Bachelor's Thesis

Survey on Regularization Methods in Continual Learning

Department of Statistics

Ludwig-Maximilians-Universität München

Jörg Schantz

Munich, March 20th, 2025

Submitted in partial fulfillment of the requirements for the degree of B. Sc.

Supervised by Dr. Julian Rodemann

Abstract

Continual learning (CL), also referred to as lifelong or incremental learning, aims to enable AI models to learn from sequentially arriving data without forgetting previously acquired knowledge. A fundamental challenge in CL is catastrophic forgetting, where new tasks interfere with past learning. Regularization-based methods provide a promising approach to mitigating this issue by imposing constraints on model updates. This thesis presents a survey of regularization techniques in CL, distinguishing between parameter-space and function-space regularization. Parameter-space methods, such as Elastic Weight Consolidation (EWC) and Synaptic Intelligence (SI), restrict parameter changes based on importance measures like the Fisher Information Matrix. Function-space approaches, including Learning without Forgetting (LwF) and Functional Regularization for Continual Learning (FRCL), aim to maintain consistency in model outputs. Additionally, sparse penalties and dynamically expandable architectures are explored as means to balance stability and plasticity. By analyzing the strengths and limitations of these methods, this thesis contributes to a deeper understanding of regularization in CL and its role in extracting importance measures in artificial neural networks (ANNs).

Contents

List of Figures

1 Introduction

The internet has opened many doors for humanity one of those being the apparent access to infinite information. Everyday people around the world generate over 400 million terabyte of new data [32]. This well of information has recently been adopted to train prominent artificial intelligence agents, like GPT-3 [15]. Training these models even once is very expensive [16] which makes it hard for developers to keep up with the seemingly unending flow of new data. Aside from this financial factor, the physical limitations of storage and computation time for all of these data files pose another problem for AI developers and demonstrates the need for selective model editing without retraining. On a smaller scale, AI often needs to adapt to more specialized use cases [42]. Pre-trained AI models, for example in smart watches, need to be able to adapt to their owner's habits in order to be fully functional. Continual learning (CL), often referred to as lifelong learning or incremental learning, aims to solve ease these limitations for AI by dividing all available and future data into distinct sets, which are then processed sequentially [42].
Training on distinct data sets, puts developers in front of a new problem: how can they make sure AI does not forget previously learned information?
AI is built on artificial neural networks (ANN), a machine learning program, that makes decisions by mimicing a human brain [21]. Although they are inspired by humans they currently lack the ability to look inward and reflect on themselves. Ergo, people need to find ways to determine what is important information for AI in order to enable efficient ways of "remembering". Understanding how ANNs make decisions is also important to build trust and confidence in AI [39].
Their use cases go far beyond suggesting a new song for your playlist or correcting typos in a rushed text message. We have started to implement AI to drive cars or help with medical examinations. These tasks often come with variations, like driving in different weather conditions or recognizing more than one kind of tumor. It would be desirable to have a single AI that can learn these task when required and maybe even use previously

learned information to help with the new training process. Like children first learn the alphabet and then use this knowledge to read and write texts, AI should leverage prior tasks to solve new ones.

Throughout this thesis I want to explore the possibilities and limitations of regularization as a CL method with some selected examples, as well as how advances in this field can contribute to a better understanding of ANNs. There have been thorough surveys on CL as a whole [5, 42, 43], which provide a more complete overview of this field. This thesis separates itself from them by diving deeper into individual methodologies in order to demonstrate how regularization in CL has helped to extract importance measures in ANNs and which tasks can even be learned in such a setting.

2 Neural Networks (NN)

Although continual learning is a general modeling concept, applicable in statistical inference as well as pattern driven prediction algorithms, it is mostly used in a machine learning context, more specifically in artificial neural networks (ANN). They are algorithms based on the functionality of a human brain and often designed for scenarios where data is seen in real-time, e.g. stock market predictions or power control systems. Du and Swamy [8] and Fahrmeir et al. [12] provide an overview to this topic, which builds the foundation of this thesis.

The simplest form of an ANN is a single linear classifier, called one-neuron perceptron, that devides a vector x into two classes using a so-called activation function $h(\cdot)$. The neuron's input is given by

$$\sum_{i=1}^{n} w_i x_i + c = w^\top x + c \tag{1}$$

where n is the number of observations, w a weight vector assigned to x and c the decision threshold. The two class regions are separated by the hyperplane

$$w^\top x + c = 0 \tag{2}$$

. Using multiple neurons with the same activation function creates a one-layer perceptron and enables classification for more than two classes with the input

$$\sum_{k=1}^{m}\sum_{i=1}^{n} w_{k,i}x_i + c = (w_1^\top x + c, ..., w_m^\top x + c)^\top = W^\top x + c \tag{3}$$

, where W is the $n \times m$ weight matrix and m the number of classes. Given h as the logistic function, a one-layer perceptron is equal to a multinomial logit model. Composing l layers of neurons, Feed Forward NN (FFNN), allows for a more abstract representation of the data and finer class boundaries. Commonly, these layers are referred to as hidden layers, since only the input and output are observable. The unknown weight matrices $W_1, ..., W_l$ and the decision threshold c are obtained by minimizing the empirical Risk of the network

$$\mathcal{R}(f) = \sum_{i=1}^{n} L(f(x_i, \theta), y_i) \tag{4}$$

, where θ are the unknown parameters and $L(\cdot)$ a loss function, which measures the discrepancy between the predicted values $\hat{y}_i = f(x_i, \theta)$ and the true values y_i. Risk minimization usually makes use of gradient-decent methods. They are numerical algorithms that converge to a local minimum in ε-neighborhood of the solution of parameters.

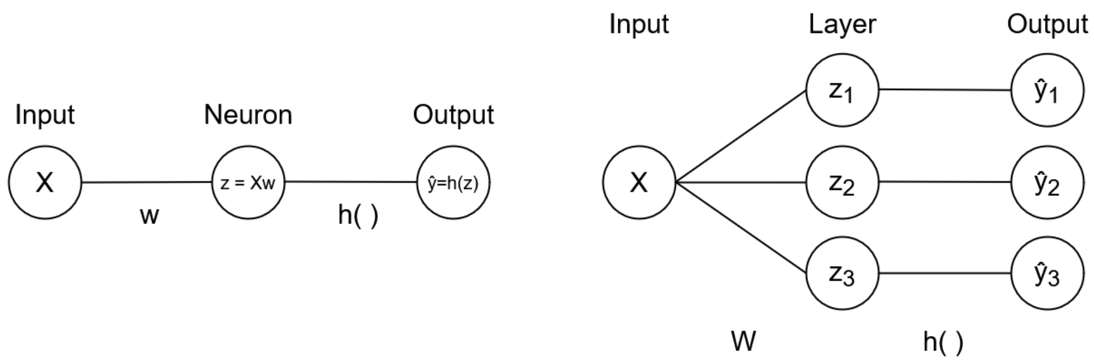

Figure 1: A one-neuron perceptron (left) and a one-layer perceptron with 3 neurons (right).

In the following, modeling of a probability distribution over a data sample and learning how to predict the next output for an unseen data point will both be referred to as a *task*.

