Simple Continual Learning Strategies for Safer Classifers

Ashish Gaurav[1] **Sachin Vernekar**[1] **Jaeyoung Lee**[2]

Vahdat Abdelzad[2] **Krzysztof Czarnecki**[2] **Sean Sedwards**[2]

[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
University of Waterloo

Abstract

Continual learning is often confounded by "catastrophic forgetting" that prevents neural networks from learning tasks sequentially. In the case of real world classification systems that are safety-validated prior to deployment, it is essential to ensure that validated knowledge is retained. In this work, we propose methods that build on existing unconstrained continual learning solutions, which increase the model variance to better retain more of the existing knowledge (and hence safety). We demonstrate the improved performance of our methods against popular continual learning approaches, using variants of standard image classification datasets.

1 Introduction

Machine learning using neural networks has achieved considerable success in applications such as image recognition, game-playing, content recommendation and healthcare (LeCun, Bengio, and Hinton 2015). Most of these applications require large amounts of training data and careful selection of architecture and parameters. Importantly, the learned systems often have to adapt to changing real-world requirements, and therefore require re-training. Under these circumstances, it is usually desired to retain performance on previous tasks while learning to perform well on new tasks. This is what constitutes continual learning (McCloskey 1989).

Any strategy used for continual learning has to balance *plasticity* (the ability to learn new tasks) and *stability* (the ability to remember previous tasks). This is the well discussed stability-plasticity dilemma (Parisi et al. 2019). This dilemma can be explained in terms of the bias-variance tradeoff, another well known concept in statistical learning (Geman, Bienenstock, and Doursat 1992). In this context, (model) variance characterizes the span of solutions that can be realized with a neural network. Adding bias reduces the variance of the model and can produce a better solution than the unbiased case (Gigerenzer and Brighton 2009). With no bias, the continual learning model is plastic (has high variance); with strong bias, the model is stiff with respect to learning new tasks (has low variance). In this work we propose ways of increasing the variance, without the model becoming too plastic.

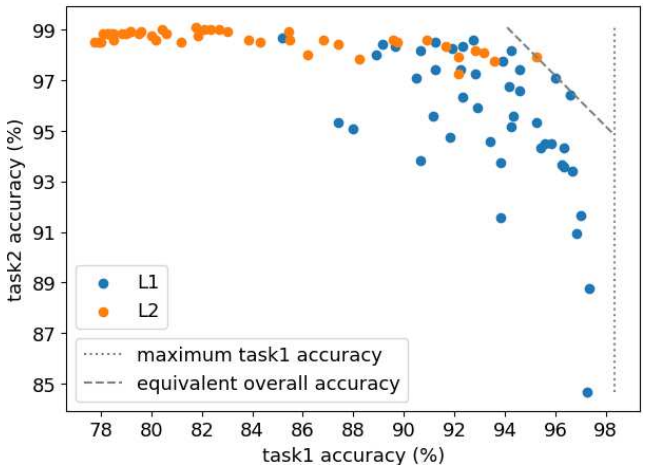

Figure 1: The nature of solutions obtained with L_2 vs L_1 constraints on Sim-EMNIST with 2 tasks, 5 seeds and different strengths ($\lambda \in [1, 10^4]$). L_1 solutions have higher variance and preserve the previous behavior more strongly than traditional L_2 strategies. See Section 3.1.

We summarize the existing continual learning approaches into three broad categories:

Architectural approaches (Yoon et al. 2018; Li et al. 2019) incrementally grow the network to learn the new task through the added capacity, that is, the untrained weight parameters. While adding new parameters increases the complexity of the model, these approaches either freeze some parameters or force the network's output to stay the same, which reduces the model variance.

Regularization approaches (Kirkpatrick et al. 2016; Zenke, Poole, and Ganguli 2017; Wiewel and Yang 2019; Chaudhry et al. 2018) assume a fixed network architecture and regularize changes to crucial weights, so that the network can learn to perform well on the new task by changing less significant weights. The regularized setup has an overall loss that can be broken down into a loss term for the current tasks, and a regularization term to stay close to previously found configurations. The regularization term restricts the

span of solutions the model can achieve.

Memory approaches (Lopez-Paz 2017; Nguyen et al. 2018) store examples from each task being learned and then learn a new task while simultaneously maximizing performance on each of the stored memories. Seeing more data per update or constraining the update to move in a direction that does not degrade performance on the stored memories reduces the variance of the model.

Our work focuses on regularization approaches for continual learning. We note that performance in previous work is often judged with respect to average validation accuracy across tasks. While good average validation accuracy is the most common metric to judge forgetting, our principal concern is safety. In safety-critical systems, it may not be acceptable to maintain an average validation accuracy at the cost of trading previously certified decisions for possibly correct decisions that have not been checked. Likewise, the calibration of a system may require that all classifier predictions, good or bad, remain the same. Therefore, in the present work, we consider exactly what has been forgotten and what has been learned.

In what follows, we propose continual learning methods that emphasize retaining previously learned tasks. In Section 3, we propose setting an upper bound on the absolute amount of forgetting that can occur over a certain dataset, and show that this produces a weaker bias than in KL-approaches, elastic weight consolidation (EWC) and synaptic intelligence (SI). In Section 4.1, we propose using the per-parameter importance computed through this upper bound to perform regularized continual learning. In Section 4.2, we modify the learning procedure to ensure that the upper bound on forgetting never exceeds a pre-specified amount. In Section 4.3, we extend the existing EWC algorithm to achieve a similar trend of network preservation. Finally, in Section 5, we evaluate and discuss the proposed techniques on variants of MNIST, EMNIST and CIFAR100 datasets. We observe that different strengths of biases may be useful for different datasets, due to the nature of the solutions achieved by the different variants.

