Fixing Neural Networks by Leaving the Right Past Behind

Ryutaro Tanno
Microsoft Research Cambridge, UK
rytanno@microsoft.com

Melanie F. Pradier
Microsoft Research Cambridge, UK
melanief@microsoft.com

Aditya Nori
Microsoft Research Cambridge, UK
Aditya.Nori@microsoft.com

Yingzhen Li
Imperial College London, UK
yingzhen.li@imperial.ac.uk

Abstract

Prediction failures of machine learning models often arise from deficiencies in training data, such as incorrect labels, outliers, and selection biases. However, such data points that are responsible for a given failure mode are generally not known a priori, let alone a mechanism for repairing the failure. This work draws on the Bayesian view of continual learning, and develops a generic framework for both, identifying training examples which have given rise to the target failure, and fixing the model through erasing information about them. This framework naturally allows leveraging recent advances in continual learning to this new problem of model repairment, while subsuming the existing works on influence functions and data deletion as specific instances. Experimentally, the proposed approach outperforms the baselines for both identification of detrimental training data and fixing model failures in a generalisable manner.

1 Introduction

Machine learning (ML) models often exhibit unexpected failures once deployed in the "wild". Recent lines of research aim to alleviate different well-known shortcomings in supervised models, such as vulnerability to annotation errors [1] and adversarial attacks [2], sensitivity to data shifts [3], and biases to underrepresented subgroups [4, 5, 6]. However, it is often challenging to anticipate beforehand all plausible failure scenarios, and protect against them pre-emptively. This motivates developing a technique that is able to *repair a model on demand*, as new failure cases arise in practice.

Undesirable behaviours of ML models commonly stem from defects in the training data. However, it is unclear how to detect the causes of such failures automatically, rendering a manual troubleshooting necessary. Furthermore, once the problems are uncovered, one would still need to design fixes, which typically involve further data curation/collection, and model retraining/redesigning from scratch. Executing the above steps demand not only time, but also mature expertise in the relevant ML areas, a scarcity in the present job market.

This work introduces an approach to identifying a set of most detrimental training examples that have caused failure cases observed at test time, and to subsequently repairing the model on these failures by deleting those culprits. At the basis of both cause identification and repairment steps is the approximation of *"counterfactual" posterior distribution* where some training examples are assumed absent. We formalise this as a Bayesian *continual (un)learning* problem [7], where the above counterfactual posterior is estimated by deleting the evidence of selected training data from the current posterior. We note that, while other factors (e.g. model class and optimisation) may play a role, we focus on "data debugging" and investigate, to what extent, prediction failures could be remedied by only intervening on data and updating the model accordingly.

Fig. 1 gives an overview of the proposed approach for model repairment, which operates in two steps by 1) identifying causes of failure among training data, and 2) updating the model by erasing the "memories" of those harmful examples. Importantly, the proposed framework is agnostic to *why* a particular

Preprint. Under review.

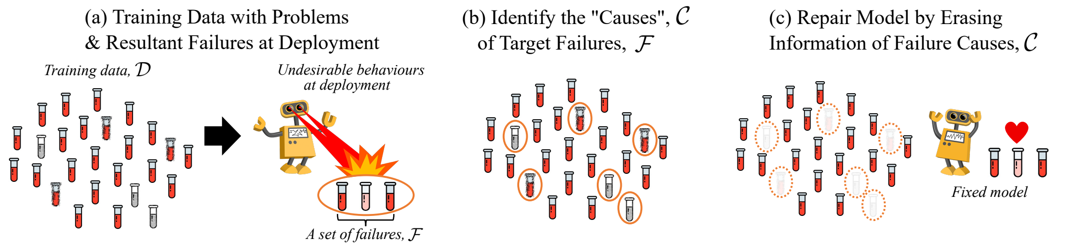

(a) Training Data with Problems & Resultant Failures at Deployment
Training data, $\mathcal{D}$
Undesirable behaviours at deployment
A set of failures, $\mathcal{F}$
(b) Identify the "Causes", $\mathcal{C}$ of Target Failures, $\mathcal{F}$
(c) Repair Model by Erasing Information of Failure Causes, $\mathcal{C}$
Fixed model

Figure 1: (a) Real-world datasets are often fraught with issues such as annotation noise, low-quality inputs, anomalies, and acquisition biases (e.g. demographic imbalances). Such issues may lead to undesirable performance of the trained models in deployment. Our approach aims at repairing such models by (b) identifying detrimental training examples which have caused the target failures, and then (c) erasing efficiently the *memories* of those examples from the models.

datapoint hurts model performance, handling both, issues in the input data and/or labels. We only require specification of the set of failed cases for which we wish to improve performance.

Our contributions: We develop a framework for *repairing* machine learning models by erasing memories of detrimental datapoints. The framework connects both identification and removal of detrimental data under a (Bayesian) continual learning perspective, which brings forth practical benefits. Firstly, the framework subsumes works on influence function [8] and data deletion [9] as specific examples, which are developed independently, and our work reveals their close connections and limitations. Secondly, the generality of our formulation allows translating any continual learning method into this model repairment setting, and opens doors to further research. In particular, we extend Elastic Weight Consolidation [10] – a specific continual learning algorithm – to cause identification and data removal, and demonstrate improvements over the prior works in a variety of settings where training data are contaminated with annotation and/or input noise.

2 Model Repairment by Data Deletion

Let us consider a prediction model $p(y|\boldsymbol{\theta},\boldsymbol{x})$ that returns a probability distribution of the output y given an input $\boldsymbol{x}$. We make the i.i.d. modelling assumption and denote $p(\boldsymbol{\theta}|\mathcal{D})$ as the posterior distribution over the model parameters $\boldsymbol{\theta}$ given training data $\mathcal{D}=\{(\boldsymbol{x}^{(n)},y^{(n)})\}_{n=1}^{N}$. At test time, the posterior predictive distribution is used to infer label $y^\star$ given a new sample $\boldsymbol{x}^\star$:

$$p(y^\star|\boldsymbol{x}^\star,\mathcal{D})=\int p(y^\star|\boldsymbol{x}^\star,\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}, \tag{1}$$

where $p(y^\star|\boldsymbol{x}^\star,\boldsymbol{\theta})$ is the likelihood term for sample $(x^\star,y^\star)$. While we use approximate posteriors in practice, we focus on the exact case for now to formalise the problem.

Imagine that, in deployment, this model makes incorrect predictions in certain situations. After collecting a set of examples from such failure mode in the test set[1], $\mathcal{F}=\{(\boldsymbol{x}_f^{(n)},y_f^{(n)})\}_{n=1}^{N_f}$, we would like to *repair* the model, such that it improves performance on the failure set $\mathcal{F}$ and similar future cases. We also argue that a successful repairment should also *maintain* a similar level of performance on the rest of test examples. These two objectives of model repairment are analogous to those of a *medical treatment*, in that both aim to fix a specific problem while leaving the “healthy” part as intact as possible. A further discussion of different aspects, including generalisation, efficiency and specificity properties is provided in Appendix A.1.

In this work, we assume that the main reason for such failures $\mathcal{F}$ is due to the existence of detrimental examples in the training data $\mathcal{D}$, for example, noisy labels, low-quality inputs and/or group imbalances. Our hypothesis is that, by removing these harmful datapoints and adapting the model accordingly, the model can be repaired to return correct predictions for datapoints in $\mathcal{F}$ as well as similar future cases. We acknowledge that there might be other reasons for a model to make wrong predictions on $\mathcal{F}$, such as optimisation and model biases, which are outside the scope of this work. Addressing data-based failures is complementary to accounting for other problems and is of relevance regardless, as datasets typically come with unexpected issues no matter how much we curate them beforehand.

[1]There may be multiple different ways in which the model fails, and the failure cases may consist of several groups. In practice, one type of mistake might incur more costs than others (e.g. in certain medical applications, *false negative* is more costly than *false positive*). Here we assume that we have identified at least one failure type that we would like to fix.

We formulate the process of **model repairment** as the following two steps (see Fig. 1 for an illustration):

1. **Cause identification:** Identify a set of detrimental datapoints $\mathcal{C}$ in the training data $\mathcal{D}$ that contributed the most to the failure set $\mathcal{F}$.
2. **Treatment:** Given the set of failure causes $\mathcal{C}$, adapt the model to predict correctly on the failure set $\mathcal{F}$, while maintaining performance on remaining test examples.

Below we describe a formal framework for performing these two steps. In the first phase of cause identification, we need to define a measure of how much a subset of training examples $\mathcal{C}=\{(\boldsymbol{x}_c^{(n)},y_c^{(n)})\}_{n=1}^{N_c}\subset\mathcal{D}$ is responsible for the failure cases $\mathcal{F}$. To this end, we propose to check how much the posterior predictive distribution on $\mathcal{F}$ changes as a result of deleting $\mathcal{C}$ from the training data:

$$r(\mathcal{C}):=\log p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})-\log p(\mathcal{F}|\mathcal{D}), \tag{2}$$

where $p(\mathcal{F}|\mathcal{D})$ and $p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$ are the posterior predictive distributions before and after removing a subset of training examples $\mathcal{C}$. If a given $\mathcal{C}$ leads to a large value of $r(\mathcal{C})$ in Eq. 2, it would mean that removing $\mathcal{C}$ from $\mathcal{D}$ would have improved the performance of the Bayesian predictive inference on the failure set $\mathcal{F}$. The first *causal identification* step thus entails finding a subset $\mathcal{C}$ with the maximal log-density ratio $r(\mathcal{C})$. In the second *treatment* step, we can directly adopt $p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$ as the updated predictive distribution that needs to be inferred, as it confers the largest improvement over the failure cases. In other words, for both the search of detrimental datapoints and repairment of the model, we need to compute $p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$.

The central computational question is, therefore, concerned with the *efficient calculation of* $p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$ *without retraining the model from scratch.* The reasons for avoiding retraining are: (1) in the cause identification step, it is computationally prohibitive as retraining needs to be done for every subset $\mathcal{C}\subset\mathcal{D}$; and (2) in the treatment step, the resulting retrained model may be drastically different from the original model one would like to fix (due to, e.g. noises in the SGD-based optimisation and parameter re-initialisation), and may lose other desirable properties that one may wish to retain. In this work, we thus argue for continual learning approaches [11] to approximate $p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$, given their computational efficiency and their natural ability to maintain model behaviour. We now elaborate on the mathematical details of the two steps.

