CONTINUAL LEARNING THROUGH HAMILTON EQUATIONS

Alessandro Betti
Inria, CNRS, Lab I3S, Maasai Team
Université Côte d'Azur, Nice, France
alessandro.betti2@unisi.it

Lapo Faggi
DINFO, University of Florence
Florence, Italy
lapo.faggi@unifi.it

Marco Gori
DIISM, University of Siena
Siena, Italy
marco.gori@unisi.it

Matteo Tiezzi
DIISM, University of Siena
Siena, Italy
mtiezzi@diism.unisi.it

Simone Marullo
DINFO, University of Florence
Florence, Italy
simone.marullo@unifi.it

Enrico Meloni
DINFO, University of Florence
Florence, Italy
enrico.meloni@unifi.it

Stefano Melacci
DIISM, University of Siena
Siena, Italy
mela@diism.unisi.it

ABSTRACT

Learning in a continual manner is one of the main challenges that the machine learning community is currently facing. The importance of the problem can be readily understood as soon as we consider settings where an agent is supposed to learn through an online interaction with a data stream, rather than operating offline on previously prepared data collections. In the last few years many efforts have been spent in proposing both models and algorithms to let machines learn in a continual manner, and the problem still remains extremely challenging. Many of the existing works rely on re-adapting the usual learning framework inherited from classic statistical approaches, that are typical of non-continual-learning oriented problems. In this paper we consider a fully new perspective, rethinking the methodologies to be used to tackle continual learning, instead of re-adapting offline-oriented optimization. In particular, we propose a novel method to frame continual and online learning within the framework of optimal control. The proposed formulation leads to a novel interpretation of learning dynamics in terms of Hamilton equations. As a case study for the theory, we consider the problem of unsupervised optical flow estimation from a video stream. An experimental proof of concept for this learning task is discussed with the purpose of illustrating the soundness of the proposed approach, and opening to further research in this direction.

1 INTRODUCTION

In this paper we focus on the problem of letting an artificial agent learn from a continuous stream of information. In this specific learning setting, data is not regarded as a collection (a set) of points which are then used to approximate statistical quantities such as functional risks, as happens in the vast majority of Statistical Machine Learning problems (Vapnick, 1998). Processing a data stream implicitly requires to take into consideration its temporal structure. This is radically different from SGD-like algorithms (Robbins & Monro, 1951; Bottou, 2010), where samples of shuffled information are leveraged to enforce stochastic approximations of optimality conditions. SGD-like methods aim to give a convenient approximation of problems in which a batch of data is available for large-scale optimization and, as such, they are generally not suitable for handling continual learning problems.

Due to the relevance of the continuous/lifelong scenario for a broad class of learning problems, a large number of scientific papers on continual learning has been written in the past few years. Many of the recently proposed methods (see the surveys of Parisi et al. (2019); Delange et al. (2021); Mai et al. (2022)) exploit classical statistic-based tools. Notably, task incremental learning formalizes the general idea of lifelong learning by assuming that there is a sequence of problems, where the goal is to control a statistical risk at each step of the sequence without having access to the data available at previous steps (Delange et al., 2021). In this way, each of the tasks of the sequence can be basically

regarded as a standard machine learning problem with the big caveat that the sequence itself plays a fundamental role in the overall learning process. While "time" in task-incremental literature has usually been regarded as a discrete index that labels the sequence of tasks (i.e., rather than a variable that is synchronized with the time of the environment), other approaches have been proposed to interpret the learning process as a set of evolution laws, referred to as Cognitive Action Laws (CALs), for the learning agent (Betti et al. (2019), Tiezzi et al. (2020)). In such approach, the temporal variables couple the updates of the model with the changes in the environment, like it happens in physics. From a global perspective the problem can be described in terms of a functional index, similar to the *action* of Classical Mechanics, where the learning quality is evaluated by an integral performed in the temporal direction.

In this paper, we aim at establishing a new connection between learning and optimal control problems (Evans, 1983). We show how this link naturally emerges as soon as we start to use neural recurrent-like architectures to model the learning agent. More precisely, we argue that the weights of a neural networks can be regarded as control parameters for the temporal dynamics of the neuron outputs which, in turn, can be considered to be the state of a control problem. With these identifications, summarized in Table 1, the learning problem is interpreted as the mechanism that *steers* the weights of the networks based on an index that takes into account the environmental information. Exactly as it happens for CALs, the objective that guides the learning process is expressed by means of a scalar functional that matches the *cost functional* in optimal control theory (Bardi et al., 1997). We remark that our proposal is general, even if we will focus on models based on neural networks. Finally, let us note that, despite few formal similarities, what we are proposing here has very few intersections with Neural ODEs (Chen et al., 2018). Similar considerations hold for works that further extended the Neural ODE model, such as Hamiltonian Neural Networks (Greydanus et al., 2019). Gnecco et al. (2015) also tackled a class of online learning problems using an optimal control inspired approach. However their method does not consider the incremental aspects of learning, since consecutive samples are assumed to be mutually independent and they enter the cost functional through random matrices whose statistical properties are assumed to be known. The (linear) relationship between inputs and the corresponding labels is also assumed to be known and time independent. In our work, samples originating from the input stream are non-i.i.d., and non-stationarities in the generating distribution may be also considered. Kim & Yang (2020) developed a Q-learning method for continuous-time dynamical systems exploiting an analogue of the Hamilton-Jacobi-Bellman equation. As in classical Q-learning-based Reinforcement Learning, the state-action value function is learnt through several interactions with the environment. In contrast, in the method we propose here, Hamilton equations are solved in an online fashion.

The paper is organized as follows. In Section 2 we lay down some notation and we define useful quantities for the remainder of the paper. In Section 3 we describe the general setting of optimal control theory that we want to take into account and we subsequently show how it can be used to model learning problems. In Section 4 a case study based on the problem of online prediction of the optical flow is presented, including an experimental proof of concept. Finally, in Section 5 we briefly discuss the advantages of the approach and we point out the main difficulties and problems that still need to be faced.