2 Background

Notation We denote by $|\mathbf{x}|$ a vector with the same dimensions as $\mathbf{x}$, such that each element in $|\mathbf{x}|$ is the absolute of the corresponding element in $\mathbf{x}$. By $\mathbf{x} \cdot \mathbf{y}$ we denote the inner product of $\mathbf{x}$ and $\mathbf{y}$. $|\mathbf{x}|^2$ denotes the element-wise square of vector $\mathbf{x}$, that is, $|\mathbf{x}|^2 := \mathbf{x} \odot \mathbf{x}$ (Hadamard product). $\|\mathbf{x}\|_1$ and $\|\mathbf{x}\|_2$ denote the standard L_1 and L_2 norms (scalars) of vector $\mathbf{x}$. For a matrix M, the operation $\textbf{diag}(M)$ produces a vector that consists of the leading diagonal of M. For a scalar x, $\text{sign}(x)$ represents the signum function, which produces 1 if $x > 0$, 0 if $x = 0$ and -1 otherwise. We denote by $1 : i$ the sequence of indices $1, 2, \ldots, i$. The vector of weights $\boldsymbol{\theta}$ after training tasks $1 : i$ is denoted $\boldsymbol{\theta}^*_{1:i}$. The j-th element of $\boldsymbol{\theta}^*_{1:i}$ is denoted $\theta^*_{1:i,j}$.

2.1 Classification

The objective of classification is to maximize performance (validation accuracy) for a given dataset $\mathcal{D}$. This can be achieved by using a ReLU feedforward neural network (final layer is followed by softmax instead of ReLU) with weights $\boldsymbol{\theta}$ as the method of function approximation. More concretely, given an input x, such a network outputs a set of positive numbers ≤ 1 that approximate how likely x belongs to class m:

$$\{P_{\boldsymbol{\theta}}(y = m|x)\}_{m=1}^{M} \tag{1}$$

M is the number of classes and the softmax function ensures that the values sum to 1. For notational simplicity, we denote $P_{\boldsymbol{\theta}}(y = m|x)$ as $P^m_{\boldsymbol{\theta}}(\cdot|x)$. If the ground truth for x is $y = g$, where $(1 \leq g \leq M)$, then we use the shorthand $P_{\boldsymbol{\theta}}(\cdot|x) := P^g_{\boldsymbol{\theta}}(\cdot|x)$ for the predicted likelihood of label g.

In this work we use the cross entropy loss, or the negative log likelihood loss, $\mathcal{L}(\boldsymbol{\theta})$, defined over an example (x, y) as:

$$\mathcal{L}(\boldsymbol{\theta}) = -\log P_{\boldsymbol{\theta}}(\cdot|x) \tag{2}$$

Gradient descent based optimizers that train deep feedforward networks to minimize an objective use the negative gradient of such a loss over a batch of examples, i.e., $-\mathbf{E}_x[\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})]$, to compute the change in weights per optimization step.

2.2 Continual Learning

Let there be n datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$ such that dataset $\mathcal{D}_i$ has K_i examples:

$$\mathcal{D}_i = (\mathbf{X}_i, \mathbf{Y}_i) = (\{x_i^{(k)}\}_{k=1}^{K_i}, \{y_i^{(k)}\}_{k=1}^{K_i}) \tag{3}$$

For any task i, the weights achieved at the end of task i ($\boldsymbol{\theta}^*_{1:i}$) should also retain performances on tasks $1, 2, \ldots, i-1$. Therefore, notionally, $\boldsymbol{\theta}^*_{1:i}$ should minimize the cross entropy loss over datasets $\mathcal{D}_{1:i} := \mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_i$.

The continual learning objective can be naively achieved by training on examples from all relevant datasets (*joint training*). Joint training quickly becomes expensive as the number of tasks grow, but typically has the best performance across all tasks (Li and Hoiem 2017).

2.3 Regularization Approaches

Regularized approaches to continual learning assume a fixed capacity of the network and jointly optimize two objectives per task through a combined loss $\mathcal{L}(\boldsymbol{\theta})$ as follows:

$$\mathcal{L}(\boldsymbol{\theta}) := \mathcal{L}_i(\boldsymbol{\theta}) + \lambda G_{1:i-1}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1}) \tag{4}$$

$$\boldsymbol{\theta}^*_{1:i} = \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{5}$$

G is the regularization loss.

To incorporate the knowledge for an example (x, y), the optimization step computes a gradient update:

$$\Delta\boldsymbol{\theta} \propto -\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{6}$$

Note that many optimization methods use additional strategies to converge faster to a solution, like momentum (Sutskever et al. 2013), adaptive gradients (Duchi, Hazan, and Singer 2011) and moment estimation (Kingma and Ba 2014). However, on their own they do not offer any particular advantage for continual learning. Hence, for simplicity, we just discuss the first order gradient update on the

regularized continual learning objective, that is $-\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$. This gradient update is sufficiently informative to provide useful insights about how these methods preserve existing knowledge.

With the regularized loss, the gradient update breaks down into two components: $-\nabla_{\boldsymbol{\theta}}\{\mathcal{L}_i(\boldsymbol{\theta})\}$ and $-\nabla_{\boldsymbol{\theta}}\{\lambda G_{1:i-1}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1})\}$. The former can be interpreted as the *plasticity update*, which tries to optimize the current task's objective, while the latter can be interpreted as the *stability update*, which tries to maintain existing knowledge.

Plasticity Update Throughout this work, we use the cross-entropy loss (2) to compute the plasticity update. The cross-entropy objective is a suitable plasticity update, since it produces exponentially strong updates as $P_{\boldsymbol{\theta}}(\cdot|x) \to 0$.

Stability Update Most regularization methods in the literature use a second order Taylor approximation on the following KL divergence as the regularization loss $G_{1:i-1}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1})$:

$$\begin{aligned}&\mathrm{KL}\big[\, p(\boldsymbol{\theta}|\mathcal{D}_{1:i-1}) \,||\, p(\boldsymbol{\theta}|\widetilde{\mathcal{D}}_{1:i-1}) \,\big] \\ &\approx \frac{1}{2}|\Delta\boldsymbol{\theta}|^2 \cdot \mathbf{diag}(F) \end{aligned} \tag{7}$$

Here, $F \equiv F(\boldsymbol{\theta}^*_{1:i-1}, \mathcal{D}_{1:i-1})$ refers to the the empirical Fisher information matrix, evaluated at $\boldsymbol{\theta}^*_{1:i-1}$, and $p(\cdot)$ refers to the weight distributions, the maximum likelihood estimate of which is found by the optimizer. While learning $\mathcal{D}_i$, from a Bayesian perspective, $p(\boldsymbol{\theta}|\mathcal{D}_{1:i-1})$ represents the prior weight distribution and $p(\boldsymbol{\theta}|\widetilde{\mathcal{D}}_{1:i-1})$ represents the posterior weight distribution after seeing some more data from $\mathcal{D}_i$. We refer the reader to (Kirkpatrick et al. 2016; Huszár 2018) for a better understanding of this approximate formulation.

