Balancing the Stability-Plasticity Dilemma with Online Stability Tuning for Continual Learning

Anton Lee
AI Institute, University of Waikato
Kirikiriroa, New Zealand
al183@students.waikato.ac.nz

Dr. Heitor Murilo Gomes
School of Engineering and Computer Science
Victoria University of Wellington
AI Institute, University of Waikato
Wellington, New Zealand
heitor.gomes@waikato.ac.nz

Dr. Yaqian Zhang
AI Institute, University of Waikato
Kirikiriroa, New Zealand
yaqian.zhang@waikato.ac.nz

***Abstract*—Balancing the stability-plasticity dilemma is an omnipresent challenge in continual learning. The dilemma is that the ability of a model to learn new knowledge (plasticity) comes at the expense of the ability to remember past knowledge (stability) and vice versa. Some continual learning algorithms incorporate a constant hyper-parameter to control this trade-off. We argue that the trade-off should be dynamically tuned rather than kept constant. We propose a method to dynamically balance stability and plasticity in a semi-online and fully online manner. Our method is applicable to a range of underlying regularisation based strategies and class incremental problems. Our experiments in three continual learning benchmarks suggest that dynamic stability tuning methods can improve accuracy and decrease forgetting of continual learning strategies throughout their lifetime.**

***Index Terms*—continual learning, class incremental, stability plasticity dilemma, hyper-parameter tuning, regularization, feedback control**

I. INTRODUCTION

Humans and animals can learn piece after piece of knowledge sequentially. Learning to drive a car does not override our ability to ride a bike. Sequentially learning knowledge is taken for granted in humans, but it does not yet exist to a satisfactory degree in artificial neural networks. Instead, artificial neural networks tend to suffer from "catastrophic forgetting" where previously learnt knowledge is interfered with by new knowledge[1]. That is, an artificial neural network would forget how to ride a bike by learning to drive a car. Continual learning (CL) aims to overcome this limitation in artificial neural networks [2].

Researchers often characterise catastrophic forgetting with the stability-plasticity dilemma[2]–[4]. The dilemma is that plasticity (ability to learn) comes at the expense of stability (ability to remember) and vice-versa. An overly plastic network will catastrophically forget. A fully stable network is entrenched and cannot learn. This paper explores the benefits of varying the stability-plasticity trade-off across time.

Class-incremental is a continual learning problem concerned with learning from a sequence of experiences (a group of classes)[3]. We focus on class incremental because it is a nontrivial and widely used continual learning problem. As in real-life, past experiences cannot be revisited. In the strictest sense, only a single pass through the data is allowed in online continual learning [5]. Relaxing this in the semi-online setting allows repeating only the current experience. This setting is also called offline continual learning [6]. We propose distinct online and semi-online stability tuning algorithms.

The regularisation family of continual learning strategies attempts to overcome catastrophic forgetting by constraining updates of weights through the loss function [2]. Ideally, these constraints would change the learning trajectory such that it finds a minima shared by every class in each experience. These constraints provide stability while ideally avoiding entrenchment. Different formulations of constraints lead to different regularisation strategies [7]–[9]. Compared to other families, regularisation strategies typically have a single hyper-parameter representing the stability plasticity dilemma directly. For this reason, regularisation strategies are preferred for dynamic stability tuning.

Regularisation based strategies typically assume that the trade-off between stability and plasticity is constant throughout training. We however propose methods to tune it dynamically. The core idea echos research into biological neural networks: "The brain is particularly plastic during critical periods of early development in which neural networks acquire their overarching structure driven by sensorimotor experiences. Plasticity becomes less prominent as the biological system stabilizes through a well-specified set of developmental stages" [2]. Our research objective is to **explore ways to dynamically adjust the stability-plasticity trade-off in different regularization based continual learning strategies to solve class-incremental problems.**

Hyper-parameter search spaces are often sizable. The best hyper-parameter depends on the network architecture, contents of an experience, training regime, and even the state of the network. All of these factors impact the choice of a good hyper-parameter. Confounding things further both the network state and data change throughout the model's lifetime. This necessitates an effective way to tune hyper-parameters dynamically.

This paper[1] explores three approaches to varying stability and hence plasticity, each building on observations and limitations of the previous approach. Any strategy with a hyper-parameter that represents stability is a suitable candidate. We focus on three underlying regularisation strategies with suitable hyper-parameters to show this generality: **Elastic Weight Consolidation (EWC)**[8], **Synaptic Intelligence (SI)**[7], and **Learning Without Forgetting (LWF)**[9]. We wrap these underlying strategies with the three proposed tuning approaches. Semi-online Stability Decay (OSD) decays stability when encountering the entrenchment effect by repeating experiences. Building on OSD, we propose Semi-online Stability Tuning (OST) that grows and decays stability to find a "Goldilocks" zone by repeating experiences. Finally, Cybernetic Online Stability Tuning (COST) is fully online, employing all the lessons learned. COST dynamically adapts stability quickly throughout and without the need to reattempt experiences. Consequently, COST is much less computationally expensive than OSD and OST. We go one step further and propose a simple novel continual learning strategy derived from COST, which only tunes the learning rate, called COST-LR.