Both ideas are closely related, because inferred statistical models can be used to deduce long term behavior or to make a best guess about unobserved data.

3 Framework

Throughout literature continual learning, in a statistical sense, means modeling a joint probability distribution, which is allowed to be infinitely expanded [43]. T samples $D_t, t \in 1, ..., T$ from different distributions $\mathbb{P}_t$ are processed sequentially. A single sample has the form $D_t = (X^{(t)}, y^{(t)}) \in (\mathbb{R}^{n_t \times d_t}, \mathbb{R}^{n_t})$ with $X^{(t)}$ being the covariate matrix and $y^{(t)}$ the dependent variable. The $y^{(t)}|X^{(t)}$ are assumed to be conditionally independent but not necessarily identically distributed [43]. The joint distribution $\mathbb{P}^{(t)}$ of $Y^{(t)}|\mathcal{X}^{(t)} = \{y^{(i)}|X^{(i)}\}_{i=1}^{t}$ a model should have learned after seeing the t-th samples is then

$$\mathbb{P}^{(t)}(Y^{(t)}|\mathcal{X}^{(t)}) = \prod_{i=1}^{t} \mathbb{P}_i(y^{(i)}|X^{(i)}) \tag{5}$$

. Each tuple $(D_t, \mathbb{P}_t)$ may correspond to a distinct regression or classification task that is to be learned. The goal is to train a single model which is able to perform well on all tasks, although training happens sequentially and cannot necessarily revisit prior data samples. This systematic constraint implies that CL algorithms risk *forgetting* previous tasks which will be explained in subsection 3.2. For ease of notation, columns of a covariate matrix will be referred to as features and the dependent variable as label(s).

3.1 Scenarios

In regards to the label space Bidaki et al. [5] and Wang et al. [43] differentiate between eight CL scenarios:

Task-incremental learning (TIL), *Class-incremental learning* (CIL), *Task-Free continual learning* (TFCL) and *Online contiunal learning* (OCL) algorithms all aim to learn a distinct set of tasks, while providing a task identity, if not stated otherwise [5, 43].

$$y^{(i)} \cap y^{(i+1)} = \emptyset \tag{6}$$

TIL allows task individual output layers or the training of separate models for each task. The challenge then is less about forgetting but finding a healthy balance between prediction accuracy and model complexity [41].
CIL restricts this approach by only training one model, which is introduced stepwise to different classification tasks. CIL only provides task identity during training [41]. For example, with samples t an agent learns to classify hats or gloves and with sample $t + 1$ shirts or pants. When testing, it is then also required to classify hats or shirts.
TFCL does not provide any task identity to the model and only focuses on labels [2].
OCL limits its sample sizes to one and focuses on real-time training [5, 43].
Domain-incremental learning (DIL) algorithms seek to learn multiple tasks that share the same label space [5]. For example, first learning to drive during sunny weather and later while it is rainy. One could view this as a version of task-incremental learning, where task identity is secondary as all tasks have the same data labels. Thus, design based strategies to inhibit forgetting are not possible [41].

$$y^{(i)} = y^{(i+1)} \nRightarrow \mathbb{P}(y^{(i)}) = \mathbb{P}(y^{(i+1)}) \tag{7}$$

Instance-incremental learning (IIL) algorithms learn one common task for all training samples [5, 43]. This is a special case of DIL where a model learns the distribution of one "domain" while only ever accessing snippets of the total available data. For example, each sample contains new real-world photographs of cats to classify. Assuming OCL only learns one task, OCL is a special case of IIL where each data point is seen in sequence.

$$y^{(i)} = y^{(i+1)}, \mathbb{P}(y^{(i+1)}) = \mathbb{P}(y^{(i+1)}) \Rightarrow \mathbb{P}(Y^{(t)}) = \mathbb{P}(y^{(i)}) \tag{8}$$

Blurred Boundary continual learning (BBCL), in contrast to all others so far, allows partially overlapping label spaces [5, 43].
Continual Pre-training (CPT) aims to improve knowledge transfer with sequentially arriving pre-training data [5, 43].
As presented, CL can occur in various environments. The connecting problem is memorizing an old task without blocking out new ones and vice versa. In particular, after each training step, the neural network needs to update its weight parameters in a way that the new weights can describe the relationship between all $(y^{(i)}, X^{(i)})$ pairs.

3.2 Stability-Plasticity Trade-off

The challenge of continual learning is to strike a balance between stability and plasticity. Models should retain knowledge of past tasks: stability, while being flexible enough to incorporate information from new data: plasticity. The sequential training nature of CL changes the weights acquired form learning task A to accommodate for a new task B. This abrupt loss of information is called catastrophic forgetting [14, 33, 34, 36]. A naive approach to solving this dilemma would be storing and replaying data to the network with each training step. This is impractical because the amount of data needed to be stored is proportional to the number of tasks learned.
Evron et al. [11] define forgetting as

$$F(t) = 1/t \sum_{i=1}^{t} \|X^{(i)} w^{(t)} - y^{(i)}\|^2 \tag{9}$$

They have analyzed catastrophic forgetting in linear regression under the assumptions that values of X are bounded by 1, tasks are jointly realizable with a bounded (by 1) norm, and there are more parameters than observations in each sample. Realizability assumes the existence of true model weights s.t. $y^{(t)} = X^{(t)} w^*$ [38]. This enables them to focus only on minimizing the distance between new and old model weights. In their work they find an upper bound for forgetting

$$\sup F(t) = \sup 1/t \sum_{i=1}^{t} \|(I - Q_i)Q_t...Q_1\|^2 \tag{10}$$

where Q_i are the projections onto the solution spaces of $w^{(i)}$,
i.e. $Q_i := I - X^{(i)\top}(X^{(i)}X^{(i)\top})^{-1}X^{(i)}$.
So far many methods of minimizing catastrophic forgetting have been developed. Their core ideas can be summarized to *Replay* methods [3, 7, 37], *Optimization* methods [22, 29, 35], *Architecual* methods [10, 13, 31] and *Regularization* methods, which will be discussed in section 5.
In order to evaluate how well these methods work Lopez-Paz and Ranzato [29] suggest three performance metrics which measure how well a model can use learned tasks to solve new ones and how much knowledge of old tasks needs to be "unlearend" / forgotten to learn new ones.