2 Background and Notation

Let us consider a space of data Ω and a given trajectory $z\colon [0,T] \to \Omega$ that maps a temporal instant that belongs to the temporal horizon $[0,T]$ (with T potentially infinite) onto the space of data. We want to tackle the general problem of learning a model[1] $\mu\colon (\mathbb{R}^N)^{[0,T]} \times [0,T] \to \mathbb{R}^n$, where N is the number of the parameters of the model and n the size of its output.[2] Given a trajectory of parameters $t \mapsto \alpha(t)$ and a temporal instant $t \in [0,T]$, the considered model returns a prediction $\mu(\alpha, t) \in \mathbb{R}^n$. For example, a feedforward neural network could be seen as particular case of the model μ when $\alpha(t)$ are the weights of the network at time t and $z(t)$ is the input data at the same instant. In general, μ can make predictions of a sample that are based on the whole trajectory of parameters. As we have already anticipated in Section 1, here we consider what we refer to as "recurrent" models, that means that $t \mapsto \mu(\alpha, t)$ is a solution of a differential equation with unknown $y(\cdot)$,

$$\dot{y}(t) = f(y(t), \alpha(t), z(t)), \qquad t \in [0,T], \tag{1}$$

where $f\colon \mathbb{R}^n \times \mathbb{R}^N \times \Omega \to \mathbb{R}^n$ is a Lipschitz function. We call this a recurrent model, or recurrent architecture, because equation 1, in discrete time, is exactly the usual update equation for the state of a recurrent neural network.

The other ingredient that we need to properly define in order to have a well-posed learning problem is a loss function. In what follows we denote with $\ell\colon \mathbb{R}^n \times \mathbb{R}^N \times \Omega \to \overline{\mathbb{R}}_+$ the function that given a prediction $y \in \mathbb{R}^n$, a set of parameters $w \in \mathbb{R}^N$ and an input z returns the loss value $\ell(y, w, z)$ which is a local-in-time measure of the quality of

[1] Here with the symbol B^A, where A and B are sets, we indicate the space of all maps $f\colon A \to B$

[2] In this general setting the expression "learn a model" means finding the appropriate trajectory of parameters $t \mapsto \alpha(t) \in \mathbb{R}^N$

the predictions of the model, given the current value of the parameters. A traditional way to measure the performance of the learner is by means of the *sequential risk*

$$\frac{1}{T}\int_0^T \ell(y(t,\alpha),\alpha(t),z(t))\,dt \tag{2}$$

where $t \mapsto \alpha(t)$ is the trajectory of model parameters that defines the learning process, and where $y(t,\alpha)$ is a solution of equation 1. Notice that the dependence of the quantity in equation 2 on α comes both from the explicit dependence of the loss (that may require some explicit regularization on the parameters) but also, and in a crucial way, through the model, hence the notation $y(t,\alpha)$. In the next section we will discuss how it is possible to control the sequential risk in equation 2 by means of optimal control techniques. We refer the reader to Table 1 for a list of some common terms used in optimal control and their equivalent in the context of learning.

More precisely, we develop an approach that uses optimal control to tackle the described learning problem in the context of Online Continual Learning. Even if our work is mostly related to a domain incremental setting (Mai et al., 2022), an extension to more specific task/class incremental settings can be readily achieved by considering an additional explicit temporal dependence on index ℓ. It is also worth noting that the classical notion of online learning refers to all those cases in which samples are processed one after the other, i.e., in a sequential manner. On the other hand, Online Continual Learning specifically focuses on non-i.i.d. streams of data and aims to overcome catastrophic forgetting that characterizes neural networks in continual scenarios. The model is also required to learn seeing each sample only once. In the domain incremental setting, non-stationary generating data distributions are also considered (Mai et al., 2022).

Table 1: Relations between some common terms in the context of optimal control and in the one of machine learning.

Optimal Control	**Continual Learning**
State	Model outputs
Control variable	Model parameters/weights
Lagrangian	Loss function
Cost functional	Sequential risk

3 Hamiltonian Learning

An optimization problem involving the index of equation 2 can be solved, in the optimal control setting, embedding it into a family of problems that differs only in the time and value of the initialization. To this end we consider the following state equation

$$\begin{cases} \dot{y}(s) = f(y(s),\alpha(s),z(s)); & \text{for } s \in (t,T) \\ y(s) = x, & \text{for } s = t \end{cases} \tag{3}$$

and we will assume that f is such that for any choice of the parameter α, of the initial state x and time t and for any signal $t \mapsto z(t)$, equation 3 has a unique solution that we denote by $y_{x,t}(s,\alpha)$. Here y is the state vector, $\dot{y}$ its time derivative and $\alpha \in \mathcal{A} \subset (\mathbb{R}^N)^{[0,T]}$ a control trajectory;[3] T is the terminal time. The objective of classical fixed-time, free-endpoint optimal control theory is to find the control α^* that drives the system along the optimal state trajectory y^* minimizing a given cost functional

$$\alpha \mapsto C_{x,t}(\alpha) := \int_t^T \ell(y_{x,t}(s,\alpha),\alpha(s),z(s))\,ds + g(y_{x,t}(T,\alpha)), \tag{4}$$

where $g(\cdot)$ is the terminal cost function (Bardi et al., 1997).

In the standard dynamic programming approach the first step is to introduce the so-called *value function* u, that quantifies the minimal cost needed to reach the terminal time T starting from an arbitrary initial state x at time t:

$$u(x,t) := \inf_{\alpha\in\mathcal{A}} C_{x,t}(\alpha) \tag{5}$$

From equation 4 and equation 5 it is immediate to see that the value function satisfies

$$u(x,T) = g(x) \qquad \forall x \in \mathbb{R}^n \tag{6}$$

as a right boundary condition. Through a dynamic programming approach and under the assumption of a continuous and differentiable value function, it can be shown that u is the solution of the following non-linear partial differential equation (PDE), known as Hamilton-Jacobi-Bellman (HJB) equation:

$$u_t(x,t) = -\min_{a\in\mathbb{R}^N}\{\ell(x,a,t) + Du(x,t)f(x,a,z(t))\}, \tag{7}$$

[3] A typically assumption for the space $\mathcal{A}$ is that it is the space of Lebesgue-measurable function on $[0,T]$

together with the boundary condition of equation 6. Here, u_t (Du) stands for the partial derivative of u with respect to its second (first) argument respectively. The HJB PDE is a necessary and sufficient condition for optimality in the sense that, given the solution u of HJB with boundary condition of equation 6 and having defined $\hat{\alpha}(x,s) \in \arg\min_{a\in\mathbb{R}^N}\{\ell(x,a,z(s)) + Du(x,s)f(x,a,z(s))\}$, the optimal state trajectory $y^*(s)$ is given by the solution of the initial value problem

$$\begin{cases} \dot{y}(s) = f(y(s), \hat{\alpha}(y(s),s), z(s)); & \text{for } s \in (t,T) \\ y(s) = x, & \text{for } s = t \end{cases} \tag{8}$$

and the associated optimal control is $\alpha^*(s) = \hat{\alpha}(y^*(s), s)$. It is worthwhile to introduce the *Hamiltonian* (Bardi et al., 1997) of the system as the scalar function $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$

$$H(x,p,s) := \min_{a\in\mathbb{R}^N} \mathcal{H}(x,p,a,s) \tag{9}$$

where we have set $\mathcal{H}(x,p,a,s) := \ell(x,a,z(s)) + p \cdot f(x,a,z(s))$ with $\cdot$ being the standard scalar product in $\mathbb{R}^n$. With this notation, the HJB equation in equation 7 can be written as $u_t(x,t) + H(x, Du(x,t), t) = 0$.