All methods that use this approximation will construct a stability update that resembles the form $-\mathbf{a}\cdot(\boldsymbol{\theta}-\boldsymbol{\theta}^*)$. We can easily see this for the case of two popular approaches, EWC and SI. EWC (Kirkpatrick et al. 2016) constructs a regularization loss $G_{1:i-1}(\boldsymbol{\theta})$ by multiplying each $(\theta_j - \theta^*_{1:i-1,j})^2$ by the corresponding diagonal term F_{jj} in the Fisher information matrix. The squared displacement of the weight θ_j from the previous weight $\theta^*_{1:i-1,j}$ is scaled by the Fisher importance of the weight. The stability update therefore has the form:

$$-\nabla_{\boldsymbol{\theta}}\{\lambda G_{1:i-1}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1})\} = -\lambda \sum_j F_{jj}(\theta_j - \theta^*_{1:i-1,j})$$

This update can be understood as an elastic constraint (much like a spring) which forces θ_j to be close to $\theta_{1:i-1,j}$. The strength of this stability update for θ_j depends on λ, the Fisher information term F_{jj} for θ_j and the magnitude of $(\theta_j - \theta^*_{1:i-1,j})$. The direction of this update is always towards $\theta^*_{1:i-1,j}$. SI (Zenke, Poole, and Ganguli 2017) uses a similar form as EWC, but calculates the per parameter importance Ω^i_j in a different way. Specifically, Ω^i_j is defined by how much θ_j has previously affected the change in total loss, and how much it (θ_j) has been modified, rather than relying on Fisher importance. Note that the form of the stability update remains similar to EWC:

$$-\nabla_{\boldsymbol{\theta}}\{\lambda G_{1:i-1}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1})\} = -2c \sum_j \Omega^i_j(\theta_j - \theta^*_{1:i-1,j})$$

This is still the elastic constraint, except that the per-parameter importance is Ω^i_j, rather than the Fisher importance. There are other approaches that use a similar form of update but use a different strategy to compute the constant of regularization, for example, EWC++ (Chaudhry et al. 2018).

We note that while the elastic constraint and its variants constitute a good stability bias for continual learning, they may be stronger than is needed. Specifically, they minimize the KL divergence between the posterior and prior but do not directly consider the amount of forgetting, which affects the preservation of (validated) knowledge.

In what follows, we propose using a weaker stability bias which preserves previous knowledge more strongly.

3 Change in Classifier Prediction

In this section, we quantify the amount of forgetting as a network learns new data. This quantification provides a generic strategy to define the per-parameter importance in the stability update. More concretely, we provide an approximate upper bound on the absolute amount of forgetting over a dataset.

3.1 Absolute Amount of Forgetting

After learning a task i, the weights of the network are $\boldsymbol{\theta}^*_{1:i}$. For simplicity, let $\boldsymbol{\theta}^*_{1:i} \equiv \boldsymbol{\theta}^*$ and that afterwards, at any point in the sequential training process, the weights are at $\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}$. Assuming $\Delta\boldsymbol{\theta}$ is small, we can apply a first order Taylor approximation of the individual predicted likelihood $P^m_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}}$ in the neighborhood of $\boldsymbol{\theta} = \boldsymbol{\theta}^*$:

$$P^m_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}} \approx P^m_{\boldsymbol{\theta}^*} + \Delta\boldsymbol{\theta} \cdot (\nabla_{\boldsymbol{\theta}} P^m_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \tag{8}$$

The individual predicted likelihood $P^m_{\boldsymbol{\theta}}$ on an example $x \in \mathcal{D}_i$ changes by the magnitude

$$|\Delta P^m_{\boldsymbol{\theta}^*}| = |P^m_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}}(\cdot|x) - P^m_{\boldsymbol{\theta}^*}(\cdot|x)|. \tag{9}$$

On average, the magnitude change in the individual predicted likelihood $P^m_{\boldsymbol{\theta}}$ over the dataset $\mathcal{D}_i$ is given by the expectation

$$\begin{aligned}&\mathbf{E}_{(x,y)\sim\mathcal{D}_i}\left[\left|P^m_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}}(\cdot|x) - P^m_{\boldsymbol{\theta}^*}(\cdot|x)\right|\right] \\ &\leq |\Delta\boldsymbol{\theta}| \cdot \mathbf{E}_{(x,y)\sim\mathcal{D}_i}\left[\left|\nabla_{\boldsymbol{\theta}} P^m_{\boldsymbol{\theta}}(\cdot|x)\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}\right|\right] \qquad (10) \\ &\equiv \sum_j C^m_j \|\theta_j - \theta^*_{1:i,j}\|_1 \equiv u^m(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}). \qquad (11)\end{aligned}$$

At every task i, we can minimize $u^m(\mathcal{D}_{i'}, \boldsymbol{\theta}^*_{1:i'}, \boldsymbol{\theta})$ directly for each of the previous datasets i', and this minimization should mitigate catastrophic forgetting (C^m_j is parameter importance for θ_j) for the network's outputs for label m

$(1 \leq m \leq M)$. This constitutes our *minimization criterion*. We extend the notation to use u^m when computing this upper bound on P^m, and u when computing it for the ground truth.