4 Metrics

In the following each sample $D_t = (X^{(t)}, y^{(t)})$ is divided into a training split $D_t^{(train)} = (X^{(t)}_{(train)}, y^{(t)}_{(train)})$ and a testing split $D_t^{(test)} = (X^{(t)}_{(test)}, y^{(t)}_{(test)})$. The chosen splitting method is arbitrary. The training process for each sample will be conducted with $D_t^{(train)}$ and evaluation with $D_t^{(test)}$.
Lopez-Paz and Ranzato [29] come up with three different measures for model predictability, stability and plasticity. I will focus on their dynamic forms given by Díaz-Rodríguez et al. [9], because they have been adapted for in training use, i.e. they represent a model's current state after the t-th training step.
Accuracy **A** represents a models performance i.e. how well the predictions $\hat{y}^{(t)}_{(test)}$ align with the true values of $y^{(t)}_{(test)}$ for a metric μ . When $A_{i,k} = \mu(\hat{y}^{(k)}_{(test)}, y^{(k)}_{(test)}) \in [0, 1]$ is the

accuracy measured on the k-th test split after the i-th training step, then

$$\mathbf{A} = \frac{2}{t(t-1)} \sum_{i \geq k}^{t} A_{i,k} \tag{11}$$

is the average accuracy after the t-th training step over all test splits $D_k^{(test)}, k <= t$.
Backward Transfer **BWT** evaluates a models stability. The metric quantifies the influence that learning sample $D_{t+1}^{(train)}$ has on the performance over test sample $D_t^{(test)}$ [29]. Given the above mentioned individual *Accuracy* scores $A_{i,k}$

$$\mathbf{BWT} = \frac{2}{t(t-1)} \sum_{i=2}^{t} \sum_{k=1}^{i-1} (A_{i,k} - A_{k,k}) \tag{12}$$

is the average backward transfer after the t-th training step. Note that **BWT** can be negative. This property captures (catastrophic) forgetting [43].
Forward Transfer **FWT** is a metric for model plasticity. Complementary to BWT, *Forward Transfer* measures how previous training steps influence the current one. Again, the individual *Accuracy* scores are the basis for this evaluation metric. The average influence of old training steps on the model performance after the t-th step is:

$$\mathbf{FWT} = \frac{2}{t(t-1)} \sum_{i<k}^{t} A_{i,k} \tag{13}$$

.

One of the motivations for CL is limited memory space for data which is why storing a test sample for each task is limiting in the long run when the number of tasks goes to infinity. A metric that directly measures the relationship between stability and plasticity on the model is presented by Mirzadeh et al.[35]. They use the maximum eigenvalue of the loss' Hessian λ^{max} to describe the width of their approximation of the loss' minimum. They hypothesize that the *wideness* of this minimum correlates with the forgetting rate of the respective model.

Given $W^{(t)*}$ and $W^{(t+1)*}$, the optimal parameters after learning the t-th and $(t+1)$-th task and $L_t(\cdot)$ and $L_{t+1}(\cdot)$ the corresponding loss function, Mirzadeh et al. formulate the upper bound

$$F_t = L_t(W^{(t+1)*}) - L_t(W^{(t)*}) \approx \frac{1}{2}\Delta W^\top \nabla^2 L_t(W^{(t)*})\Delta W \leq \frac{1}{2}\lambda_t^{max}\|\Delta W\|^2 \qquad (14)$$

for the forgetting F_t of the t-th task. They approximate $L_t(W^{(t+1)*})$ around $W^{(t)*}$ with a second order Taylor approximation, where ∇^2 is the Hessian for L_t and ΔW the difference between $W^{(t+1)*}$ and $W^{(t)*}$. They argue that the loss can be approximated this way, because of its almost convex path around the minimum, for models that have more observations per sample than parameters.

Furthermore, ΔW is dependent on the training process of the $(t+1)$-th task, which depends on the random sample it is trained on, so one can view the differences in parameters as a random vector that follows some distribution parameterized by the eigenvalues of $\nabla^2 L_t(W^{(t)*})$ [35]. Controlling the distance between old and new weights, as well as λ^{max} of subsequent tasks seems to be the key to mitigating forgetting [35].

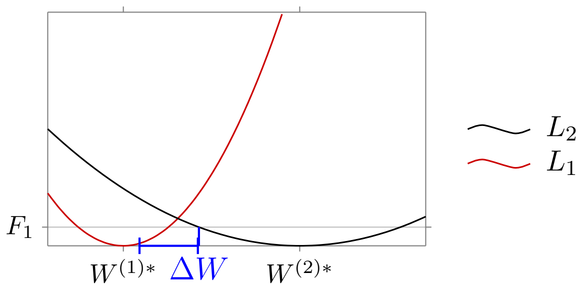

Figure 2: Depending on the curvature of L_1 and L_2, different Δw result in different forgetting rates.

5 Regularization

As mentioned in subsection 3.2, one way to address the stability-plasticity problem is the use of regularization. This approach adds a penalty term to the loss function of a model. Usually this penalty term depends on the model parameters. Later we will also see some methods that directly penalize the output of a model. I will begin by categorizing the regularization methods that I have found through out my research and present some selected examples. After this overview of current possibilities in regularization techniques, I will present approaches at unifying and generalizing them.

In [11, 25] the authors introduce a rudimentary approach to regularization in CL, called *ordinary conitunal learning*. It is used as a worst case comparison for their contribution to this field and is the basis for the upper bound on forgetting in subsection 3.2.

Despite its limitations, I believe that it is beneficial to start this section with ordinary CL. In the beginning I will reduce the complexity of continual learning to two subsequent linear regression problems, which simplifies the entry into the field.

Ordinary continual learning [11, 25, 49] assumes a CL problem with $T = 2$ linear regression models $y^{(t)} = X^{(t)}w^* + \epsilon_t, \epsilon_t \sim N(0, \sigma^2), t \in \{1, 2\}$, the task corresponding samples $D_t = (X^{(t)}, y^{(t)}) \in (\mathbb{R}^{n_t \times d}, \mathbb{R}^{n_t})$ and a covariance matrix of $\hat{w}^{(t)}$, $\Sigma_t = 1/n^{(t)} X^{(t)\top} X^{(t)}$. To estimate the first task parameters $\hat{w}^{(1)}$, ordinary continual learning algorithm performs an ordinary least square minimization over the first sample set D_1, i.e. $\hat{w}^{(1)} = (X^{(1)\top} X^{(1)})^{-1} X^{(1)\top} y^{(1)}$. In the second training sequence, ordinary continual learning fits $w^{(2)}$ to the residuals of task one with respect to $X^{(2)}$. The new parameters $\hat{w}^{(2)}$ are then:

$$\hat{w}^{(2)} = \hat{w}^{(1)} + (X^{(2)\top} X^{(2)})^{-1} X^{(2)\top} (y^{(2)} - X^{(2)} \hat{w}^{(1)}) \tag{15}$$

In their analysis of ordinary continual learning,Zhao et al. [49] show that it suffers from constant forgetting and provide the lower bound for its forgetting:

$$F_t = \mathbb{E}(\|\hat{w}^{(t)} - w^*\|_2^2) \geq \frac{\sigma^2}{\lambda_{max}^{(t)}} \tag{16}$$

where $\lambda_{max}^{(t)}$ maximum eigenvalue of Σ_t.

Since the inverse of the negative Hessian of the neg-log likelihood of $\hat{w}^{(t)}$ is an estimator for Σ_t, ordinary CL a good example for the "worst case" of CL. Neither the maximum eigenvalue of nor the distance between old and new parameters are controlled, thus forgetting happens constantly. Ergo methodologies that control at least one of them are required.

