On the Theory of Continual Learning with Gradient Descent for Neural Networks

Hossein Taheri[1], Avishek Ghosh[2], Arya Mazumdar[1]

June 2026

[1] University of California, San Diego
[2] Indian Institute of Technology, Bombay

Continual Learning with Neural Nets

- In many real-world applications, data are not static:
  - medical diagnostics: new diseases and evolving patient populations,
  - finance: market conditions shift over time,
  - robotics: agents must adapt to new environments.

- Standard learning theory often assumes i.i.d. training data from a **fixed** distribution.

- In practice, data often arrive **sequentially** and from a **non-stationary** stream of tasks.

Motivation

- Neural networks trained with gradient methods on new tasks often suffer from **catastrophic forgetting**.
- Example:
  - train on Task A → good performance on Task A,
  - then train on Task B → performance on Task A drops.
- **Goal of continual learning:** learn new tasks while preserving previously acquired knowledge.

Continual Learning Setup

We consider a sequence of tasks

$$\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K, \qquad \mathcal{D}_k = \{(x_i^{(k)}, y_i^{(k)})\}_{i=1}^{n_k}.$$

The sequential objective is to minimize

$$\min_{\theta} \sum_{k=1}^{K} \mathbb{E}_{(x,y)\sim\mathcal{D}_k} \mathcal{L}(f_\theta(x), y).$$

At task k, the parameters are updated using only the current task's data:

$$\theta_{t+1}^{(k)} = \theta_t^{(k)} - \eta \nabla_\theta \widehat{\mathcal{L}}_k(\theta_t^{(k)}).$$

After training on $\mathcal{D}_k$, we move to $\mathcal{D}_{k+1}$ without revisiting old data.

Algorithm: (Regularized) Continual Learning with GD

1. Initialize
$$w_1^{(0)} \sim \mathcal{N}(0, I_p).$$
2. For $k = 1, \ldots, K$:
   2.1 Load task-specific dataset $\mathcal{D}_k$.
   2.2 For $t = 0, \ldots, T-1$:
$$w_k^{(t+1)} \leftarrow w_k^{(t)} - \eta \left(\nabla \widehat{F}\left(w_k^{(t)}; \mathcal{D}_k\right) + \nabla_{w_k^{(t)}} \mathcal{R}\left(w_k^{(t)}, w_{k-1}^{(T)}\right) \right),$$
   2.3 Set
$$w_{k+1}^{(0)} \leftarrow w_k := w_k^{(T)}.$$
3. Return
$$w_K := w_K^{(T)}.$$

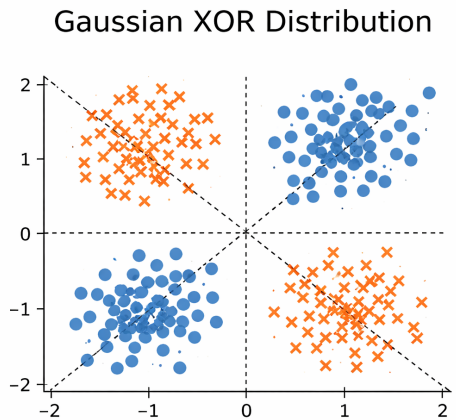
Questions

- How can we characterize **train-time** and **Test-time forgetting**?
- Is successful continual learning possible with neural nets?
- what is the impact of regularization, width or sample complexity on forgetting?

Data Model: XOR Cluster

For task k, we consider the XOR-cluster distribution:

$$x_i^{(k)} \sim \begin{cases} \frac{1}{2}\mathcal{N}(\mu_1^{(k)}, \sigma^2 I_d) + \frac{1}{2}\mathcal{N}(-\mu_1^{(k)}, \sigma^2 I_d), & y_i^{(k)} = 1, \\ \frac{1}{2}\mathcal{N}(\mu_2^{(k)}, \sigma^2 I_d) + \frac{1}{2}\mathcal{N}(-\mu_2^{(k)}, \sigma^2 I_d), & y_i^{(k)} = -1 \end{cases}$$



- This is a representative non-linear data model.
- Different tasks correspond to different cluster directions.
- It is rich enough to study forgetting in neural networks analytically.
- The theory can be extended to other data distributions as well.

