$\mathbf{0.72} \pm 0.01$	0.71 ± 0.04	$\mathbf{0.73} \pm 0.01$	$\mathbf{0.04} \pm 0.03$	$\mathbf{6.93} \pm 0.21$
			Trajectory Average		
Accumulate	0.63 ± 0.01	0.84 ± 0.02	0.69 ± 0.05	2.18 ± 0.68	8.47 ± 1.20
Refresh	$\mathbf{0.64} \pm 0.01$	$\mathbf{0.85} \pm 0.00$	$\mathbf{0.72} \pm 0.03$	$\mathbf{0.22} \pm 0.03$	$\mathbf{7.50} \pm 0.71$

The refresh rate of the approximation. In table 1, we compare the 'accumulate' and 'refresh' strategies for curvature computation using the True Fisher Information Matrix. The results indicate that refreshing the curvature at each task end improves both the overall accuracy and the fidelity of the quadratic approximation, as evidenced by lower κ loss and gradient errors. However, the performance is overall moderate. When comparing regularization to sequential training and replay (fig. 1), we observe that regularization achieves performance on par with sequential training and replay on the current task (second column) on almost all tasks. Therefore, the lower average performance is due to forgetting.

The optimization implicit bias. In table 2, we compare the three training modes: sequential, regularized (with curvature refresh), and replay on the same metrics as before. Replay significantly outperforms both other methods across all metrics. Crucially, replay scores the lowest approximation errors. What is more, the same low approximation errors are scored even when not refreshing the curvature (as shown in table 3), indicating that **replay helps maintain the model within the validity region of the quadratic approximation**

Table 2: Cross-Mode Performance Comparison: Sequential, Regularized (Refresh), and Replay. Values reported as mean ± standard deviation.

Mode	Past Avg (↑)	Curr. Acc (↑)	Reg. Acc (↑)	κ Loss (↓)	κ Grad (↓)
			Final Step		
Sequential	0.68 ± 0.06	$\mathbf{0.95} \pm 0.03$	0.69 ± 0.07	0.71 ± 0.42	5.01 ± 1.04
Regularized	0.71 ± 0.01	0.81 ± 0.17	0.70 ± 0.05	0.13 ± 0.12	7.21 ± 0.58
Replay	$\mathbf{0.91} \pm 0.02$	0.93 ± 0.04	$\mathbf{0.93} \pm 0.02$	$\mathbf{0.11} \pm 0.04$	$\mathbf{3.26} \pm 0.43$
			Trajectory Average		
Sequential	0.62 ± 0.02	$\mathbf{0.88} \pm 0.01$	0.68 ± 0.06	0.97 ± 0.36	7.32 ± 1.12
Regularized	0.64 ± 0.01	0.85 ± 0.01	0.69 ± 0.05	0.42 ± 0.11	7.83 ± 0.82
Replay	$\mathbf{0.78} \pm 0.01$	0.87 ± 0.01	$\mathbf{0.89} \pm 0.02$	$\mathbf{0.22} \pm 0.02$	$\mathbf{4.16} \pm 0.42$

From these results we also conclude that regularization only marginally improves performance over sequential training. This is further confirmed by fig. 2, which shows the training accuracy on previous tasks across training steps.

The curvature matrix approximation. ...

Table 3: Comparison of Approximation Fidelity (κ metrics) across Non-Refreshed (Stale) Methods.

	Final Step		*Trajectory Average*	
Mode	κ Loss (↓)	κ Grad (↓)	κ Loss (↓)	κ Grad (↓)
Sequential	2.34 ± 1.06	9.35 ± 1.48	3.54 ± 1.20	13.44 ± 1.37
Regularized	3.49 ± 1.47	8.50 ± 1.73	2.40 ± 0.66	8.47 ± 1.20
Replay	$\mathbf{0.46} \pm 0.18$	$\mathbf{4.74} \pm 0.73$	$\mathbf{0.44} \pm 0.16$	$\mathbf{5.33} \pm 0.93$

Table 4: Final Step Comparison: Curvature Approximations. (✓) signals refreshed curvature; (×) signals stale curvature.