II. Background

A. Elastic Weight Consolidation and Synaptic Intelligence

EWC [8], SI [7], and some other regularisation strategies follow the approximate shape of Equation 1. The key idea of these methods is to penalize change to a neural network to reduce forgetting.

$$\mathcal{L}^* = \mathcal{L} + \lambda \sum_k I_k(\theta_k - \theta_k^*)^2 \tag{1}$$

$\mathcal{L}$ is garden-variety loss, originating from the current experience. The summation is responsible for penalizing change. k is the index of a parameter. $(\theta_k - \theta_k^*)^2$ penalizes the difference between the current parameter θ_k^* and the old parameter θ_k. This constrains the model to be similar to the previous model. I_k weights the per-parameter penalty. A simple underperforming regularisation strategy is to weigh all parameters equally [8]. However, by penalizing changes based on the parameter's importance to classifying previous experiences, model stability improves without sacrificing much plasticity. EWC and SI calculate I_k to approximate each parameter's importance. EWC uses gradients to approximate importance [8]. SI calculates each parameter's importance from its contribution to decreasing loss [7]. EWC and SI weigh plasticity against stability with the λ hyper-parameter.

EWC suffers from the problem of gradient explosion [11], [12]. An improved variant of EWC is described in [11] where gradients are clipped using Huber regularization based on Huber loss - a piecewise function that starts quadratic and becomes linear. All our experiments use Huber Regularisation with a constant clipping threshold of $\beta = 0.0001$. Other approaches to stabilizing EWC have been investigated, such as [12].

This work attempts to tune the λ parameter that represents the stability plasticity trade-off.

B. Learning Without Forgetting (LWF)

LWF [9] is distinct from EWC and SI in the sense that it uses knowledge distillation [13] instead of an approximation of importance to constrain weights. LWF makes a teacher and student model by copying the student model from the previous experience. The student and the teacher ingest instances from the current task but only the student's parameters are updated. The student learns from the experience and teacher, by mimicking the teacher's output. That is to say, the loss function penalizes the differences between teacher and student output layers using soft-max cross-entropy [13]. Intuitively, the teacher model says: "that horse looks a little like a hamster that I remember", and then the student model attempts to incorporate that information.

Class-incremental trains on a subset of classes during each experience. Consequently, the output layer reserves neurons for future classes. Some implementations of LWF mask unused neurons [10]. We found empirically that this decreases accuracy. We hypothesise the cause is decreased knowledge transfer from teacher to student. The difference in performance grows for a pretrained model because the output layer is meaningful. We use this simpler and empirically less forgetful variant of LWF.

LWF has a λ hyper-parameter representing how much to favour loss from the teacher. Our method tunes LWF's λ since it represents the strategy's stability.

C. Continual Hyper-parameter Framework

De Lange, Aljundi, Masana, *et al.* [6] proposed the Continual Hyper-parameter Framework (CHF). CHF would decay stability in a semi-online manner as a general method for tuning hyper-parameters in class-incremental learning. Inspired by their algorithm, we strive to provide a continuation of their work. We experiment with growing and decaying stability in a semi-online and fully online setting.

One distinction between CHF and our method is we hold the learning rate constant as a control variable. CHF uses a grid search to pick the learning rate to maximize the accuracy of the experience by disregarding other tasks. The authors of [14] showed how decaying learning rate exponentially in conjunction with dropout can outperform more complex strategies. CHF's strategy of picking a learning rate to maximize the current experience accuracy results in increased plasticity and consequently forgetting. CHF relies heavily on the continual learning strategy to provide stability to avoid catastrophic forgetting. Instead, we propose COST-LR to reduce forgetting through dynamic adjustment of the learning rate. Combining COST-LR with our other methods is plausible but is left to future work. Leaving the learning rate as a control variable allows us to focus on the interaction between continual learning strategy, tuning algorithm, and stability hyper-parameters.

[1]Code available at https://github.com/tachyonicClock/online-stability-tune. The avalanche continual learning framework is used to improve reproduciblity and simplify implementation [10].

D. Intransigence

Understanding when a model becomes excessively stable is essential to our approaches. To account for variation in task difficulty, we use a metric known as intransigence. Chaudhry, Dokania, Ajanthan, *et al.* [15] defined the metric of intransigence as the: "inability of an algorithm to learn new tasks". Intransigence measures the difference in accuracy between a naive fine-tuned model and a continual learning strategy. Chaudhry, Dokania, Ajanthan, *et al.* [15] trained the reference on the union of all tasks seen so far. Storing all instances disregards the continual learning setting and is only acceptable when intransigence is solely for analysis and evaluation. However, we wish to use the metric to drive parameter selection making cumulative training inappropriate. Instead, we employ a reference model that naively fine-tunes the weights. The method was a sufficient alternative and did not break the restrictions imposed by the continual learning problem setting.

Our conception of "stability too high" is by no means the only one. Further feature engineering might lead to the construction of a better way to measure "stability too high". For example, [16] supplies several features to a reinforcement learning algorithm which learns how to interpret them to adjust an optimizer's learning rate intelligently.