Train-/Test-time Forgetting

Train-/Test- time forgetting: Increase in empirical/population loss on an earlier task after learning later tasks.

Theorem (Forgetting bounds)

For a two-layer quadratic network with ***width*** **m**, *trained for* **T** ***GD iterations*** *per task using* **n** ***samples per task***, *the train-time forgetting after* **k** ***tasks*** *scales as*

$$\hat{\mathcal{F}}_k = \widetilde{\mathcal{O}}\left(\eta T\sqrt{\frac{k}{d^2 n}} + \eta T\frac{\sqrt{k}}{d^2\,\mathrm{polylog}(d)} + \frac{\eta^2 T^2 k^2}{\sqrt{m}}\right), \qquad \textit{(Train-time)}$$

$$\mathcal{F}_k \le \hat{\mathcal{F}}_k + \frac{\eta T e^{\frac{\eta T k}{\sqrt{m}}}}{n}. \qquad \textit{(Test-time)}$$

Sample and iteration complexities

- Train and test-time forgetting error do *not* vanish unless both sample size and network size are large enough.

- For learning k tasks without forgetting:
  $n = \widetilde{\Theta}\left(d^2 k\right), m = \Omega\left(d^8 k^4\right), T = \widetilde{\Theta}\left(d^2\right).$

Extensions to General Data Distributions

Theorem (Forgetting for general data)

Assume learning k tasks sequentially with a quadratic NN with ***width m****, trained for* ***T iterations*** *of GD per task using* ***n samples*** *per task. Train-time forgetting after learning k tasks is*

$$\hat{\mathcal{F}}_k = \left| \frac{1}{n} \sum_{x_k} \eta T x_k^\top \left(\sum_{j=k+1}^{K} A_j \right) x_k \right| + O\left(\frac{\|w_K - w_0\|^2}{\sqrt{m}} \right),$$

$A_j := \frac{1}{n} \sum_{v=1}^{n} y_j^v x_j^v \left(x_j^v\right)^\top$, *and* $\left\{ \left(x_j^v, y_j^v\right) \right\}_{v \in [n]}$ *denotes the training data for task j.*

Some prior works

- Prior works focused on linear models [Evron et al, 2024,Banayeeanzadeh et al 2024].
- We obtained closed-form bounds for quadratic NNs.
- The bounds show the role of different parameters and show that successful continual learning is possible even without regularization.