2.1 Step I: Cause Identification

Identifying the set of detrimental examples $\mathcal{C}$ requires solving the following optimisation problem

$$\mathcal{C}=\operatorname{argmax}_{\mathcal{C}'\in\mathbb{P}(\mathcal{D})}r(\mathcal{C}'), \tag{3}$$

where $\mathbb{P}(\mathcal{D})$ denotes the power set of $\mathcal{D}$. Solving this comes with multiple computational challenges.

Firstly, a naive approach would require computing the predictive distribution $p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$ — and thus, the posterior $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ — for every subset $\mathcal{C}$ of $\mathcal{D}$, which is prohibitively expensive. In the following, we present a **"predictive" approach** that removes this computational burden. The key idea is to notice that $p(\mathcal{D}\backslash\mathcal{C}|\boldsymbol{\theta})=p(\mathcal{D}|\boldsymbol{\theta})/p(\mathcal{C}|\boldsymbol{\theta})$ for any $\mathcal{C}\subset\mathcal{D}$ due to the i.i.d. modelling assumption. Inserting this into the Bayes' rule for computing $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ yields:

$$\log p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})=\log\int p(\mathcal{F}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})d\boldsymbol{\theta}=\log\int\frac{p(\mathcal{F}|\boldsymbol{\theta})}{p(\mathcal{C}|\boldsymbol{\theta})}\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}d\boldsymbol{\theta}-\log\frac{p(\mathcal{D}\backslash\mathcal{C})}{p(\mathcal{D})}.$$

Notice that $p(\mathcal{D}\backslash\mathcal{C})/p(\mathcal{D})=\int\frac{1}{p(\mathcal{C}|\boldsymbol{\theta})}\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}d\boldsymbol{\theta}$ again due to the i.i.d. assumption. Therefore, we can compute the log density ratio $r(\mathcal{C})=\log p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})-\log p(\mathcal{F}|\mathcal{D})$ as follows without computing $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$:

$$r(\mathcal{C})=\log\int\frac{p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})}{p(\mathcal{C}|\boldsymbol{\theta})}d\boldsymbol{\theta}-\log\int\frac{p(\boldsymbol{\theta}|\mathcal{D})}{p(\mathcal{C}|\boldsymbol{\theta})}d\boldsymbol{\theta}. \tag{4}$$

In this form, we would only need to compute the posterior $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ once, and can side-step the requirement of computing $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ for inspection of every new $\mathcal{C}\subset\mathcal{D}$, thereby removing one of the key computational bottlenecks. The detailed derivation of Eq. (4) is provided in Sec. A.2 in the appendix.

Secondly, we are still left with the combinatorial search for the best subset $\mathcal{C}$, which is also prohibitive when the size of training data $\mathcal{D}$ is large. We address this issue by a first-order Taylor series approximation of $r(\mathcal{C})$ that factorises over the samples in $\mathcal{C}$. Let us re-write the log density ratio as

$$r(\mathcal{C})=F(1,p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F}))-F(1,p(\boldsymbol{\theta}|\mathcal{D})),\quad F(\epsilon,g(\boldsymbol{\theta})):=\log\int g(\boldsymbol{\theta})e^{-\epsilon\log p(\mathcal{C}|\boldsymbol{\theta})}d\boldsymbol{\theta}. \tag{5}$$

Algorithm 1 Model Repairment

```
Input: training data $\mathcal{D}$; failure cases $\mathcal{F}$; approximate posterior $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$; likelihood $p(\boldsymbol{z}|\boldsymbol{\theta})$
Output: failure causes $\mathcal{C}$, "repaired" posterior $q^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}(\boldsymbol{\theta})$

# Step I: Cause Identification
Update posterior: Apply a continual learning method to obtain $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ by fitting the failure set $\mathcal{F}$
Compute influences of training examples on $\mathcal{F}$: Calculate $\tilde{r}(\boldsymbol{z}) \forall \boldsymbol{z} \in \mathcal{D}$ (Eq. (7))
Find failure causes $\mathcal{C}$: Return the examples with positive influence, $\mathcal{C} \leftarrow \{\boldsymbol{z} \in \mathcal{D}: \tilde{r}(\boldsymbol{z}) > 0\}$

# Step II: Treatment
Delete information of $\mathcal{C}$: Apply a continual (un)learning method to the original posterior $q(\boldsymbol{\theta})$, and obtain the posterior on the corrected data $q^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$
```

Note that $F(0, g(\boldsymbol{\theta})) = 0$ for any well-defined distribution $g(\boldsymbol{\theta})$. Assuming $g(\boldsymbol{\theta})$ is a distribution, we perform a Taylor expansion of $F(\epsilon, g(\boldsymbol{\theta}))$ around $\epsilon = 0$: $F(\epsilon, g(\boldsymbol{\theta})) = -\epsilon \mathbb{E}_{g(\boldsymbol{\theta})}[\log p(\mathcal{C}|\boldsymbol{\theta})] + \mathcal{O}(\epsilon^2)$. This results in an approximate log density ratio $\hat{r}(\mathcal{C}) \approx r(\mathcal{C})$ by plugging the Taylor expansion into Eq. (5):

$$\hat{r}(\mathcal{C}) := \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\log p(\mathcal{C}|\boldsymbol{\theta})] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})}[\log p(\mathcal{C}|\boldsymbol{\theta})]. \tag{6}$$

Assuming that data are i.i.d., and defining $\boldsymbol{z} = (\boldsymbol{x}, y)$, the above approximation can be expressed as the sum of individual log density ratios, $\hat{r}(\mathcal{C}) = \sum_{\boldsymbol{z} \in \mathcal{C}} \hat{r}(\boldsymbol{z})$, where each term is given by

$$\hat{r}(\boldsymbol{z}) = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\log p(\boldsymbol{z}|\boldsymbol{\theta})] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})}[\log p(\boldsymbol{z}|\boldsymbol{\theta})], \quad p(\boldsymbol{z}|\boldsymbol{\theta}) = p(y|\boldsymbol{x}, \boldsymbol{\theta}).$$

With this approximation, in order to find a subset $\mathcal{C}$ that leads to the maximal $\hat{r}(\mathcal{C})$, it suffices to compute $\hat{r}(\boldsymbol{z})$ for every training example $\boldsymbol{z} \in \mathcal{D}$ and find the top K examples with largest $\hat{r}(\boldsymbol{z})$ values, thereby reducing the search space from $\mathcal{O}(|\mathcal{D}|!)$ choices to only $\mathcal{O}(|\mathcal{D}|)$ choices.

In practice, the causes of failure will correspond to examples $\boldsymbol{z}$ with $\hat{r}(\boldsymbol{z}) > 0$. Intuitively, $\hat{r}(\boldsymbol{z})$ measures how much the posterior predictive moments of $\boldsymbol{z}$ changes when the model is further trained on $\mathcal{F}$, i.e., computation of $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$. If the moment difference is positive, i.e., $\hat{r}(\boldsymbol{z}) > 0$, it means that the example $\boldsymbol{z} \in \mathcal{D}$ is a conflicting evidence against the test examples in $\mathcal{F}$; conversely, if $\hat{r}(\boldsymbol{z}) < 0$, then $\boldsymbol{z}$ and $\mathcal{F}$ are aligned. See Algorithm 1 for specification of the whole procedure.

Lastly, for non-linear models (e.g. neural networks), approximate posteriors $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta} \mid \mathcal{D})$ and $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ are needed due to intractability of the exact posteriors. We assume that $q(\boldsymbol{\theta})$ is available after training, and suffers from the prediction failures $\mathcal{F}$. As recomputing the posterior $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta})$ from scratch can be expensive, we propose to use a *continual learning* technique [11] and obtain this quantity by updating the original posterior $q(\boldsymbol{\theta})$. Finally, we use the following metric $\tilde{r}(\boldsymbol{z})$ in practice to calculate the detrimental impact of each training datapoint on the failure set, $\mathcal{F}$:

$$\tilde{r}(\boldsymbol{z}) := \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{z}|\boldsymbol{\theta})] - \mathbb{E}_{q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta})}[\log p(\boldsymbol{z}|\boldsymbol{\theta})] \tag{7}$$

and approximate the failure causes $\mathcal{C}$. This metric is generic, and the detail implementation depends on the specifics in which both $q(\boldsymbol{\theta})$ and $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta})$ are computed, e.g. MLE/MAP point estimates, Laplace approximation, variational inference, etc. We end by providing two concrete examples: the first shows that the well-known *linear influence function* [8] is a specific instance of Eq. (7); and the second is derived by extending a continual learning method, known as *Elastic Weight Consolidation* (EWC) [10] to cause identification, and is a key methodological development in our work.

Example 1 (Linear Influence Function): Our proposed metric in Eq. (7) recovers the *linear influence function* from Koh & Liang [8] when point estimates are used for $\boldsymbol{\theta}$. Assume that the model is trained on data $\mathcal{D}$ with parameters $\hat{\boldsymbol{\theta}}$, which corresponds to an approximation of MLE/MAP estimates, i.e., $q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \approx p(\boldsymbol{\theta}|\mathcal{D})$. After observing the set of failures $\mathcal{F}$, a point estimate of $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ is obtained by performing a single update of natural gradient ascent [12] on the log likelihood of $\mathcal{F}$ with step size $\gamma > 0$:

$$\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}} \approx \hat{\boldsymbol{\theta}} + \gamma \hat{\boldsymbol{F}}^{-1}_{\hat{\boldsymbol{\theta}}} \nabla_{\hat{\boldsymbol{\theta}}} \log p(\mathcal{F}|\hat{\boldsymbol{\theta}}) \tag{8}$$

where $\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}$ is the empirical Fisher information matrix. This means

$$\tilde{r}(\boldsymbol{z}) = -\gamma \nabla_{\hat{\boldsymbol{\theta}}} \log p(\mathcal{F}|\hat{\boldsymbol{\theta}})^{\top} \hat{\boldsymbol{F}}^{-1}_{\hat{\boldsymbol{\theta}}} \nabla_{\hat{\boldsymbol{\theta}}} \log p(\boldsymbol{z}|\hat{\boldsymbol{\theta}}), \tag{9}$$

if defining the updated posterior as $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta})=\delta(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}})$. The negation of the above equation coincides with the linear influence function (Eq. (2) in [8]) when the failure set is assumed to be $\mathcal{F}=\{\boldsymbol{z}^{\star}\}$ and $\gamma=1$.

