Candidate number: 244106

Supervisor: Dr Christopher Buckley

Computation in Physical Media: Investigating Biologically-Plausible Learning in Visco-Elastic Materials via Predictive Coding

---

Submitted to the Department of Informatics

University of Sussex

In Partial Fulfilment of the Requirements

For the Degree of Master of Science

23rd August 2022

This report is submitted as part requirement for the degree of Master of Science in Artificial Intelligence and Adaptive Systems at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Acknowledgements

I'd like to thank Chris both for the fascinating lectures in my first semester, and for all the inspiration and guidance over the course of this project. I must also thank Paul for all of our chats in your office. I am very grateful for all of the advice you offered me and I wish you all the best in your PhD. Thank you also to both my parents for supporting me through this year, and the WoW for all the chaotic good.

Abstract

The question of how life emerged on this planet is very much a fundamental one, and the ‘vast gulf’ between prebiotic chemistry and the simplest biology renders this a challenging problem to address (Walker & Davies, 2013). Recent evidence has shown that computation can be achieved in visco-elastic materials using a learning rule inspired by a biologically-plausible energy-based algorithm known as equilibrium propagation (Stern et al., 2021). However, we argue that this learning rule is limited in biological plausibility given the need for storing memory of previous network states, as well as the requirement for external supervision in defining the targets during training. We therefore use recent work by Millidge et al. (2022) in energy-based models to derive a new learning rule based on contrastive hebbian learning in predictive coding networks that is more biologically-plausible, and could conceivably be conducted fully-autonomously in a real physical system without the need for external supervision. We show that this learning rule, which we term generalised prospective configuration (GPC), is capable of learning desired network responses given target inputs. We argue that this learning rule provides tentative evidence for a simplified model for the way in which computation could plausibly emerge in organic materials.

Keywords: predictive coding, energy-based models, prospective configuration

Table of Contents

Page

1 Introduction

In the words of Hopfield, the reason that the biological world consists of physical systems that seem ‘entirely foreign’ to physics is because it is ‘physics plus information’ (Hopfield, 1994). Critically, Hopfield argues that the origin of this striking difference lies in the evolutionary value that has been placed in being able to predict the future in an external world that is both spatially and temporally heterogeneous. It is this emergent ability which is fundamental to life’s progression towards increasing computational capacity, crucially allowing organisms to overcome the cost of complexity (Seoane & Solé, 2018).

The study of the origins of emergent computation in physical substrates is a question that best fits the goals and motivations of artificial life research. The use of ‘emergent’ in this sense is the idea that certain properties of a system amount to more than the sum of their parts. The field of artificial life is concerned with the capturing of complex system dynamics by both identifying and recapitulating the fundamental underlying rules which govern such a system (Heiney et al., 2020). For instance, Langton (1990) identified three fundamental criteria that must be supported in a physical substrate in order for the emergence of computation to be achievable: these being the transmission, storage and modification of information. The paper then goes on to present evidence that a system whose dynamics are in the transitional phase between ordered and chaotic appears to be a key condition under which such ‘computational primitives’ are expected to be supported in a physical system. While identifying such criteria is important, the next step must be to find plausible paradigms by which learning can be achieved in a substrate. Such paradigms must form some kind of closed-loop system in which learning is completely autonomous and adaptive (Ma et al., 2018).

Work in morphological computation offers an interesting insight into the kind of primitive non-neural computation you can achieve in physical media. Morphological computation can be succinctly summarised as the idea that “certain processes are performed by the body that otherwise would have to be performed by the brain” (Pfeifer and Bongard, 2006). Computational tasks are essentially outsourced onto analogical physical systems (such as compliant soft bodies) in order to increase the efficiency of control.

This work has given rise to an interesting line of research which frames morphological computation as a particular branch of reservoir computing (RC). RC is a paradigm by which the dynamics of the analogical physical system (known as a ‘reservoir') work to project inputs into high-dimensional spaces, where separating the features then becomes a simple linear task (Seoane, 2019). This is therefore a cheap method by which continuous temporal data can be processed without the need for complex training algorithms. Recent work by Johnson et al. (2016) has taken this blurring between body and brain further, in which they use this paradigm to explore the information-processing potential of mechanical mass-spring-damper networks in performing cognitive tasks such as the active discrimination of shapes.

Recent research in the field of metamaterials and statistical physics has shown that computation can be achieved in visco-elastic materials, as modelled by mechanical networks. Stern et al. (2021) present a method for the automated design of materials with desired functionality through a process of learning in which the material adapts to external inputs in situ. They demonstrate that their ‘generalised coupled learning’ framework is capable of training physical networks to perform classification on a benchmark machine learning task, although as we argue later, this achievement has been somewhat inflated. Despite acknowledging biological neural networks as the inspiration for their local (and therefore scalable) learning, the work of Stern et al. (2021) has been framed very much in terms of a machine learning framework in which the minimisation of loss is discussed. However, we believe this work actually appears to have more resonance with biologically-plausible algorithms in neuroscience literature, emulating the work of energy-based models (EBMs) which have been shown to approximate backpropagation under certain conditions (Millidge et al., 2022). Generalised coupled learning is a novel formulation of such algorithms in the sense that the learning process utilises the fact that mechanical networks naturally minimise an internal energy function purely by virtue of their physical dynamics.

While Stern et al. (2021) maintain that their learning framework is autonomous and scalable, given its use of a local learning framework in which component parts of the system adapt to local information alone, we argue that this overlooks other hurdles towards autonomy. Training using coupled learning requires two important steps which limits the autonomy of this paradigm: first of all, there is a need for some form of external memory in order to store information about previous energy states of the system before a

learning update can be applied. Secondly, an external trainer is needed to clamp targets at particular values based on a supervised loss calculation. This second step additionally leads to another disadvantage in that it renders the system sensitive to noise. For these reasons, this learning rule presents distinct challenges for autonomous implementation in a real physical system, and is therefore a biologically-implausible method by which unsupervised learning could have emerged in physical substrates.

Given this strong link to biologically-plausible learning algorithms, we revisit the work of Stern et al. (2021) in light of the deep similarity we recognise between the equations of motion for mass-spring-damper (MSD) networks (used to model viscoelastic materials) and the dynamics of generalised predictive coding networks. We therefore propose a new learning rule for learning within MSD networks derived from contrastive hebbian learning in generalised predictive coding networks, which is more physically implementable and theoretically more robust to noise. We refer to this rule as generalised prospective configuration (GPC) and argue that this more biologically realisable rule offers a way in which unsupervised learning of environmental associations could have plausibly emerged in organic materials. This paper therefore seeks to replicate the work of Stern et al. (2021) and implement their generalised coupled learning rule for the task of learning desired responses to inputs in MSD networks, before going on to assess whether we can successfully implement our own learning rule, GPC.

Overall, we find that we can successfully learn desired responses in a model of visco-elastic materials (MSD networks) through the application of GPC. The process by which this is learned is more biologically-plausible than the generalised couple learning framework presented in Stern et al. (2021), adhering to the criteria for intelligence set out by Ma et al (2018) in which both a discriminative and generative model combine to become an efficient, stable and adaptive closed-loop systems capable of autonomous learning.

2 Hopfield Networks

Alongside the rapid evolution of feedforward models and the burgeoning field of classical machine learning during the 1980s, an alternative and more biologically-plausible class of models were proposed, spearheaded by the work of John Hopfield. Inspired by abstract thermodynamic systems in which an energy function is locally minimised by component parts, Hopfield proposed a type of recurrent artificial neural network in which one could encode memories as the stable low-energy states of the network (Hopfield, 1982). The nodes in the network take a discrete binary value (+1 or -1) and have weighted connections to other nodes within the network. Essentially, each of the nodes wants to change its states based on the sum of its own inputs. If the weighted sum is less than some threshold b, then the node becomes -1, and if higher it becomes 1. We can think of higher energy as reflecting inconsistency between the weighted connections of the network with the states each node would like to be in. Through the application of this simple (and crucially local) update rule, one can evolve the behaviour of the network over time and see it settle to a minimum energy state. Central to this model is the idea that, like the Ising model in physics, you can describe the system by its Hamiltonian function - that is, a value of the total energy of the system by summing the energy between all component variables (Brush, 1967). The Hopfield energy of a network is described by the following function:

$$E = -\frac{1}{2}x^T W x - b^T x \quad (1)$$

Where W is the weight matrix describing the connections between nodes x, b is the bias (threshold) and where $W_{ii} = 0$ (i.e. there are no self-loops in the network). Applying the update rule described above essentially amounts to performing gradient descent on this energy function, and this should agree with the update rule if we add in step size, η:

$$\frac{\partial E}{\partial x_j} = -\sum_{i \neq j} x_i w_{ij} - b_j ,$$

$$\frac{dx_j}{dt} = -\eta \frac{\partial E}{\partial x_j},$$

$$= \eta\left(\sum_{i \neq j} x_i w_{ij} - b_j\right) \quad . \qquad (2)$$

Hopfield's main idea was that we can set the weights in such a way so as to memorise a pattern in the low energy state such that inputting a partially complete pattern (i.e. initialising the the network with some part of the memorised binary string of node values) should lead to the network produce the most likely match from memory through application of the update rule. Hopfield termed this a content addressable memory (CAM), proposing a method by which you can fill in partial completions as well as perform error detection (Hopfield, 1982).