Experiments on the Gaussian XOR Cluster

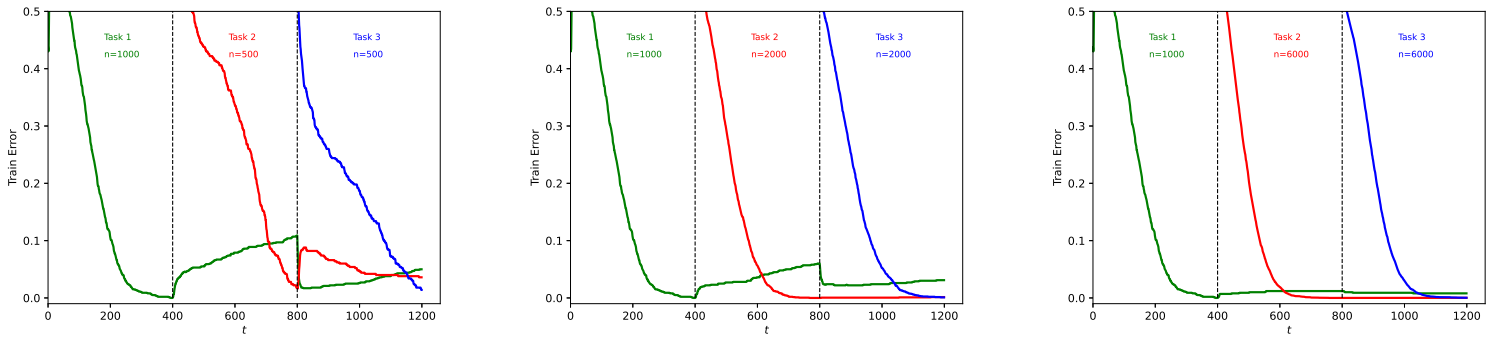


Figure 1: Train error for each task vs. iterations for the XOR cluster with $K = 3$ tasks. We fix $n = 2500$ for the first task and increase the sample size of the second and third tasks across figures. Increasing the sample size *decreases forgetting* for *previous tasks*.

Experiments on MNIST

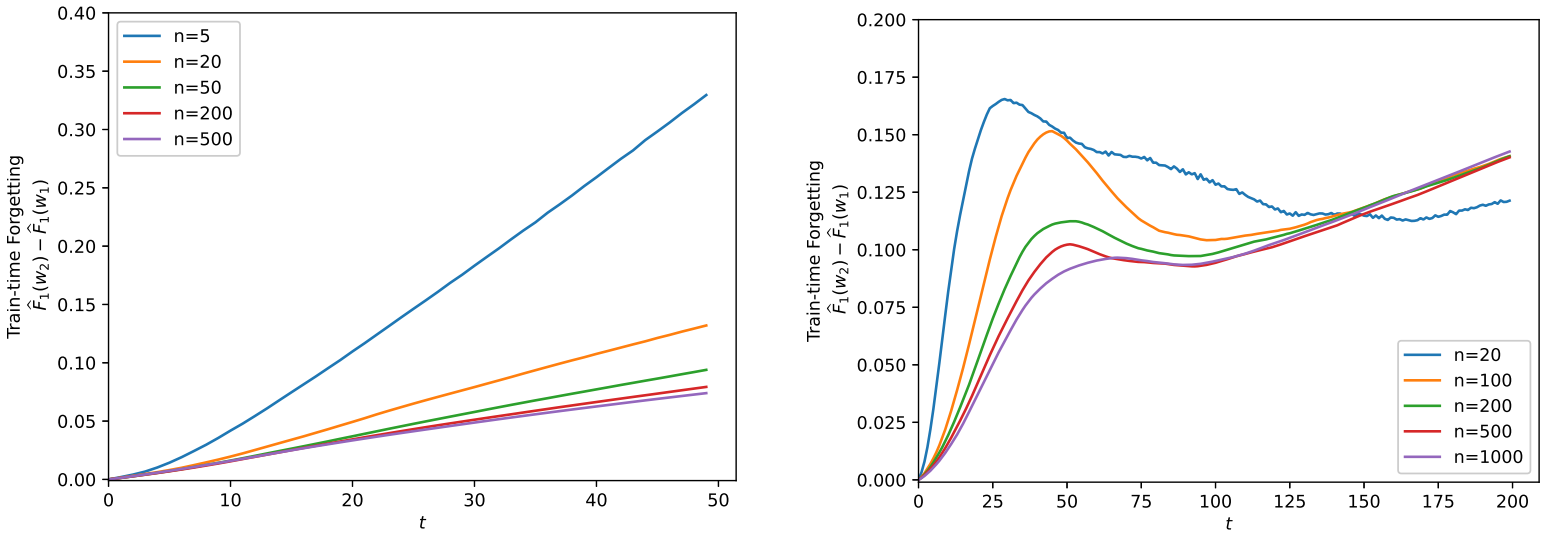


Figure 2: Train-time forgetting for task 1 during training on task 2 for different sample sizes from the MNIST dataset. Increasing n for the second task decreases forgetting of the previous task.

Conclusions

The main message: Continual learning is not just an empirical phenomenon. For different models and distributions, we can mathematically study when learning new tasks causes forgetting of old ones.

Future directions:

- Characterizing when knowledge transfer and forgetting happens depending on data distribution.

- Forgetting bounds for feature learning.