A Weaker Constraint If this absolute amount of forgetting is a part of the stability update, then the stability update corresponds to a weaker constraint:

$$-\nabla_{\boldsymbol{\theta}}\{\lambda G_{1:i}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1})\} = -\nabla_{\boldsymbol{\theta}}\{\lambda u^m(\mathcal{D}_{i-1}, \boldsymbol{\theta}^*_{1:i-1}, \boldsymbol{\theta})\}$$

$$\equiv -\nabla_{\boldsymbol{\theta}}\left\{\lambda \sum_j C_j^m \|\theta_j - \theta^*_{1:i-1,j}\|_1\right\} \tag{12}$$

$$= -\lambda \sum_j C_j^m \operatorname{sign}(\theta_j - \theta^*_{1:i-1,j}) \tag{13}$$

Unlike the elastic constraint, the strength of this stability update for θ_j depends on λ and on C_j^m, but not on the magnitude of $(\theta_j - \theta^*_{1:i-1,j})$; the direction of this update is always towards $\theta^*_{1:i-1,j}$. This is therefore a weaker stability bias.

To understand the nature of the solutions obtained by this weaker update, we consider a simple two task example (Sim-EMNIST, but for two tasks). Using the method described in Section 4.1, we plot the task 1 and task 2 accuracies on the x and y axes and obtain a figure as shown in Figure 1. We observe that as the strength of λ increases, more of the task 1 accuracy is preserved, but at some point the performance on task 2 decreases. Of the two variants, the L_1 constraint obtains sparse solutions, which are almost always more restrictive on forgetting task 1. This is by design, since the goal is to minimize forgetting on previous task(s) more strongly, as explained later. The vertical dotted line corresponds to the maximum accuracy achieved at the end of training task 1. The dashed diagonal line represents the line across which the best equivalent overall accuracy will lie ($x + y = c$ for some constant c).

The point at which the performance on task 2 decreases corresponds to the optimum of the stability-plasticity curve. To understand this curve, in Figure 2 we plot the final average validation accuracy for this two task example across a wide range of λ. As can be observed, for both the L_1 and L_2 versions of absolute expected amount of forgetting, there are different values of λ at which the approach achieves (joint) best stability and plasticity. This is achieved with lower values of λ for the L_1 variant than for the L_2 variant.

Analogy with Regression Models The regularized continual learning objective has a similar shape to the objective in biased regression models. Regression models typically minimize an ordinary least-squares objective OLS($\boldsymbol{\theta}$) but suffer from a high variance problem. The most popular ways to reduce this variance is through LASSO and ridge methods, which introduce a bias onto the estimate of regression weights $\boldsymbol{\theta}$ through a regularization term. LASSO adds an L_1 term $\|\boldsymbol{\theta}\|_1$ as the regularization term, while ridge adds an L_2 term $\|\boldsymbol{\theta}\|_2^2$ as the regularization term. The effect of these biases is that the solution satisfies $\|\boldsymbol{\theta}\|_1 \leq t$ or $\|\boldsymbol{\theta}\|_2^2 \leq t$, for some small t, with LASSO producing a sparse solution, that is, a solution with fewer non-zero weights. From a Bayesian standpoint, this is because LASSO shrinkage is equivalent to a Laplace prior, while ridge shrinkage is equivalent to a Gaussian prior. Since a Laplace prior is more peaky towards its mean, the solution is more likely to be the mean (and near the mean) than with a Gaussian prior.

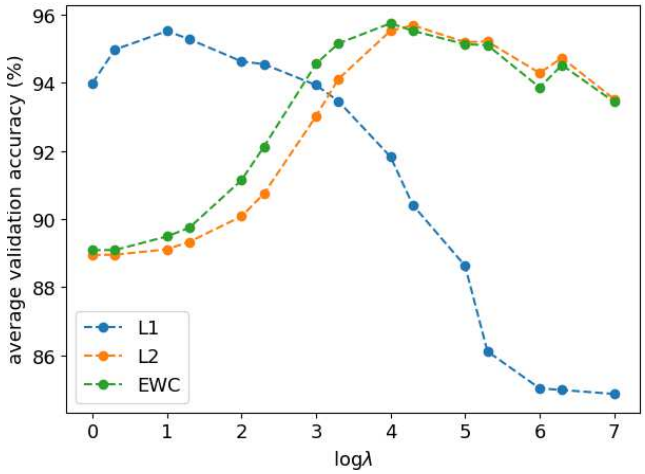

Figure 2: The effect of λ on the final average validation accuracy for DM-L_1 (Case III), DM-L_2 (Case III) and EWC, for Sim-EMNIST, two tasks, 10 seeds. The different approaches achieve the optimum joint stability and plasticity at different strengths of λ.

Similarly, regularized continual learning minimizes the loss for the most recent task $\mathcal{L}_i(\boldsymbol{\theta})$, while ensuring that $G_{1:i-1}(\boldsymbol{\theta}, \boldsymbol{\theta}^*_{1:i-1}) \leq t$. When using the approximate KL-divergence for G, it yields the Bayesian interpretation (as explained in EWC) that the KL-divergence between the prior $p(\boldsymbol{\theta}|\mathcal{D}_{1:i-1})$ and the posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:i})$ should be $\leq t$ while simultaneously minimizing the cross-entropy loss. When using the absolute amount of the forgetting, it can be understood as trying to keep the absolute amount of forgetting from previous datasets within a certain t. With the L_1 version of the constraint, more θ_j are likely to not change than in the L_2 version.

Equivalence with KL Approaches If the classification behavior corresponding to the ground truth label is desired to be preserved, the minimization criterion has a similar form to KL approaches, which use a loss based on the KL-approximation stated in (7). More concretely, the squared L_2 version of our upper bound is smaller than the second-order approximate KL divergence (with the diagonal of the empirical Fisher). This means that when using the expectation term instead of the Fisher, the stability updates are smaller in magnitude, and thus, less strong. This interpretation of per-parameter updates achieves a slightly higher model variance,

which is aligned with our goal. The proof is as follows:

$$\begin{aligned}
&\mathbf{E}_{(x,y)\sim\mathcal{D}_i}\left[\left(P_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}}(\cdot|x) - P_{\boldsymbol{\theta}^*}(\cdot|x)\right)^2\right] \\
&(\approx) \leq |\Delta\boldsymbol{\theta}|^2 \cdot \mathbf{E}_{(x,y)\sim\mathcal{D}_i}\left[\left|\nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\cdot|x)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}\right|^2\right] \\
&\leq |\Delta\boldsymbol{\theta}|^2 \cdot \mathbf{E}_{(x,y)\sim\mathcal{D}_i}\left[\left|\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\cdot|x)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}\right|^2\right] \\
&= \sum_j F_{jj}(\theta_j - \theta_j^*)^2
\end{aligned}$$