5.1 Regularization of Parameter Space

5.1.1 Quadratic Penalties

Expanding on the naive *ordinary continual learning* approach, Li et al. [25] suggest an adaptation of the Ridge penalty for continual learning, dubbed *continual ridge regression* (CRR) [25, 49]. For estimating $w^{(2)}$, they introduce a Ridge-like penalty term

$$\text{pen}_R(w) = \lambda \|w - \hat{w}^{(1)}\|_2^2 \tag{17}$$

which centers the new weights around the previously estimated $\hat{w}^{(1)}$ instead of 0. Instead of using penalized least squares, the authors decide to perform a penalized mean squared error regression.

$$\begin{aligned} \hat{w}^{(2)} &= \arg\min_w \frac{1}{n}\|y^{(2)} - X^{(2)}w\|_2^2 + \text{pen}(w) \\ &= (X^{(2)\top}X^{(2)} + \lambda n I)^{-1}(X^{(2)\top}y^{(2)} + \lambda n \hat{w}^{(1)}) \end{aligned} \tag{18}$$

Unlike regular ridge regression, CRR is unbiased. The proofs for an unbiased CRR can be found in Appendix A.2. Li et al. acknowledge that centering around $\hat{w}^{(1)}$ enables a more stable learning environment, compared to ordinary CL, but still struggles when tasks are too dissimilar. Another reason CRR has difficulties learning dissimilar tasks, is that all dimensions of w are penalized equally throughout all tasks, i.e.

$$\text{pen}_R(w) = (w - \hat{w}^{(1)})^\top \lambda I (w - \hat{w}^{(1)}) \tag{19}$$

. When learning a joint probability distribution, as in DIL, the information contained in D_t about w can vary across coordinates. As a solution to this, Zhao et al. [49] propose a generalized quadratic penalty for linear regression tasks, which allows individual regularization strengths in all directions of w.

From now on I will no longer restrict Cl to only two tasks.

Generalized l2-regression (GR) [49] extends CRR and ordinary CL. It is asymptotically equivalent to an unrestricted model, i.e. a model that can access all data samples at the same time. Zhao et al. define the unrestricted estimation error $\mathcal{L}(\cdot)$ over all tasks as:

$$\mathcal{L}(\hat{w}^{(T)}) = \sum_{i}^{d} \frac{\sigma^2}{\alpha_i^{(1)} n^{()} + ... + \alpha_i^{(T)} n^{(T)}} \tag{20}$$

with $\alpha_i^{(t)}$ being the i-th eigenvector of Σ_t. Note that this error is monotonously decreasing as T gets bigger, thus no forgetting [49]. GRs goal is to find a matrix $\Lambda^{(t)}$ for (19), which properly accommodates a samples contribution to each $\hat{w}_i$ so that the combined estimation error of $L(\hat{w}^{(t)})$ converges to $\mathcal{L}(\hat{w}^{(T)})$. The authors are able to prove that for $\Lambda^{(t)}$, a diagonizable matrix with $\Lambda^{(t)} = U \Delta U^\top, \Delta = \text{diag}(\delta_1, ...\delta_d)$, the loss $L(\hat{w}^{(t)})$ is bounded by $\mathcal{L}(\hat{w}^{(T)})$ if the diagonal values of Δ are

$$\delta_i = \frac{\sigma^2/(U_i^\top w^*)^2 + \alpha_i^{(1)} n^{(1)} + ... + \alpha_i^{(t-1)} n^{(t-1)}}{n^{(t)}} \tag{21}$$

For large $n^{(t)}$, $\frac{\sigma^2/(U_i^\top w^*)^2}{n^{(t)}}$ becomes small enough to be neglected and [49] approximate $\tilde{\Lambda}^{(t)} = \frac{1}{n^{(t)}} \sum_{i=1}^{t-1} n^{(i)} \Sigma_i$.

In this linear setting, the Σ_i are equivalent to the Hessian and Fisher information matrix of the loss function. This means that every changew_i is penalized proportionally to the information sample D_t contains about its previous state.

Zhao et al. [49] demonstrate how powerful regularization can be. However, that all tasks share the individual solution w^* is not always realistic. Kirkpatrick et al. [24] tackle this problem by taking a Bayesian look at the joint distribution over all tasks.

One of the most influential regularization approaches for CL is the *elastic weight consolidation* penalty (EWC) by Kirkpatrick et al. [4, 20, 25, 28, 40, 46, 48, 49]. They suggest measuring weight importance via the Fisher information matrix. Kirkpatrick et al. justify this approach through a probabilistic view of neural networks. They no longer want to find the parameters that best fit the data pattern, but find the most probable model weights, depending on a given data sample. Using Bayes' Rule and the assumption of independent samples, they express the conditional probability $\mathbb{P}(w|\mathcal{D}^{(t)}), \mathcal{D}^{(t)} = \{D_1, ..., D_t\}$ of the weights as [1]

$$\log \mathbb{P}(w|\mathcal{D}^{(t)}) = \log \mathbb{P}(D_t|w) + \log \mathbb{P}(w|\mathcal{D}^{(t-1)}) - \log \mathbb{P}(D_t) \tag{22}$$

and point out that all of the information about all prior tasks is in $\mathbb{P}(w|\mathcal{D}^{(t-1)})$, but is unavailable due to the sequential training constraint. To overcome this problem, the authors approximate the missing posterior as a Gaussian with expected value $\hat{w}^{(t-1)}$ and precision matrix $F = diag(\sum_{i<t} \mathcal{I}_i(w_1), ..., \sum_{i<t} \mathcal{I}_i(w_d))$ where $\mathcal{I}_i(w_j), j \in \{1, ..., d\}$ are the Fisher information of w_i from the i-th training step, thus $\mathbb{P}(w|\mathcal{D}^{(t-1)}) \dot{\sim} N(\hat{w}^{(t-1)}, F^{-1})$. The resulting penalty function is a weighted Ridge penalty, where the squared deviation from the previous parameters is weighed against its Fisher information:

$$\text{pen}_{EWC}(w) = \frac{\lambda}{2}(w - \hat{w}^{(t-1)})^\top F(w - \hat{w}^{(t-1)}) \tag{23}$$

Note that if the loss is chosen as the negative log-likelihood, EWC is equal to the 2nd Taylor approximation of a generalized forgetting rate in (14). In this case the Fisher information matrix is equivalent to the Hessian of the loss. In general the EWC penalty encourages gradient decent to follow along trajectories of w_i with low Fisher information which are thus less prone to forgetting.