The process by which memories can be encoded within the network is done by setting the weights according to a learning rule. Setting the weight from node i to j involves running through all of the patterns and looking at the two different states of the nodes which that weight is between, and then adding up how many times they are in the same state, and how many times they are in opposite states, and then averaging across those. In other words, W_{ij} is the 'average coactivation' between nodes i and j. This should be reminiscent of the idea behind Hebbian learning: 'neurons that fire together, wire together' (Hebb, 1949). Whenever both nodes are active, it increases the weighting between them while also working to reduce the magnitude of the weights between nodes that are in opposite states.

Like the node state update rule, the weight update (learning) rule can also be derived from the energy function, by performing gradient descent on the energy function with respect to the weights:

$$\frac{\partial E}{\partial W_{ij}} = -\sum_{i \neq j} x_i x_j,$$

$$\frac{dW_{ij}}{dt} = -\eta \frac{\partial E}{\partial W_{ij}},$$

$$= \eta\left(x_i x_j,\ i \neq j\right) . \quad (3)$$

Since their inception, Hopfield networks have proved a useful framework with which to view memory storage and retrieval in the cortex (Rizzuto & Kahana, 2001). They have proven useful in optimization problems too, with early work showing that continuous Hopfield networks (where nodes are not constrained to binary states) can be applied to benchmark problems such as the travelling salesman problem (Hopfield and Tank, 1985). The power behind Hopfield networks comes from the analogue computation of non-linearly activated neurons and the interconnectivity between them. An analog computational system such as this can compute solutions rapidly owing to its ability to adjust many interacting variables at once (Jackson, 1960). This notion of Hopfield energy drives a lot of the theory behind the practice of unsupervised learning. We don't have an output, so we cannot compute a global loss which we can minimise. Instead we come up with this idea of energy, and this gives us something to optimise (Orchard, 2021).

3 Energy-Based Models

This early work into biologically-plausible learning within recurrent neural networks provided the foundations for a framework that has since been termed energy-based models (EBMs), in which the degrees of freedom within the system are optimised so as to find a minimum of a global energy function. The significance of these models is that they offer a way in which we might address the "credit-assignment problem" (Millidge et al, 2022). This concept comes from neuroscience and refers to the idea that learning in the brain needs to calculate the contribution that each neuron makes to the output behaviour, and then adjust the neuron's activity based on this assigned credit (or blame). This is very much a non-trivial problem in hierarchical recurrent neural networks given that neurons can only change their synaptic activity in response to interactions with their local connected neighbourhood (Richards & Lillicrap, 2019). In machine learning, this task is performed by a widely used algorithm known as backpropagation in which the gradient of the loss with respect to the weights is calculated at the output layer, and through application of the chain rule, is propagated backwards through the network (Rumelhart et al., 1986). However, this algorithm lacks biological-plausibility given that it demands that neurons be

differentiable, there be separate forward and backward passes during which weights can be copied, and weight updates based on non-local information (Shrestha et al., 2019). Interestingly, in a recent paper, Millidge et al. (2022) outline their theory in which they explicate the conditions under which EBMs have been shown to approximate backpropagation while satisfying the constraint of local learning, attempting to unify these largely disparate algorithms under a shared mathematical property.

Before going on to explore their findings and examine some different examples of EBM algorithms, we must first address an important question which might arise from examining this work in biological-plausibility: if we are able to implement non-biologically plausible methods in software (such as weight-copying) then why not? First of all, copying nature's use of local learning rules means that the approach is scalable while the time needed in order to calculate the gradients of some kind of global loss function increases significantly with network size (Stern et al., 2021). In addition to this, local rules have the benefit of not requiring information about the overall network topology, therefore leading to overall system level properties that we tend to see in biology such as fault tolerance and graceful degradation in place of the catastrophic failure that can be observed in artificial networks (Stork, 1989). While training recurrent networks (with cyclic structures) has in part been solved by backpropagation-through-time, this is largely restricted to hierarchical architectures, and as we know the mammalian cortex involves dense and highly recurrent connections in what is more of a heterarchical organisation (Lillicrap & Santoro, 2019; Avena-Koenigsberger et al., 2018). The topological flexibility of EBMs, on the other hand, has been demonstrated in recent work in predictive coding networks, an EBM algorithm which acts to minimise variational free energy. In a novel formulation termed *PC Graphs*, Salvatori et al. (2022) show that learning can be performed on arbitrary graph topologies, both for fully-connected and reduced networks. A final motivation behind examining the constraints that biology imposes (that is particularly relevant for our own interests in this paper) is for the purpose of studying plausible processes by which the emergence of computation and cognition could have conceivably come about in organic materials. It is necessary that such processes are topology-agnostic, unsupervised and simple and EBMs are therefore a compelling candidate framework in that they appear to satisfy such constraints.

The following section describes two examples of EBM algorithms before going on to explore Millidge et al.'s (2022) theoretical framework uniting many disparate algorithms under a general mathematical property of these models related to their free phase equilibrium.

3.1 Contrastive Hebbian Learning

As explored in the previous chapter, pure hebbian learning works to strengthen co-active neurons. However, a problem associated with this learning rule for continuous Hopfield networks (that is Hopfield networks whose neurons can hold continuous 'graded' values as opposed to the binary state neurons first explored by Hopfield) is that this can cause many of the weights to increase exponentially in a positive feedback loop (Hopfield, 1984). Contrastive hebbian learning (CHL) emerged to circumvent this issue, and works by updating the weights based on two different states of the network. The first phase (the 'free' state) is one in which the inputs are fixed at their value while the rest of the network is allowed to update to an equilibrium state. It is here that an anti-hebbian update rule is performed where connections between co-active neurons are made less probable. The second phase (the 'clamped' state) involves clamping the target nodes at their desired values and running to a second fixed point before performing a hebbian update rule where connections between co-active neurons are made more probable. The change in the weights is therefore,

$$\frac{dW}{dt} = \bar{x}\bar{x}^T - xx^T , \tag{4}$$

where $\bar{x}$ is the state of the nodes at the clamped state equilibrium and x is the state of the nodes at the free state equilibrium This process is somewhat intuitive as we are essentially strengthening the associations that are responding to the target in a correct manner, and weakening those responding incorrectly. It makes the clamped state more probable by lowering its energy while the free state is made less probable by increasing its energy. We are performing gradient descent with respect to the weights on the contrastive loss function, C, which is simply the difference between the standard Hopfield energies (Eq. 1) at the free state equilibrium E^F and the clamped equilibrium state E^C:

$$C = E^C - E^F . \quad (5)$$

3.2 Equilibrium Propagation

Equilibrium propagation (EP) was introduced as a novel method to train continuous Hopfield networks (Scellier & Bengio, 2017). It uses a learning rule based on CHL in which there are two phases (free and clamped) but rather than clamping the target to its desired value, we clamped the target at a nudged version of the free phase target value by a small amount towards the desired target. This nudge size is controlled by a hyperparameter λ.

In the free phase we are essentially minimising a global energy function (E) which could be Hopfield energy for instance, while in the clamped (or 'nudged') phase, we are minimising a different energy function $E + \lambda L$ where L corresponds to the supervised loss function in the output layer and is scaled by λ (where $\lambda << 1$). The update rule is defined as follows:

$$\frac{dW}{dt} = \frac{1}{\lambda}\left[\frac{\partial E^C}{\partial W} - \frac{\partial E^F}{\partial W}\right], \quad (6)$$

which amounts to the difference between the partial derivatives of the two equilibrium states with respect to the weights. This update rule converges to backpropagation as $\lambda \to 0$ (Scellier & Bengio, 2017).

3.3 General EBM Framework

Millidge et al. (2022) describe how this relationship to backpropagation in EBMs comes from the fact that the total energy function describing the system can be divided into two parts. We can describe two components: the output layer energy, L, and the internal energy, I:

$$E = I + \lambda L , \quad (7)$$

where the energy at the output layer is scaled by the parameter λ. In accordance with the principle of locality, this internal energy, I, can necessarily be further decomposed into local energies (for each layer if we are discussing a hierarchical network for instance). When we run the network to the free phase

equilibrium, $\lambda = 0$ as the output can vary freely without the actual target value having any impact on the dynamics of the network. Running the network to a fixed point requires gradient descent on the overall energy, and given that $\lambda = 0$, this means that $E = I$. Now if, from this point, we then clamp the targets to either a clamped or nudged-clamped state (meaning $\lambda > 0$ and the target affects the dynamics), the change in the node states is effectively the gradient of the supervised loss because the $\frac{\partial I}{\partial x}$ is zero at the end of the free phase equilibrium:

$$\begin{aligned} \frac{dx}{dt} &= -\frac{\partial E}{\partial x}, \\ &= -\frac{\partial I}{\partial x} - \lambda \frac{\partial L}{\partial x}, \\ &= 0 - \lambda \frac{\partial L}{\partial x}, \\ &= -\lambda \frac{\partial L}{\partial x}. \end{aligned} \tag{8}$$

Therefore, we can say that given the clamped state equilibrium is being found after being started from this first free phase equilibrium state, the dynamics of the nodes reflects the negative gradient of loss, thereby propagating this loss throughout the network. While the node states remain close to this free phase fixed point, the dynamics of the network should predominantly reflect the supervised loss and therefore approximate the loss gradients obtained in backpropagation. Millidge et al. (2022) termed this limit in which the clamped equilibrium converges to the free state equilibrium the *infinitesimal inference limit* and different EBM algorithms vary in the way in which they reach this limit.