3.2 Extension to Different Use Cases

The formulation of the upper bound in terms of the output label allows for its adaptation to different use cases. Specifically, real world systems have different motivations for continual learning, which are dependent on the *requirement* specification. For example, in an outlier detection system, it may be desired to preserve the reject class more strongly than the rest of the classes, since retaining knowledge about the outlier examples is of the highest priority. Therefore, more generally, depending on the requirement, it may be desired to preserve the network behavior expressed by all or some of the output neurons. We identify four such use cases, which we illustrate in Figure 3 and describe below:

Case I We can preserve the entire set of predicted likelihoods from $\boldsymbol{\theta}^*$ to $\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}$, which penalizes changes to any individual predicted likelihood. This is the most restrictive version of the criterion and can be achieved by regularizing a sum over $1 \leq m \leq M$ of the individual changes:

$$u_{\mathrm{I}}(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) := \sum_{1\leq m\leq M} u^m(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) \qquad (14)$$

Case II We can preserve the change in predicted likelihood for the confident label(s) at $\boldsymbol{\theta}^*$, which typically corresponds to the highest individual probability in $\{P^m_{\boldsymbol{\theta}^*}(\cdot|x)\}_{m=1}^M$. This may be desired in tasks related to safety-critical systems, where a network has been safety-calibrated at deployment and now needs to add some more knowledge without violating previously satisfied safety calibrations. To achieve this, we can first compute an expectation over $|(\Delta P^m_{\boldsymbol{\theta}^*})P^m_{\boldsymbol{\theta}^*}|$ rather than the formulation in (9):

$$\mathbf{E}_{(x,y)\sim\mathcal{D}_i}\left[\left|(\nabla_{\boldsymbol{\theta}} P^m_{\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*})P^m_{\boldsymbol{\theta}^*}\right|\right] \equiv \bar{u}^m(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) \quad (15)$$

Since the network's outputs represent the confidence of the x being a certain label, highly confident $P^m_{\boldsymbol{\theta}^*}$ ends up contributing more to the expectation. Then, a sum over $1 \leq m \leq M$ of the confidence weighted upper bound $\bar{u}^m$ can be used:

$$u_{\mathrm{II}}(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) := \sum_{1\leq m\leq M} \bar{u}^m(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) \qquad (16)$$

Case III We can preserve the absolute change in predicted likelihood for the ground truth by directly regularizing $u_{\mathrm{III}}(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) := u(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta})$. This corresponds to the typical motivation, which is to maintain a stable average validation accuracy.

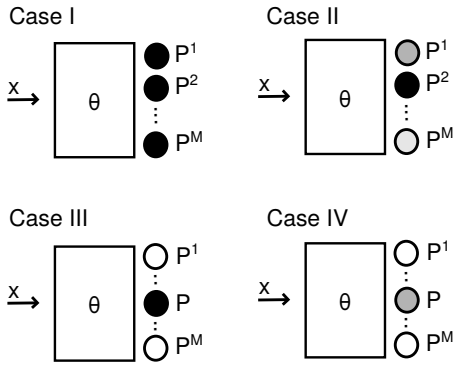

Figure 3: Use cases I-IV. The intensity of the output neurons denote their contribution. Case I preserves all the outputs. Case II preserves the outputs in proportion to their confidence. Case III preserves only the ground truth output. Case IV preserves the ground truth in proportion to its confidence.

Case IV We can partially preserve the change in predicted likelihood for the ground truth, that is, penalize the change $P_{\boldsymbol{\theta}^*}(\cdot|x) = 1 \rightarrow P_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}}(\cdot|x) = 0$, but allow the change $P_{\boldsymbol{\theta}^*}(\cdot|x) = 0 \rightarrow P_{\boldsymbol{\theta}^*+\Delta\boldsymbol{\theta}}(\cdot|x) = 1$ for the ground truth predicted likelihood. This applies the penalty only when a correctly classified x at $\boldsymbol{\theta}^*$ (high confidence on the ground truth) becomes incorrectly classified (lower confidence on the ground truth) at $\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}$. Using previously defined notation, this yields:

$$u_{\mathrm{IV}}(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) := \bar{u}(\mathcal{D}_i, \boldsymbol{\theta}^*_{1:i}, \boldsymbol{\theta}) \qquad (17)$$

4 Strategies and Applications

With a bound on absolute forgetting, we have a quantification of the degree of preserved knowledge, and hence a quantification of the degree of preserved validated knowledge. In this section, we provide a few strategies to directly minimize this quantification.

4.1 Regularizing the Amount of Forgetting

The upper bound(s) on the amount of forgetting can be directly used in the stability update. With the formulation now being understood in terms of the amount of forgetting, we need to ensure that the sum of absolute amounts of forgetting is $\leq t$ from all previous datasets. We can encode this objective by choosing a $G_{1:i-1}(\boldsymbol{\theta})$ in (4) as follows:

$$G_{1:i-1}(\boldsymbol{\theta}) := \sum_{1\leq i'\leq i-1} g(\mathcal{D}_{i'}, \boldsymbol{\theta}^*_{1:i'}, \boldsymbol{\theta}) \equiv \sum_{1\leq i'\leq i-1} g_{i'}$$

Here, g can be either $u_{\mathrm{I}}, u_{\mathrm{II}}, u_{\mathrm{III}}, u_{\mathrm{IV}}$ for each of the previous datasets, depending on the requirement. After finishing the training for a particular dataset $\mathcal{D}_i$, $g(\cdot)$ can be com-

puted and added to $G_{1:i-1}(\boldsymbol{\theta})$ to produce $G_{1:i}(\boldsymbol{\theta})$. With this definition, the objective can optimized as expressed in (5).