Solving a PDE on the whole domain $\mathbb{R}^n \times [0,T]$ is not easy to handle. Fortunately, a very elegant idea from PDE theory is particularly handy to evaluate the solution of equation 7 along a trajectory $s \mapsto y(s)$. The basic idea is to look for curves along which the PDE reduces to a system of ODEs. We will argue that in our case these equations are easier to handle and to interpret. This general method is known as the *method of characteristics* and can be applied when the Hamiltonian H is smooth (for a comprehensive treatment of the modern theory of dynamic programming we refer to the book of Bardi et al. (1997)). First of all, we define the *costate* p as the trajectory

$$p(s) = Du(y(s), s), \tag{10}$$

then the method of characteristics assure us that if $y(\cdot)$ is optimal and H is differentiable then $y(\cdot)$ and $p(\cdot)$ satisfy a set of differential equations known as *Hamiltonian system*:

$$\begin{cases} \dot{y}(s) = D_p H(y(s), p(s), s) & s \in (t,T); \\ \dot{p}(s) = -D_x H(y(s), p(s), s) & s \in (t,T), \end{cases} \quad \text{and} \quad \begin{cases} y(s) = x & s = t; \\ p(s) = Dg(y(s)) & s = T. \end{cases} \tag{11}$$

Here $D_p H$ and $D_x H$ are the partial derivatives of H with respect to its second and first argument respectively. The boundary conditions on p follow directly from its definition in equation 10, from the definition of value function in equation 5, and from the boundary condition on u in equation 6. Indeed, since $u(x,T) = g(x)$, then $p(T) = Dg(y(T))$. The converse also holds, i.e., given a solution of equation 11 it is possible to exhibit a solution of the HJB equations. Notice that the first differential equation in y must be also consistent with equation 3. If we let for every fixed y and p in $\mathbb{R}^n$ and for each $s \in (t,T)$, $\alpha(y,p,s) \in \arg\min_{a\in\mathbb{R}^N} \mathcal{H}(y,p,a,s)$, then it is not difficult to see that $D_p H(y(s),p(s),s) = f(y(s), \alpha(y(s),p(s),s), s)$. Beside describing the characteristics of equation 7, the Hamiltonian system in equation 11 has another useful interpretation that also gives us a better understanding of the "geometric" meaning of the costate p. This other way of looking at the Hamiltonian equations comes from a direct interpretation of the optimization problem for the cost $C_{x,t}$ under the constraint in equation 3 and it is known in the literature as Pontryagin's Maximum Principle (PMP) (see Bardi et al. (1997)). The PMP suggests to look at the costate p as a Lagrange multiplier associated to the dynamical constraint that defines the evolution of the state, and in this sense can be seen as a sort of delta error that is propagated backward in time (a concise and clear explanation can be found in Evans (1983)).

Hence it is clear that with the choices $t = 0$ and $g = 0$ equation 11 gives us a method, rooted in dynamic programming, to tackle the learning problems discussed in Section 2. This gives us a direct way to interpret a large class of learning problems defined over a continuous stream of data in the framework of optimal control. A particularly interesting case arises, as we will better explore in Section 4 when we assume that the function $f(y,a,\cdot)$ in equation 3, for all $y \in \mathbb{R}^n$ and $a \in \mathbb{R}^N$, is a neural network. In this scenario the control parameters that "steer" the learning agent predictions are the weights of a neural network. The gradient-based learning algorithms, commonly used for FNNs, are replaced by equation 11 with $t = 0$ and $p(T) = 0$. Once the argmin of $\mathcal{H}$ has been evaluated, the corresponding optimal (y,p) trajectories are computed using equation 11 (again with the choices $t = 0$ and $p(T = 0)$)—in Section 4 we will propose a general method to achieve exactly this when a closed form of the Hamiltonian is not available.

3.1 Assumptions and Limitations

The right boundary condition on $p(T) = 0$ triggers several important considerations. Since in most Machine Learning applications it is not feasible to enforce the condition by direct numerical methods (because of the complexity of the model and the length of the temporal interval), we must rely on other approaches that satisfy causality requirements. The importance of this condition can hardly be overstated and every sound theory of learning should seriously take into

account this issue. In this work, we propose to tackle such a problem by leveraging the possibility to actively *modify the data from the input stream* without changing the essence of the learning problem, with the purpose of promoting solutions with $p \to 0$ as $t \to T$. The main intuition behind this idea comes from the interpretation of the costate given by the PMP: An appropriate filtering of the input could loosen the dynamical constraint in equation 3 so that the reaction needed to enforce this constraint (which is proportional to p given its interpretation as a Lagrange multiplier) could be made arbitrarily small. In Section 4 we will show a concrete example in the case of optical flow estimation. Nevertheless, a general and complete analysis of this very crucial point is beyond the scope of this work and it should be the subject of future explorations.

4 Case study: Learning to Estimate Optical Flow

In this section we show how the theory described in Section 3 can be used to formulate and solve an unsupervised learning problem concerning the estimation of optical flow. Broadly speaking, optical flow is the apparent motion that can be detected between a pair of consecutive frames in a two dimensional retina. However, despite the recent major improvements in optical flow estimation (Fischer et al., 2015; Ilg et al., 2017; Zhai et al., 2021), a well posed mathematical definition of this quantity is still missing (Verri, 1987; Verri & Poggio, 1989; Aubert & Kornprobst, 2006). For the purposes of this paper we will restrict ourselves to the classical notion of optical flow as it has been introduced by Horn and Schunck in a seminal paper published at the beginning of the eighties (Horn & Schunck, 1981). The basic idea behind their work is to use the local-in-time constancy of a *brightness* intensity function that encodes a video stream $b\colon \Sigma \times [0,T] \to [0,1]$ where Σ is a subset of $\mathbb{R}^2$, in order to find a constraint that characterizes (although not in a complete fashion, as we will see) the so-called optical flow. The brightness constancy assumption basically means that $\forall t_0 > 0$ there exists a $\tau > 0$ such that for every $x_0 \in \Sigma$ we can define the trajectory $\gamma_{x_0}\colon [t_0, t_0+\tau] \to \Sigma$ that maps $t \mapsto \gamma_{x_0}(t) \in \Sigma$ with $\gamma_{x_0}(t_0) = x_0$ for which

$$b(\gamma_{x_0}(t), t) = b(x_0, t_0), \qquad \forall t \in [t_0, t_0+\tau]; \tag{12}$$

Under the assumption of smoothness of b we can equivalently express the above constraint in an infinitesimal form,

$$b_t(x_0, t_0) + v(x_0, t_0) \cdot Db(x_0, t_0) = 0, \tag{13}$$

where $v(x_0, t_0) := (d\gamma_{x_0}/dt)(t_0)$ is the optical flow, $\cdot$ is the standard scalar product in $\mathbb{R}^2$ and b_t and Db are the partial derivatives of b with respect to its second (time) and first (space) arguments, respectively.