Zenke et al. [48] argue that a static estimate of parameter importance between training

[1]Kirkpatrick et al. [24] only provide equation (22) for 2 tasks. The expansion for t tasks ,which is used here, can be found in Appendix A.1.

steps is not enough and suggest a dynamic solution, *synaptic intelligence* (SI), along the loss' gradient. Similar to EWC, GR and CRR they impose a quadratic penalty on the loss:

$$\text{pen}_{SI}(w) = \frac{\lambda}{2}(w - \hat{w}^{(1)})^\top H(w - \hat{w}^{(1)}) \tag{24}$$

where H is the diagonal of the Hessian of the current loss $L(X^{(2)}, y^{(2)}, w)$.
The next example is the *memory aware synapses* (MAS) penalty [1]. Similar to EWC and SI it focuses on task disjoint CL. To further improve the idea of gradient based importance, the authors consider the model output $h(\cdot)$ and measure its sensitivity to changes in the model parameters. The importance matrix Ω holds the mean gradients over all samples of the squared L2-normed outputs i.e. $\Omega^{(t)} = 1/n \sum_{i \leq n} \nabla \| h(x_i^{(t)}, \hat{w}^{(t)}) \|_2^2$. With the resulting penalty function:

$$\text{pen}_{MAS}(w) = \lambda(w - \hat{w}^{(t-1)})^\top \Omega^{(t-1)}(w - \hat{w}^{(t-1)}) \tag{25}$$

Aljundi et al. [1] aim to provide a flexible importance measure, which can be calculated on any representative data set, since it does not depend on the model loss.
Generally regularization in CL, with a squared penalty, restricts large deviations from the previously estimated parameters if these $w_i^{(t-1)}$ were important to prior learnings. Given some importance matrix A, the generalized squared-l2 penalty for the next training step is:

$$pen_{l2}(w) = \lambda(w - \hat{w}^{(t)})^\top A(w - \hat{w}^{(t)}) \tag{26}$$

Although FIM and Hessian at large are not identical, Benzing [4] demonstrate how SI and MAS are still linked to FIM. Yin et al. [46] provide a unifying analysis of squared penalties in CL. They conclude that the difference between the true loss over all tasks and its approximation depends on two factors: first a sample effect, which is negligible for increasing $n^{(t)}$, and second the technical error of the approximation. Thus they encourage more accurate approximations. Furthermore, Liu et al. [27] point out that the diagonal approximation of the Fisher information matrix has potential to lead gradient decent "off-path" and use rotations of the parameter space to adjust.

5.1.2 Sparse Penalties

The Ridge-like penalties provide control over a model's stability. Because their approximations become increasingly inaccurate with each new training step, they encourage smaller steps away from the current state of the model as training continues [46]. All of the algorithms presented so far imply that all parameters are, to some degree, useful across all tasks. *Adaptive Group Sparsity based Continual Learning* (AGS-CL) [23] and the *Dynamically Expandable Network* algorithm (DEN) [47] question this and suggest a Grouped-LASSO penalty. The parameter groups are determined by the neurons they connect to or come from. This way the model can benefit from already established weights and simultaneously use free neurons to fit task specific parameters.

AGS-CL argues that a selective use of network nodes could be beneficial when dealing with unrelated tasks. Their idea is to influence model plasticity via Grouped-LASSO regularization. Similar to the already seen penalties, the AGS-CL algorithm makes use of an importance matrix $\Omega \in \mathbb{R}^{l\times\nu}$ to decide whether weights connecting to a node should be protected or not.

$$\Omega^{(t)} = \Omega^{(t-1)} + \sum_{j=1}^{l}\sum_{k=1}^{\nu}\left[\frac{1}{n^{(t)}}\sum_{i=1}^{n^{(t)}}\mathrm{ReLU}_{j,k}(x_i^{(t)})\right] \tag{27}$$

With $\mathrm{ReLU}_{j,k}(x) = \max(0, x)$ as the activation function of node k in layer j, the task specific penalty term is then:

$$\begin{aligned}\mathrm{pen}_{AGSCL}(W) =& \mu \sum_{j,k\leq l,\nu} \mathrm{id}(\Omega_{j,k}^{(t-1)} = 0)\|W_{j,k}\|_2 \\ &+ \lambda \sum_{j,k\leq l,\nu} \mathrm{id}(\Omega_{j,k}^{(t-1)} > 0)\|W_{j,k} - \hat{W}_{j,k}^{(t-1)}\|_2\end{aligned} \tag{28}$$

All nodes with importance 0 are penalized directly and thus their weights might get estimated to 0. Deviations from very important nodes, i.e. nodes with high Ω-value, can be prevented completely due to the second penalty term.

Note that AGS-CL relies on a fixed network structure. With increasing task number, the ability "to freeze" entire nodes could potentially stop learning all together, if new tasks are too unrelated to prior knowledge. For example, a classifier with t output nodes needs to learn $t + 1$ categories.

DEN acknowledges this problem [47]. In order to provide a higher degree of plasticity, DEN identifies the task related sub-networks of the model and retrains it to evaluate if additional nodes are required to adequately adjust to the new data. To identify which established network parameters are useful to the new task, DEN uses a standard LASSO penalty on the weights directly connecting to the output layer. In case the sub-network's loss is too high, DEN gradually expands each layer with additional nodes, which are again selected via grouped-LASSO. The final network is then again estimated via Ridge-regularization as in (17).

Overall DEN aims to keep a sparse network for individual tasks through the abuse of prior knowledge and simultaneously is able to adapt to unrelated tasks by expanding its network structure when needed. This relates DEN closely to methods that directly influence a network's structure and hints that the targeted combination of CL algorithms could balance out individual weaknesses.

Another way of controlling changes in a model arises when looking at its outputs. Such methods generally use some representation of the old model to guide the new learning process. Since CL has clear training stages, these approaches can be seen as offline knowledge distillation [17], where the previous model acts like a teacher to the current one.

5.2 Regularization of Function Space

Li and Hoiem [26] provide with *Learning without Forgetting* (LwF) one of the earliest advances in this field. LwF assumes (multi-) image-classification tasks and specifies a logistic loss function $L(y, h(x)) = -y \log h(x)$ and the softmax activation function $h(x) = \exp(x_i)/\sum_{i \leq \#classes} \exp(x_i)$ in the output layer. In detail, LwF makes a prediction $y_o^{(t)}$, based on the old model, for the new sample before the next training cycle. These predictions are then used to penalize the fitted values $\hat{y}_o^{(t)}$ from the new model:

$$\text{pen}_{LwF}(W) = \lambda \sum_{i=1}^{\#classes} -y_{o,i}^{(t)} \log \hat{y}_{o,i}^{(t)} \tag{29}$$

Considering the predefined loss above, then pen_{LwF} could be viewed as a 2nd parallel learner. It is important to note that $\hat{y}_o^{(t)}$ and $y_o^{(t)}$ are not the actual values but some weighted version of themselves, in order to help the new network to better approximate old outputs (Knowledge Distillation loss [19]). LwF also makes some structural adjustments with each training step by adding a number of fully connected neurons to the output layer if the new sample contains unseen class labels. Without this, the network would force predictions in the same label space for all tasks and thus not be able to adjust to the expanding multinomial distribution of $\mathbb{P}(Y^{(t)}) = \prod_{i=1}^{t} \mathbb{P}(y^{(i)})$.
Instead of relying on an estimate of the previous true fitted values $\hat{y}^{(1)}, ..., \hat{y}^{(t-1)}$, Wang et al. [44] suggest with *Deep Retrieval and Imagination* (DRI) storing a small sample M of all previous tasks. In their work, they also focus on image classification with known labels (CIL or DIL). To ensure that M is representative of all $t-1+1$ tasks, the authors use a resampling strategy inspired by Welling [45] when adding new data points to it. During training DRI does not only aim to learn the combined sample $D^{(t)} \cup M$ but also just M. The additional penalty terms center new model outputs of already seen tasks

around their previous fittings, which are calculated on a copy of the old model.

$$\begin{aligned}\hat{W}^{(t)} =& \arg\min_{W} L(W, D^{(t)} \cup M) \\ &+ \beta L(W, M) + \frac{\alpha}{n^{(M)}} \sum_{i}^{n^{(M)}} \| f(W, x_i^{(M)}) - f(W^{(t-1)}, x_i^{(M)}) \|_2^2\end{aligned} \tag{30}$$

In their work Wang et al. come to a very similar conclusion, as Yin et al. [46] in their analysis of generalized l2 penalties in CL. The discrepancy between the true joint training error and their surrogate loss consists yet again of an approximation error and a finite-sample effect. They alleviate the approximation error by padding M with generated data from itself.