Example 2 (Elastic Weight Consolidation): The generality of Eq. (7) permits any continual learning method of one's choice for estimating the updated posterior $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta})\approx p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ after observing failure samples $\mathcal{F}$. Here we illustrate how EWC [10] as a continual learning method can be adopted in the context of cause identification. EWC approximates $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ by first performing the Laplace approximation of the original posterior $p(\boldsymbol{\theta}|\mathcal{D})$ around the point estimate $\hat{\boldsymbol{\theta}}$, and subsequently finding the MAP solution of $\boldsymbol{\theta}$. Formally, $\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}$ is obtained by maximising the objective below w.r.t. $\boldsymbol{\theta}$ via SGD (see Appendix A.3 for details):

$$\log p(\mathcal{F}|\boldsymbol{\theta})-\frac{N}{2}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^{\top}\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})-\frac{\lambda}{2}||\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}||_2^2, \tag{10}$$

where the off-diagonal elements of $\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}$ are dropped for memory reason in practice. Defining $q(\boldsymbol{\theta})=\delta(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})$ and $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta})=\delta(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}})$, we have that

$$\tilde{r}(\boldsymbol{z})=\log p(\boldsymbol{z}|\hat{\boldsymbol{\theta}})-\log p(\boldsymbol{z}|\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}). \tag{11}$$

To compute the above for each datapoint $\boldsymbol{z}\in\mathcal{D}$, we only need to solve the optimisation problem of Eq. (10) by SGD once. We refer to this version of $\tilde{r}(\boldsymbol{z})$ as *EWC-influence function*.

Comparison: The EWC-influence function generalises the linear influence approach. To see this, we derive the fixed point of the EWC objective Eq. (10) w.r.t. $\boldsymbol{\theta}$ (where we set $\lambda=0$):

$$\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}+N^{-1}\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}^{-1}\nabla_{\boldsymbol{\theta}}\log p(\mathcal{F}|\boldsymbol{\theta}). \tag{12}$$

Then the update in Eq. (8) that is implicitly used by linear influence function can be viewed as (damped) one-step fixed-point iteration update initialised at $\hat{\boldsymbol{\theta}}$ for solving the fixed-point equation. As EWC-influence update (Eq. (11)) is obtained by using the optimum of Eq. (10), it is arguably more accurate than linear influence function (Eq. (9)) for measuring the (detrimental) effect of a datum $\boldsymbol{z}$ to the model failures $\mathcal{F}$.

2.2 Step II: Treatment

Once the causes $\mathcal{C}$ of the failures $\mathcal{F}$ are identified among the training data $\mathcal{D}$, we seek to repair the model $q(\boldsymbol{\theta})$ by erasing the memories of $\mathcal{C}$. We formalise this problem as the computation of the posterior $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$, i.e., $\mathcal{C}$ is absent from the training data $\mathcal{D}$. A naive approach would re-run approximate inference on the whole "corrected" dataset $\mathcal{D}\backslash\mathcal{C}$ to obtain an approximate posterior $q^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}(\boldsymbol{\theta})\approx p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ which can be time consuming. But more fundamentally, by doing so, the obtained $q^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}(\boldsymbol{\theta})$ may be unrelated to the original $q(\boldsymbol{\theta})$ based on which $\mathcal{C}$ were identified, due to, e.g. non-convex SGD optimisation issues. Moreover, this "model replacement" approach may not maintain other good properties of the original model $q(\boldsymbol{\theta})$.

Analogous to cause identification (Sec. 2.1), we propose to employ the continual learning approach to estimate efficiently the modified posterior $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$. Applying Bayes' rule and some algebraic manipulations yield $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})\propto p(\boldsymbol{\theta}|\mathcal{D})/p(\mathcal{C}|\boldsymbol{\theta})$ (see Appendix A.4). Therefore the information about $\mathcal{C}$ can be removed by scaling the current posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by the inverse of $p(\mathcal{C}|\boldsymbol{\theta})$ and re-normalising. In other words, we can treat the approximation of $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ as a continual learning task, where the task is to "unlearn" the datapoints in $\mathcal{C}$ while using the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ as the prior. In practice, the target model to be fixed corresponds to the approximate posterior $q(\boldsymbol{\theta})\approx p(\boldsymbol{\theta}|\mathcal{D})$. Therefore continual (un)learning is done by

$$q^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}(\boldsymbol{\theta})\propto q(\boldsymbol{\theta})/p(\mathcal{C}|\boldsymbol{\theta})\approx p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C}). \tag{13}$$

The above approximation can be carried out with different approximate inference techniques such as MLE/MAP point estimate, Laplace approximation [10] and variational inference [7]. At the end of this section, we also provide a few examples to concretise this process.

We highlight the deep connection between the "predictive approach" for cause identification (Sec. 2.1), and the continual (un)learning for data deletion in the treatment step. They share the key idea of editing the posterior distribution $q(\boldsymbol{\theta})$ via continual learning. This can be seen from the factor graph interpretation of posterior distributions, where posterior editing just corresponds to editing the selected factors in the graph, e.g. insertion of $p(\mathcal{F}|\boldsymbol{\theta})$ in cause identification, and deletion of $p(\mathcal{C}|\boldsymbol{\theta})$ in treatment.

Example 1 (Fine-tuning on Corrected Data): Given a point estimate of model parameters $\hat{\boldsymbol{\theta}}$, i.e., $q(\boldsymbol{\theta})=\delta(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})$, a simple way to approximate $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ is to fine-tune on the corrected dataset $\mathcal{D}\backslash\mathcal{C}$

and update the point estimate. The new $\hat{\boldsymbol{\theta}}^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}$ of the repaired model are obtained by maximising the log-likelihood $\log p(\mathcal{D}\backslash\mathcal{C}|\boldsymbol{\theta})$ via SGD, starting from $\hat{\boldsymbol{\theta}}$.

Example 2 (Newton Update Removal): Guo *et al.*[9] proposed a Newton update based method for data deletion. This method reduces to a specific form of Eq. (13) when using negative log-likelihood as its loss:

$$\hat{\boldsymbol{\theta}}^{\star\star}_{\mathcal{D}\backslash\mathcal{C}} \approx \hat{\boldsymbol{\theta}} - \gamma \hat{\boldsymbol{F}}^{-1}_{\hat{\boldsymbol{\theta}}} \nabla_{\hat{\boldsymbol{\theta}}} \log p(\mathcal{C}|\hat{\boldsymbol{\theta}}). \tag{14}$$

Here information about $\mathcal{C}$ gets deleted by performing a single-step natural gradient descent on their log likelihood [12]. Also, notice the similarity with the way linear influence [8] is computed in Eq. (8), illuminating the relation between cause identification and treatment steps.

Example 3 (EWC for data deletion): The update rule in Eq. (13) for data deletion is amenable to any continual learning approaches. For example, given model parameters $\hat{\boldsymbol{\theta}}$, EWC-based deletion obtains new parameters $\hat{\boldsymbol{\theta}}^{\star\star}_{\mathcal{D}\backslash\mathcal{C}}$ by maximising the following objective (see Appendix A.4):

$$-\log p(\mathcal{C}|\boldsymbol{\theta}) - \frac{N}{2}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^\top \hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}) - \frac{\lambda}{2}||\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}||^2_2 \tag{15}$$

where the first term seeks to remove information about $\mathcal{C}$ while the remaining terms discourage parameters from deviating from the original values. Contrasting this with Eq. (10) again reveals the connection between EWC methods for cause identification and treatment steps.

Comparison: Similar to the comparison made in the cause identification part, EWC for data deletion also generalises the Newton update removal (Eq. (14)). This can again be shown by deriving (damped) one-step fixed point iterative update starting from $\hat{\boldsymbol{\theta}}$ to approximate the fixed point of Eq. (15) when $\lambda = 0$. As EWC for deletion uses SGD to approximate optimum of Eq. (15), it is arguably better than Newton update removal for erasing the effects of detrimental examples $\mathcal{C}$, while better maintaining performance on other cases.

3 Related work

Model Editing. There is a recent surge of interest in developing targeted updates to correct model's undesirable behaviours, while leaving other desired properties intact. As naive fine-tuning methods often lead to overfitting to the failure examples and accuracy degradation on others, various strategies have been proposed. For example, Zhu *et al.*[13] employ a simple regularization technique to minimize parameter changes during the fine-tuning phase. Subsequent works [14, 15] advocate for a functional regularisation instead, e.g. KL divergence in the output space, to achieve better regularisation. These lines of work, additionally, propose to use meta-learning [16] to learn to edit the target model, where the latest meta-learning approach is proposed by Mitchell *et al.*[17]. Another promising approach [18] performs weight editing so that features of a specific concept (e.g. snow) map to the features of another (e.g. road). A commonality among these approaches is the focus on direct model edits for correction. Our work takes an orthogonal and under-explored angle where the aim is to "edit" the data instead, by identifying and removing harmful examples which cause failures — in turn, this difference makes our framework complementary to these model-editing approaches.

Continual learning. Continual learning is an active research area with a related but broader scope than model repairment, which aims to develop methods that adapt the model for future tasks while maintaining model performance on previously learned tasks [11]. We focus on a more targeted problem in this work, yet introduce a framework that allows the use of any continual learning approach for model repairment. Our experiments presents EWC [10] as a practical instantiation of the framework. One can also leverage improvements over EWC such as online EWC [19], or other regularisation-based methods that are motivated by Bayesian learning principles, such as variational continual learning [7, 20, 21], synaptic intelligence [22], and orthogonal gradient descent [23]. As approximations to $r(\mathcal{C})$ rely on accurate posterior approximations, advances in Bayesian continual learning methods are expected to improve the practical effectiveness of model repairment under our framework.

Data Selection and Valuation. Multiple techniques have been introduced for selecting "influential" training examples on a chosen metric (e.g. test accuracy), such as influence functions [8, 24, 25, 26, 27, 28, 29], Shapley value-based approaches [30, 31, 32] and probability of sufficiency [33]. Within the category of influence functions, two representative approaches include linear influence function [8] and SGD-influence [27]. The former approach performs one-step update only, thus, while efficient, it may be less accurate in reflecting the influence of a datum $\boldsymbol{z}$. The latter approach computes a projected difference between $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}$ but with $\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}$ obtained by running SGD fine-tuning on training data without $\boldsymbol{z}$. Thus SGD-influence is computationally inefficient. Compared to both baselines, our EWC-influence approach achieve the best in both worlds:

it produces more accurate influence estimates than linear influence due to better optimisation, while it is more efficient than SGD-influence as it requires only one optimisation procedure on the given failure set $\mathcal{F}$.