Of course, condition of equation 12 holds up to a certain degree when there are no occlusions and no dramatic changes in the light conditions. More than that, as we mentioned earlier, equation 13 is not sufficient to identify the optical flow since it only determines its projection along Db. Hence a selection method is needed and the one adopted by Horn and Schunck, as many other approaches as well (Aubert & Kornprobst, 2006) relies on a regularization imposed through a calculus of variation method. More precisely, they looked for a solution of the problem

$$\inf_{v \in V} \big(A^b_t(v) + S_t(v)\big) \tag{14}$$

where V is an appropriate functional space, $A^b_t(v) := \int_\Omega \big(b_t(x,t) + v \cdot Db(x,t)\big)^2\, dx$, enforces the constraint of equation 13, while S imposes smoothness. Additional terms can be added to condition the extraction of the flow where the brightness is constant. The form of S may vary depending on the kind of regularity that we want to require in the solution, however here as stated we take into account the regularization term used in the original work of Horn & Schunck (1981): $S_t(v) = \int_\Sigma \|Dv^1\|^2 + \|Dv^2\|^2\, dx$, where $\|\cdot\|$ is a norm and Dv^1 and Dv^2 are the spatial gradients of the components of the optical flow. The solution of problem in equation 14, as the subscripts in the notation suggest, gives the optical flow at time t. Hence, the quantity $A^b_t(v) + S_t(v)$, when appropriately written on a discretized retina, will serve (with an appropriate modification) as a loss function of the form that we considered in Section 2 and Section 3, where the input signal is the video stream $t \mapsto b(\cdot, t)$ together with its temporal variation $t \mapsto b_t(\cdot, t)$.

To this end. let us discretize the spatial coordinates by considering a two dimensional grid of $P = d\,h$ pixels indexed by $1 \le i \le d$ and $1 \le j \le h$. Then, the brightness field on this discrete grid can be represented by the map $s \mapsto b_{ij}(s)$ where i and j refers to the coordinates of the pixel. In the same way we represent with $s \mapsto v_{ij}(s) \in \mathbb{R}^2$ the optical flow field. We furthermore assume that the optical flow is computed by the following model

$$\begin{cases} \dot v_{ij}(s) = f_{ij}(v(s), w(s), z(s)) & s \in [0,T] \\ v_{ij}(0) = 0\,, \end{cases} \tag{15}$$

where $z(s) = (b(s), \dot b(s))$, with $\dot b_{ij}(s)$ being the discretized version of the temporal variation of the brightness b_t. As we can clearly see, the computational model for v is exacly of the form of equation 3, where v replaces y, and w

are used in place of α. Here we make the additional assumptions, as we anticipated in Section 3, that $w(s)$ are the weights of the neural network and they represent the control variables of the dynamical system under consideration. Moreover, in what follows we assume that $f(v, w, s) = c(-v + F(z(s), w))$, where, $F(\cdot, w)$ is a feedforward neural newtwork for all $w \in \mathbb{R}^N$ with $F \colon \Omega \times \mathbb{R}^N \to \mathbb{R}^n$, while c is a positive constant. Notice that, for $c \to \infty$ one recovers a feed-forward computation for the optical flow, that is $v(s) = F(z(s), w(s))$. With this definitions the Horn and Schunck functional in equation 14 can be translated into the loss function

$$\sum_{i=1}^{d}\sum_{j=1}^{h} \left| \dot{b}_{ij}(s) + v_{ij}(s) \cdot (\mathrm{D}\, b)_{ij}(s) \right| + \sum_{i=1}^{d}\sum_{j=1}^{h} |\mathrm{D}\, v^1_{ij}| + |\mathrm{D}\, v^2_{ij}|,$$

with the choice of L_1 norm for both the regularization term and the term in A^b_t, and where D is the discrete gradient operator. However, as it is immediate to realize, this loss function has the peculiarity that it does not explicitly depends on the control parameters w (but it will depend on w through the state v). In order to avoid any eventual ill-posedness of the problem due to this aspect, we decided to add an additional regularization by including a term that imposes a second-order version of the equation 12. More precisely, we add the constraint $d^2 b(\gamma_x(s), s)/ds^2 = 0$, for all $x \in \Sigma$ and $t \in [0, T]$. Notice that this constraint is trivially fulfilled when the brightness constancy assumption is exactly satisfied, but at the same time it may serve as regularization when the constrains are softly imposed through a loss function. A direct calculation shows that this constraint can be rewritten in terms of v as

$$b_{tt}(x,t) + v(x,t) \cdot b_{xx}(x,t)v(x,t) + 2b_{tx}(x,t) \cdot v(x,t) + b_x(x,t) \cdot (v_t(x,t) + v(x,t) \cdot Dv(x,t)) = 0,$$

where b_{tt}, b_{xx} and b_{tx} are the second-order partial derivatives of b and v_t is the partial derivative of v with respect to its second argument (time). Finally we end up with the following loss

$$\begin{aligned}\ell(v, w, (b, \dot{b})) = \frac{1}{P}\sum_{i=1}^{d}\sum_{j=1}^{h}\Big(\left| \dot{b}_{ij} + v_{ij} \cdot (\mathrm{D}\, b)_{ij} \right| + |\mathrm{D}\, v^1_{ij}| + |\mathrm{D}\, v^2_{ij}| + \lambda_{der} \left| \ddot{b}_{ij} + v_{ij} \cdot (\mathrm{D}^2\, b)_{ij} v_{ij} \right. \\ \left. + 2(\mathrm{D}\, \dot{b})_{ij} \cdot v_{ij} + (\mathrm{D}\, b)_{ij} \cdot (f_{ij}(v, w, (b, \dot{b})) + v_{ij} \cdot (Dv)_{ij}) \right| \Big),\end{aligned} \tag{16}$$

where D^2 is the discrete Hessian, $v \cdot Dv = (v \cdot \mathrm{D}\, v^1,\ v \cdot \mathrm{D}\, v^2)'$ and λ_{der} is an hyperparameter of the model. The Hamiltonian of the system is then consistently defined as

$$H(v, p, s) = \min_{w \in \mathbb{R}^N} \mathcal{H}(v, p, w, s) := \min_{w \in \mathbb{R}^N} \left\{ \ell(v, w, (b(s), \dot{b}(s))) + \sum_{i=1}^{d}\sum_{j=1}^{h} p_{ij} f_{ij}(v, w, (b(s), \dot{b}(s)) \right\}. \tag{17}$$