As a final example for function space regularization, I want to present the *functional regularization for continual learing* (FRCL) algorithm by Titsias et al. [40]. They take a Bayesian perspective on NN, such that the model outputs $\hat{y}^{(t)}$ are a random vector depending on the output function $f^{(t)}(\cdot)$. In contrast to prior Bayesian approaches to CL, FRCL only focuses on the outermost parameter layer, connecting to the output, and optimizes everything else: $f^{(t)}(x^{(t)}) = w^{(t)\top} g(x^{(t)}; \theta)$ where $g(x; \theta)$ is the composition of all hidden layers and their parameters θ. In each training step, they approximate the task specific output distributions $\mathbb{P}(y^{(i)})$ with a Gaussian Process (GP) [30] $p(y^{(t)} | f^{(t)}(X^{(t)}))$ and store a small collection of its inducing points $\tilde{D}_t = (\tilde{X}^{(t)}, \tilde{y}^{(t)} = f^{(t)}(\tilde{X}^{(t)}) \in \mathbb{R}^{d \times m^{(t)}} \times \mathbb{R}^{m^{(t)}}$. These inducing points are found via minimizing the Kullback-Leibler divergence KL from their approximated distribution to the true GP. The base line model then minimizes

$$L^{(t)}(\theta, w) = L_t(D^{(t)}, w^{(t)}, \theta) + \sum_{i=1}^{t-1} \frac{n^{(i)}}{m^{(i)}} L_i(\tilde{D}_i, w^{(i)}, \theta) \tag{31}$$

where each L_i is a task specific loss and $m^{(i)}$ the size of the inducing sample of task i. This approach makes a couple of assumptions and approximations about different parts of the network, which will be summarized in the remaining part of this subsection, as well as a non-trivial transformation to get to the explicit loss function.

Titsias et al. begin by assuming a normal prior on the outer-layer parameters $p_\theta(w^{(t)}) = N(0, \sigma_w^2 I)$ with its variational approximation $q(w^{(t)}) = N(w^{(t)}|\mu_{w^{(t)}}, \Sigma_{w^{(t)}})$, as well as a linear kernel [30] $K_{X^{(t)}} = \sigma_w^2 g(X^{(t)};\theta)^\top g(X^{(t)};\theta)$ for the network. Where $\mu_{w^{(t)}} \in \mathbb{R}^t$ and $\Sigma_{w^{(t)}} = C_{w^{(t)}} C_{w^{(t)}}^\top$ with $C_{w^{(t)}}$ is a square lower triangular matrix with positive diagonal elements. With this, the general posterior distribution over all function values is

$$p(y^{(t)}|f^{(t)}(X^{(t)})) = \int p\left(y^{(t)}|w^{(t)\top} g(x^{(t)};\theta)\right) q(w^{(t)}) dw^{(t)} \tag{32}$$

The authors acknowledge that regularization could already happen with this expression of the posterior. They argue that it would be a very expensive process and make use of the above mentioned approximation via inducing points $\tilde{D}_t$ to reduce the required storage space per task. These points are learned by minimizing the Kullback-Leibler divergence KL from $q(w^{(t)})$ to $p_\theta(w^{(t)})$. Now the task specific loss L_t becomes the evidence lower bound (ELBO) [6] for D_t and is maximized over θ and $q(w^{(t)})$. The authors transform the ELBO to the following equation, referencing multiple other papers. For convenience $w^{(t)\top} g(x^{(t)};\theta)$ will be referred to as $f^{(t)}$ further on.

$$L_t(\theta, q(w^{(t)})) = \sum_{i=1}^{n^{(t)}} \mathbb{E}_{q(w^{(t)})}\left[\log p(y_i^{(t)}|f_i^{(t)})\right] - \mathrm{KL}\left(q(w^{(t)}) \| p_\theta(w^{(t)})\right) \tag{33}$$

After this set up, we can focus on summarizing the last step, finding the penalty term for FRCL. Considering the learned inducing points $\tilde{D}_t$ and the joint distribution $p(y^{(t)}, f^{(t)}, \tilde{y}^{(t)})$, the true posterior can now be written as

$$p(y^{(t)}|f^{(t)}) = p_\theta(f^{(t)}|\tilde{y}^{(t)}, y^{(t)}) p_\theta(\tilde{y}^{(t)}|y^{(t)}) \tag{34}$$

which can now be approximated by the GP $p_\theta(f^{(t)}|\tilde{y}^{(t)}) q(\tilde{y}^{(t)})$, with $p_\theta(f^{(t)}|\tilde{y}^{(t)}) = N(f^{(t)}|\mu_{f^{(t)}}, \Sigma_{f^{(t)}})$ and $q(\tilde{y}^{(t)}) = N(\tilde{y}^{(t)}|\mu_{\tilde{y}^{(t)}}, C_{\tilde{y}^{(t)}} C_{\tilde{y}^{(t)}}^\top)$, only depending on the variational distribution over $\tilde{y}^{(t)}$. Thus FRCL needs new task samples to be able to explain the approximated posteriors $q(\tilde{y}^{(t)})$ of every prior tasks so far seen. This leads to the penalty term:

$$\text{pen}_{FRCL}(\theta, q(w^{(t)})) = -\sum_{j=1}^{t-1} \text{KL}(q(\tilde{y}^{(j)}) \| p_\theta(\tilde{y}^{(j)})) \tag{35}$$

with $p_\theta(\tilde{y}^{(t)}) = N(\tilde{y}^{(t)}|0, K_{\tilde{X}^{(t)}})$. Recalling that $\tilde{y}^{(t)}$ still depends on the distribution over $w^{(t)}$, its distributional parameters are $\mu_{\tilde{y}^{(t)}} = g(\tilde{X}^{(t)};\theta)^\top \mu_{w^{(t)}}$ and $C_{\tilde{y}^{(t)}} = g(\tilde{X}^{(t)};\theta)^\top C_{w^{(t)}}$. Ultimately, the algorithm stores $(\tilde{X}^{(t)}, \mu_{\tilde{y}^{(t)}}, C_{\tilde{y}^{(t)}})$ to memorize task t for future training cycles.

As we have seen, FRCL mitigates forgetting via saving an approximate posterior of the current task's probability distribution and regularizes coming tasks through a KL penalty term. The authors acknowledge that their approach has similarities to replay-based methods, further strengthening the assumption that a combination of different methods could be the key to successful CL.