Data Deletion. The detrimental data removal in the treatment step is related to *data deletion*, a rapidly developing field of machine learning research [34, 9, 35, 36, 37, 38]. Closest to our work is variational Bayesian unlearning [39] which extends variational Bayes to data deletion settings. But the connection to continual learning is not explicitly made, and it is limited to applications in logistic regression and sparse Gaussian processes. In general, the main focus of existing data deletion research is to preserve data privacy, and datapoints to be removed are assumed provided. On the contrary, in this work, we focus on the repairment of models and propose a unified procedure not only to remove data but also to identify which ones to do so.

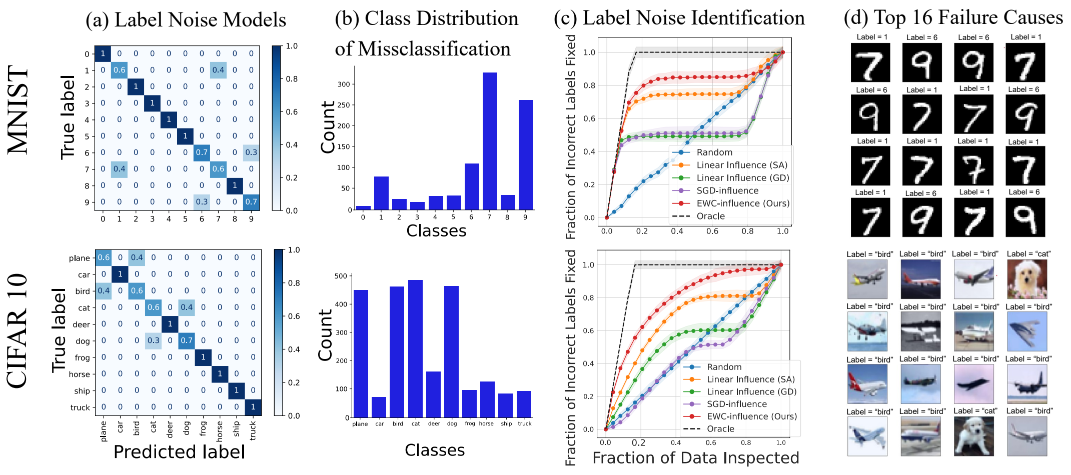


Figure 2: Results on cause identification in the presence of annotation noise. (a) shows the confusion matrices used to simulate class-dependent label noise on MNIST and CIFAR-10. (b) shows the class distribution of the misclassified examples for a single run. (c) plots how much of the identified causes match the samples with incorrect labels for different approaches. The shade represents the standard deviation computed from 5 different runs. (d) shows the top 16 causes of the failures as ranked by EWC-influence.

4 Experiments

We evaluate the efficacy of the proposed framework in a) identifying the causes of target prediction failures in Sec. 4.1, and b) repairing the original model by erasing the memories of such causes in Sec. 4.2. We use augmented versions of MNIST and CIFAR-10 datasets with simulated annotation and input noise. Such controlled experiments are performed for creating "ground truths" of failure causes – necessary for validating the quality of identification methods – and for testing the method in a variety of settings.

Baselines. For the cause identification task, we compare our approach (*EWC-influence*) against the linear influence function [8] and *SGD-influence* [27]. To avoid expensive computation of $\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}^{-1}$, Koh & Liang [8] introduced two efficient approximations to the Hessian-vector product $\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}^{-1}\nabla_{\boldsymbol{\theta}}\log p(\mathcal{F}|\boldsymbol{\theta})$; the first solves $\arg\min_v\{v^T\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}^{-1}\nabla_{\boldsymbol{\theta}}v - \log p(\mathcal{F}|\boldsymbol{\theta})^Tv\}$ with gradient descent (GD), while the second uses an iterative algorithm for stochastic approximation (SA) from [40]. We implement these two variants (GD & SA) of linear influence in Pytorch, and use the original implementation for SGD-influence. For the model treatment task, we compare our method (*EWC-deletion*) against Newton update removal [9]. This method again requires computing a Hessian-vector product for which we employ the same stochastic approximation technique [40]. To isolate the evaluation of cause identification and treatment, we further consider in Section 4.1 *fine-tuning* on $\mathcal{D}\backslash\mathcal{C}$ as another repairment strategy, which would return the best repairment result if the set $\mathcal{C}$ correctly captures the detrimental datapoints. Lastly, we set the prior term λ to zero in eq.(10) and eq. (15) to ensure fair comparison with linear influence and Newton update removal.

Common Set-up. We train the base classification models on the training split of the "augmented" MNIST and CIFAR-10 datasets. For MNIST, we use 6% (3000 samples) of the original training set to make the task more challenging. We use instances of CNNs throughout and train them using the Adam optimiser [41]. The architecture and training details can be found in Appendix C. For evaluation, we separate the test set $\mathcal{T}$ into the set of misclassified examples, $\mathcal{F}$ ("*failure set*") and the others, $\mathcal{T}\backslash\mathcal{F}$ which are correctly classified ("*remaining set*"). We further split the failure set into *query*, $\mathcal{F}_q$ and *holdout*, $\mathcal{F}_h$ sets, where we only use the for-

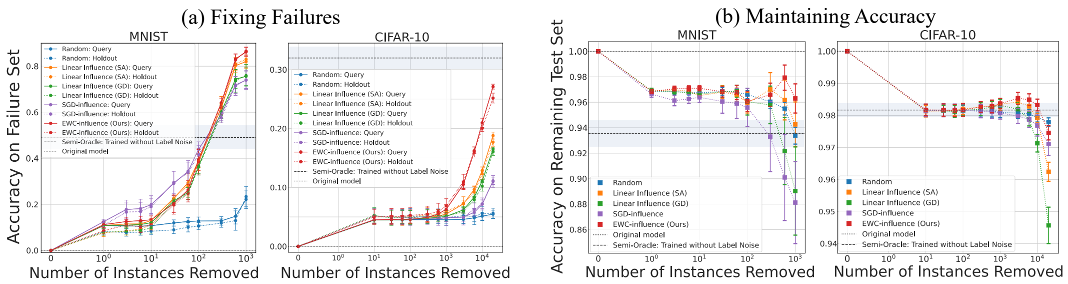


Figure 3: Comparison of the quality of identified causes in the presence of annotation noise. The impact of gradually removing samples in MNIST and CIFAR-10 datasets in the order of influence values $r(z)$ are measured on the failure sets (holdout and query) in (a), and on the remaining test set in (b). We also plot the performance of another reference ("semi-oracle") that is the original model fine-tuned on the training data without the label noise instances. The means and standard deviations of all quantities are calculated over 5 different runs.

mer to identify failure causes $\mathcal{C}$, and use the latter to quantify how generalisably the removal of $\mathcal{C}$ can amend the failure cases. We stress that $\mathcal{F}_q$ is used for cause identification only, but not for further model adaptation.

4.1 Identifying Failure Causes

Annotation Noise. To induce test prediction failures, for the training set we randomly flip labels between semantically similar classes (e.g. 1 and 7 for MNIST, and cats and dogs for CIFAR-10) according to the confusion matrices in Fig. 2(a). As a result, the classes of miss-classified test examples are concentrated on those classes with label noise as depicted in Fig. 2(b).

To measure the accuracy of identifying incorrectly labelled examples, we inspect the training examples $z \in \mathcal{D}$ in the descending order of $\tilde{r}(z)$ computed with $\mathcal{F}_q$ which contains 50% of the miss-classified test cases, and calculate the fraction of incorrectly labelled datapoints in inspected examples. Fig. 2(c) shows that EWC-influence identifies more failure causes earlier on compare to other methods, and is the closest match to the "Oracle" baseline which has full knowledge of samples with wrong labels. Fig. 2(d) shows that the top few causes according to EWC-influence are the samples with incorrect labels, while the least harmful ones are the images of the same classes but with the correct labels as shown in Fig. 7 in Appendix B.

As stipulated in Sec. 2, a set of identified causes $\mathcal{C}$ is of higher quality if removing them leads to a larger gain in accuracy on the failure set while maintaining performance on others. To measure such *quality of causes*, we fine-tune the base model on $\mathcal{D}\backslash\mathcal{C}$ and report the accuracy on the failure query set $\mathcal{F}_q$, the holdout failure set $\mathcal{F}_h$ as well as the remaining test set, $\mathcal{T}\backslash\mathcal{F}$. Results in Fig. 3 suggest that removing failure causes according to EWC-influence yields the highest increase in accuracy on the failure set $\mathcal{F}$ without hurting performance in the remaining test set $\mathcal{T}\backslash\mathcal{F}$. We also note that all of the methods are able to fix the failures better than randomly removing datapoints, and more interestingly, for MNIST, when enough causes are erased ($\approx 10^3$), all methods even surpass the case in which all label noise instances are removed. This result implies that, while annotation noise is a major detrimental factor, the prediction failures also arise from other types of harmful examples.

Lastly, we evaluate the sample efficiency of cause identification by reducing the size of the query set $\mathcal{F}_q$. Fig. 6 in Appendix B shows that all approaches degrade gracefully in repairment performance as the query size gets smaller, but overall EWC-influence still remains the best in terms of label noise detection and repairment accuracy on the failure set and the remaining set.

Random Input Noise. In this experiment, we inject synthetic outliers into MNIST and CIFAR-10 and test the quality of cause identification. We select a set of target classes — 1, 7, 6, 9 for MNIST and plane, bird, cat, dog for CIFAR-10 — and randomly corrupt 30% of the images in those classes by adding salt-and-pepper in MNIST and Gaussian noise in CIFAR-10. The top row in Fig. 4 shows examples, and those corrupted images constitute roughly 12% of the whole training set. Sec. C in the appendix provides details.