Determining the minimum of $\mathcal{H}$ in a closed form is infeasible, so we suggest to approximate this condition through a gradient descent approach on $\mathcal{H}$ performing M update steps of the form $w_{m+1}(s) = w_m(s) - \eta_w \mathcal{H}_\omega(v(s), p(s), w_m(s), s)$, where $\mathcal{H}_\omega$ is the gradient of $\mathcal{H}$ with respect to its third argument (the parameters of the network), before switching to the next frame and updating state and costate variables through the approximated Hamiltonian dynamic in equation 11 which in our current notation reads:

$$\begin{cases} \dot{v}_{ij}(s) = f_{ij}\left(v(s), w_M(s), s\right); & \text{(18a)} \\ \dot{p}_{ij}(s) = -D_x \mathcal{H}_{ij}(v(s), p(s), w_M(s), s)\,. & \text{(18b)} \end{cases}$$

Assuming temporal smoothness, we could expect the minimum of $\mathcal{H}$ to remain almost constant between consecutive frames so that, instead of performing a fixed amount M of steps, we may introduce a threshold on the average of $\sum_i \mathcal{H}^2_{\omega_i}$ to save computational resources when not needed.

As anticipated in the previous section, a serious issue arises when considering the boundary conditions $v_{ij}(0) = 0$, $p_{ij}(T) = 0$ that should come with the approximated Hamiltonian equations 18a, 18b. Our aim is to proceed forward in time, in conjunction with the temporal dynamic dictated by the external video stream, and nothing guarantees the fulfilment of the right boundary condition on the costate variables when initializing them with an arbitrary value at the beginning of the time horizon. To tackle this problem, we thus propose to actively *modify the input data* to enforce the fulfillment of the proper costate boundary condition, inspired by works in the direction of facilitating the learning process in neural networks via input modification (Marullo et al. (2021), Marullo et al. (2022)). Input modification should be enough relevant to be able to influence costate dynamic and, at the same time, enough delicate to not completely disrupt the original signal since optimality of the solution would be referred to the modified input. Moreover, it is worth nothing that we have to keep the dynamic of the modified input well separated from the one of the original dynamical system (15): we can not introduce any additional control variables steering the modified input in an optimal way since, otherwise, we should face supplementary boundary conditions for the costates associated

with these new state variables. Then, to implement input modification in the specific case under analysis, we settled for a classical online scheme as explained below.

First and foremost, let initialize the costate variables to be null at the beginning of the stream: $p_{ij}(0) = 0$. Input modification can then be realized through a second neural network G that takes as input the original frame $b(s)$ and whose weights at time s are collectively indicated with $\theta(s)$. The modified input β thus can be expressed as $\beta_{ij}(s) = G_{ij}(\theta(s), b(s))$. Having initialized the costate variables to be null, on way to automatically satisfy the boundary condition $p_{ij}(T) = 0$ would be to keep them constant along the entire horizon. We thus propose to determine the weights θ of neural network G by minimizing the following index:

$$\Upsilon(\theta) = \frac{1}{P} \sum_{i=1}^{d} \sum_{j=1}^{h} \left(\frac{1}{2} \left(b_{ij}(s) - G_{ij}(\theta, b(s)) \right)^2 + \frac{\lambda_{\dot{p}}}{2} D_x \mathcal{H}_{ij}^2 \right). \tag{19}$$

The fist term encourages modified frames to be close to the original ones according to the Euclidean distance, while the second term fosters the temporal derivative of the costate variables to be null. In essence, the scalar factor $\lambda_{\dot{p}}$ regulates the trade-off between input reconstruction and the approximate optimality of the solution. Let us note that $D_x \mathcal{H}_{ij}^2$, being function of the frames as expressed by equation 17, hides an implicit dependence on the weights θ. From now on we will assume that the inputs b and $\dot{b}$ used to predict the optical flow are replaced with their modified versions β and $\dot{\beta}$.

4.1 Experimental Proof of Concept

We are now going to analyze, through a practical example, the predictions of the proposed Hamiltonian learning paradigm of this paper. To this end, we consider a very simple input video $b_{ij}(s)$ depicting a squared gray-scale tile of size 150×150 pixels that is moving along four different directions with constant velocity and forming a closed squared-shaped loop. Each frame has a spatial resolution of 256×256 pixels and the background is uniform/black. The temporal resolution of the video is 20 frame per second (fps) for a total of 400 frames, 100 for each direction in which the square is moving. In all our experiments, we have considered 3 complete loops of the tile obtained by repeating the video, for a total of 1200 frames. The tile moves 1 pixel each frame and its motion is represented associating to the direction of the flow a certain hue, while its magnitude corresponds to pixels' value (Baker et al., 2007). For better visualization, we normalized the flow magnitude with its maximum value at each frame. In the previous sections we have considered the time variable as a continuous one and now we have to discretize the equations of interest 18a and 18b that describe the dynamic of state and costate variables. To this end, time is discretized according to the temporal quantization of the input stream, that is $s = k\Delta t$ where $k \in \mathbb{N}$ and $\Delta t = 1/\text{fps} = 1/20$. The discrete counterparts of $v_{ij}(s)$ and $p_{ij}(s)$, as well as all the other quantities of interest, will be represented through the use of an additional upper index k, that is $v_{ij}(s = k\Delta t) \to v_{ij}^k$ and $p_{ij}(s = k\Delta t) \to p_{ij}^k$.
To end up with numerically stable algorithms, the temporal derivative in the state equation 18a has been approximated with a single-step backward finite difference (backward Euler method), that is $\dot{v}_{ij}^{k+1} = (v_{ij}^{k+1} - v_{ij}^k)/\Delta t$. This usually results in an implicit numerical scheme that guarantees stability at the price of an higher computational cost. Nevertheless, in our specific case in which $f(v, w, s) = c(-v(s) + F((\beta(s), \dot{\beta}(s)), w(s)))$, we can invert by hand the resulting scheme obtaining a closed form update equation for v_{ij}^{k+1}

$$v_{ij}^{k+1} - v_{ij}^k = c\Delta t \left(-v_{ij}^{k+1} + F_{ij}((\beta^{k+1}, \dot{\beta}^{k+1}), w^{k+1}) \right) \to v_{ij}^{k+1} = \frac{1}{1 + c\Delta t} \left(v_{ij}^k + c\Delta t F_{ij}((\beta^{k+1}, \dot{\beta}^{k+1}), w^{k+1}) \right). \tag{20}$$