III. Semi-online Stability Decay (OSD)

Algorithm 1 Semi-Online Stability Decay

Require: $0 < \alpha < 1$ decaying factor, $0 < p < 1$ accuracy drop margin, λ stability hyper-parameter, θ_t previous model, D_{t+1} experience data

$A_{ref} \leftarrow finetune(\theta_t, D_{t+1})$ ▷ Reference accuracy
loop
  $A, \theta_{t+1} \leftarrow clStrategy(\theta_t, D_{t+1}, \lambda)$ ▷ Train model
  if $A < (1-p)A_{ref}$ **then** ▷ A outside drop margin?
    $\lambda \leftarrow \alpha \times \lambda$ ▷ Decay stability
  else
    return θ_{t+1} ▷ To next experience
  end if
end loop

OSD based on CHF [6], described in Algorithm 1, works by decaying an underlying continual learning algorithm's stability. OSD starts by obtaining the accuracy, A_{ref}, of the unconstrained fine-tuned model. A_{ref} represents the accuracy if the model has maximum plasticity. The accuracy metrics used for tuning are calculated before back-propagation, in an interleaved-test-then-train[2] manner. A hold-out test set could also be appropriate, but we favour an interleaved-test-then-train evaluation technique because it more naturally fits the online setting and necessitates less evaluation. To clarify, accuracy is only calculated in this way for tuning purposes. For evaluation metrics it is calculated with a hold-out test set. Utilizing the drop margin p, OSD makes a comparison between the accuracy of the unconstrained model, A_{ref}, and the constrained model A. If it is outside of the acceptable drop margin p we assume that the algorithm has stopped learning and has become entrenched. We compensate by decaying stability and repeating the experience. Stability is decayed appropriately once accuracy is inside the drop margin again. This approach to detecting entrenchment is inherited from CHF and relates to the concept of intransigence that drives COST.

- α - **Decay factor** exponentially decays stability. α controls precision at the cost of speed. We used 0.8 for all our experiments.
- λ - **Stability** is the underlying algorithm's stability at the start. It should be intentionally too high since it will be decayed to a suitable value.
- p - **Accuracy drop margin** governs when an algorithm is considered entrenched.

[2]Also known as prequential evaluation. Patterns are used for testing before they are trained on. Prequential evaluation is inappropriate when instances are trained multiple times because the model may overfit.

IV. Semi-online Stability Tuning (OST)

The major limitation of CHF and OSD is the assumption that stability should only ever decrease. This characteristic becomes evident when plotting LWF and SI stability over time in Figure 6; stability is decayed to a minimum early and then held constant. As a result, we hypothesise that it may be advantageous to decay and grow stability for each experience. This hypothesis led to Semi-Online Stability Tuning (OST).

Algorithm 2 Semi-Online Stability Tuning

Require: $0 < \alpha_{change} < 1$ change factor, $0 < p < 1$ accuracy drop margin, λ stability hyper-parameter, θ_t previous model, D_{t+1} experience data

$\alpha_{decay} \leftarrow 1 - \alpha_{change}$
$\alpha_{grow} \leftarrow (\alpha_{decay})^{-1}$

$A_{ref} \leftarrow finetune(\theta_t, D_{t+1})$ ▷ Reference accuracy
$A, \theta_{tmp} \leftarrow clStrategy(\theta_t, D_{t+1}, \lambda)$

if $A > (1-p)A_{ref}$ **then** ▷ Above drop margin?
  while $A > (1-p)A_{ref}$ **do**
    $\lambda \leftarrow \lambda \times \alpha_{grow}$ ▷ Grow stability
    $\theta_{best} \leftarrow \theta_{tmp}$ ▷ Save previous model
    $A, \theta_{tmp} \leftarrow clStrategy(\theta_t, D_{t+1}, \lambda)$
  end while
  $\lambda \leftarrow \lambda \times \alpha_{decay}$ ▷ Undo the last stability growth
else
  while not $A > (1-p)A_{ref}$ **do**
    $\lambda \leftarrow \lambda \times \alpha_{decay}$ ▷ Decay stability
    $A, \theta_{best} \leftarrow clStrategy(\theta_t, D_{t+1}, \lambda)$
  end while
end if
return θ_{best} ▷ To next experience

OST (Algorithm 2) starts by using α_{change} to define α_{decay} and α_{grow}. These values define how quickly stability

is grown or decayed. OST is similar to OSD as both use the reference model to determine the accuracy if the model was unconstrained with maximum plasticity. The underlying strategy's accuracy determines if λ should grow or decay. If the strategy's accuracy is inside the drop margin, OSD decides to grow stability until it leaves the drop margin (lines 7-13). Otherwise, it determines it should decay stability nearly identically to OSD (lines 14-18). The goal is a model with maximum stability while still having enough plasticity to be within the drop margin.

- α_{change} - **Change factor** defines the speed of growth and decay. We found 0.2 to work well corresponding to $\alpha_{decay} = 0.8$ $\alpha_{grow} = 1.25$. Similar to Decay Factor in OSD, this parameter is mostly unimportant it is used to tune precision vs speed.
- λ - **Stability** is the starting stability. A nonzero guess at the median is sufficient to tune this parameter. A worse guess will result in the algorithm taking longer to run.
- p - The **accuracy drop margin** is similar to OSD's accuracy drop margin. However, it will differ from OSDs since it defines growth and decay.

V. Cybernetic Online Stability Tuning (COST)

A major constraint of OST, OSD, and CHF is that they revisit and reattempt experiences, requiring a "semi-online" setting. Semi-online is undesirable because it discards unsuccessful parameter tuning attempts. Another limitation is that real world experiences are continuous with no clear distinction between them. As such, we cannot rely on the knowledge of experience boundaries, necessitating the ability to learn within an experience rather than between them. Furthermore, the stability plasticity trade-off is likely to change throughout an experience as the model learns. COST addresses these issues by becoming a fully online method that considers the stability plasticity trade-off throughout an experience without the need to reattempt anything.