To evaluate which version of the stability bias (L_1 or L_2) is appropriate for knowledge preservation, we conduct experiments with the L_1, L_2 and the elastic-net variants of the objective. The L_1 and L_2 variants are described in Section 3.1, and the elastic-net variant is a weighted combination of these two variants (Zou and Hastie 2005); we use an equally weighted combination, which we refer to as $E0.5$. We refer to the family of these methods as *direct minimization* (DM) strategies.

4.2 Finer Control over Forgetting

Minimization through the Lagrange method of multipliers produces a minimum for $\mathcal{L}_i(\boldsymbol{\theta})$, such that the absolute amount of forgetting over datasets $\mathcal{D}_{1:i-1}$ is bounded by some t, that is $G_{1:i-1}(\boldsymbol{\theta}) \leq t$. Without any additional assumptions on the solution, the value of t is determined by the minimization procedure. However, it is possible to alter the minimization procedure to achieve more control over the amount of forgetting t, and hence the degree of preservation. While this introduces a stronger bias and thereby reduces the variance of the model, it also gives us a hyperparameter to find the precise t at which the solution is optimal for a certain stability bias.

Specifically, let us assume that we do not want $G_{1:i-1}(\boldsymbol{\theta})$ to exceed some c. We can attach a scalar identity term $\mathbb{I}$ to the cross-entropy objective as follows:

$$\mathcal{L}_i(\boldsymbol{\theta}) \cdot \mathbb{I}(G_{1:i-1}(\boldsymbol{\theta}) \leq c) + \lambda G_{1:i-1}(\boldsymbol{\theta}) \tag{18}$$

As long as $G_{1:i-1}(\boldsymbol{\theta}) \leq c$, this training objective is equivalent to the description in (4) and (5). When $G_{1:i-1}(\boldsymbol{\theta}) > c$, the training objective directly minimizes $G_{1:i-1}(\boldsymbol{\theta})$ until $G_{1:i-1}(\boldsymbol{\theta}) \leq c$. This forces the optimizer to re-focus on finding a solution that first satisfies $G_{1:i-1}(\boldsymbol{\theta}) \leq c$, and thereby forces the optimization to not exceed a certain c (stability is enforced). A stricter objective can be obtained by individually considering the different g within G:

$$\mathcal{L}_i(\boldsymbol{\theta}) \cdot \mathbb{I}(g_1(\boldsymbol{\theta}) \leq c_1, \cdots, g_{i-1}(\boldsymbol{\theta}) \leq c_{i-1}) + \lambda G_{1:i-1}(\boldsymbol{\theta}) \tag{19}$$

In our experiments, we use this stricter objective, maintaining a forgetting threshold (c_i) per task i. We initialize $c_i \leftarrow c^{(1)}$ and then incrementally increase this forgetting threshold as we see more tasks, that is, $c_i \leftarrow c_i + c^{(2)}$ per new task seen. Through the hyperparameter search, we can obtain a minimum c which produces the best solution for a given stability bias.

4.3 Fisher Freezing

With any regularization strategy, all the weights are always updated, even if the changes to some weights are very small. This can perturb sensitive weights, such as early layer weights (Raghu et al. 2017). Even if this perturbation is only small, small perturbations can add up over multiple tasks and eventually affect the classifier likelihood irreversibly.

The upper bound of change in classifier likelihood for a dataset $\mathcal{D}_i$ is dependent on two terms (see (10)), $|\Delta\boldsymbol{\theta}|$ and the expectation of the absolute gradients. To minimize the change in classifier likelihood, we may opt to minimize $|\Delta\boldsymbol{\theta}|$ more conventionally, by freezing the most important weights. This reduces the magnitude of $|\Delta\boldsymbol{\theta}|$ and therefore results in a smaller change in classifier predictions. Other strategies in the literature have tried similar approaches (Serra et al. 2018). Note that this method directly enforces sparsity in $|\Delta\boldsymbol{\theta}|$.

Specifically, we compute the Fisher information matrix $F(\boldsymbol{\theta}^*_{1:i})$ and choose the top p-percentile parameters $\boldsymbol{\theta}_p \subseteq \boldsymbol{\theta}$. For these parameters, we ensure that the optimizer does not update their values. To assess the effects of this kind of freezing separately from the aforementioned L_1 criterion, we freeze weights on EWC. In our experiments, we refer to this method as *EWC-p*.

As an example, we plot the task 1 vs task 2 accuracy in Figure 4 for EWC with freezing. The nature of solutions with increasing p is similar to that depicted in Figure 1. An increase in p therefore reduces the plasticity of the model, which is expected. This shows that a higher span of solutions (higher variance) can be achieved directly through EWC, by introducing this hyperparameter p.

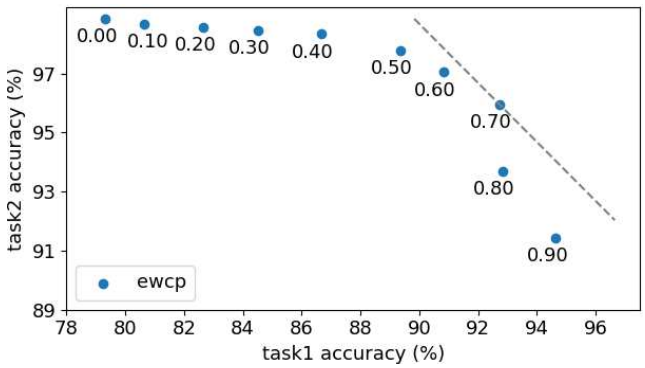

Figure 4: Fisher freezing on EWC applied to Sim-EMNIST; average of 5 seeds with $\lambda = 1$. As we freeze more weights, task 1 accuracy increases, while task 2 accuracy decreases. The trend of solutions is similar to that obtained using L_1.

5 Experiments

We evaluate our proposed methods and compare their performance with other popular KL based approaches in continual learning. We evaluate the following strategies:

Baseline Trained with just the likelihood loss, that is, no regularization (no stability bias).

EWC Accumulated Fisher information matrices and combined quadratic losses, as described in (Kirkpatrick et al. 2016; Huszár 2018; Kirkpatrick et al. 2018). We implemented this method from scratch, following the description in the related papers. Note that EWC was previously compared to the unweighted L_2 regularization $\sum_j (\theta_j - \theta^*_{1:i-1,j})^2$. This is different to the L_2 variant described in this work, which has a per-parameter importance.