The corresponding scheme is stable whatever the value of c is. Costates' dynamic 18b has been instead quantized through a forward finite difference, resulting in the following update equation[4]

$$p_{ij}^{k+1} = p_{ij}^k - \Delta t \, D_x \mathcal{H}^k(v^k, p^k, w_M^k). \tag{21}$$

In the proposed practical implementation, neural network F has six convolutional layers with LeakyReLU activation functions and zero biases. The last layer has a linear activation, while convolutional filters have size 5 and the number of the feature maps is $(16, 32, 32, 32, 32, 2)$ over the 6 layers. Zero-padding is performed in order to keep the original spatial resolution of the input. The network G for input modification is a shallow one made up by just a single convolutional layer with filter size 3; we leave the analysis of more comple networks to future studies. Weights are initialized to be zero and there is no bias so that the initial output β_{ij}^0 of the network is null. Additionally, all the spatial derivatives in the Hamiltonian function have been approximated through central finite differences, while first

[4]The upper index k in $\mathcal{H}^k(v^k, p^k, w_m^k)$ represents the explicit temporal dependence on time in its continuous counterpart $\mathcal{H}(v(s), p(s), w_m(s), s)$.

and second-order temporal derivatives of the modified signal have been approximated through backward differences (see equation 17 and the definition of $\ell(v(s), w(s), (\beta(s), \dot{\beta}(s)))$ in equation 16). The workflow of the model is summarized in Algorithm 1.

Algorithm 1 Workflow of our Model

Initialize: State and costate variables are initialized to be null: $v_{ij}^0 = 0$, $p_{ij}^0 = 0$ for $1 \leq i \leq d$ and $1 \leq j \leq h$. Weights w of the neural network F are randomly initialized near zero. Weights θ of the second shallow neural network G are initialized to be zero. There are no biases for both the networks. With the chosen architecture for G, this makes the initial modified input null as well: $\beta_{ij}^0 = 0$.

Repeat until $0 < k < \lfloor T/\text{fps} \rfloor$**:**

1. Considering the current frame b^k, evaluate the modified frame $\beta_{ij}^k = G_{ij}(\theta^k, b^k)$ and, through finite difference approximations, all the required spatial and temporal derivatives to evaluate $\mathcal{H}^k$ (see equation 17 and the definition of $\ell(v(s), w(s), (\beta(s), \dot{\beta}(s)))$ in equation 16).
2. Evaluate the output of neural network $F_{ij}((\beta^k, \dot{\beta}^k), w_m^k)$ and the current state variable v_{ij}^k through the discretized Hamiltonian equation 20. Evaluate $\dot{v}_{ij}^k$ as $c(-v_{ij}^k + F_{ij}((\beta^k, \dot{\beta}^k), w_m^k)$ and the required spatial derivatives of v_{ij}^k as well through finite difference approximations. We can now compute $\mathcal{H}^k(v^k, p^k, w_m^k)$. Note that $m = 0$ during this first iteration.
3. Perform a gradient descent step on $\mathcal{H}^k(v^k, p^k, w_m^k)$ with respect the weights of F: $w_m^k \rightarrow w_{m+1}^k$. In our implementation, we have exploited an adaptive learning rate (Adam) and the corresponding optimizer is re-initialized each frame switch.
4. Considering the new weights w_{m+1}^k of network F, repeat steps 2 and 3 for a fixed amount of M iterations or until a threshold on the gradients $\mathcal{H}_\omega^k(v^k, p^k, w_m^k)$ is reached. Considering step 2, note that v_{ij}^k is evaluated only once during the first iteration $m = 0$. The final weights of F at temporal step k are indicated as w_M^k.
5. Update the costate variables according the discretized Hamiltonian equation 21. Update the weights θ of neural network G through a single gradient step on $\Upsilon(\theta)$ defined by equation 19.

The outcome of our experimental activity is reported in Figure 1 and Figure 2. We first investigate how the input gets modified and how motion is predicted in the considered stream. The first and second rows of Figure 1 report such information focusing on the second lap of the tile, thus when the input stream has been already processed for a while.[5] Input modification reduces mostly to an overall reduction of pixels intensities, and this is somewhat expected considering the chosen simple architecture of network G (see also the third row of Figure 2). Focusing on the second row of Figure 1, we can observe that the predicted flow direction is consistent with the one from the original video stream originating from the classical offline and non-continual learning based Horn and Schunck approach (third row). The ground-truth motion associated to the original video is also reported in the last row. This confirms that the model is learning to predict motion in a meaningful way, even if the modulus of the predicted flow has been found to be quite far from the ground-truth in the tested configurations (see also the first row of Figure 2). Comparing the second and third rows of Figures 1 we can also notice that the proposed model needs some more time to adapt to new directions with respect to the offline estimation. In Figure 2 we investigate the values of the quantities of interest in the considered setting, in function of different choices of the main hyper-parameters ($\lambda_{\dot{p}}$, c and λ_{der} in the first, second and last column respectively). We reported the average modulus of the predicted flow (first row), the temporal evolution of the cost function of equation 16 (second row), the largest pixel intensity of the modified frames (third row) and the average costate norm (last row). In the first three rows we also included, as red dashed lines, the corresponding reference values: the average ground-truth motion (first row) is around 0.34 in the considered stream; the largest pixel intensity (second row) is of course equal to 1; in case of a flawless motion estimation consistent with the brightness invariance condition of equation 13, the loss ℓ (third row) should be zero (neglecting the spatial regularization term); the values of the costate variables (last row) has not a clear reference value, since they are required to vanish just at the end of the considered time horizon. We can notice that, as expected, the more $\lambda_{\dot{p}}$ is higher, the more the effects of input modifications are stronger and pixel intensity gets reduced (third row - first column), while the costate variables increase in a slower manner (last row - first column). In this limit, modified frames are duller and, correspondingly, the loss ℓ is also smaller (second row - first column), while a less evident trend can be found in terms of predicted flow

[5] In this case, the chosen hyper-parameters are $c = 1e03$. $\lambda_{der} = 1e\text{-}02$, $\lambda_{\dot{p}} = 1e09$ and weights w_m^k are updated initializing (frame-by-frame) the optimizer with a base learning rate of $1e\text{-}05$ until a threshold of $3e\text{-}05$ on the average of $\sum_i \mathcal{H}_{w_i}$ is reached (usually with less than 10 iterations). In the worst cases, to facilitate convergence, the base learning rate is manually reduced by a factor 10 after the first 30 iterations. A minimum of 5 iterations is always performed up to a maximum of 60. The input is modified with a learning rate of $1e\text{-}02$.