We propose the application of feedback control to the problem of dynamically adjusting stability. A feedback controller is a component that monitors the outputs of a process and adjusts the process's inputs to achieve desired results [17]. For example, when driving a car, the pressure on the accelerator is adjusted to achieve the desired speed by measuring the current speed. The use of feedback makes these systems robust to uncertainties and perturbations, such as a car going over a hill [17]. This property is desirable for our proposed methods because of the many stochastic and unpredictable factors in the learning process.

Our controller, called COST (Algorithm 3), works by tuning stability based on intransigence. The set point acts as a goal and is similar to OST/D's drop margin (shown in Figure 1). When intransigence exceeds the set point, COST reduces stability proportionally. Dropping below the set-point increases the stability proportionally. Our controller thus maintains a sweet spot. We use a simple proportional controller rather than the more generalized proportional, integral and derivative control (PID). PID was experimented with briefly, but we found it unnecessary and introduced more parameters. Figure 6 shows how COST tunes stability and thus plasticity over time.

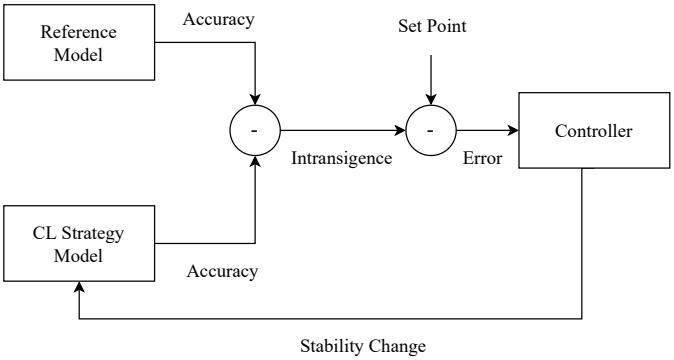

Fig. 1. Cybernetic Online Stability Tune Control Loop

We failed to find a relevant example of hyper-parameter tuning feedback controllers for neural networks in the literature. We speculate that this uniqueness is due to researchers' view of training as a static process rather than a dynamic one [18].

Algorithm 3 Cybernetic Online Stability Tuning

The following is run on each minibatch in an experience.

Require: k_p Proportional gain. sp Set point to bias towards stability.

1: $A_{ref} \leftarrow finetune(...)$
2: $A \leftarrow clStrategy(...)$
3: $I \leftarrow A_{ref} - A$ ▷ Calculate Intransigence
4: $e_t = sp - I$ ▷ Bias towards stability
5: $\lambda \leftarrow \lambda + k_p \times e_t$ ▷ Apply "*force*"
6: **Continue to next mini-batch**

Fig 1 shows diagrammatic representation of this algorithm.

VI. COST-LR

Mirzadeh, Farajtabar, Pascanu, *et al.* [14] showed the importance of training regime on the stability plasticity dilemma. A neural network will be less forgetful if its learning is slowed, which comes at the cost of decreased plasticity. Therefore, we hypothesise that using COST to adjust the learning rate of a fine-tuning network could balance stability and plasticity. The result is a novel continual learning strategy.

COST-LR works by using a small negative k_p to force the learning rate to decrease as intransigence decreases, thus increasing stability.

VII. Experiments

We consider the following two metrics in the experiments:

- **Accuracy** - The accuracy for **only** the classes trained on so far.
- **Forgetting** - The difference between the accuracy when the experience was first learnt and the current accuracy [15].

A. Benchmarks & Models [3]

1) Split MNIST: A simple single-headed feed-forward network (Layer sizes: 28×28-512-10) with drop out $p = 0.5$. Trained on split MNIST, where the dataset is split into 10 experiences of 1 class each. Order and composition of experiences are randomized, increasing variance but demonstrating greater generality.

2) CORE50 NC: Pretrained single-headed ResNet18 trains to classify the CORE50 NC benchmark. CORE50 is a continual learning benchmark proposed by [19] to represent continual learning problems in robotic vision. They provide 3 distinct scenarios, of which we use the class-incremental one, or "NC", scenario. Order and composition follows the predefined random orders in [19].

3) Split CIFAR100: We train a pretrained single-headed ResNet18 model on CIFAR100. Splitting CIFAR100 into 10 experiences of 10 classes yields an exceptionally challenging continual learning benchmark. It is a common benchmark used by [3], [15]. Masana et al [3] notes that single-headed regularisation methods perform poorly on this benchmark. The cause of this difficulty is that class-incremental continual learning is challenging, especially when lots of classes and many experiences are present. Classes similar to each other increase the challenge because they are more likely to interfere with each other. For example, if clouds and planes are in different experiences, the first will likely be catastrophically forgotten when learning the next. Similarly, to split MNIST, the order and composition is randomized.

B. Hyper-parameter Choice

Grid search is used to select hyper-parameters. Usage of grid search is a notable relaxation of the continual learning setting because continual learning is characterised by not revisiting past experiences. However, it is necessary to pick hyper-parameters that best represent the performance of each approach. Table I shows the search grid. Note that we consider the training regime control variables and thus hold the learning rate constant during the whole CL training process, except for COST-LR that explicitly tunes the learning rate.