4 Computation in Physical Networks

What is interesting about physical networks such as mass-spring-damper (MSD) networks, flow networks and variable resistor networks is that the system is set up in such a way that the physical degrees of freedom naturally change over time to minimise an overall energy function. For instance, in a MSD network, the position of each node adjusts in order to obtain a force balance and thereby minimise the total elastic energy of the entire system to ultimately reach an energy minimum (equilibrium state) (Stern

et al., 2021). Crucially, this process occurs locally, where each node is adjusting its positions in response to its local neighbourhood without the need for global information about the state of the entire network. This evolution towards a low energy state follows the dynamics of EBM models purely by virtue of the physics of the network.

4.1 Mass-Spring-Damper Network Dynamics

Let's examine the dynamics of a two-dimensional MSD network, a kind of physical network which has classically been used to model the visco-elastic behaviour of organic materials (Xu et al., 2018). These consist of a graph, $G(V, E)$, of point-masses (or nodes), $V = \{x_1, x_2, \dots, x_n\}$, connected by linear damped hookean springs, $E = \{s_1, s_2, \dots, s_j\}$. The nodes of the network are sparsely connected in such a way that each mass is only locally connected to its neighbouring nodes within the network (see Fig.1).

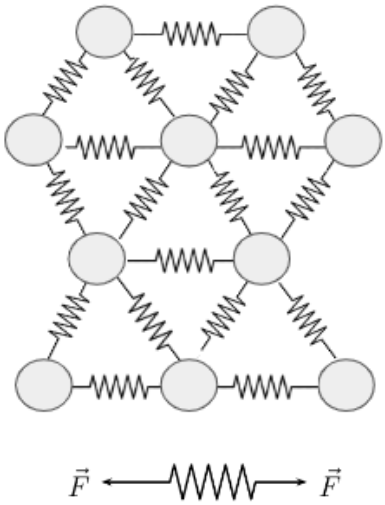


Figure 1: A mass-spring-damper network of size N=10, S=18. The masses are locally connected by massless linear springs with an associated equilibrium length, L_j, and spring constant, k_j.

Each spring therefore connects a pair of nodes and has an associated equilibrium length l_j and spring constant k_j. For the sake of simplicity, the effect of gravity and friction can be ignored, and we take each node to have a mass of one. The force of each spring is given by Hooke's law:

$$F = -k_j(r_j - L_j). \tag{9}$$

The acceleration acting upon each node n is therefore proportional to the sum over all points m connecting to that node of the stretch (or compression) of the connecting springs from their associated equilibrium lengths. There is also an added damping term in which the acceleration of the node is damped in proportion to the velocity of the node. The magnitude of this damping effect is controlled by the damping constant, c:

$$\ddot{x}_n = -c\dot{x}_n - \sum_m k_{nm}(r_{nm} - L_{nm}), \tag{10}$$

where r_{nm} is the euclidean distance between two nodes x_m and x_n connected by the spring.[1] The system can therefore be described in terms of the elastic energy of each spring j, and the total elastic energy is the sum of the energies of all component springs (Stern et al., 2021):

$$E_j = \frac{1}{2}k_j(r_j - L_j)^2,$$

$$E_{total} = \sum_{j \in springs} E_j. \tag{11}$$

As a result of this damping force, the dynamics of MSD networks evolve over time to minimise the total elastic energy of the network, for a network of j springs.

$$\frac{dx}{dt} = -\frac{\partial E}{\partial x} \tag{12}$$

[1] See Mocz (2021)

4.2 Generalised Coupled Learning

Stern et al. (2021), present a learning rule specifically for implementing within such physical networks as these MSD networks. While they very much introduce this general approach framed in a machine learning context, we notice that their work appears to have greater resonance with the EBM paradigm and neuroscience literature surrounding biologically-plausible forms of learning. The authors' generalised coupled learning (GCL) paradigm is very much inspired by the process of EP, and has simply been applied to the context of physically instantiated networks. The following section describes GCL and contextualises within general EBM theory (Millidge et al., 2022).

The learning process involves two sets of variables: the physical degrees of freedom (these are what naturally change in order to minimise a scalar energy function, so node positions in MSD networks), and the learning degrees of freedom (in MSD networks we can either take these to be the equilibrium lengths or the spring constants). Unlike EP with neural networks, in MSD networks, we clamp an input spring at a given strain rather than a node value, therefore the boundary conditions are defined on the edges of the network rather than the vertices. As in EP, we can obtain the free phase equilibrium by clamping the inputs at a given strain S_T and running to equilibrium. We must then store this energy state to be used in the weight update later. We next obtain the nudged clamped strain S_T^C by nudging the value of the target spring slightly away from the free state strain S_T^F towards the target strain S_T using the following equation:

$$S_T^C = S_T^F + \eta\left[S_T - S_T^F\right], \tag{13}$$

where η is the nudge size and is assumed to be small. Once these two energy states have been obtained, we then update the learning degrees of freedom based on the partial derivative of the contrastive energy function (which is the difference in elastic energy between the free state (E^F) and the nudged-clamped state (E^C). This update rule can be derived for either of the two learning degrees of freedom:

$$E_j = \frac{1}{2}k_j(r_j - L_j)^2,$$

$$\frac{dL_j}{dt} = \frac{\alpha}{\eta}\frac{\partial}{\partial L_j}\{E^F - E^C\},$$

$$= \frac{\alpha}{\eta}K_j\{\left(r_j^F - L_j\right) - \left(r_j^C - L_j\right)\}. \quad (14)$$

$$\frac{dk_j}{dt} = \frac{\alpha}{\eta}\frac{\partial}{\partial k_j}\{E^F - E^C\},$$

$$= \frac{\alpha}{2\eta}\{\left(r_j^F - L_j\right)^2 - \left(r_j^C - L_j\right)^2\}. \quad (15)$$

Despite the authors' motivations behind GCL, namely the engineering of materials with desired functionality without the need for external evaluation (as the material is capable of adapting autonomously in situ), their approach still requires significant external intervention. For one, an external trainer is needed to calculate the nudge-clamped value of the target springs (using Eq.13). The need for end-to-end supervision effectively makes this paradigm an open-loop system in which learning cannot be fully autonomous (Ma et al., 2018). In addition to this, some form of external memory is needed in order to store the energy of the system at the free state before a weight update is performed. Therefore not only do Stern et al. (2021) perhaps overstate the autonomy of their learning paradigm with respect to metamaterials design, but GCL also appears to fall short of the criteria that we are interested in for the purposes of this paper: a plausible method by which unsupervised learning could have arisen in prebiotic chemistry. While in a later paper, Dillavou et al. (2022) attempt to mitigate some of these shortcomings associated with the need for memory storage in GCL by simply having a set of identical "twin" networks, we argue that this remains a cumbersome approach to implement physically and continues to be biologically unrealistic. Given these challenges that GCL appears to face, we now turn to an alternative EBM, predictive coding, and explore how this framework may offer solutions to some of these limitations.

5 Predictive Coding

Predictive coding has emerged from neuroscience literature as a potential unifying framework of cortical functioning, in which the activity of the cortex is viewed as the product of the minimisation of error between incoming sensory data and the brain's predictions of these signals (Friston, 2005). In this way, prediction can be seen to essentially act as a natural unsupervised learning objective for the brain (Rao and Ballard, 1999). Predictive coding can be formulated as an energy-based model, and one which operates in accordance with the general EBM framework to approximate backpropagation at the infinitesimal inference limit (Millidge et al., 2022). Much of the literature surrounding predictive coding has been in relation to hierarchically organised neural networks (Whittington & Bogacz, 2017). However, recent work has shown that a generalised form of predictive coding approximates backpropagation along arbitrary computational graphs (Millidge, Tschantz & Buckley, 2022).

The energy function that is minimised in predictive coding networks is known as *variational free energy* is described by the following equation for a network of N nodes:

$$E_{PC} = \frac{1}{2}\sum_{i=1}^{N}\left(x_i - f(\sum_j W_{ij}x_j)\right)^2 , \quad (16)$$

in which the error ε_i is equal the term inside the brackets $(x_i - f(\sum_j W_{ij}x_j))$ and amounts to the summed weighted difference between the node value, x_i, and all connected nodes, x_j. In predictive coding, the change in the activity of the nodes over time amounts to the minimization of the partial derivative of the energy function with respect to the state nodes:

$$\frac{dx_i}{dt} = -\frac{\partial E}{\partial x_i},$$

$$= -\gamma\frac{\partial E}{\partial x_i},$$

$$= \gamma\,(-\varepsilon_i + f'(x_i)\sum_{k=1}^{n}\varepsilon_k W_{k,i})\,. \tag{17}$$

The weight update can be derived from gradient descent on the partial variational free energy with respect to the weights:

$$\frac{dW_{i,j}}{dt} = -\frac{\partial E}{\partial W_{i,j}},$$

$$= -\alpha\left(\frac{\partial E}{\partial W_{i,j}}\right),$$

$$= \alpha\,(\varepsilon_i\, f(x_j))\,. \tag{18}$$

It is worth noting that while both differential equations (Eq. 17 and Eq. 18) can run simultaneously, the time constant for the weight update is much larger than the one for updating the node values. This means that the earning is much slower than the physical dynamics of the network, therefore the nodes converge to equilibrium before the weight update is performed. This is known as the assumption of quasistaticity in learning and is also present in GCL (Stern et al., 2021).