SI Synaptic Intelligence strategy as described in (Zenke, Poole, and Ganguli 2017), using the original code released by the authors.

DM-I, II, III, IV Proposed in Section 4.1, a strategy that directly regularizes the amount of forgetting, for L_1, L_2 and the elastic-net variant.

DM-I, II, III, IV with fine control Proposed in Section 4.2, similar to the previous strategy but with additional hyperparameters that allow finer control over forgetting; evaluated for L_1, L_2 and the elastic-net variant.

EWC-p Freezing strategy described in Section 4.3; implemented on EWC.

5.1 Training Methodology

Training for each strategy is performed on feedforward ReLU networks with 2 hidden layers ($h = 128, \eta = 0.0001$), for 20 epochs. For hyperparameter search, we evaluate all methods on a single random seed, then choose a parameter that has the highest average validation accuracy. The final results (mean and standard deviation) are averaged over 5 seeds, using the best parameter. Table 1 shows the performance (final average validation accuracy) of the proposed methods.

We use the Adam optimizer for our experiments. Constants searched for EWC include $\lambda \in \{1, 10^1, 10^2, 10^3, 10^4\}$. For DM-I, II, III, IV (with and without finer control), we searched for $\lambda \in \{1, 10^1, 10^2, 10^3, 10^4\}$, $c^{(1)} \in \{0.025, 0.05, \cdots 0.10\}$ and $c^{(2)} \in \{0.0, 0.025, 0.05, \cdots 0.10\}$. For the CIFAR100 datasets, we searched for $\lambda \in \{1, 10^1, 10^2, \cdots 10^7\}$. For EWC-p, we searched for $p \in \{0.1, 0.2, 0.3, \cdots 0.9\}$. For SI, we searched for $c \in \{0.01, 0.1, 0.5, 1, 2\}$ and $\zeta \in \{0.001, 0.01, 0.1, 1\}$.

For the CIFAR100 experiments we use the embeddings from a pretrained Resnet-v1 model which achieves $\approx 70\%$ accuracy on the 100-class classification.

5.2 Datasets

We evaluate on the following datasets:

Permuted MNIST 5 task version, where every task is a 10-class classification on the MNIST dataset with permuted pixels; used in (Kirkpatrick et al. 2016; Zenke, Poole, and Ganguli 2017; Nguyen et al. 2018; Li et al. 2019).

Split MNIST 5 tasks, where every task is a 2-class classification. The tasks are labels 0/1, 2/3, 4/5, 6/7, and 8/9 from the MNIST dataset; used in (Chaudhry et al. 2018; Wiewel and Yang 2019).

Similar EMNIST Hand-picked labels from the EMNIST dataset such that the classification tasks look approximately similar; 4 tasks, 3-class classification, task labels are 2/O/U, Z/8/V, 7/9/W, and T/Q/Y.

CIFAR100 Realistic image dataset with 5 tasks; 3-class classification; task labels are 0/1/2, 3/4/5, 6/7/8, 9/10/11, and 12/13/14.

Similar CIFAR100 As CIFAR100, but tasks are chosen from superclasses such that the labels per task correspond to the coarse classes. We chose the coarse classes to be "aquatic mammals", "food containers" and "household furniture". Since each coarse class contains 5 superclasses, this corresponds to 3-class classification spanning over 5 tasks.

Two of the datasets of our choice are similar continual datasets. The effect of task similarity has been discussed previously (Kemker et al. 2018), but these discussions consider permutated data to be similar tasks. On the other hand, our investigation considers different datasets to be similar if the labels draw from some common superclass distribution of data. This is of practical significance since in real world classification systems, we usually desire our classification ability to persist when the newer classification tasks are labelwise similar to the previous data. For example, we expect a classifier that classifies between cars and motorbikes to easily (continually) learn the distinction between trucks and bikes.

Additionally, we note that a lot of the work in continual learning is evaluated on incremental classes, but with incremental classes the network is expected to remember just the labels that have been seen over multiple tasks. However, remembering multi-class classification requires remembering the differences between all the classes in each task. This is our rationale for the choice of few-class classification spanning over 4 or 5 tasks.

5.3 Results

Our numerical results are given in Table 1. We report the following insights:

Baselines and Existing Approaches As expected, the baselines for all the datasets incur catastrophic forgetting. This forgetting is less for the similar datasets Sim-EMNIST and Sim-CIFAR100, because the classification tasks are related, that is, learning the first task is enough to perform decently on following tasks. EWC and SI significantly improve upon the baseline accuracies.

DM-L_1 vs DM-L_2 We find that in almost all the datasets (an exception being Sim-EMNIST) the L_1 variant finds a better overall solution over multiple tasks, compared to the L_2 variant. In the case of fine control, this still holds on almost all the datasets, but the exception is Sim-CIFAR100. Note that even in the exceptional cases, the mean accuracies differ by a very small amount.

Fine Control We expect the fine-control version of DM-L_1 and DM-L_2 to perform the best in all datasets, because with the correct hyperparameters it can find the optimum which jointly maximizes both plasticity and stability. We indeed observe this for the grayscale datasets, where the improvements are good, but this does not hold in the case of the CIFAR100 datasets. We speculate that this is because of the relatively coarse granularity of the hyperparameter search for $c^{(1)}$ and $c^{(2)}$. Since the λ search for the CIFAR100 datasets was already computationally expensive, we chose not to repeat the search with finer granularity. Nevertheless, the best values for the fine methods are still better than EWC and SI.