TABLE I
CHOICE OF GRIDS FOR GRID-SEARCH

	Constant Stability		
EWC	2k, 4k, 8k, 16k, 32k, 64k, 128k, 256k		
SI	1k, 2k, 4k, 8k, 16k, 32k, 64k		
LWF	0.5, 1, 2, 4, 8, 16, 32, 64, 128		
	Semi-Online Stability Decay		
	Initial Stability	**Decay Factor**	**Drop Margin**
EWC	100k[4]		
SI	100k	0.8	0.1, 0.2, 0.3, 0.4
LWF	50		
	Semi-Online Stability Tune		
	Initial Stability	**Change Factor**	**Drop Margin**
EWC	10k		
SI	10k	0.2	0.1, 0.2, 0.3, 0.4
LWF	10		
	Cybernetic Online Stability Tune		
		Proportional Gain	**Set Point**
EWC		5k, 10k	
SI		1k, 2k	0.1, 0.2, 0.3, 0.4
LWF		1, 2	
COST-LR		-1e-3, -1e-4, -1e-5	

TABLE II
AVERAGE FINAL ACCURACY. 10X RUNS.

		Mean Accuracy $\pm$ 1σ (n=10)			
CL Strategy	**Dataset**	**Constant**	**OSD**	**OST**	**COST**
Finetuning	**mnist**	17.5±5.2			**37.7±5.9**
	cifar	**9.1±0.7**			8.7±1.6
	core	21.0±1.4			**29.2±1.8**
EWC	**mnist**	47.2±9.0	48.5±8.5	42.3±7.0	**63.6±4.6**
	cifar	9.6±2.9	10.6±2.9	10.6±2.4	**12.3±2.8**
	core	31.6±2.6	35.1±3.5	34.3±3.2	**36.8±3.5**
LWF	**mnist**	41.4±4.0	28.0±6.2	44.8±8.2	**57.9±4.5**
	cifar	25.1±1.1	18.0±1.0	27.7±2.3	**30.5±1.7**
	core	50.1±1.6	46.9±2.0	52.0±1.6	**53.0±2.2**
SI	**mnist**	46.1±2.9	23.9±5.1	37.8±9.1	**54.3±4.7**
	cifar	10.3±2.7	11.3±3.0	**12.7±3.2**	8.4±4.1
	core	32.7±0.9	32.9±2.0	34.2±2.3	**36.0±2.3**

VIII. RESULTS

A. Performance Analysis

Figure 3 and Table II show the mean final accuracy and standard deviation of 10 runs. On the MNIST benchmark, COST is substantially better than other methods. Compared to employing a constant stability, dynamic stability with COST achieves large performance gains over all three baseline CL algorithms (EWC:+16.6%, LWF:+16.5%, SI:+8.2%). In contrast, dynamic stability with OSD and OST fail to improve the performance over the baselines with fixed stability and even hurt the performance in some cases. In CIFAR and CORE, COST also outperforms most other methods. OST is generally better than holding stability constant, while OSD is usually worse.

Besides analyzing the final accuracy at the end of training, we are also interested in the the accuracy changes during the whole continual learning process. In an online setting, the notion of *"accuracy at the end"* is barely meaningful because online algorithms inhabit a hypothetical never-ending stream of experience. Figure 2 shows the accuracy over each experience during the whole continual learning process. Every approach starts with high accuracy before dropping as increasingly more classes are included in the test set. An ideal algorithm would have a nearly horizontal line (no forgetting), whereas the worst case is the fine-tuning red line (catastrophic forgetting). OST and COST tend to be closer to this ideal (most notably in MNIST).

COST performs less than optimally if improperly parameterized. As an example, take COST SI on CIFAR100 benchmark. In this case, the final accuracy has a high variance and a mean lower than naively fine-tuning. Reducing proportional gain would reduce variance because COST will act less aggressively. This improper parameterization is likely a result of running a single iteration over the grid during grid-search.

[3]Batch Size: 64. Epochs: 1. Learning Rate [0.001 (MNIST, CORE), 0.005 (CIFAR100)]. **Single-headed evaluation** following advice from [15].

[4]MNIST uses 500000 in order to be "too stable". This is caused by the MNIST experiment using a model that can be more aggressively constrained.