We use a subgroup of failures in the target classes as the query $\mathcal{F}_q$ to compute influence values. Surprisingly, the second rows in Fig. 4(a) and (b) show that EWC-influence largely avoids selecting the corrupted images as the top 1000 causes for MNIST and the top 20000 causes for CIFAR-10. However, the third and the fourth rows show that removing those causes results in the best treatment performance on failures while maintaining the performance at a level similar to other baselines. In fact, removing all the input noise and retraining is not able to fix the failures by much, indicating that EWC-influence is able to correctly avoid these relatively harmless outliers and detect other more harmful causes. Fig. 8 in Appendix B visualises the

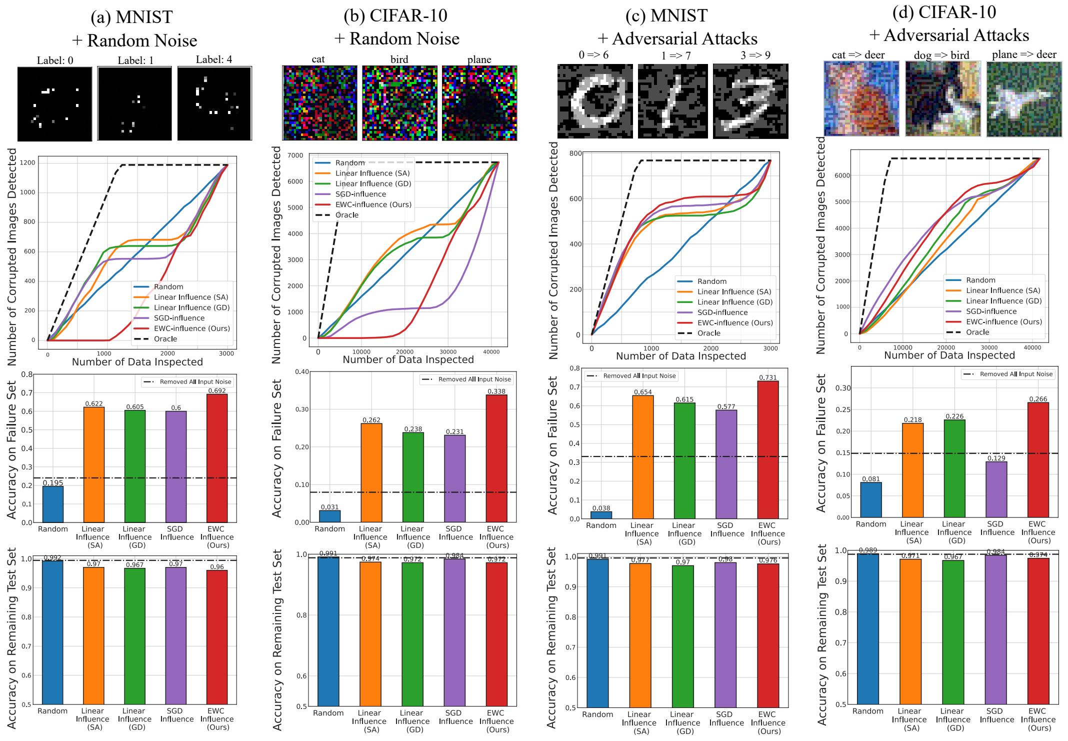


Figure 4: Results on cause identification in the presence of different input noises. From top to bottom, we show i) examples of corrupted samples (synthetic proxy for potential causes of failure), ii) how many of the identified causes correspond to samples corrupted with input noise, iii) and iv) performance in failure holdout set $\mathcal{F}_h$ and remaining test set when removing the top 1000/20000 identified causes in MNIST/CIFAR-10. The influence values are calculated with respect to 50% of test-time failure cases that belong to the classes that suffer from input noise. EWC-Influence identifies "harmful" (adversarial) input noise better than random while avoiding "harmless" (random) input noise.

most harmful examples identified by EWC-influence. Many of them appear to be ambiguous instances in non-target classes, e.g. wonky digits, close-up views of vehicles, a real instance with incorrect label [42], etc.

Adversarial Poisoning. To simulate input noise that can induce test-time failures, we introduce contaminated data by randomly corrupting 30% of the training images in those previously mentioned target classes. These poisoned datapoints are adversarial images crafted by the fast gradient sign method (FGSM) [43] on a separate set of victim models trained on the original clean datasets, and they are labelled by the classes predicted by the victim models. The poisoned datasets are then used to train the base models that are used for evaluation of cause identification.

Fig. 4(c) and (d) show that most of the influence functions detect the corrupted samples better than the "random" baseline. The dashed lines in the third row show that removing all of the corrupted inputs lead to a significant gain in accuracy on the holdout failure set in comparison with the random noise setting, illustrating the larger extent of harms caused by data poisoning. However, most of the identification methods still outperform this reference by a large margin. This suggests again for the presence of other influential samples, and EWC-influence is able to pick up the most important ones, judging by the accuracy on the failure set.

Speed Comparison. Table 1 in Appendix B shows the total run-time of cause identification methods on a single GPU for their best sets of hyper-parameters selected based on the treatment accuracy on the failure set. For both datasets, EWC-influence achieves comparable or shorter run time than the baselines.

4.2 Comparison of Treatment Methods

We evaluate the performance of different deletion-based methods for treatment introduced in Sec. 2.2 on MNIST and CIFAR10 datasets with simulated annotation noise, used in the previous section. We run both EWC-deletion (ours) and Newton update removal [9] methods with early stopping based on the query set accuracy, and experiment with different hyper-parameter settings (see Sec. C in Appendix) to achieve different trade-offs between failure set accuracy and remaining set performance. Here EWC-influence is used to identify the causes, and the top 15% examples were removed by the respective deletion methods. Such

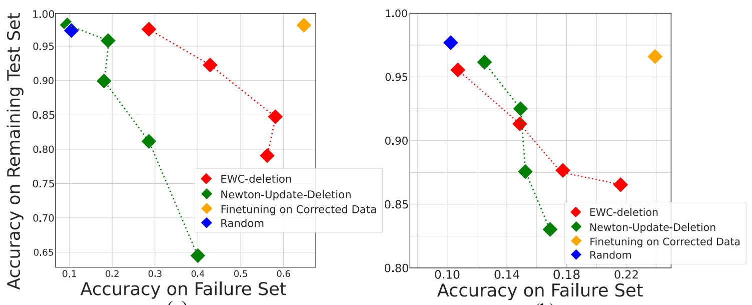


Figure 5: Comparison of deletion-based treatment methods on (a) MNIST and (b) CIFAR-10. For Newton-update-deletion and EWC-deletion, we plot multiple results for varying hyper-parameters to visualise the trade-off between the accuracy on the failure set and the remaining set. The closer to the top right corner, the more desirable.

trade-off is shown in Fig. 5, where fine-tuning on $\mathcal{D}\backslash\mathcal{C}$ is included as an "upper-bound" reference for data deletion performance. On MNIST, EWC-deletion attains a considerably better trade-off between treatment and maintenance compared to Newton-update-deletion, and is much closer to the fine-tuning reference. For CIFAR-10, EWC-deletion beats the Newton-update deletion by 5% in the best failure accuracy while the order reverses for the best accuracy on the remaining test set but with less than 1% difference.

5 Conclusions

In this work, we develop a generic framework for *repairing* machine learning models by erasing memories of detrimental datapoints. The framework consists of two key components, that are, the mechanism for identifying the "causes" in training data which are responsible for the given failures, and the adaptation method for fixing the model by removing information about them. Two components are connected under the Bayesian view of continual (un)learning, which brings forth several practical benefits. Firstly, the framework subsumes some recent works on influence function and data deletion as specific examples, and elucidate their limitations. Secondly, the generality of our approach allows leveraging recent advances in continual learning in this new problem of model repairment. In particular, we extend Elastic Weight Consolidation to cause identification and information removal, and demonstrate empirically its competitive performance in both tasks. Future work will investigate the values of adapting more recent continual learning approaches, and also study other types of data correction mechanisms (e.g. label correction, sample/label acquisition).

Acknowledgements

We would like to thank Ozan Oktay, Stephanie Hyland, and Ted Meeds at Microsoft Research Cambridge and Jin Chen at University College London for their valuable feedback on an early version of this work.

References

[1] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[2] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

[3] Pang Wei Koh, Shiori Sagawa, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[4] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[5] Irene Chen, Fredrik D Johansson, and David Sontag. Why is my classifier discriminatory? In *Advances in Neural Information Processing Systems*, pages 3539–3550, 2018.

[6] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[7] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[8] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[9] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

[10] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[11] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[12] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[13] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.

[14] Anton Sinitsin, Vsevolod Plokhotnyuk, Sergei Popov, and Artem Babenko. Editable neural networks. *arXiv preprint arXiv:2004.00345*, 2020.

[15] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. 2021.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[17] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. *CoRR*, 2021.

[18] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[19] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[20] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. In *Advances in Neural Processing Information Systems*, 2020.

[21] Noel Loo, Siddharth Swaroop, and Richard E Turner. Generalized variational continual learning. In *International Conference on Learning Representations*, 2021.

[22] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[23] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[24] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems*, pages 5254–5264, 2019.

[25] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training examples via relative influence. *arXiv preprint arXiv:2003.11630*, 2020.

[26] Ryan Giordano, William Stephenson, Runjing Liu, Michael Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1139–1147, 2019.

[27] Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with sgd. In *Advances in Neural Information Processing Systems*, pages 4213–4222, 2019.

[28] Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3382–3390, 2019.

[29] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*, 2021.

[30] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, 2019.

[31] Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*, pages 272–281. PMLR, 2019.

[32] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

[33] Aleksandar Chakarov, Aditya Nori, Sriram Rajamani, Shayak Sen, and Deepak Vijaykeerthy. Debugging machine learning tasks. *arXiv preprint arXiv:1603.07292*, 2016.

[34] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019.

[35] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3518–3531, 2019.

[36] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models: Algorithms and evaluations. *arXiv preprint arXiv:2002.10077*, 2020.

[37] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. *arXiv preprint arXiv:2007.02923*, 2020.

[38] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *arXiv preprint arXiv:2106.04378*, 2021.

[39] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. *arXiv preprint arXiv:2010.12883*, 2020.

[40] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization in linear time. *stat*, 1050:15, 2016.

[41] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[42] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR)*, 70:1373–1411, 2021.

[43] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

A Algorithmic details

A.1 Problem formulation for model repairment

This section extends the discussion on the problem formulation of model repairment. Specifically, under the modelling assumptions presented in the main text, the goals for model repairment are the following:

- For a set of "failure cases" $\mathcal{F} = \{\boldsymbol{z}_f = (\boldsymbol{x}, y)\}$ where the model with (Bayesian) predictive inference makes wrong predictions, repair the model to make correct predictions on $\mathcal{F}$ and similar cases.
- For a set of "benchmark cases" $\mathcal{B} = \{\boldsymbol{z}_b = (\boldsymbol{x}, y)\}$, maintain a given level of prediction accuracy after model repairment.