5.1 Prospective Configuration

Millidge et al. (2022) show it is possible to formulate a CHL algorithm that takes in the variational free energy function from predictive coding networks in place for Hopfield energy. What is interesting about predictive coding networks is that the free phase equilibrium leads to the complete minimization of prediction errors, and therefore the gradient of the variational free energy with respect to the weights at the free phase is zero. This means that the contrastive hebbian update in predictive coding will simply be the gradient at the clamped state equilibrium:

$$\frac{dW_{i,j}}{dt} = \frac{1}{\lambda}\left[\frac{\partial E^C}{\partial W_{i,j}} - \frac{\partial E^F}{\partial W_{i,j}}\right],$$

$$= \frac{1}{\lambda}\left[\frac{\partial E^C}{\partial W_{i,j}} - 0\right],$$

$$= \frac{1}{\lambda}\frac{\partial E^C}{\partial W_{i,j}}. \quad (19)$$

Song et al. (2022) term this learning rule *prospective configuration,* given the fact that the updates consolidate the network's response to external stimuli by moving them towards the values that would have yielded the correct classification of the input.

5.2 Generalised Prospective Configuration

Interestingly, the equation for the variational free energy of a generalised predictive coding network looks strikingly similar to the energy function describing the elastic energy of a MSD network:

$$E_j = \sum_{j=1}^{S} \frac{1}{2} k_j (r_j - L_j)^2. \quad (20)$$

$$E_{PC} = \frac{1}{2}\sum_{i=1}^{N}\left(x_i - f(\sum_j W_{ij} x_j)\right)^2. \quad (21)$$

Given this similarity, we propose that we can apply the same logic by which the learning rule for predictive coding networks is obtained, but instead derive it from the elastic energy function, thus providing us with a new method for implementing learning within a network of MSD networks as a computational medium. We shall refer to this new learning rule as generalised prospective configuration (GPC).

If we take the partial derivative of this energy function with respect to the equilibrium lengths, *L*, we obtain the following update rule:

$$E_j = \frac{1}{2}k_j(r_j - L_j)^2,$$

$$\frac{\partial E}{\partial L_j} = K_j(r_j - L_j),$$

$$\frac{dL_j}{dt} = -\alpha\left(\frac{\partial E}{\partial L_j}\right),$$

$$= \alpha K_j(r_j - L_j). \quad (22)$$

We can likewise also take the partial derivative of the elastic energy with respect to the spring constants as an alternative weight update rule:

$$E_j = \frac{1}{2}k_j(r_j - L_j)^2,$$

$$\frac{\partial E}{\partial L_j} = K_j(r_j - L_j),$$

$$\frac{dk_j}{dt} = -\alpha\left(\frac{\partial E}{\partial k_j}\right),$$

$$= \alpha(r_j - L_j)^2. \quad (23)$$

5.3 Advantages of Generalised Prospective Configuration

There are a number of distinct advantages for implementing the GPC learning rule rather than GCL. First of all, GPC only requires the application of a clamped phase alone, without the need to first take the free phase state and then 'store' this information in some form of external memory whilst the clamped state is found. There is also no need for the element of external supervision needed to obtain a 'nudged' version of the target value based on a supervised loss; we simply clamp the targets at the desired value. Hence, learning within the GPC is both more plausible from the perspective of biological implementation while also being more computationally efficient.

Another advantage of GPC is that it should, in principle, be more robust to noise within the system. Stern et al. (2021) acknowledge the problems that would be associated with trying to perform GCL in a real implementation of a physical network. These reservations relate to the idea that if the nudge magnitude is very small, the difference between the free and clamped states of the networks will be very small and therefore this will lead to any noise in the system larger than these differences to significantly interfere with the learning process. The authors suggest that there is therefore a trade-off that must be made in the nudge size to ensure that the learning process is not dominated by noise but also yields an effective estimate of the supervised loss gradients. This tradeoff, however, is not present in GPC as the clamped state is normally far from the free state as it is fully clamped at the target value and not just a nudged version of the target. We therefore suggest that GPC offers a more succinct and reliable way of learning that naturally lends itself to noise robustness.This is important when actually attempting to physically implement a system like this as there will undoubtedly be noise introduced in measurements in the real world. Furthermore, this adds to making GPC a more promising candidate with respect to a learning rule that could plausibly have given rise to the kind of early computational primitives in organic materials.

6 Methods

We therefore propose to replicate the results of Stern et al. (2021) before attempting to implement GPC as an alternative learning rule for computation in physical networks.

6.1 Modelling Mass-Spring-Damper Networks

The recurrent physical networks used in our investigations are two-dimensional mass-spring-damper (MSD) networks (see Appendix A.1 for details). In an attempt to loosely replicate the spatially connected patterns of disordered flow networks in the original paper, the nodes of the network are sparsely connected in a triangular lattice pattern (see Fig. 2). It is possible to simulate this system adhering to these physically realistic dynamics in which we can calculate the acceleration upon each node and integrate over time to find the velocity and again to update the node positions at the new time step (see Sec. 4.1). However, for our simulations we chose to employ an alternative method. We note that the evolution of the damped system over time amounts to gradient descent on the partial derivative of the energy function

with respect to the physical degrees of freedom (node positions). Therefore, rather than evolving the system in accordance with the physical rules governing the system, we can instead treat the total elastic energy of the network as a loss function and perform gradient descent on the positions of the nodes, thereby ignoring the need for damping. While the gradient descent may not be exactly physically realistic, it is a formally equivalent process which still finds the minimum energy (equilibrium) state of the network (see Fig. 3a). With the use of automatic differentiation packages such as PytTorch, this process is comparatively computational efficient.[2] This method was employed in a recent paper describing a soft-body physics simulator for the purpose of investigating deep reinforcement locomotion controllers (Rojas et al., 2021).

6.2 Clamping

In order to clamp a spring at a particular value, we first calculate the length we need to make the spring (in relation to its equilibrium length) such that it has a desired strain. The strain of a spring is given by the following equation:

$$S_j = \frac{r_j}{L_j} - 1 . \quad (24)$$

This can be rearranged such that the new length is given by:

$$r_j = L_j(S_T^C + 1) , \quad (25)$$

where S_T^C is the target (clamped) strain value. We then insert a solid rod of this length into the network which cannot move (this can be done by setting a spring whose length matches its equilibrium length, and has very high spring constant,[3] k_j), meaning that the energy needed to move the associated nodes at either end of the rod is very high, therefore, during gradient descent on the energy function the nodes do not

[2] This automatic differentiation of the energy function was performed using the Adagrad optimizer in the PyTorch package with a learning rate of 0.1 in order to run the network to equilibrium.

[3] The value of k used for clamping target springs in this method was 1e10

move (see Fig. 2). The strain of the target spring can then remain constant during the inference phase (see Fig. 3).

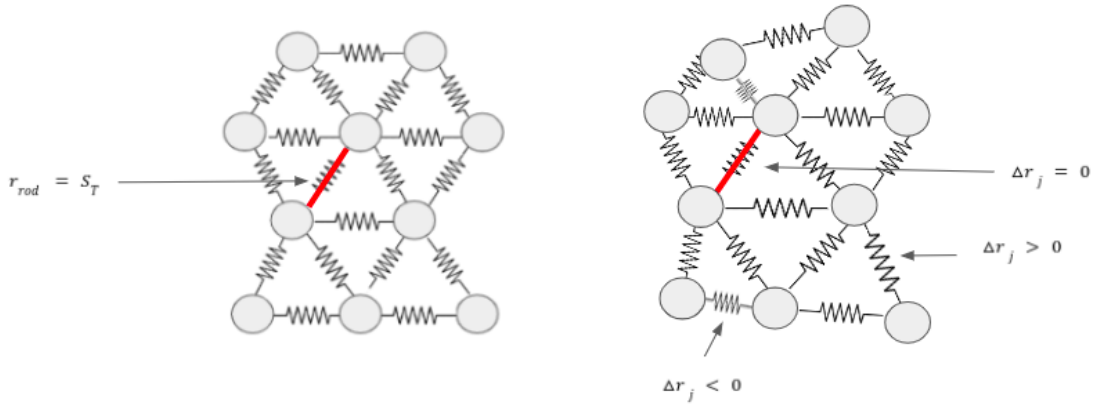


Figure 2. a) A MSD network in which the rod (in red) has clamped the target spring to a desired strain, S_T. b) When all other node positions are allowed to vary freely during inference (meaning that the spring lengths can either be compressed or stretched) there will be no change in length of the clamped spring.