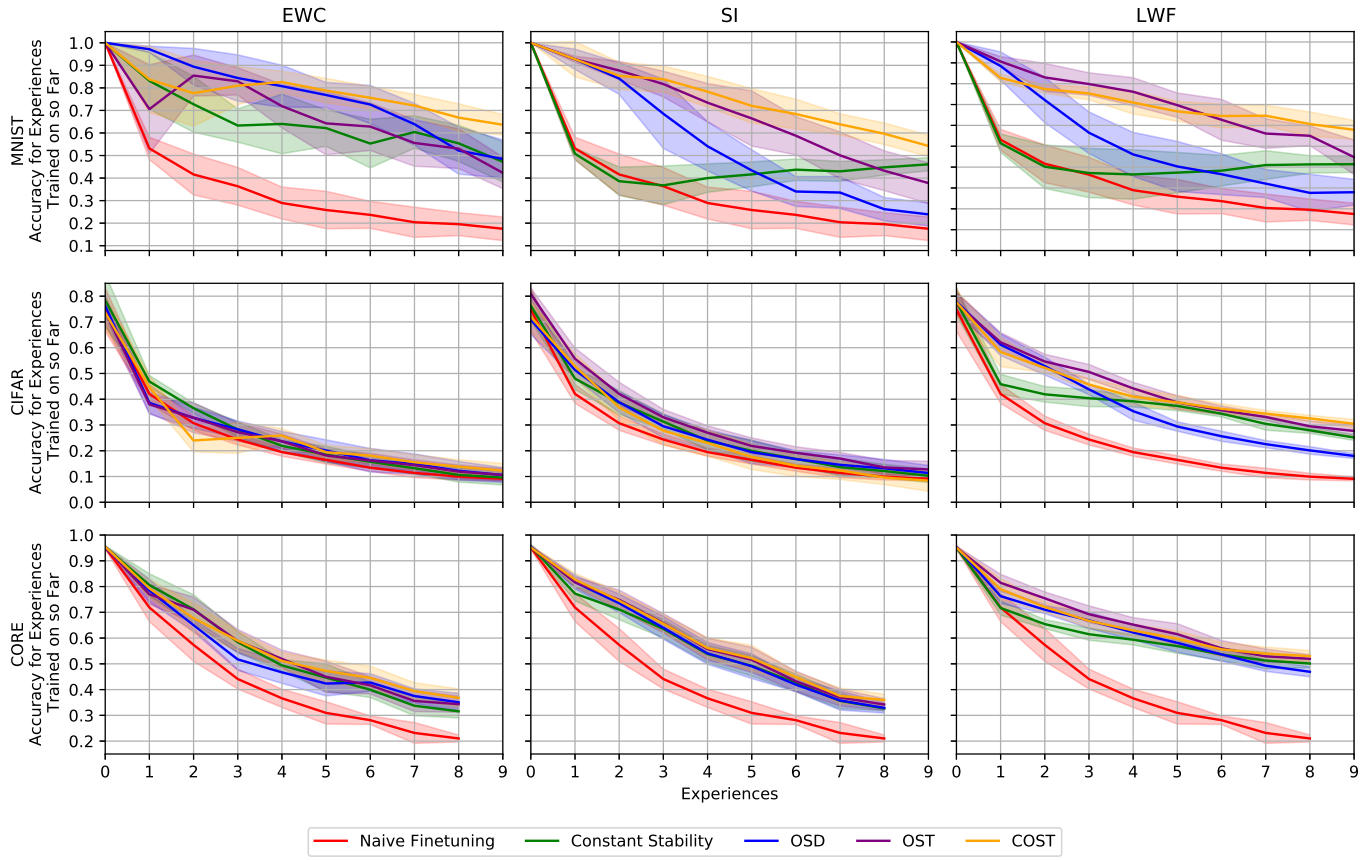

Fig. 2. **Stability Tuning Variant Comparisons over Experiences** - Shows different tuning approaches' accuracy on only the experiences they have seen with 1 standard deviation shaded from 10 runs. Red shows fine-tuning suffering from catastrophic forgetting when naively used for continual learning. All continual learning methods should beat naive fine-tuning. The green line is the underlying strategy when stability is constant. Dynamic tuning needs to outperform the green line to demonstrate a benefit. COST SI on MNIST is an ideal result where our methodology has low forgetting and is consistently better throughout the model's lifetime. CIFAR100 and CORE50 are challenging benchmarks for class incremental single-headed neural networks making the differences in approaches visually unclear. Despite this, dynamic tuning has modest improvements throughout the model's lifetime and decreased forgetting (See Figure 4 and 5), improving final accuracy in some cases (See Figure 3)

Grid search creates a bias towards high accuracy at the end, at the expense of negatively affecting earlier experiences. Constant stability methods MNIST SI and CIFAR LWF illustrate this (Figure 2). Their accuracy follows an "L" shape. For example, MNIST SI almost completely forgets the first experience dropping to around 0.5. Finally, accuracy flattens out to make them perform well by the end of the experience stream. Our dynamic methods do not suffer from this effect, where COST and OST can achieve better performance throughout learning, a desirable feature in online environments.

To further understand the behavior of continual learning algorithms, we analyze the "forgetting" behavior in Figure 4. Interestingly, although some models have similar final accuracy in CIFAR100 dataset as shown in Figure 5, the amounts of forgetting of different algorithms are actually quite different. As shown in Figure 4, COST usually has a relatively low amount of forgetting. This finding is also reflected visually in the EWC confusion matrix (Figure 5). COST EWC has a similar accuracy level to constant EWC but lower level of forgetting. This suggests that COST is more focused on "stability" rather than plasticity as a consequence of the set point and proportional gain on this data set.

B. Dynamic Stability

To further investigate the advantage of COST over OSD and OST, we visualize the dynamic changes of stability parameters for different tuning methods in Figure 6. Decay-based algorithms (OSD), except for EWC perform worse than their static counterparts. OSD decays stability whenever it starts suffering from the "entrenchment effect". EWC is more prone to the effect since it sums penalties calculated with all past experience's parameters. If λ is not decreased the stability will increase over time.[5] LWF and SI do not have the same naturally increasing stability, instead the stability is often set low early on and then kept constant, see Figure 6. Subsequently, OSD SI and OSD LWF perform worse than holding stability constant.

[5]Online EWC [20] may mitigate this by storing one importance matrix which is updated and decayed.