Method	P-MNIST	S-MNIST	Sim-EMNIST	CIFAR100	Sim-CIFAR100
Baseline	55.63 (1.04)	63.36 (0.38)	75.38 (1.15)	37.48 (5.74)	76.25 (0.49)
EWC	93.86 (0.30)	70.85 (2.65)	89.65 (2.99)	61.70 (2.41)	83.76 (2.06)
SI	92.64 (0.75)	78.30 (2.65)	91.41 (1.21)	62.08 (1.34)	83.67 (0.88)
EWC-p	94.47 (0.26)	72.02 (2.80)	89.15 (2.95)	65.41 (3.36)	**85.00 (1.35)**
DM-L_1 (best)	95.02 (0.30)	77.24 (1.96)	88.65 (2.97)	65.61 (4.88)	84.33 (1.64)
DM-$E0.5$ (best)	94.98 (0.23)	**80.30** (2.09)	88.90 (1.93)	**65.91** (2.15)	84.39 (0.97)
DM-L_2 (best)	94.27 (0.32)	71.23 (3.43)	89.53 (2.64)	61.24 (2.07)	83.64 (0.60)
DM-L_1 (best, fine)	**95.07** (0.13)	80.04 (1.88)	**92.95** (0.76)	65.77 (4.80)	83.87 (1.77)
DM-$E0.5$ (best, fine)	95.05 (0.19)	**80.30** (2.10)	91.62 (1.06)	65.25 (1.81)	84.53 (0.97)
DM-L_2 (best, fine)	94.35 (0.29)	68.99 (0.85)	89.45 (2.89)	61.24 (2.07)	83.91 (2.04)

Table 1: Mean and std final average validation accuracies (%) across 5 seeds for the best hyperparameters for all methods described in Section 5 and for all the datasets mentioned in Section 5.2. The details of the hyperparameter search are mentioned in Section 5.1. Methods are ranked by their mean validation accuracy across 5 seeds.

Relative Retention in Cases I–IV Given the variance in our results, we find no clear empirical order of retention using DM-L_1. While each case responds to λ in a different way, we observe that at optimal λ, the cases have approximately equal retention across all datasets. We illustrate this in Figure 5 for Sim-EMNIST, for two tasks.

The Effect of Freezing Important Weights Compared to EWC, EWC-p always produces an improved solution, except in the case of Sim-EMNIST, where it remains close to EWC. It outperforms the other methods for Sim-CIFAR100. We also note that for the grayscale datasets the best degree of preservation p is between $20\%-40\%$, while for the realistic image datasets, the best degree of preservation is around $60\% - 80\%$. This is in agreement with the strength of the stability bias λ for the realistic image datasets for EWC, which are also high, meaning that the realistic images require stronger preservation to register improvements.

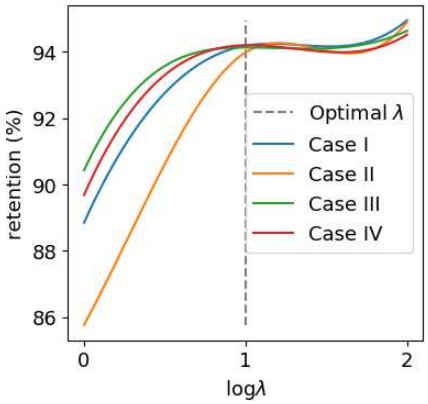

Figure 5: Average performance (smoothed) of Cases I–IV on Sim-EMNIST; 2 tasks with DM-L_1, 5 seeds ($\lambda \in [1, 10^2]$). Retention is defined as the percentage of classifier predictions on task 1 that stay the same after task 2 has been trained.

6 Conclusions

In the context of real world classification systems, catastrophic forgetting may not just cause performance deterioration, but also cause a loss of safety-validated knowledge. Existing (regularization) strategies to mitigate catastrophic forgetting typically minimize the elastic criterion, which can produce non-sparse solutions and require a costly hyperparameter search for the appropriate penalty weight.

In this paper, we re-formulated the continual learning problem to directly minimize an approximate upper bound on the absolute amount of forgetting. We found that directly minimizing this upper bound produces a weaker bias for continual learning, thereby resulting in a higher model variance and stronger preservation of past classification knowledge. This, in turn, can be seen as a stronger retention of safety-validated knowledge. We demonstrated different variants of network preservation corresponding to different system requirements, showed how to achieve the joint stability-plasticity optimum and also proposed a simple modification to EWC to achieve a similar degree of retention. Finally, through experiments on grayscale and colored datasets, we also demonstrated that better preservation of past knowledge can often produce solutions with better performance.

References

Chaudhry, A.; Dokania, P. K.; Ajanthan, T.; and Torr, P. H. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 532–547.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Geman, S.; Bienenstock, E.; and Doursat, R. 1992. Neural networks and the bias/variance dilemma. *Neural computation* 4(1):1–58.

Gigerenzer, G., and Brighton, H. 2009. Homo heuristicus: Why biased minds make better inferences. *Topics in Cognitive Science* 1(1):107–143.

Huszár, F. 2018. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences of the United States of America* 115 11:E2496–E2497.

Kemker, R.; McClure, M.; Abitino, A.; Hayes, T. L.; and Kanan, C. 2018. Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N. C.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2016. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America* 114 13:3521–3526.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2018. Reply to huszár: The elastic weight consolidation penalty is empirically valid. *Proceedings of the National Academy of Sciences* 115(11):E2498–E2498.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436.

Li, Z., and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40(12):2935–2947.

Li, X.; Zhou, Y.; Wu, T.; Socher, R.; and Xiong, C. 2019. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, 3925–3934.

Lopez-Paz, D. 2017. Marc'aurelio ranzato. *Gradient episodic memory for continuum learning. NIPS*.

McCloskey, M. W. 1989. Catastrophic interference in connectionist networks: The sequential learning problem" the psychology.

Nguyen, C. V.; Li, Y.; Bui, T. D.; and Turner, R. E. 2018. Variational continual learning. In *International Conference on Learning Representations*.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*.

Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Dickstein, J. S. 2017. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2847–2854. JMLR. org.

Serra, J.; Suris, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 4555–4564.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, 1139–1147.

Wiewel, F., and Yang, B. 2019. Localizing catastrophic forgetting in neural networks. *arXiv preprint arXiv:1906.02568*.

Yoon, J.; Lee, J.; Yang, E.; and Hwang, S. J. 2018. Lifelong learning with dynamically expandable network. In *International Conference on Learning Representations*. International Conference on Learning Representations.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3987–3995. JMLR. org.

Zou, H., and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)* 67(2):301–320.