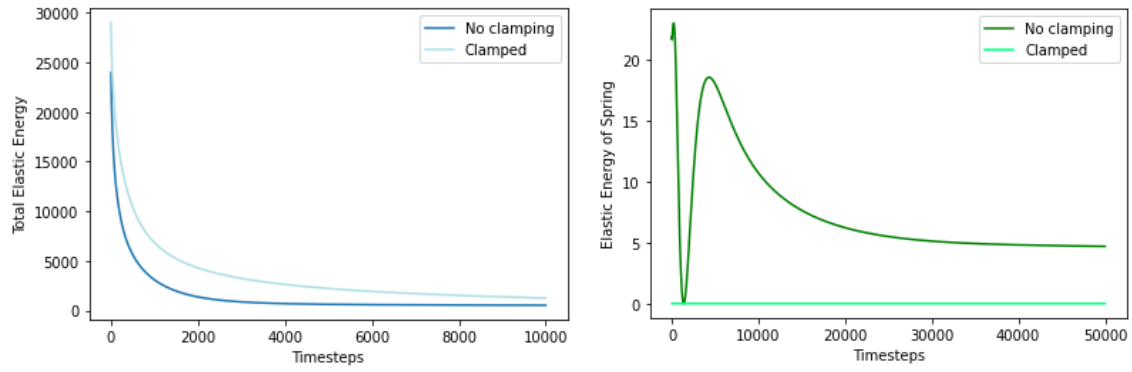


Figure 3: a) The total elastic energy of an MSD network (N=10, S=18) as a function of time (dt = 0.1) as the network is run to equilibrium via gradient descent on the partial energy with respect to the node positions. The graph compares the total elastic energy when one spring is clamped at a given length versus no clamping, showing the global minimum for the clamped network is higher given that one of the springs is not allowed to adjust its length to

locally minimise elastic energy. b) The elastic energy of a single spring of that network as a function of time. The energy settles to a local minimum in the context of its neighbouring springs when it is not clamped, while the elastic energy of the clamped remains constant as it cannot change length.

6.3 Supervised Classification in Stern et al. (2021)

Stern et al. (2021) claim to successfully perform a benchmark machine learning classification task (the classification of images of handwritten digits using the MNIST dataset) with the use of their coupled learning rule. However, this somewhat overstates the complexity of the task actually performed for the following reasons. It should be noted that the paper drastically simplifies the task by not only reducing the training data to only zeros and ones (which importantly are the most very different shapes and therefore intuitively should be easily distinguishable), but the authors also perform principal components analysis (PCA) on the data prior to training. The purpose of PCA is to reduce the dimensionality of input data (Howley et al., 2005). For example, each image in the MNIST dataset is 28 x 28 pixels, and therefore has a total of 784 dimensions. PCA is based upon the foundational idea that high-dimensional data can often be re-represented in a much lower dimensional code, provided that the data lies near a linear manifold in high-dimensional space. If such a manifold can be found, we can essentially express each datapoint in terms of where it lies on this manifold, significantly reducing the dimensionality. Importantly, this process of re-representation to a lower dimensional subspace should not cause too much information about the original datapoint to be lost given that by necessity the data varies little in the directions orthogonal to the manifold (Hinton, 2017).

Stern et al. (2021) state that they take the top 25 principle components of the MNIST dataset. To put the power of this technique into perspective, we found that using the top 25 PCs, we could train a very simple linear support vector regression model to an accuracy of 98.2%, while using only two principal components, we achieved an impressive 99.1% (see Table 1). One could argue that this preprocessing step largely takes the wind out of the fact that such networks were able to perform MNIST classification.

Dimensionality of data (as reduced by PCA)	SVR Accuracy
2	0.991
10	0.980
25	0.982

Table 1. Accuracy of a linear support vector regression (SVR) trained on 50 training examples for different numbers of principal components of the MNIST dataset.

Another limitation to the performance of the coupled learning rule in Stern et al (2021) with regards to supervised classification is the vast number of training iterations needed to achieve what we have determined to be a task which is better defined as a simple regression rather than classification. Each epoch consists of 100 iterations of the learning rule with a different training “image” (set of 25 PCA components for a digit) presented at each iteration. The experiment runs over 5000 iterations, which is therefore 50,000 iterations of the learning rule for an entire training cycle. This is a significant amount of time given that, with increasing size and (depending on the architecture of the network) the time for a physical network to converge to equilibrium (the relaxation time) increases (Fancher & Katifori, 2021).

6.4 Allostery

We therefore attempt to replicate the initial and arguably most simple task undertaken by Stern et al. (2021): that of learning allosteric responses within a network. The concept of allostery arises from biology, referring to the regulatory phenomenon by which the binding of a molecule locally to a site on a macromolecule (commonly a protein) can have an effect on the activity of that macromolecule at a distal site. This is usually caused by the coupling of conformational changes in the protein (Motlagh et al., 2014).

The mechanical equivalence to this kind of functionality can be achieved in MSD networks through the clamping of a ‘source’ spring at a desired strain, which should then yield the corresponding desired strain at a distant ‘target’ spring. Prior work into the designing of allosteric responses within MSD networks has been conducted by Hexner et al. (2020), in which the authors employ a supervised learning approach and

show that through the application of certain stresses (the stretching and compressing of springs), desired responses at distant sites can be achieved. Stern et al. (2021) show that they can produce allosteric responses in both mechanical and flow networks through the application GCL. They randomly allocate 10 input strains to 10 random source springs, and then randomly allocate 3 springs to be the targets (with associated desired target strains). They demonstrate that they are successfully able to learn these desired responses in networks of size 512 and 1024 by either modifying the equilibrium lengths or the spring constants, with the equilibrium length based learning yielding the most consistent results (with less variation in error reduction during training).

We therefore implement a scaled down version of this biologically inspired task on smaller networks with fewer source and target springs, and attempt to replicate the original paper's findings using GCL on the spring equilibrium lengths given that they had the most success using this choice of learning degree of freedom, before applying GPC.

7 Experiments

7.1 Learning Allosteric Responses with Generalised Coupled Learning

We test the coupled learning framework on MSD networks of size N=10 and S=18 from a fixed initialised starting position with random gaussian noise added to the node positions for each independent trial. The details of the MSD network implementation in Stern et al. (2021) is much more sparse than what is given in the original paper for GCL on flow networks. Therefore, given that the paper shows that the learning rule is successful with both uniformly initialised and randomly initialised learning degrees of freedom (which for the elastic networks is both the equilibrium lengths, L_{ij}, and the spring constants, k_{ij}), we choose to implement the with randomly initialised L and k to show the generality of the learning framework.[4] In order to simplify our experiments, we randomly chose just one source spring and one target spring (compared to the 10 source and 3 targets in the original paper). The corresponding strain of the source spring is chosen randomly from a Gaussian distribution $S_S = N(5, 1)$ and as in the original

[4] These were initialised to take random values between 0-20 and 0-50 respectively.

paper, we subsequently chose a random target spring strain scaled by the source strain.

$S_T = N(5,\ 0.2\sum S)$. We measure the difference between the actual target spring output and the desired target value using the following error function:

$$C = \sum_T \frac{1}{2}\left[S_T^F - S_T\right]^2, \tag{26}$$

where S_T^F is the strain obtained in the free state equilibrium, while S_T is the target strain value we want to obtain.

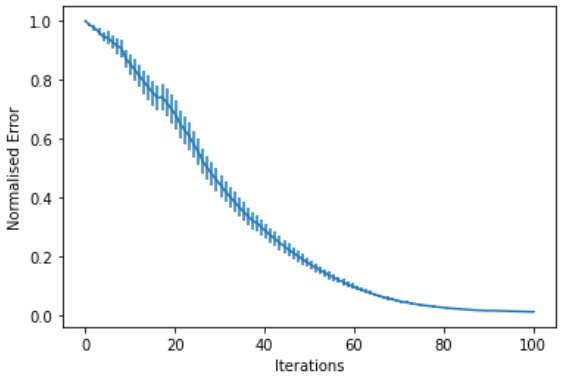


Figure 4: Training MSD networks using GCL modifying the equilibrium lengths. The mean average error (scaled by the initial error value) of 10 independent trials is shown as a function of learning rule iterations.

Fig. 4 shows the mean average error of 10 independent trials of the GCL in MSD networks, and demonstrates the success of GCL in learning allosteric responses. The error, C was reduced by orders of magnitude during the training process and in all trials but one the normalised error converges to zero (see Fig. A in appendix). Despite their being no evidence of this kind of clarification in Stern et al. (2021), we chose to make sure that the responses that had been learned were not of a trivial kind in which the target spring simply learns to be a certain strain regardless of the input strain (essentially reflecting a complete decoupling between the two springs within the network). As shown in Fig. 5, the allosteric response

learned in this example trial was indeed a non-trivial coupling, as reflected in the fact that when we change the input strain in the 'trained' network, the target strain indeed changes too. This was apparent for all trials (see Fig. B in appendix A.2).

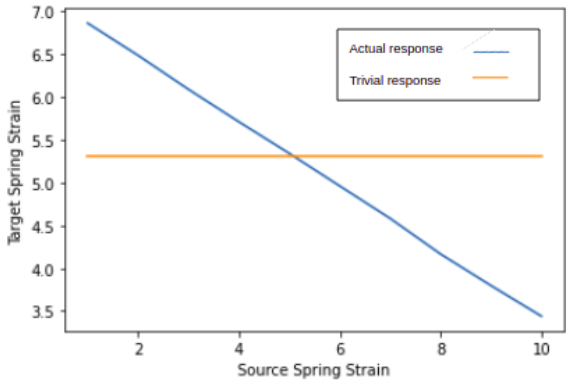


Figure 5: The change in target spring strain with the application of different source strains (in blue) versus the expected target spring response if the learning was of a trivial 'decoupled' kind (in orange) for a single independent trial.

7.2 Learning Allosteric Responses with Generalised Prospective Configuration

The next step was to repeat the above experiments exactly, but implement generalised prospective configuration rather than GCL.