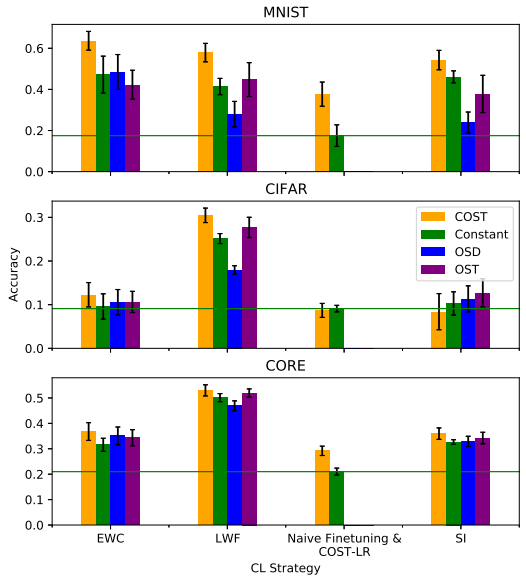

Fig. 3. **Average final accuracy** - Mean final accuracy and 1 standard deviation of 10 runs. All approaches should exceed the green line that indicates accuracy from naively fine-tuning. LWF with COST is the best performing model overall. COST-LR is a notable improvement over simple naive fine-tuning. Figure 4 and Figure 5 provide a supplemental picture of the effects of dynamic tuning.

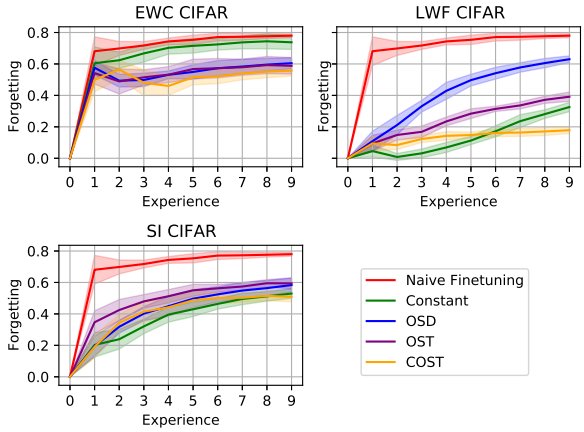

Fig. 4. **CIFAR100 Stability Tuning Variant Forgetting** - Shows the mean forgetting of 10 runs, with one standard deviation shaded.

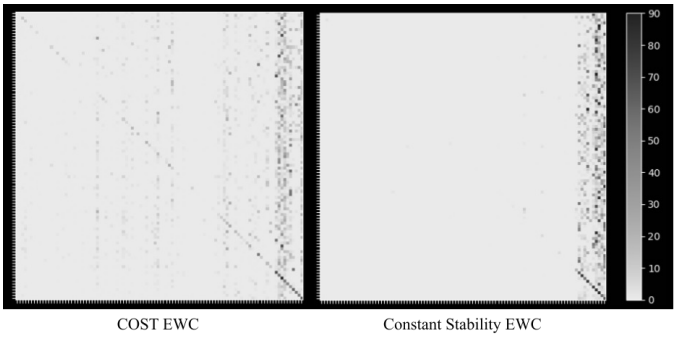

Fig. 5. **CIFAR100 EWC Confusion Matrix (CM)** - Earlier classes are on the left and recent classes are on the right in two figures. The diagonal represents correct classifications. The further back in time a class is learnt, the more likely it is catastrophically forgotten. Despite balancing the stability plasticity dilemma, we have not overcome it. As a result, both of the above CM have similar overall accuracy despite different amounts of forgetting.

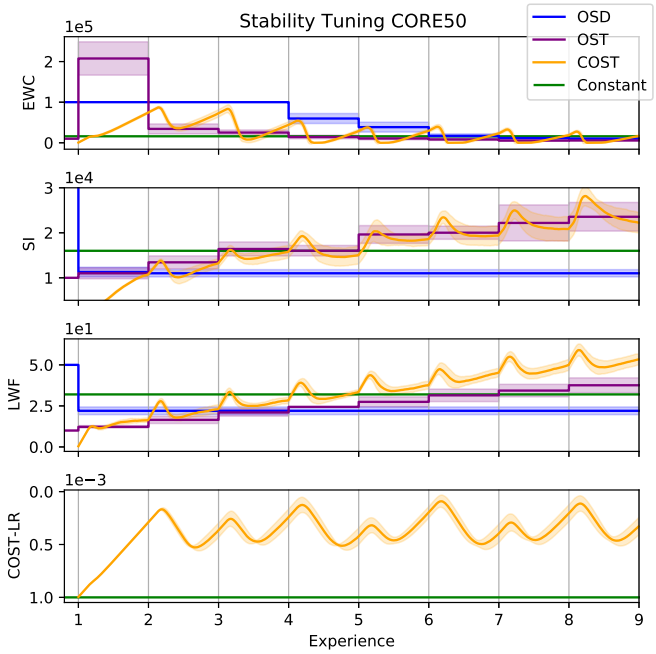

Fig. 6. **Stability Change Over Experiences** - Shows the mean stability parameter over time using various tuning approaches and underlying algorithms on 10 runs. The shaded region shows one standard deviation. Vertical lines represent class boundaries. COST-LR has it's y-axis inverted since learning rate is inversely related to stability. Stability Tuning on CORE50 is highly similar to the other datasets. In general we see stability increase overtime and decrease at the start of each experiences.