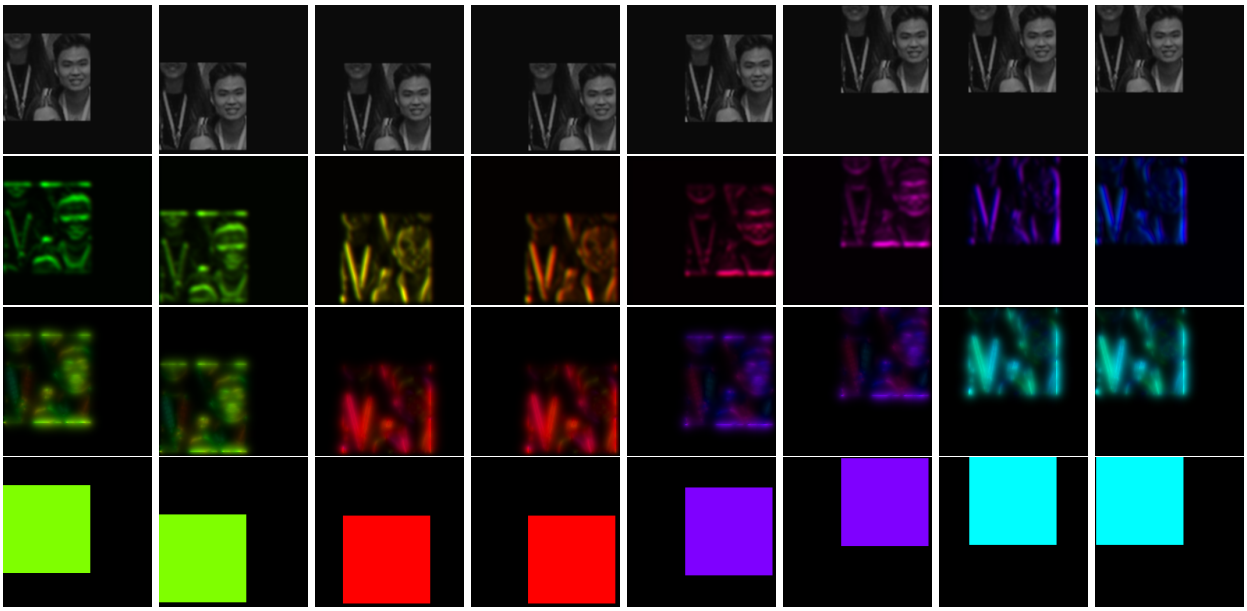

Figure 1: Some modified frames β^k along with the corresponding predicted optical flow v^k during the second lap of the tile (between frames 400 and 800). The last two rows show the Horn-and-Schunck and ground-truth motion from the original video stream, respectively. Motion is represented through classic color space encoding (Baker et al., 2007) (green: down; red: left; purple: up; light-blue: left)

(first row). A similar effect is also present considering the second column that is about c, when focusing on the second and third rows. Conversely, a larger c implies larger costate variables (last row - second column) that is the opposite trend with respect to the one observed for increasing $\lambda_{\dot{p}}$. Other than a smaller loss function ℓ, duller modified frames are also related to smaller predicted flows. This is particularly noticeable considering higher λ_{der} values as shown in the third column (first and third rows).

5 Conclusions and Future Work

In this paper we have approached the problem of continual online learning from a stream of data with the use of optimal control theory. The dynamic of the learning agent is expressed in terms of Hamilton equations for state and costate variables. From this standpoint, state variables correspond to the output of the model we aim to learn, while control variables stand for model parameters. State and control trajectories found by solving the control problem are optimal in terms of the overall considered time horizon. We argue that this might be clearly helpful in facing the issue of *catastrophic forgetting* that plagues existing lifelong learning agents. On the other hand, finding the optimal control solution requires the satisfaction of specific boundary conditions at the end of the time horizon, that can not be exactly fulfilled when proceeding forward in time in conjunction with the external stream of data. We proposed to face this issue actively modifying the external input to foster the constancy of costate variables for the proper fulfillment of the right boundary conditions, establishing in turn a trade-off between optimality of the discovered solution and how far the modified stream is from the original signal. Requiring the costates to have a trivial dynamic may be, in turn, too restrictive and different approaches could be more effective, such as for example forcing the vanishing of the costates only asymptotically through the help of additional degrees of freedom characterizing the dynamic of the network, the cost functional or the modified input. Nevertheless, we leave an in-depth analysis of this problem for future work, along with the study of other continual setting such as the task and class incremental scenarios. We applied the proposed ideas to the problem of unsupervised optical flow estimation from a given video stream. Qualitative results foster future investigations on the line of the presented approach. From this perspective, it could be interesting to control the temporal variation of the weights ($\dot{\omega}(s) = \beta(s)$) along with the addition of a simple regularization term in the cost functional $\frac{1}{2} m_\beta \beta(s)^2$ instead of controlling the weights themselves. As a result, at the cost of adding costate variables p_ω, we avoid the numerical approximation of the minimum of the hamiltonian involved in Algorithm 1. We leave the deepening of these aspects to further studies.