There are further considerations for executing and evaluating model repairment in practice.

Generalisation and efficiency of model repairment In practice the number of failure cases might be large or even infinite. For example, an image classification model that fails on a test input with Gaussian noise may also fail on all the other inputs with such level of noise. So we need to consider both, *generalisation* and *efficiency* aspects of model repairment. Here, good generalisation means that the failure is fixed, not only for observed failure cases but also for future cases for which the model would make the same type of failure before fixes. On the other hand, efficiency of repairment considers the number of failure examples required for the model repairment method to fix a particular type of failure.

To describe both concepts in more details, in addition to a set of known/observed failure examples $\mathcal{F}$ collected by the users, we need to define the set of *unknown/unobserved* failure examples $\mathcal{F}_U$. If examples in $\mathcal{F}$ and $\mathcal{F}_U$ are similar, then good *generalisation* of a model repairment algorithm means that a model repaired by such method using information from $\mathcal{F}$ should produce correct predictions for instances in $\mathcal{F}_U$. On the other hand, a model repairment method is *efficient* if it only needs a small set $\mathcal{F}$ of collected failure cases to achieve good generalisation of repairment.

Specificity of model repairment Furthermore, there may be multiple different scenarios in which the model fails, and therefore the set of failure cases may consist of several groups, i.e.,

$$\mathcal{F} = \sqcup_{m=1}^{M} \mathcal{F}^{(m)}, \quad \mathcal{F}_U = \sqcup_{m=1}^{M} \mathcal{F}_U^{(m)}, \tag{16}$$

where $\mathcal{F}^{(m)}$ denotes the observed failure cases of failure type m and $\mathcal{F}_U^{(m)}$ represents unobserved failure cases of the same type. Targeting a specific type of mistake and repairing it one at a time may be desirable in practice. One type of mistake might incur more costs than others (e.g., in medical applications, false negative is generally more costly than false positive), so users might have different priorities for different types of errors to be fixed. This is especially the case when there exists a trade-off between fixing different types of errors, again the false negative vs false positive trade-off is a prevalent example.

Repairment by identifying and removing detrimental training data There can be many different reasons for a model with Bayesian predictive inference making wrong predictions on $\mathcal{F}$. In this work, we assume that *the main reason is due to the existence of detrimental datapoints in* $\mathcal{D}$. Our hypothesis is that, by removing/correcting these detrimental datapoints and adapting the model with them, the model can be repaired to return correct labels for datapoints in $\mathcal{F}$. Base on the above hypothesis, the model repairment process contains the following steps:

1. **Cause identification:** Identify a set of detrimental data points $\mathcal{C}$ in the training data $\mathcal{D}$ that contributed the most to the failure set $\mathcal{F}$.
2. **Treatment:** Given the set of failure causes $\mathcal{C}$, adapt the model to predict correctly on the failure set $\mathcal{F}$, while maintaining performance on other examples which were correctly predicted previously.

A.2 The "predictive approach" for cause identification

We use the following function to describe the contribution of a subset $\mathcal{C} \subset \mathcal{D}$ to the model failures on examples in $\mathcal{F}$:

$$r(\mathcal{C}) = \log\left(\frac{p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})}{p(\mathcal{F}|\mathcal{D})}\right). \tag{17}$$

A naive approach would compute $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ for all subsets of $\mathcal{D}$ with all possible correction methods, which is prohibitively expensive. Instead we present a **"predictive" approach** that removes this computational burden. First, notice that we make the i.i.d. modelling assumption which means that one can write the likelihood term as follows:

$$p(\mathcal{D}|\boldsymbol{\theta}) = p(\mathcal{D}\backslash\mathcal{C}|\boldsymbol{\theta})p(\mathcal{C}|\boldsymbol{\theta}), \quad \forall \mathcal{C} \subset \mathcal{D}. \tag{18}$$

This allows us to expand the log evidence $\log p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})$ as

$$\begin{aligned}
\log p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C}) &= \log\int p(\mathcal{F}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})d\boldsymbol{\theta} \\
&= \log\int p(\mathcal{F}|\boldsymbol{\theta})\frac{p(\mathcal{D}\backslash\mathcal{C}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D}\backslash\mathcal{C})}d\boldsymbol{\theta} \qquad \text{(Bayes' rule)} \\
&= \log\int p(\mathcal{F}|\boldsymbol{\theta})\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{C}|\boldsymbol{\theta})p(\mathcal{D}\backslash\mathcal{C})}d\boldsymbol{\theta} \qquad \text{(by Eq. (18))} \\
&= \log\int \frac{p(\mathcal{D})}{p(\mathcal{D}\backslash\mathcal{C})}\cdot\frac{p(\mathcal{F}|\boldsymbol{\theta})}{p(\mathcal{C}|\boldsymbol{\theta})}\cdot\frac{p(\mathcal{D},\boldsymbol{\theta})}{p(\mathcal{D})}d\boldsymbol{\theta} \qquad \text{(multiplying } \frac{p(\mathcal{D})}{p(\mathcal{D})} \text{ and rearranging terms)} \\
&= \log\int \frac{p(\mathcal{F}|\boldsymbol{\theta})}{p(\mathcal{C}|\boldsymbol{\theta})}p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} + \log\frac{p(\mathcal{D})}{p(\mathcal{D}\backslash\mathcal{C})}. \qquad \text{(Bayes' rule)}
\end{aligned}$$

Then we can rewrite the log density ratio as

$$\begin{aligned}
r(\mathcal{C}) &= \log\left(\frac{p(\mathcal{F}|\mathcal{D}\backslash\mathcal{C})}{p(\mathcal{F}|\mathcal{D})}\right) \\
&= \log\int\frac{p(\mathcal{F}|\boldsymbol{\theta})}{p(\mathcal{C}|\boldsymbol{\theta})}p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} - \log p(\mathcal{F}|\mathcal{D}) + \log p(\mathcal{D}) - \log\int p(\mathcal{D}\backslash\mathcal{C}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \\
&\qquad\qquad \text{(by definition of marginal distributions)} \\
&= \log\int\frac{1}{p(\mathcal{C}|\boldsymbol{\theta})}\frac{p(\mathcal{F}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})}{p(\mathcal{F}|\mathcal{D})}d\boldsymbol{\theta} - \log\int\frac{p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{C}|\boldsymbol{\theta})}\frac{p(\boldsymbol{\theta})}{p(\mathcal{D})}d\boldsymbol{\theta} \\
&\qquad\qquad \text{(by Eq. (18) and rearranging terms)} \\
&= \log\int\frac{p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})}{p(\mathcal{C}|\boldsymbol{\theta})}d\boldsymbol{\theta} - \log\int\frac{p(\boldsymbol{\theta}|\mathcal{D})}{p(\mathcal{C}|\boldsymbol{\theta})}d\boldsymbol{\theta}. \qquad \text{(Bayes' rule)}
\end{aligned} \tag{19}$$

By doing so, instead of computing $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$ for every possible subset $\mathcal{C}$, the "predictive approach" only requires computing $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ once. As shown in the main text, with (approximations of) the two posteriors $p(\boldsymbol{\theta}|\mathcal{D})$ and $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$ at hand, the log density ratio $r(\mathcal{C})$ can be efficiently approximated by Monte Carlo and/or further approximations described in the main text that employ Taylor expansions. This approach is "predictive" in the sense that the influence of candidate set $\mathcal{C}$ is evaluated by computing "predictions" $p(\mathcal{C}|\boldsymbol{\theta})^{-1}$ on them using the two posterior distributions, which is different from existing approaches that compute predictions on $\mathcal{F}$ using approximations to the modified posterior $p(\boldsymbol{\theta}|\mathcal{D}\backslash\mathcal{C})$.

A.3 Objective for EWC-influence

Recall in the main text the first-order Taylor series approximation to $r(\mathcal{C})$ is

$$r(\mathcal{C}) \approx \sum_{\boldsymbol{z}\in\mathcal{C}}\hat{r}(\boldsymbol{z}), \quad \hat{r}(\boldsymbol{z}) = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\log p(\boldsymbol{z}|\boldsymbol{\theta})] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})}[\log p(\boldsymbol{z}|\boldsymbol{\theta})].$$

Therefore the "predictive approach" for computing $r(\mathcal{C})$ as well as the approximated form require the computation of (approximate) posteriors $p(\boldsymbol{\theta}|\mathcal{D})$ and $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$. This can be achieved using continual learning: we assume the model has been trained on $\mathcal{D}$ and an approximation $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$ has been obtained. Then the current task for continual learning is to adapt the trained model on the failure cases $\mathcal{F}$, which leads to an adapted approximation $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$.

Elastic Weight Consolidation (EWC) [10] is a continual learning algorithm that can be interpreted as updating the maximum a posteriori (MAP) approximation to the posterior given new tasks. To see this, first in this approach the q posteriors are assumed to be delta measures. In other words, $q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$ where $\hat{\boldsymbol{\theta}}$ is the parameters of the trained model, and $q^{\star}_{\mathcal{D},\mathcal{F}}(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}})$ where as we shall see $\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}$ is the parameter obtained by running EWC using $\mathcal{F}$. With these assumptions, the EWC-influence is defined as:

$$r(\mathcal{C}) \approx \sum_{\boldsymbol{z} \in \mathcal{C}} \hat{r}(\boldsymbol{z}) \approx \sum_{\boldsymbol{z} \in \mathcal{C}} \tilde{r}(\boldsymbol{z}), \quad \tilde{r}(\boldsymbol{z}) = \log p(\boldsymbol{z} | \hat{\boldsymbol{\theta}}) - \log p(\boldsymbol{z} | \hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}).$$

It remains to discuss the optimisation procedure for obtaining $\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}}$. As motivated, we consider MAP approximations to the posterior, which seeks to find the maximum of the log posterior $\log p(\boldsymbol{\theta} | \mathcal{D}, \mathcal{F})$. Notice that by Bayes' rule:

$$p(\boldsymbol{\theta} | \mathcal{D}, \mathcal{F}) \propto p(\mathcal{F} | \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) \quad \Rightarrow \quad \log p(\boldsymbol{\theta} | \mathcal{D}, \mathcal{F}) = \log p(\mathcal{F} | \boldsymbol{\theta}) + \log p(\boldsymbol{\theta} | \mathcal{D}) + \text{constant}.$$