Matrix	**Curvature**	**Upd.**	**Past Avg** (↑)	**Reg. Acc** (↑)	κ **Loss** (↓)	κ **Grad** (↓)
Block	Fisher	✓	$0.69 \pm$ —	$0.70 \pm$ —	$0.17 \pm$ —	$6.92 \pm$ —
		×	0.68 ± 0.02	0.65 ± 0.08	2.22 ± 0.03	7.87 ± 1.63
	True Fisher	✓	0.71 ± 0.01	0.69 ± 0.07	0.16 ± 0.15	7.42 ± 0.63
		×	$0.66 \pm$ —	$0.63 \pm$ —	$2.87 \pm$ —	$9.12 \pm$ —
Diag	Fisher	✓	0.65 ± 0.03	0.65 ± 0.05	0.19 ± 0.07	7.13 ± 0.26
		×	0.65 ± 0.00	0.65 ± 0.02	1.52 ± 0.37	6.73 ± 0.27
	True Fisher	✓	0.68 ± 0.04	0.65 ± 0.02	0.33 ± 0.10	6.85 ± 0.56
		×	0.67 ± 0.02	0.64 ± 0.02	1.49 ± 0.20	7.33 ± 0.27
Full	Fisher	✓	0.67 ± 0.06	0.66 ± 0.09	0.33 ± 0.33	7.68 ± 1.47
		×	0.70 ± 0.03	0.67 ± 0.08	2.28 ± 0.47	7.55 ± 1.54
	True Fisher	✓	$\mathbf{0.72} \pm 0.01$	$\mathbf{0.73} \pm 0.01$	$\mathbf{0.05} \pm 0.03$	$\mathbf{6.93} \pm 0.21$
		×	0.68 ± 0.04	0.67 ± 0.10	3.49 ± 1.47	8.50 ± 1.73

4 On the failure modes of regularization.

okay and finally we ask: what is causing regularization to fail so poorly? Clearly one option is the trust region violation, another option is that the approximation is wrong to start with because we are not at a minimum, and the last option is simply that the regularization strength is not well tuned.

So, how would we expect regularization to behave? basically it should prevent the optimization dynamics from going in the directions that are deemed important by the model. So if we look at the projection onto these directions we should see a lower alignment than without regularization... what do we see? So does regularization work as expected? if yes and still breaks -¿ it's really the trust region if no -¿ then regularization in itself is not working as expected and needs to be fixed

if 2, then we try with projection. This should work. But if we still do not see an improvement then it is the trust region that is the problem.

Probably even fixing all the above we still get a bit of difference to replay, due to the trust region.

5 Is there a difference between samples?

6 Learning regimes: regularization works for the lazy.

Figures

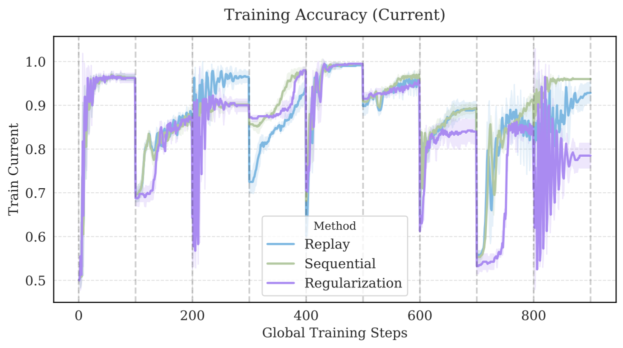

Figure 1: Training Accuracy on the Current Task across Training Steps for Different Methods.

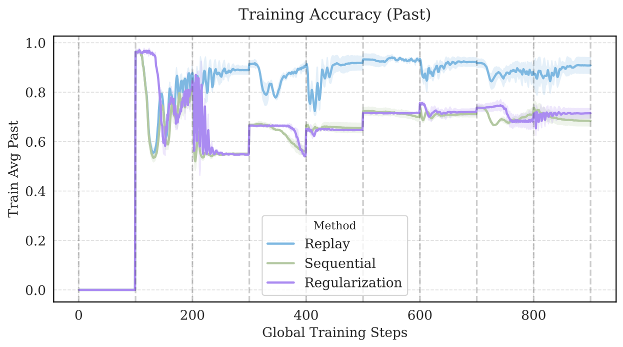

Figure 2: Training Accuracy on the Previous Tasks across Training Steps for Different Methods.