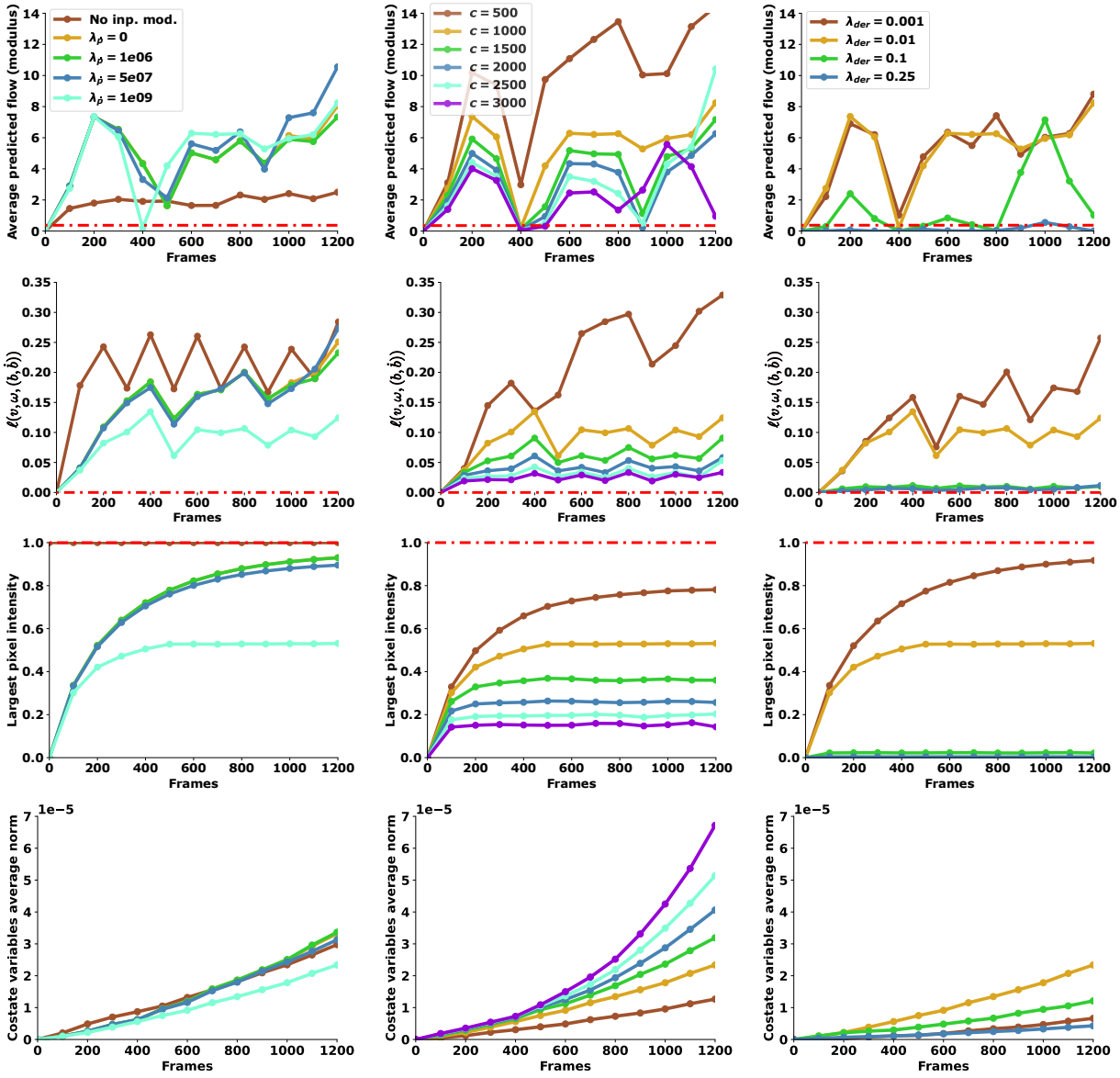

Figure 2: Evolution of various quantities of interest as function of time, measured in term of # frames. Along the first column, c and λ_{der} have been fixed to the constant value $c = 1e03$ and $\lambda_{der} = 1e\text{-}02$ while $\lambda_{\dot{p}}$ is varying. Along the second column, $\lambda_{\dot{p}} = 1e09$ and $\lambda_{der} = 1e\text{-}02$ with varying c, while in the last column $c = 1e03$, $\lambda_{\dot{p}} = 1e09$ and λ_{der} is changing. Red dashed lines in first three rows represent reference values in case of a flawless estimation of the motion of the original video stream. Since costates are always almost null on the considered time horizon (fourth row), the evolution of the hamiltonian is almost identical to the one of the cost function ℓ (second row) and, for this reason, the hamiltonian has not been reported.

ACKNOWLEDGMENTS

This work was partly supported by the PRIN 2017 project RexLearn, funded by the Italian Ministry of Education, University and Research (grant no. 2017TWNMH2) and by the French government, through the 3IA Côte d'Azur, Investment in the Future, project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

REFERENCES

Gilles Aubert and Pierre Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer Science & Business Media, 2006.

Simon Baker, Stefan Roth, Daniel Scharstein, Michael J. Black, J.P. Lewis, and Richard Szeliski. A database and evaluation methodology for optical flow. In *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007. doi: 10.1109/ICCV.2007.4408903.

Martino Bardi, Italo Capuzzo Dolcetta, et al. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer, 1997.

Alessandro Betti, Marco Gori, and Stefano Melacci. Cognitive action laws: The case of visual features. *IEEE transactions on neural networks and learning systems*, 31(3):938–949, 2019.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Lawrence C Evans. An introduction to mathematical optimal control theory version 0.2. *Lecture notes available at http://math. berkeley. edu/˜ evans/control. course. pdf*, 1983.

Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

Giorgio Gnecco, Alberto Bemporad, Marco Gori, Rita Morisi, and Marcello Sanguineti. Online learning as an lqg optimal control problem with random matrices. In *2015 European Control Conference (ECC)*, pp. 2482–2489. IEEE, 2015.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

Jeongho Kim and Insoon Yang. Hamilton-jacobi-bellman equations for q-learning in continuous time. In *Learning for Dynamics and Control*, pp. 739–748. PMLR, 2020.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.10.021. URL https://www.sciencedirect.com/science/article/pii/S0925231221014995.

Simone Marullo, Matteo Tiezzi, Marco Gori, and Stefano Melacci. Friendly training: Neural networks can adapt data to make learning easier. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Simone Marullo, Matteo Tiezzi, Marco Gori, and Stefano Melacci. Being friends instead of adversaries: Deep networks learn from data simplified by other networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

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

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Matteo Tiezzi, Stefano Melacci, Alessandro Betti, Marco Maggini, and Marco Gori. Focus of attention improves information transfer in visual features. In *NeurIPS 2020-34th Conference on Neural Information Processing Systems*, 2020.

Vladimir N Vapnick. *Statistical learning theory*. Wiley, New York, 1998.

A. Verri. Against quantititive optical flow. *Proc. First Int'l Conf. Computer Vision, London*, pp. 171–180, 1987. URL https://ci.nii.ac.jp/naid/20000149413/en/.

A. Verri and T. Poggio. Motion field and optical flow: Qualitative properties. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(5):490–498, May 1989. ISSN 0162-8828. doi: 10.1109/34.24781. URL https://doi.org/10.1109/34.24781.

Mingliang Zhai, Xuezhi Xiang, Ning Lv, and Xiangdong Kong. Optical flow and scene flow estimation: A survey. *Pattern Recognition*, 114:107861, 2021.