Computing the first term $\log p(\mathcal{F} | \boldsymbol{\theta})$ in the MAP objective is straightforward given the i.i.d. modelling assumption. For the second term, as $\log p(\boldsymbol{\theta} | \mathcal{D})$ is intractable, the EWC approach constructs a Laplace approximation to it by assuming the trained model parameter $\hat{\boldsymbol{\theta}}$ as a MAP point estimate of $p(\boldsymbol{\theta} | \mathcal{D})$. In detail, a Laplace approximation to the posterior is

$$\log p(\boldsymbol{\theta} | \mathcal{D}) \approx \frac{1}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\top} \boldsymbol{H}[\log p(\hat{\boldsymbol{\theta}} | \mathcal{D})] (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) + \text{constant}. \tag{20}$$

where $\boldsymbol{H}[f(\boldsymbol{\theta})]$ denotes the Hessian matrix of a twice-differentiable function $f(\boldsymbol{\theta})$ with respect to parameters $\boldsymbol{\theta}$. Further decomposing the Hessian term yields:

$$\boldsymbol{H}[\log p(\hat{\boldsymbol{\theta}} | \mathcal{D})] = \boldsymbol{H}[\log p(\mathcal{D} | \hat{\boldsymbol{\theta}}) + \log p(\hat{\boldsymbol{\theta}}) - \log p(\mathcal{D})] \tag{21}$$

$$= \underbrace{\boldsymbol{H}[\log p(\mathcal{D} | \hat{\boldsymbol{\theta}})]}_{\approx -N \hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}} + \underbrace{\boldsymbol{H}[\log p(\hat{\boldsymbol{\theta}})]}_{\text{Prior term}} \tag{22}$$

where N is the total number of samples in $\mathcal{D}$ and $\hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}}$ is an empirical estimate of the *Fisher information matrix*. Assuming the zero-mean isotropic Gaussian prior $p(\boldsymbol{\theta})$ with precision $\lambda \boldsymbol{I}$, the second term becomes $\boldsymbol{H}[\log p(\hat{\boldsymbol{\theta}})] = -\lambda \boldsymbol{I}$. Substituting these results back into Eq. (20) gives us:

$$\log p(\boldsymbol{\theta} | \mathcal{D}) \approx -\frac{N}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\top} \hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) - \frac{\lambda}{2} ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_2^2 + \text{constant}. \tag{23}$$

Combining terms, the optimisation task for EWC adaptation using failure set is:

$$\hat{\boldsymbol{\theta}}^{\star}_{\mathcal{D},\mathcal{F}} = \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{F} | \boldsymbol{\theta}) - \frac{N}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\top} \hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) - \frac{\lambda}{2} ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_2^2, \tag{24}$$

which is as presented in the main text. We further note that in the original EWC formulation, the empirical Fisher information is further simplified to contain *diagonal* entries only — with millions of model parameters as is typically the case with neural networks, as computing the full matrix is expensive.

A.4 Objective for EWC-deletion

Assuming that the detrimental datapoints $\mathcal{C} \subset \mathcal{D}$ have been identified, the next step for model repairment is *treatment* where we seek to remove the influence of $\mathcal{C}$ to the model. This is done by computing (an approximation to) $p(\boldsymbol{\theta} | \mathcal{D} \backslash \mathcal{C})$ which can also be done via continue learning. To see this, we can show by using Bayes' rule and Eq. (18) again,

$$\log p(\boldsymbol{\theta} | \mathcal{D} \backslash \mathcal{C}) = \log \frac{p(\mathcal{D} \backslash \mathcal{C} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D} \backslash \mathcal{C})} = \log \frac{1}{p(\mathcal{C} | \boldsymbol{\theta})} + \log \underbrace{\frac{p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})}}_{= p(\boldsymbol{\theta} | \mathcal{D})} + \log \frac{p(\mathcal{D})}{p(\mathcal{D} \backslash \mathcal{C})}. \tag{25}$$

This means that finding the MAP estimate of $p(\boldsymbol{\theta} \mid \mathcal{D} \setminus \mathcal{C})$ is equivalent to maximising $-\log p(\mathcal{C} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta} \mid \mathcal{D})$ w.r.t. $\boldsymbol{\theta}$. With further approximation to the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ using the Laplace method as discussed in section A.3, one can write the optimisation task for EWC-deletion as presented in the main text, namely:

$$\hat{\boldsymbol{\theta}}^{\star\star}_{\mathcal{D} \backslash \mathcal{C}} = \arg\max_{\boldsymbol{\theta}} -\log p(\mathcal{C} | \boldsymbol{\theta}) - \frac{N}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\top} \hat{\boldsymbol{F}}_{\hat{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) - \frac{\lambda}{2} ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_2^2. \tag{26}$$

B Additional Results

B.1 Sample Efficiency of Cause Identification

Fig. 6 compares the sample efficiency of different approaches for cause identification on MNIST and CIFAR-10 datasets with annotation noise. More specifically, we want to understand how many failure cases in the query set $\mathcal{F}_q$ we need to see for cause identification to work, i.e., how performance of each approach varies as we reduce the size of the query set. Here, all detrimental examples i.e., $\tilde{r}(z) < 0$ are removed from the training data and the corresponding metrics are measured. EWC-influence approach performs the best in terms of precision, recall, and accuracy on holdout failure set $\mathcal{F}_h$ even as we decrease the size of the query set. All approaches exhibit a stable behavior w.r.t the size of query set. The biggest drop occurs in accuracy on the remaining test set $\mathcal{T} \setminus \mathcal{F}$: EWC-influence presents a stronger degradation in small sample regimes (around 5% of the original failure set) but without dropping below other approaches.

For the failure accuracy on MNIST, even when the query set is as small as 5% of the original one (≈ 20 examples), all approaches display improvement over the semi-oracle that is trained after removing all instances of annotation errors: this means that all methods, in a sample-efficient manner, not only correct the synthetically added annotation errors, but they also remove other harmful examples that are naturally present to begin with.

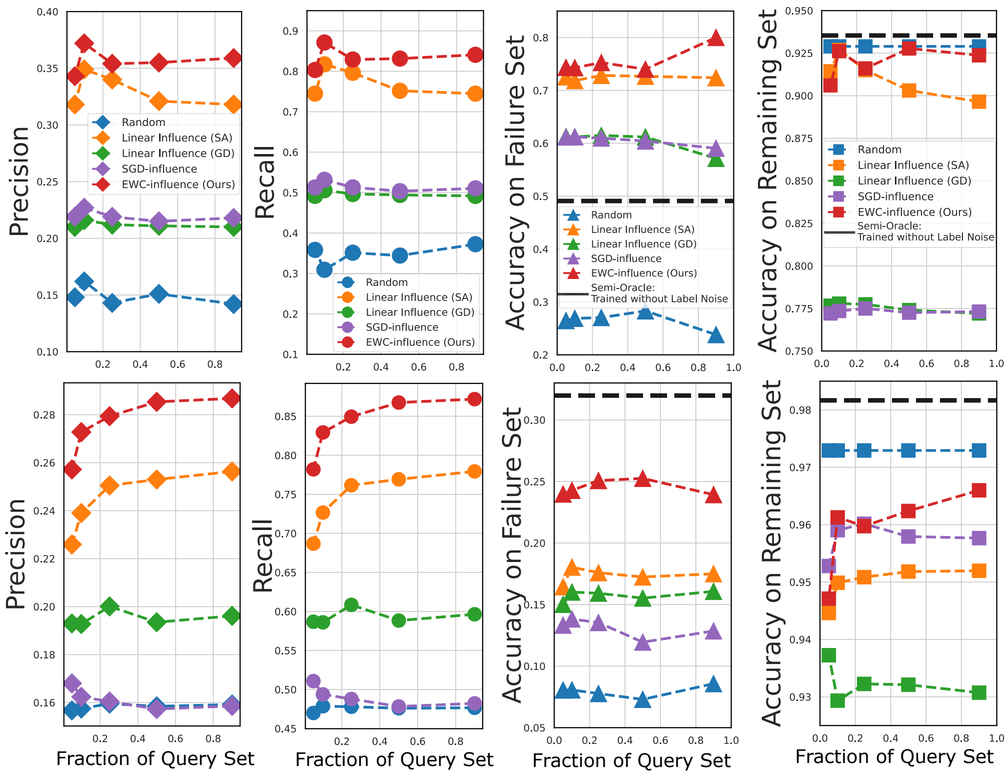


Figure 6: Comparison of sample efficiency for cause identification performance on (top) MNIST and (bottom) CIFAR-10 datasets with class-dependent annotation noise. On the x-axis of each sub-figure, we vary the size of the query failure set $\mathcal{F}_q$ used for cause identification. From left to right, (a) precision, (b) recall, (c) accuracy on holdout failure set $\mathcal{F}_h$, and (d) accuracy on the remaining test set.

B.2 Speed Comparison

Table 1 shows the total computation time of the proposed EWC-influence and other baselines for cause identification on a single Tesla K80 GPU with 12GB of RAM. Note that we rely on the publicly available implementation of SGD-influence[2], and that we have implemented our own version of linear influence functions in Pytorch. Overall, EWC-Influence is either as fast or faster as other baselines, achieving one order of magnitude speed boost compared to SGD-influence in all cases. For CIFAR10 where a considerably larger base model is used, EWC-influence is consistently faster than other approaches, around twice faster than linear influence methods. For MNIST, EWC-influence attains similar computation time as linear influence approaches.

Table 1: Comparison of total computation time for cause identification.

Experiment	**Method**	**Time (s)** (MNIST)	**Time (s)** (CIFAR10)
Label Noise	Linear Influence (SA)	16.4	1322.4
	Linear Influence (GD)	11.2	996.2
	SGD-Influence	185.1	8301.1
	EWC-Influence (Ours)	**9.8**	**496.3**
Random Input Noise	Linear Influence (SA)	10.8	1141.1
	Linear Influence (GD)	15.7	978.4
	SGD-Influence	188.2	8285.6
	EWC-Influence (**Ours**)	11.0	**527.5**
Adversarial Poisoning	Linear Influence (SA)	14.3	1312.0
	Linear Influence (GD)	**10.0**	984.2
	SGD-Influence	177.29	7803.7
	EWC-Influence (**Ours**)	14.5	**528.9**

B.3 Qualitative Results

Annotation noise. In the main text, Fig. 2(d) shows examples with annotation noise ranked as most harmful according to EWC-influence. Similarly, Fig. 7 shows examples ranked as least harmful according to EWC-influence. All of these examples correspond to non-corrupted examples with correct labels.