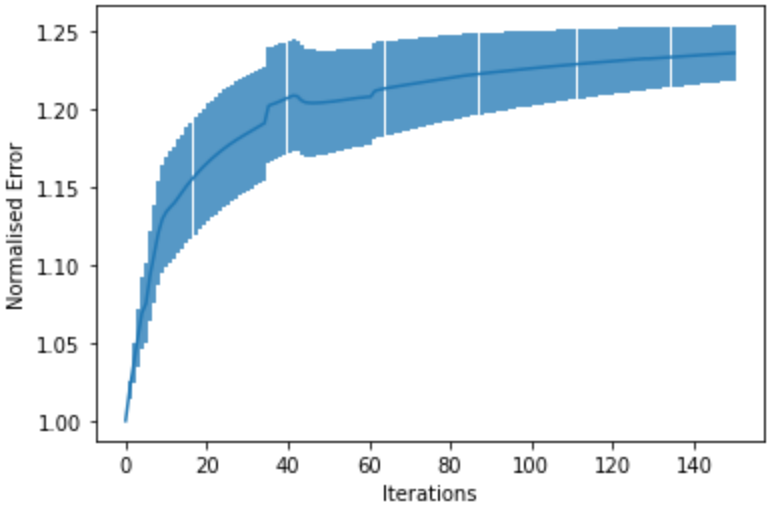


Figure 6: Training MSD networks using PC modifying the equilibrium lengths. The mean average error (scaled by the initial error value) of 10 independent trials is shown as a function of learning rule iterations.

We notice that the performance of the inference learning rule is very poor, and upon first examination, the network does not appear to be learning anything. The error increases on average, while the standard deviation is very large reflecting a large variation in trajectories for the different trials (see Fig. 6). We therefore investigate this issue in order to attempt to understand the failings of the learning rule. As shown in Fig. 7, we first confirm that the total energy and the partial energy with respect to the equilibrium lengths is correctly being descended upon. This was apparent for all trials (see Fig. C in appendix A.3).

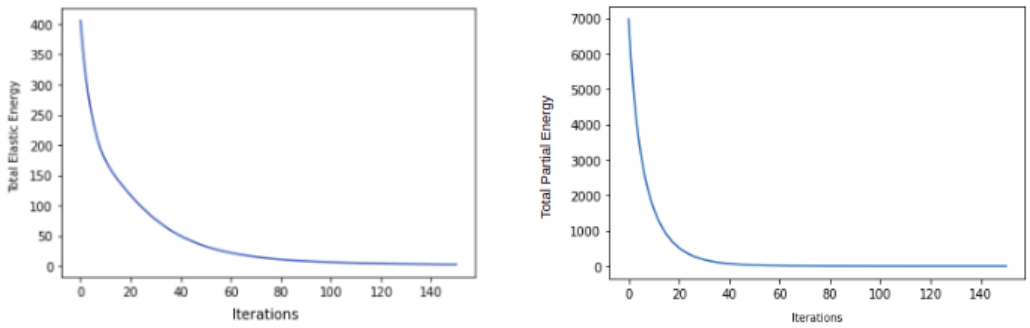


Figure 7: a) The total elastic energy as a function of training iterations and b) partial elastic energy with respect to the equilibrium lengths as a function of training iterations for a single independent trial.

Next, we examine the spring lengths during the training process and notice that the spring lengths do change over time to become nearer to the length of the clamped rod (and therefore length of the target). If we plot the normalised distance of the target spring length to the target clamped value, we see that this new error reduces by around 50% (see Fig. 8). This is a great improvement to the strain-based error we see in Fig. 6.

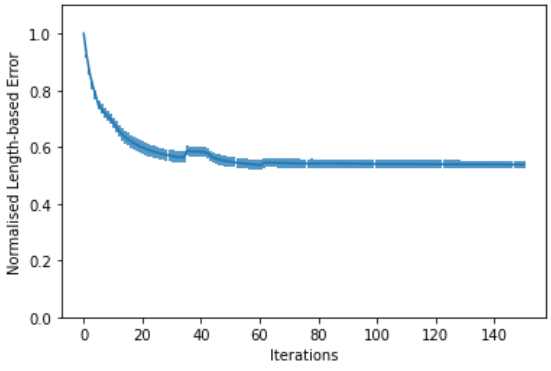


Figure 8: The mean average distance of target spring length from clamped target spring length (scaled by the initial distance value) of 10 independent trials is shown as a function of learning rule iterations.

Finally, we investigate what is happening to the equilibrium lengths of the target spring during the training process. Interestingly, if we plot the distance between the equilibrium length of the target spring and the value of the strain (i.e. the clamped length) of the target spring, and then scale this by the initial distance for each trial, this error converges to zero (see Fig. 9). It appears as though the learning of particular strains within the network cannot be achieved using GPC, however effective learning of desired equilibrium lengths can be accomplished (Fig. 9), and by virtue of this, some partial learning of the spring lengths (Fig. 8). With the application of this learning rule, the equilibrium length of the target spring simply learns to match the length of the target (in the clamped state). As we did with the networks trained with GCL, we confirmed that this coupling was non-trivial for both the lengths and the equilibrium lengths for all trials (see Fig. B appendix A.3).

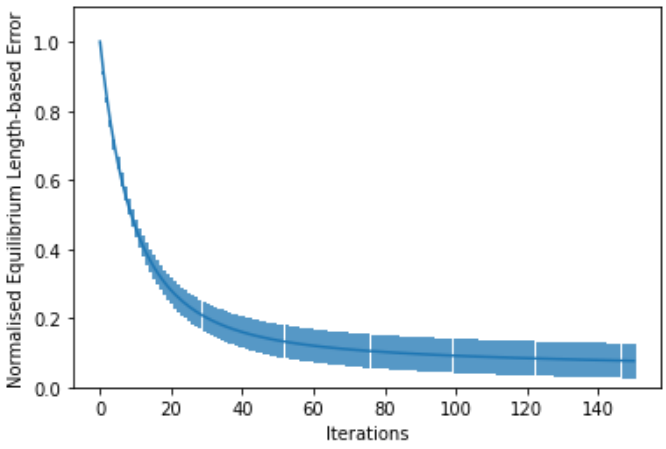


Figure 9: The mean distance of the target spring equilibrium length from the clamped equilibrium length value (scaled by initial distance) of 10 independent trials is shown as a function of learning rule iterations.

We therefore re-examine our prospective configuration learning rule in order to gain some kind of intuition as to why that might be. First of all, we note that our method of clamping in the GPC framework, at no point does the induced strain on the target node have any effect on the network, but only the length of the clamped rod. The reason for this is based on the fact that the concept of strain reflects the distance that the spring is stretched or compressed from its equilibrium length. The whole idea of clamping (or

fixing) a strain is in part problematic, as the introduction of a rod of a fixed length cannot reflect a strain, only a length. We maintain the idea of a strain, as the rod is designed to be set at such a length that it pushes or pulls the target spring in relation to its equilibrium length such that it is set to its desired strain. However, given that during GPC this fixed rod length remains unchanged throughout the learning process, if the target spring updates its equilibrium length then the induced strain changes too. The problem with the fact that GPC has no free state equilibrium in which the target spring can freely vary means that at no point does this 'induced' strain actually have a chance to have an effect on the network dynamics, it is only ever the target length of the clamped rod.

If we examine Eq. 23, we see that in order to reduce the partial energy with respect to the equilibrium lengths, we would want to reduce the value inside the brackets ($r_j - L_j$) which can be done by bringing the equilibrium length towards the actual length. It therefore makes intuitive sense that the network would learn to simply match the equilibrium length the target springs to their clamped length of the target, converging the sum of the values inside the brackets to zero and reducing the partial energy. Therefore, the equilibrium length of the target spring simply learns to be the length that the spring gets clamped at. This means that we can never induce a gradient that points in the direction of a desired strain, but can only point in the direction of the desired length - therefore with this method it is not feasible to learn specific strains. Unless, of course, we do not allow the target spring to learn, thereby fixing its equilibrium length, and meaning any changes in its length induced by hidden spring changes will therefore have an effect on its strain. We therefore repeat the above experiments in which we do not allow the updating of the target spring during the weight update phase.

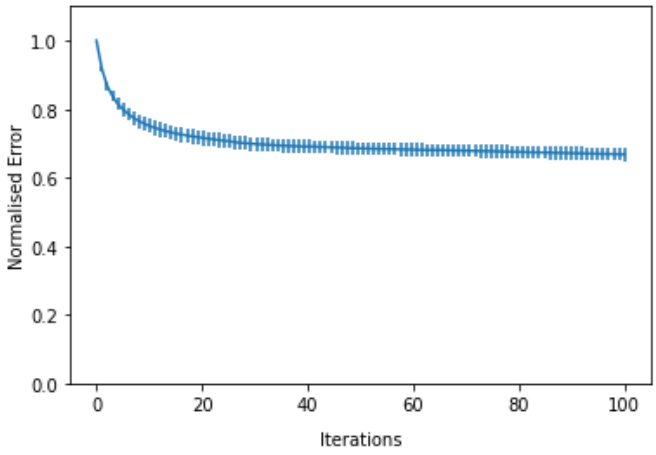


Figure 10: The mean normalised strain-based error for 10 independent trials as a function of learning iterations in which the target spring is not allowed to learn during the learning process.

As shown in Fig. 10, we achieve a reduction of the (strain-based) error - i.e. the learning of a desired strain - however, this only improves the target strain on average 33% towards the desired strain. This is because the rest of the network is working off length based gradients of the target spring and not strain based gradients.