6 Notes on Evaluation

At last it is important to note that this thesis does not provide information about the compared performance of each of the presented regularization techniques. Despite the fact that all authors have conducted their own experiments, a formal comparison, through survey work alone, is not possible at the moment. Each research team has used different data sets and the overlap of used reference models is very small. Additionally, ordinary CL, CRR and GR serve a strictly theoretical purpose, since their assumptions strongly restrict their real world applications. The teams behind them have only provided experimental proof of concepts via simulated data and acknowledge their limitations.

Nevertheless, the conducted experiments highlight the influence of EWC on the field. It is used as a reference model for the majority of the presented regularization methods [1, 23, 40, 47, 48].

Another point worth highlighting is that MINST and CIFAR are the most used data bases for the showcased methodologies [23, 44, 47, 48]. This raises the question if their overuse biases research towards a race for the ”best fit”.

7 Conclusion

As we have seen regularization on parameters is a powerful tool to stabilize a continual learning algorithm. Stability is a desirable quality when tasks are similar but can lead to a learning collapse if they are not. Direct regularization of parameters could lead to unwanted generalization problems for future tasks, thus slowing down the rate at which new tasks can be learned. These shortcomings could be alleviated by using the outputs for parameter estimation. Penalizing deviations in the output might lead to an overall more flexible learner. Such approaches require extra memory to store information about past tasks. This drawback can be mitigated by effective sampling to min-max memory size and retained information or incorporating a second generative model. Efforts on balancing stability and model plasticity have also been made with sparse penalties, which aim to reduce individual network connection or even entire nodes to 0. They reduce the task specific complexity of the model in order to achieve better generalization across all tasks. Requiring tasks which can be generalized well in the first place which explains why the linear task assumption is so widespread. Yet in order to learn a distinct set of tasks, like in CIL, shared label spaces seem too restrictive for one fixed network. Therefore, some allow for an expansion of the network to accommodate for new task specific requirements, such as a larger output space or general model plasticity. Overall the advantages and flaws of regularization in CL demonstrate the two fundamental criteria [43] for successful CL, intra- and inter-task generalizability.

A Appendix

A.1 Expansion of eq. 2 in [24] for t samples (22)

Let $D_i, i \in \{1, ..., t\}$ be t independent samples, as described in section 3, $\mathcal{D}^{(t)} = \{D_1, ..., D_t\}$ the joint samples and $w \in \mathbb{R}^d$ a weight vector. Then the conditional probability

$$\begin{aligned}\mathbb{P}(\mathcal{D}^{(t)}|w) &= \frac{\mathbb{P}(D_1, ..., D_t, w)}{\mathbb{P}(w)} \\ &= \frac{\mathbb{P}(D_1, ..., D_{t-1}|D_t, w)\mathbb{P}(D_t, w)}{\mathbb{P}(w)} \\ &= \mathbb{P}(D_1, ..., D_{t-1}|w)\mathbb{P}(D_t|w)\end{aligned} \tag{36}$$

This we plug into the Bayes' Rule for the posterior $\mathbb{P}(w|\mathcal{D}^{(t)})$ and get

$$\begin{aligned}\mathbb{P}(w|\mathcal{D}^{(t)}) &= \frac{\mathbb{P}(\mathcal{D}^{(t)}|w)\mathbb{P}(w)}{\mathbb{P}(\mathcal{D}^{(t)})} \\ &= \frac{\mathbb{P}(D_1, ..., D_{t-1}|w)\mathbb{P}(D_t|w)\mathbb{P}(w)}{\mathbb{P}(\mathcal{D}^{(t)})} \\ &= \frac{\mathbb{P}(w|D_1, ..., D_{t-1})\mathbb{P}(D_t|w)}{\mathbb{P}(D_t)}\end{aligned} \tag{37}$$

The approximate Gaussian for the posterior $\mathbb{P}(w|D_1, ..., D_{t-1})$ of all prior tasks is then $N(w, (\sum_{i=1}^{t-1} \text{diag}(F_i))^{-1})$ using the chain rule for independent Fisher information $F_i = \mathcal{I}_{D_i}(w)$.

A.2 CRR is unbiased

Proof. Let $D_1 = (X^{(1)}, y^{(1)})$ and $D_2 = (X^{(2)}, y^{(2)})$ be two conditionally independent linear regression problems, with $y^{(i)} \sim N(X^{(i)}w^*, \sigma^2 I), i \in \{1, 2\}$. Since $\hat{w}^{(1)}$ is estimated OLS, we already know that it is unbiased so we directly continue with derivation of $\hat{w}^{(2)}$:

$$\begin{aligned}
\hat{w}^{(2)} &= \arg\min_{w} \frac{1}{n}\|y^{(2)} - X^{(2)\top} w\|_2^2 + \text{pen}(w) \Leftrightarrow \\
0 &= \nabla[\frac{1}{n}\|y^{(2)} - X^{(2)\top} w\|_2^2 + \text{pen}(w)] \\
&= \nabla[\frac{1}{n}(y^{(2)} - X^{(2)}w)^\top (y^{(2)} - X^{(2)}w) + \lambda(w - \hat{w}^{(1)})^\top (w - \hat{w}^{(1)})] \\
&= -\frac{2}{n}X^{(2)\top} y^{(2)} + \frac{2}{n} w X^{(2)\top} X^{(2)} + 2\lambda w - 2\lambda \hat{w}^{(1)} \Rightarrow \\
\hat{w}^{(2)} &= (X^{(2)\top} X^{(2)} + \lambda n I)^{-1}(X^{(2)\top} y^{(2)} + \lambda n \hat{w}^{(1)})
\end{aligned} \tag{38}$$

Define $A := (X^{(2)\top} X^{(2)} + \lambda n I)^{-1}$, then $\mathbb{E}[\hat{w}^{(2)}]$ is

$$\begin{aligned}
\mathbb{E}[\hat{w}^{(2)}] &= \mathbb{E}[(X^{(2)\top} X^{(2)} + \lambda n I)^{-1}(X^{(2)\top} y_2 + \lambda n \hat{w}^{(1)})] \\
&= A\left(\mathbb{E}[X^{(2)\top}(X^{(2)} w^{(2)} + \varepsilon^{(2)})] + \lambda n w^*\right) \\
&= A(X^{(2)\top} X^{(2)} w^* + \lambda n w^*) \\
&= AA^{-1} w^* \\
&= w^*
\end{aligned} \tag{39}$$

□

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget, 2018. URL `https://arxiv.org/abs/1711.09601`.

[2] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-free continual learning, 2019. URL `https://arxiv.org/abs/1812.03596`.

[3] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning, 2019. URL `https://arxiv.org/abs/1903.08671`.

[4] F. Benzing. Unifying regularisation methods for continual learning, 2021. URL `https://arxiv.org/abs/2006.06357`.

[5] S. A. Bidaki, A. Mohammadkhah, K. Rezaee, F. Hassani, S. Eskandari, M. Salahi, and M. M. Ghassemi. Online continual learning: A systematic literature review of approaches, challenges, and benchmarks, 2025. URL `https://arxiv.org/abs/2501.04897`.

[6] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Apr. 2017. ISSN 1537-274X. doi: 10.1080/01621459.2017.1285773. URL `http://dx.doi.org/10.1080/01621459.2017.1285773`.