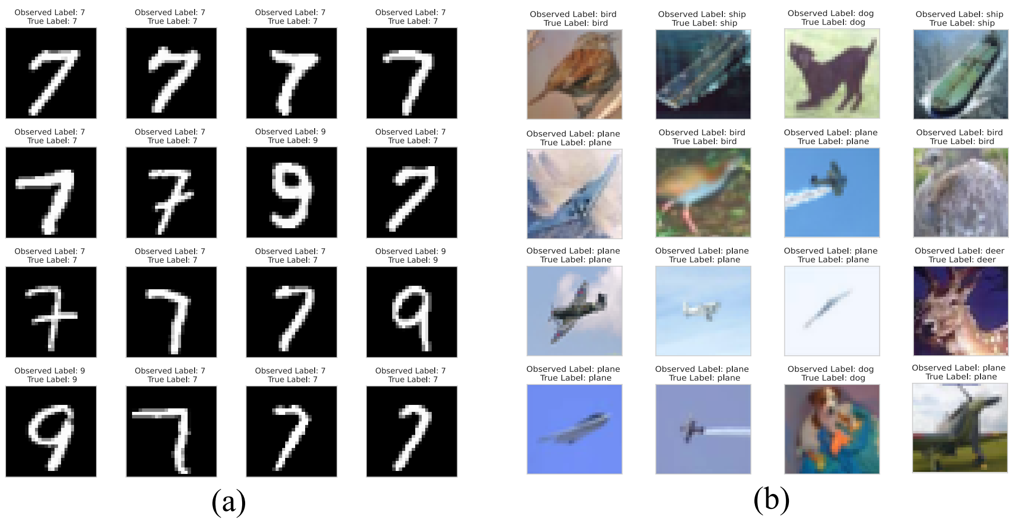
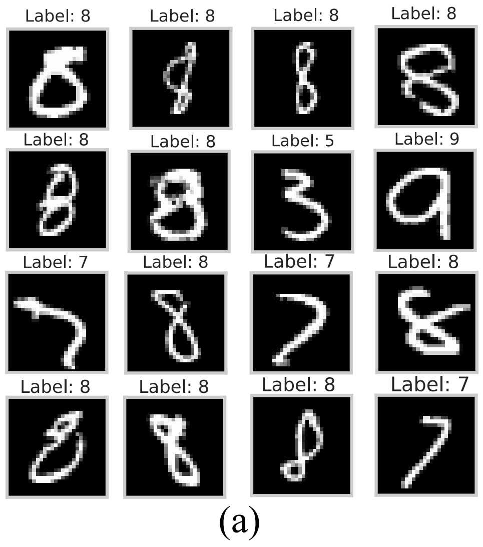

Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 9 True Label: 9
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 9 True Label: 9
Observed Label: 9 True Label: 9
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
Observed Label: 7 True Label: 7
(a)
Observed Label: bird True Label: bird
Observed Label: ship True Label: ship
Observed Label: dog True Label: dog
Observed Label: ship True Label: ship
Observed Label: plane True Label: plane
Observed Label: bird True Label: bird
Observed Label: plane True Label: plane
Observed Label: bird True Label: bird
Observed Label: plane True Label: plane
Observed Label: plane True Label: plane
Observed Label: plane True Label: plane
Observed Label: deer True Label: deer
Observed Label: plane True Label: plane
Observed Label: plane True Label: plane
Observed Label: dog True Label: dog
Observed Label: plane True Label: plane
(b)

Figure 7: Examples of 16 **least** harmful examples for MNIST and CIFAR-10 with annotation noise as ranked by EWC-influence. None of the selected examples were corrupted with annotation noise.

[2] `https://github.com/sato9hara/sgd-influence`

Random input noise. Fig. 8 shows examples that are ranked highest (most harmful) by EWC-influence for datasets contaminated with random input noise. Recall that the target classes of the input noise are 1, 6, 7, 9 for MNIST and plane, bird, cat, dog for CIFAR10, and the rest of the images are free of such noise. First of all, we observe that the most detrimental examples belong to the non-target classes. As shown in the main text, while the input noise itself may not harm the performance of the model by much, the sample size of clean images in the target classes is still smaller as a result of noise injection—such group imbalance can be rectified by sub-sampling the dominant group, for example, by removing those identified detrimental data points. Secondly, many of them appear to be ambiguous instances in non-target classes. It is worth noting that for MNIST, one of the identified examples is interestingly a real instance of 3 that is incorrectly labelled as 5, which is also reported by Northcutt *et al.*[42].



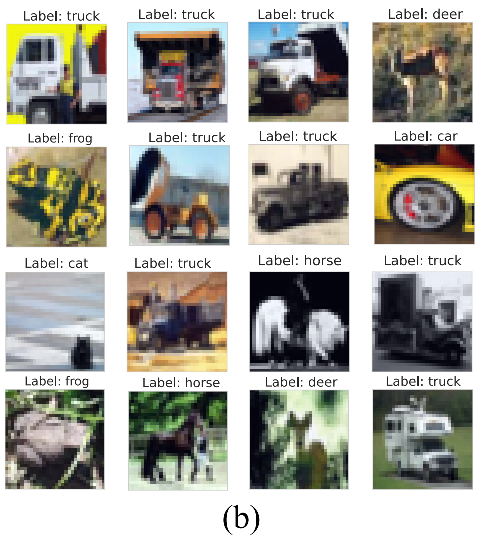
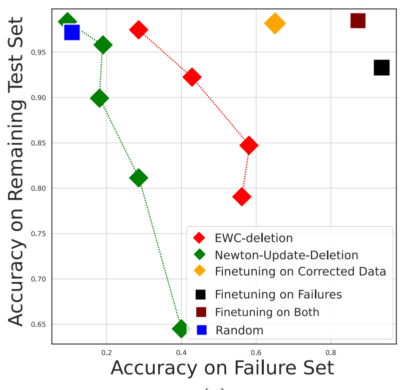


Figure 8: Examples of 16 most harmful examples for MNIST and CIFAR-10 with random input noise as ranked by EWC-influence. None of the selected examples were corrupted with random input noise.

B.4 Treating Models by Forgetting Detrimental Past and Learning from Present Mistakes

While the primary goal of this work is to investigate how many prediction errors can be remedied by identifying harmful training data and removing them, one could alternatively use the labelled failure cases directly to adapt the model. Fig. 9 shows our preliminary results where we compare the deletion based methods to an approach that fine-tunes the model directly on the failure query set $\mathcal{F}_q$ with an L2-norm based locality constraint $||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_2^2$ [13] and its combination with the best deletion-based approach, that is, fine-tuning on the corrected dataset $\mathcal{D}\backslash\mathcal{C}$. As with other experiments, early stopping is performed based on the loss on a portion of the query set $\mathcal{F}_q$. We see that while fine-tuning on $\mathcal{F}_q$ (black points) leads to a higher accuracy (on the holdout failure set $\mathcal{F}_h$) than fine-tuning on $\mathcal{D}\backslash\mathcal{C}$ (yellow points), the accuracy on the remaining test set $\mathcal{T}\backslash\mathcal{F}$ is worse, even with the weight constraint — by 6% on MNIST and 21% on CIFAR-10. This issue of over-fitting to the failure cases is also reported in recent works such as [15, 14]. Importantly, by combining the two approaches (brown points), we can attain the best trade-off between the failure and the maintenance accuracy, indicating the complementarity between the proposed data correction methods and such fine-tuning approaches.



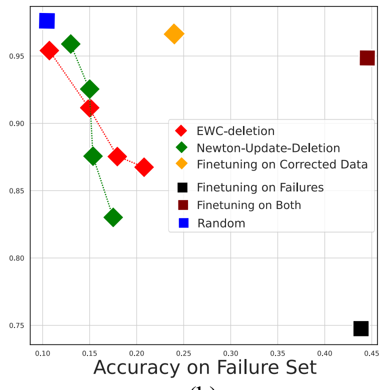


Figure 9: Comparison of deletion-based treatment methods and the direct fine-tuning on the failures on (a) MNIST and (b) CIFAR-10 datasets with noisy annotations.

C Experimental Details

Datasets we perform our experiments on the MNIST digit classification task and the CIFAR-10 object recognition task. The MNIST dataset consists of 60,000 training and 10,000 testing examples, all of which are 28×28 grayscale images of digits from 0 to 9 (10 classes). The CIFAR-10 dataset consists of 50,000 training and 10,000 testing examples, all of which are 32×32 coloured natural images drawn from 10 classes. Both datasets are preprocessed by subtracting the mean, but no data augmentation is used. For MNIST, to make the task more challenging, we randomly select 3000 examples from the training split and train the base models while the entire test set is used for evaluation.

Architecture Details For MNIST, the base classifier was defined as a CNN architecture comprised of 4 convolution layers, each with 3×3 kernels follower by Relu. The number of kernels in respective layers are $\{32,32,64,64\}$. After the first two convolution layers, we perform 2×2 max-pooling, and after the last one, we further down-sample the features with Global Average Pooling (GAP) prior to the final fully connected layer. For CIFAR-10, we used a 50-layer ResNet [44].

Optimisation For all experiments, we employ the same training scheme unless otherwise stated. We optimize parameters using Adam [41] with initial learning rate of 10^{-3} and $\beta = [0.9, 0.999]$, with minibatches of size 64 and train for max 100 epochs with early stopping with a patience of 5, that is, training is stopped after 5 epochs of no progress on the validation set (10% of the training set). For computing both EWC-influence and EWC-deletion, we also employed the same training scheme but applied early stopping based on the performance on a validation split (10%) of the failure query set $\mathcal{F}_q$.

In Fig. 5 in Sec. 4.2, we present the performance of Newton update removal and EWC-deletion (ours) with different hyper-parameter settings. For Newton-update deletion [9], we vary the step size $\gamma > 0$ of the gradient ascent by scaling the second term in Eq. (14). For EWC-deletion (our method), we vary the amount of weight regularisation — the second term in Eq. (15) — by scaling it by $2/\gamma N$ where N denotes the number of training datapoints, and $\gamma > 0$. We also set the strength of the prior term to $\lambda = 0$. We run both methods for different values of γ in the range $[0.01, 0.05]$.