How, then, is it that GCL is able to learn strains effectively? To answer this question, we turn back to the general EBM interpretation of GCL as outlined in Millidge et al. (2022). In the GCL, the spring length and spring strain are allowed to freely vary, as we are updating the equilibrium lengths based on a supervised loss (the difference between the energies in the free and the clamped state). As in EP (see Sec.3.2), the free state provides a basis, and then the nudge essentially allows us to "point" in the direction of the gradient of the supervised loss which in our case is based on the difference between the target and actual strain. This nudged length (obtained from rearranging the equation for strain, $S_j = \frac{r_j}{L_j} - 1$ which becomes $r_j = L_j(S_T^C + 1)$ will change throughout the course of the learning process as the equilibrium lengths change (see Eq. 13). One can therefore see with our method of clamping how coupled learning can effectively provide strain-based gradients for the network to then

learn with, as the free state allows the current strain of the target node to have an influence on the network, while the inference process is only able to provide length-based gradients.

8 Discussion

Overall, we find that we have successfully replicated the results of Stern et al. (2021) in learning allosteric responses within MSD networks. We further find that through the derivation of a new learning rule based around adapting the principle of performing contrastive hebbian learning in predictive coding networks to MSD networks, we can successfully achieve learning within such networks. However, we find that we are unable to learn strain-based responses by virtue of physical constraints within the concept of clamping inputs. Instead, we find that through the application of our learning rule, we are able to successfully learn a desired equilibrium length, a property related to but not equivalent to spring strain. It is important given these results that we ask whether this limitation in strain-based learning is actually significant. We are still successfully learning some kind of computational coupling between the input and the output spring. We therefore argue that the kind of learning that we can achieve is still very much of computational importance. However, further investigation into this limitation is certainly warranted.

One important limit in the generalisability of our work is that so far we have just explored the performance of the learning rule in relation to small networks (of 10 nodes). We note that given that we have taken the quasistatic limit of generalised prospective configuration, the effect of network size of learning size is likely to differ for larger networks. This is because the time for a large network to relax to an equilibrium will increase as the size of the physical networks increase (Fancher & Katifori, 2021). As discussed by Stern et al. (2021) in relation to GCL, the time for physical networks to relax to equilibrium does not scale with the amplitude of the perturbation but is set by the speed of sound. It is important that stable equilibrium states are reached in order to allow time for gradients to be propagated through the entire system before learning is consolidated based upon such gradients. This means that the speed of learning is not necessarily something that can be addressed by increasing the learning rate as higher learning rates would lead to the breaking of this quasistatic assumption. We therefore propose that our GPC learning should be tested on larger networks in order to confirm that it is capable of scaling. The fact

that the learning paradigm is based on local learning rules does suggest however that it is likely to scale given previous literature surrounding local biological-plausible learning (Salvatori et al., 2022).

Given that this work was conducted in simulation, and that we are studying this learning rule with a view to gaining insight about plausible learning that could occur spontaneously in real physical substrates, it is worth discussing the experimental considerations for real physical implementation of GPC. For this paradigm to work, we need classes of materials which can adapt their properties (e.g. their stiffness) in response to environmental strains, such as ethylene vinyl acetate (Stern et al., 2021). With a mind to exploring real physical implementation, it would be interesting to study the effect of noise on the effectiveness of the learning rule in simulation. Given its theoretical propensity for noise robustness (as discussed in Sec. 5.3), we would expect GPC to perform much better than GCL with the introduction of noise into system measurements. The limits of this robustness would therefore be an interesting and lucrative avenue of further research.

The work in learning allosteric responses is computationally primitive compared to the kinds of complex computations that have emerged in biology. We therefore would like to see whether MSD networks are capable of learning more complex functionality with GPC (for instance, learning multiple input-output functions). The next logical step to take would be to see if we can learn simple boolean logic operations (logic gates) within these networks, before going on to see if we can perform discriminiative tasks such as supervised classification. Furthermore, given the fact that MSD networks are necessarily symmetric in the way in which the response of the target spring should be proportional to the input at the source spring, this in theory should lead the networks to have generative capabilities. We can ask whether reversing the process and clamping target springs while allowing the source spring to vary freely leads to the production of a source spring response reflective of that class. Indeed, this is a property of predictive coding networks from which GPC is derived. Evidence has shown that the distribution of weights in a restricted boltzmann machine (another EBM which learns probability distributions of inputs) can be key to changing the weighting of the model from a generative towards a classifying ability (Larochelle et al., 2012). This adjusting of the weight distribution acts to appropriately combine the discriminative and generative training objectives in order to yield a model capable of classification. The effect of the

distributions of the learning degrees of freedom (k and L) within the network is therefore something also to be explored with relation to both the discriminative and generative performance of the network.

9 Conclusion

We believe our results may shed the kind of learning paradigm that might be required for the emergence of non-neural computation in physical media, namely visco-elastic materials, although further research is certainly required to confirm the generalisability of our proposed rule. Given the fact that the targets are clamped at their desired values, and not a nudged version abstracted away from the original input, it is therefore conceivable that a material which is simply being exposed to stresses over time and then deforming according to our learning rule, might end up learning simple associative responses. We suggest our results provide an interesting basis for further investigation, and hope that this may spur on further research investigating plausible learning rules by which computation could organically emerge in physical substrates.

References

Avena-Koenigsberger, A., Misic, B. & Sporns, O. (2018) Communication dynamics in complex brain networks. *Nature Reviews Neuroscience*, 19(1), 17–33.

Brush, S. G. (1967). History of the Lenz-Ising Model. *Reviews of Modern Physics*. 39 (4): 883–893.

Dillavou., S., Stern, M., Liu A. J. & Durian, D. J. (2022). *Laboratory Demonstration of Decentralized, Physics-Driven Learning,* arXiv:2108.00275v4 [cond-mat.dis-nn]

Fancher, S. & Katifori, E. (2021). Tradeoffs between Energy Efficiency and Mechanical Response in Fluid Flow Networks, arXiv:2102.13197v1 [physics.bio-ph]

Friston, K. (2005). A theory of cortical responses. Philosophical Transactions of the Royal Society B: Biological Sciences, 360(1456), 815–836.

Hebb, D.O. (1949).*The Organization of Behavior*. Wiley & Sons.

Heiney, K., Tufte, G. & Nichele, S. (2020). *On Artificial Life and Emergent Computation in Physical Substra*tes. arXiv:2009.04518v1 [q-bio.NC].

Hexner, D., Liu, A., & Nagel, S. (2020). Periodic training of creeping solids. *Proceedings Of The National Academy Of Sciences*, *117*(50), 31690-31695.

Hinton, G. (2017). “From PCA to autoencoders” [video file]. University of Toronto. Retrieved from https://www.youtube.com/watch?v=PSOt7u8u23w.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings Of The National Academy Of Sciences*, *79*(8), 2554-2558.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings Of The National Academy Of Sciences*, *81*(10), 3088-3092.

Hopfield, J. (1994). Physics, Computation, and Why Biology Looks so Different. *Journal Of Theoretical Biology*, *171*(1), 53-60.

Hopfield, J., & Tank, D. (1985). “Neural” computation of decisions in optimization problems. *Biological Cybernetics*, *52*(3), 141-152.

Howley, T., Madden, M., O’Connell, M., & Ryder, A. (2005). The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data. *Applications And Innovations In Intelligent Systems XIII*, 209-222.

Jackson, W. (1960). *Analog computation* (pp. 5-8). McGraw Hill.

Johnson, C., Philippides, A., & Husbands, P. (2016). Active Shape Discrimination with Compliant Bodies as Reservoir Computers. *Artificial Life*, *22*(2), 241-268.

Langton, C. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, *42*(1-3), 12-37.

Larochelle, H., Mandel, M., Pascanu, R. & Bengio, Y. (2012) Learning Algorithms for the Classification Restricted Boltzmann Machine, *Journal of Machine Learning Research*, 13, 643-669.

Lillicrap, T., & Santoro, A. (2019). Backpropagation through time and the brain. Current Opinion In Neurobiology, 55, 82-89.

Ma, Y., Tsao, D. & Shum, H. Y. (2022). *On the Principles of Parsimony and Self-Consistency for the Emergence of Intelligence,* arXiv:2207.04630v3 [cs.AI]

Millidge, B., Song, Y., Lukasiewicz, T. & Bogacz, R. (2022) *Backpropagation at the Infinitesimal Inference Limit: Unifying Predictive Coding, Equilibrium Propagation, and Contrastive Hebbian Learning*, arXiv:2206.02629v3 [cs.LG]

Millidge, B., Tschantz, A., & Buckley, C. (2022). Predictive Coding Approximates Backprop Along Arbitrary Computation Graphs. *Neural Computation*, *34*(6), 1329-1368.

Mocz, P. (2021). *Create Your Own Spring Network Simulation*. Medium. Retrieved 12.08.22 from https://philip-mocz.medium.com/create-your-own-spring-network-simulation-with-python-d9246e4091e5.

Motlagh, H., Wrabl, J., Li, J., & Hilser, V. (2014). The ensemble nature of allostery. *Nature*, *508*(7496), 331-339.

Orchard, J. (2021) March 6. *Hopfield Networks* [Video file]. YouTube. Available: https://www.youtube.com/watch?v=81B-ESqgCjs&t=3s. [Accessed: August 19, 2022].

Pfeifer, R., & Bongard, J. (2006). How the Body Shapes the Way We Think: A New View of Intelligence. MIT Press.

Rao, R., & Ballard, D. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, *2*(1), 79-87.

Richards, B., & Lillicrap, T. (2019). Dendritic solutions to the credit assignment problem. *Current Opinion In Neurobiology*, *54*, 28-36.