[7] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. S. Torr, and M. Ranzato. On tiny episodic memories in continual learning, 2019. URL `https://arxiv.org/abs/1902.10486`.

[8] K.-L. Du and M. N. S. Swamy. *Neural Networks and Statistical Learning*. Springer London, 2 edition, 2019.

[9] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning, 2018. URL `https://arxiv.org/abs/1810.13166`.

[10] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach. Adversarial continual learning, 2020. URL `https://arxiv.org/abs/2003.09553`.

[11] I. Evron, E. Moroshko, R. Ward, N. Srebro, and D. Soudry. How catastrophic can catastrophic forgetting be in linear regression?, 2022. URL `https://arxiv.org/abs/2205.09588`.

[12] L. Fahrmeir, T. Kneib, S. Lang, and B. D. Marx. *Regression: Models, Methods and Applications*. Springer Berlin, 2 edition, 2022.

[13] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks, 2017. URL `https://arxiv.org/abs/1701.08734`.

[14] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(99)01294-2. URL `https://www.sciencedirect.com/science/article/pii/S1364661399012942`.

[15] f. given i=D, given=Dilip. What is chatgpt openai and how it is built: The methodology behind it. URL `https://levelup.gitconnected.com/what-is-chatgpt-openai-how-it-is-built-the-technology-behind-it-ba3e8acc1e9b`.

[16] f. given i=K, given=Katharina. The extreme cost of training ai models. URL `https://www.forbes.com/sites/katharinabuchholz/2024/08/23/the-extreme-cost-of-training-ai-models/`.

[17] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, Mar. 2021. ISSN

1573-1405. doi: 10.1007/s11263-021-01453-z. URL http://dx.doi.org/10.1007/s11263-021-01453-z.

[18] V. Hassija, A. Chamola, V.and Mahapatra, and et al. Interpreting black-box models: A review on explainable artificial intelligence. *Cogn Comput*, 16:45–74, Jan. 2024. doi: 10.1007/s12559-023-10179-8. URL https://doi.org/10.1007/s12559-023-10179-8.

[19] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

[20] F. Huszár. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(11), Feb. 2018. ISSN 1091-6490. doi: 10.1073/pnas.1717042115. URL http://dx.doi.org/10.1073/pnas.1717042115.

[21] I. (I.). Neural network. URL https://www.ibm.com/think/topics/neural-networks.

[22] K. Javed and M. White. Meta-learning representations for continual learning, 2019. URL https://arxiv.org/abs/1905.12588.

[23] S. Jung, H. Ahn, S. Cha, and T. Moon. Continual learning with node-importance based adaptive group sparse regularization, 2021. URL https://arxiv.org/abs/2003.13726.

[24] J. Kirkpatrick, R. Pascanu, N. Rabinowitza, J. Veness, A. A. R. Guillaume Desjardins, K. Milan, J. Quan, T. Ramalho, D. H. Agnieszk Grabska-Barwinska, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv:1612.00796v2*, 2017.

[25] H. Li, J. Wu, and V. Braverman. Fixed design analysis of regularization-based continual learning, 2024. URL https://arxiv.org/abs/2303.10263.

[26] Z. Li and D. Hoiem. Learning without forgetting, 2017. URL `https://arxiv.org/abs/1606.09282`.

[27] X. Liu, M. Masana, L. Herranz, J. V. de Weijer, A. M. Lopez, and A. D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting, 2018. URL `https://arxiv.org/abs/1802.02950`.

[28] N. Loo, S. Swaroop, and R. E. Turner. Generalized variational continual learning, 2020. URL `https://arxiv.org/abs/2011.12328`.

[29] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning, 2022. URL `https://arxiv.org/abs/1706.08840`.

[30] M. Ludkovski and J. Risk. *Gaussian Process Models for Quantitative Finance.* Springer Nature Switzerland, Cham, 2025. ISBN 978-3-031-80874-6. doi: 10.1007/978-3-031-80874-6. URL `https://doi.org/10.1007/978-3-031-80874-6`.

[31] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights, 2018. URL `https://arxiv.org/abs/1801.06519`.

[32] Matt and Matt. How much data is generated per day, 12 2024. URL `https://www.digitalsilk.com/digital-trends/how-much-data-is-generated-per-day/`.

[33] J. Mcclelland, B. Mcnaughton, and R. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102 3:419–457, 1995. doi: https://doi.org/10.1037/0033-295X.102.3.419.

[34] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi: https://doi.org/10.

1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[35] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh. Understanding the role of training regimes in continual learning, 2020. URL https://arxiv.org/abs/2006.06958.

[36] R. Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological RReview*, 97 2:285–308, 1990. URL https://api.semanticscholar.org/CorpusID:18556305.

[37] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning, 2017. URL https://arxiv.org/abs/1611.07725.

[38] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learing: From Theory to Algorithms*. Cambridge University Press, 2014.

[39] A. Sudmann. *Künstliche neuronale Netzwerke als Black Box*, pages 189–199. Springer Fachmedien Wiesbaden, Wiesbaden, 2020. ISBN 978-3-658-27852-6. doi: 10.1007/978-3-658-27852-6_10. URL https://doi.org/10.1007/978-3-658-27852-6_10.

[40] M. K. Titsias, J. Schwarz, A. G. de G. Matthews, R. Pascanu, and Y. W. Teh. Functional regularisation for continual learning with gaussian processes, 2020. URL https://arxiv.org/abs/1901.11356.

[41] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, Dec 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3.

[42] E. Verwimp, R. Aljundi, S. Ben-David, M. Bethge, A. Cossu, A. Gepperth, T. L. Hayes, E. Hüllermeier, C. Kanan, D. Kudithipudi, C. H. Lampert, M. Mundt, R. Pascanu, A. Popescu, A. S. Tolias, J. van de Weijer, B. Liu, V. Lomonaco, T. Tuytelaars, and G. M. van de Ven. Continual learning: Applications and the road forward, 2024. URL `https://arxiv.org/abs/2311.11908`.

[43] L. Wang, X. Zhang, H. Su, J. Zhu, Fellow, and IEEE. A comprehensive survey of continual learning: Theory and method and application, 2024. URL `https://arxiv.org/abs/2302.00487`.

[44] Z. Wang, L. Liu, Y. Duan, and D. Tao. Continual learning through retrieval and imagination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8): 8594–8602, Jun. 2022. doi: 10.1609/aaai.v36i8.20837. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20837`.

[45] M. Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, volume 382, page 141, 06 2009. doi: 10.1145/1553374.1553517.

[46] D. Yin, M. Farajtabar, A. Li, N. Levine, and A. Mott. Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint, 2021. URL `https://arxiv.org/abs/2006.10974`.

[47] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks, 2018. URL `https://arxiv.org/abs/1708.01547`.

[48] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence, 2017. URL `https://arxiv.org/abs/1703.04200`.

[49] X. Zhao, H. Wang, W. Huang, and W. Lin. A statistical theory of regularization-based continual learning, 2024. URL `https://arxiv.org/abs/2406.06213`.

Declaration of authorship

Munich, March 20th, 2025

Name