OST EWC suffers from too much regularisation early on. In Figure 6, OST EWC's stability reaches a maximum early. This must immediately be decayed in the next experience when the model becomes entrenched. Figure 2 OST EWC on MNIST shows the consequences most strongly where the accuracy drops off massively at experience 1. OST manages to compensate but some damage is already done.

One noticeable quirk of COST is the blip caused by the start of a new experience (shown in Figure 6). The blip is when stability steeply increases before steeply decreasing. The blip is caused by the reference and strategy algorithm performing similarly poorly at the start of an experience causing an increase in stability. We attempted to mitigate the blip by manually reducing stability at the start of each experience, but we found this to have little effect as COST adapts reasonably quickly.

C. Running Time

The different tuning approaches result in varying runtimes. Semi-online methods are the slowest since they can repeat experiences multiple times. OSD is 142% (mean percentage difference) slower than constant stability. OST is 248% slower because it tunes stability in two directions. The same semi-online strategy on the same data will have variations in runtime because it is unknown how many times stability should be grown or decayed until runtime. Online methods will always take the same amount of time, equivalent to a single pass through the data. As a result, COST is only 29% slower. COST's reference network is responsible for this speed decrease. COST outperforms semi-online tuning in both speed and accuracy.

D. COST-LR

Apart from tuning the stability parameter, we perform a preliminary study to investigate the effectiveness of COST in tuning other hyper-parameters. As shown in Table II and Figure 3, applying COST to tune the learning rate (COST-LR) using fine-tuning as the baseline method significantly outperforms the constant learning rate. In fact, fine-tuning with COST-LR performs surprisingly well and is even comparable to some regularization-based approaches (e.g. LWF, SI) that are specially designed for CL setting. Tuning the learning rate is tangential to the other strategies since it is plausible that algorithms can jointly tune the learning rate and other stability parameters, however, this is left as future work.

IX. Conclusion

To our knowledge, we are the first to propose a fully online tuning method for continual learning. Our methods are flexible enough to be applied to a range of regularisation strategies. Future work could experiment with other regularisation, replay, parameter isolation, and even un-invented strategies. In our preliminary work with COST-LR, we showed that other hyper-parameters could also benefit from tuning. An interesting future research direction is to investigate the joint tuning of multiple hyper-parameters across time.

The problem of hyper-parameter selection in continual learning persists. Our use of feedback gives some resilience to bad choices of hyper-parameters, but poor selection remains problematic. This limitation led us to relax the constraints of online continual learning through the use of grid search. Work is still needed towards a plug-and-play online tuning algorithm.

Tuning stability with feedback control can improve accuracy and decrease forgetting throughout a model's lifetime.

References

[1] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, p. 8, 1999.

[2] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[3] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," Oct. 2020.

[4] M. Mermillod, A. Bugaiska, and P. BONIN, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in Psychology*, vol. 4, p. 504, 2013.

[5] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars, "Online Continual Learning with Maximally Interfered Retrieval," *arXiv:1908.04742 [cs, stat]*, Oct. 2019.

[6] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[7] F. Zenke, B. Poole, and S. Ganguli, "Continual Learning Through Synaptic Intelligence," in *International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 3987–3995.

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[9] Z. Li and D. Hoiem, "Learning without Forgetting," *arXiv:1606.09282 [cs, stat]*, Feb. 2017.

[10] V. Lomonaco, L. Pellegrini, A. Cossu, A. Carta, G. Graffieti, T. L. Hayes, M. De Lange, M. Masana, J. Pomponi, G. M. van de Ven, M. Mundt, Q. She, K. Cooper, J. Forest, E. Belouadah, S. Calderara, G. I. Parisi, F. Cuzzolin, A. S. Tolias, S. Scardapane, L. Antiga, S. Ahmad, A. Popescu, C. Kanan, J. van de Weijer, T. Tuytelaars, D. Bacciu, and D. Maltoni, "Avalanche: An End-to-End Library for Continual Learning," 2021, pp. 3600–3610.

[11] L. Liu, Z. Kuang, Y. Chen, J.-H. Xue, W. Yang, and W. Zhang, "Incdet: In defense of elastic weight consolidation for incremental object detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2306–2319, Jun. 2021.

[12] A. Kutalev and A. Lapina, "Stabilizing elastic weight consolidation method in practical ml tasks and using weight importances for neural network pruning," *arXiv:2109.10021 [cs]*, Sep. 2021.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531 [cs, stat]*, Mar. 2015.

[14] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh, "Understanding the role of training regimes in continual learning," Jun. 2020.

[15] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., ser. Lecture Notes in Computer Science. Springer International Publishing, 2018, vol. 11215, pp. 556–572.

[16] C. Daniel, J. Taylor, and S. Nowozin, "Learning Step Size Controllers for Robust Neural Network Training," Feb. 2016.

[17] Å. Karl and R. Murray, *Feedback systems: An introduction for scientists and Engineers*. Princeton University Press, 2021.

[18] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama, "Machine learning for streaming data: State of the art, challenges, and opportunities," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, pp. 6–22, 2019.

[19] V. Lomonaco and D. Maltoni, "CORe50: A New Dataset and Benchmark for Continuous Object Recognition," in *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, Oct. 2017, pp. 17–26.

[20] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & Compress: A scalable framework for continual learning," *arXiv:1805.06370 [cs, stat]*, Jul. 2018.