Rizzuto, D., & Kahana, M. (2001). An Autoassociative Neural Network Model of Paired-Associate Learning. *Neural Computation*, *13*(9), 2075-2092.

Rojas, J., Sifakis, E. & Kavan, L. (2021) *Differentiable Implicit Soft-Body Physics*, arXiv:2102.05791v3 [cs.LG]

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536.

Salvatori, T., Pinchetti, L., Millidge, B., Song, Y., Bao, T., Bogacz, R. & Lukasiewicz, T. (2022) *Learning on Arbitrary Graph Topologies Via Predictive Coding*, arXiv:2201.13180v2 [cs.LG]

Scellier, B., & Bengio, Y. (2017). Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers In Computational Neuroscience*, *11*.

Seoane, L. (2019). Evolutionary aspects of reservoir computing. *Philosophical Transactions Of The Royal Society B: Biological Sciences*, *374*(1774), 20180377. https://doi.org/10.1098/rstb.2018.0377

Shrestha, A., Fang, H., Wu, Q., & Qiu, Q. (2019). Approximating Back-propagation for a Biologically Plausible Local Learning Rule in Spiking Neural Networks. *Proceedings Of The International Conference On Neuromorphic Systems*.

Song, Y., Millidge, B. G., Salvatori, T., Lukasiewicz, T., Xu, Z. & Bogacz, R. (2022). *Inferring neural activity before plasticity: A foundation for learning beyond backpropagation.* bioRxiv.

Stern, M., Hexner, D., Rocks, J., & Liu, A. (2021). Supervised Learning in Physical Networks: From Machine Learning to Learning Machines. *Physical Review X*, *11*(2).

Stork, D. G. (1989). Is backpropagation biologically plausible? *Proc. Int. Joint Conf. Neural Netw. (IJCNN),* vol. 2., 241–246.

Walker, S., & Davies, P. (2013). The algorithmic origins of life. *Journal Of The Royal Society Interface*, *10*(79), 20120869. doi: 10.1098/rsif.2012.0869

Whittington, J., & Bogacz, R. (2017). An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation*, *29*(5), 1229-1262.

Xu, L., Lu, Y., & Liu, Q. (2018). Integrating viscoelastic mass spring dampers into position-based dynamics to simulate soft tissue deformation in real time. *Royal Society Open Science*, *5*(2), 171587.

A Appendix

A.1 Experimental Details

The MSD networks were simulated using Python and all experiments were performed using Google Colaboratory remote servers with Tesla P100 GPUs and 27 gigabytes RAM.

A.2. Generalised Coupled Learning Experiments

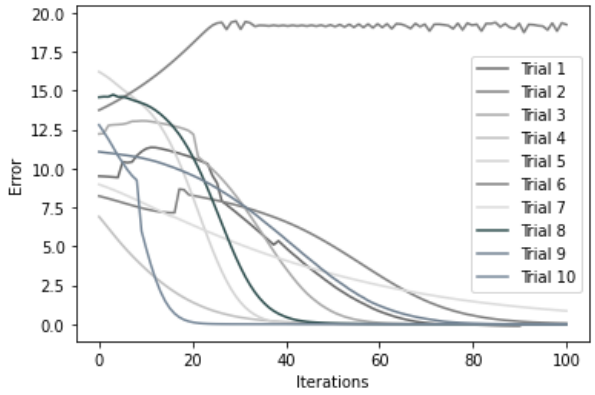


Figure A. Unscaled error as a function of learning rule iterations for 10 independent trials using GCL learning.

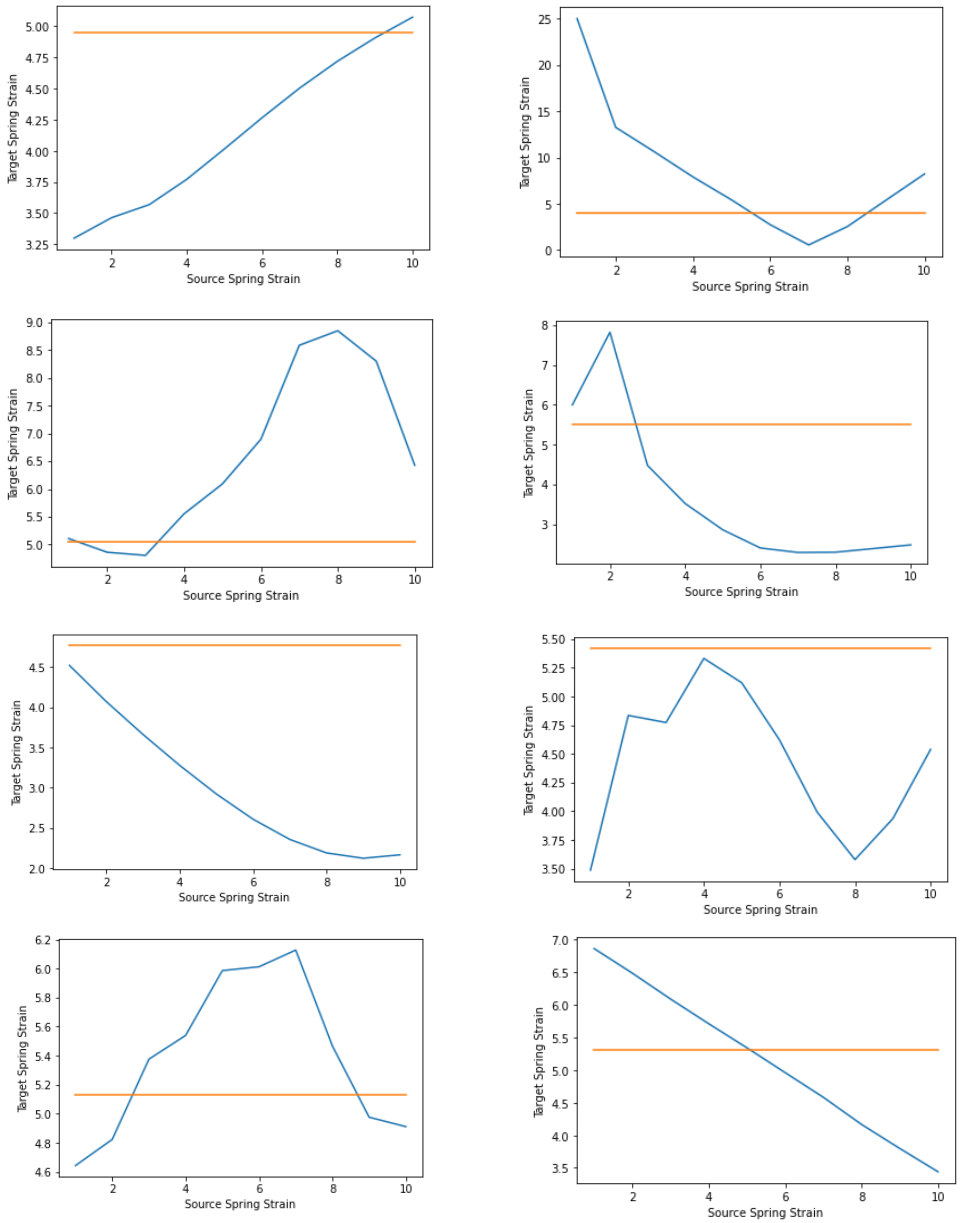


Figure B. Target spring strain in response to application of different source spring strains for 9 independent trained networks. The graphs show the networks' responses (in blue) which do not match the predicted response of the network (in orange) if the couplings learned were trivial (i.e. remained the same as the desired allosteric target response regardless of input).

A.3. Generalised Prospective Configuration Experiments

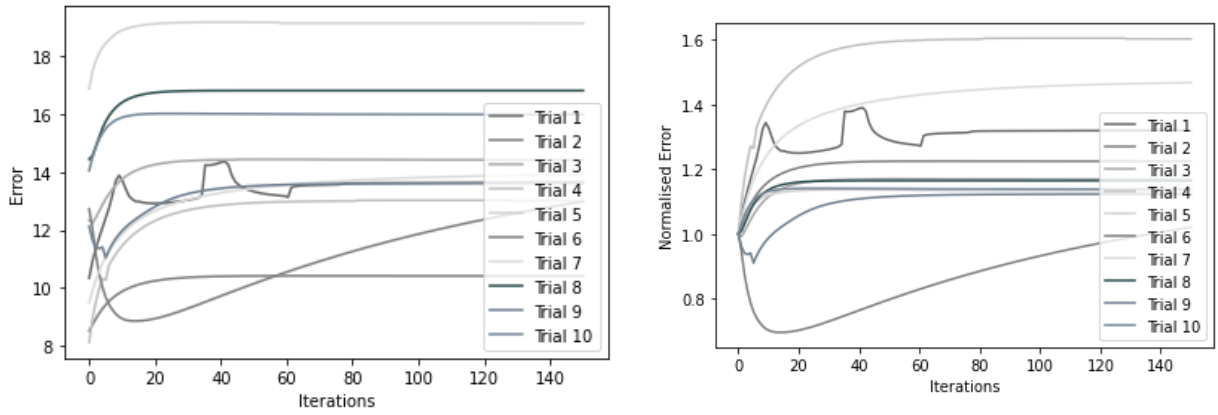


Figure A. a) Unscaled error as a function of learning rule iterations for 10 independent trials using GPC learning and b) error scaled by initial error value.

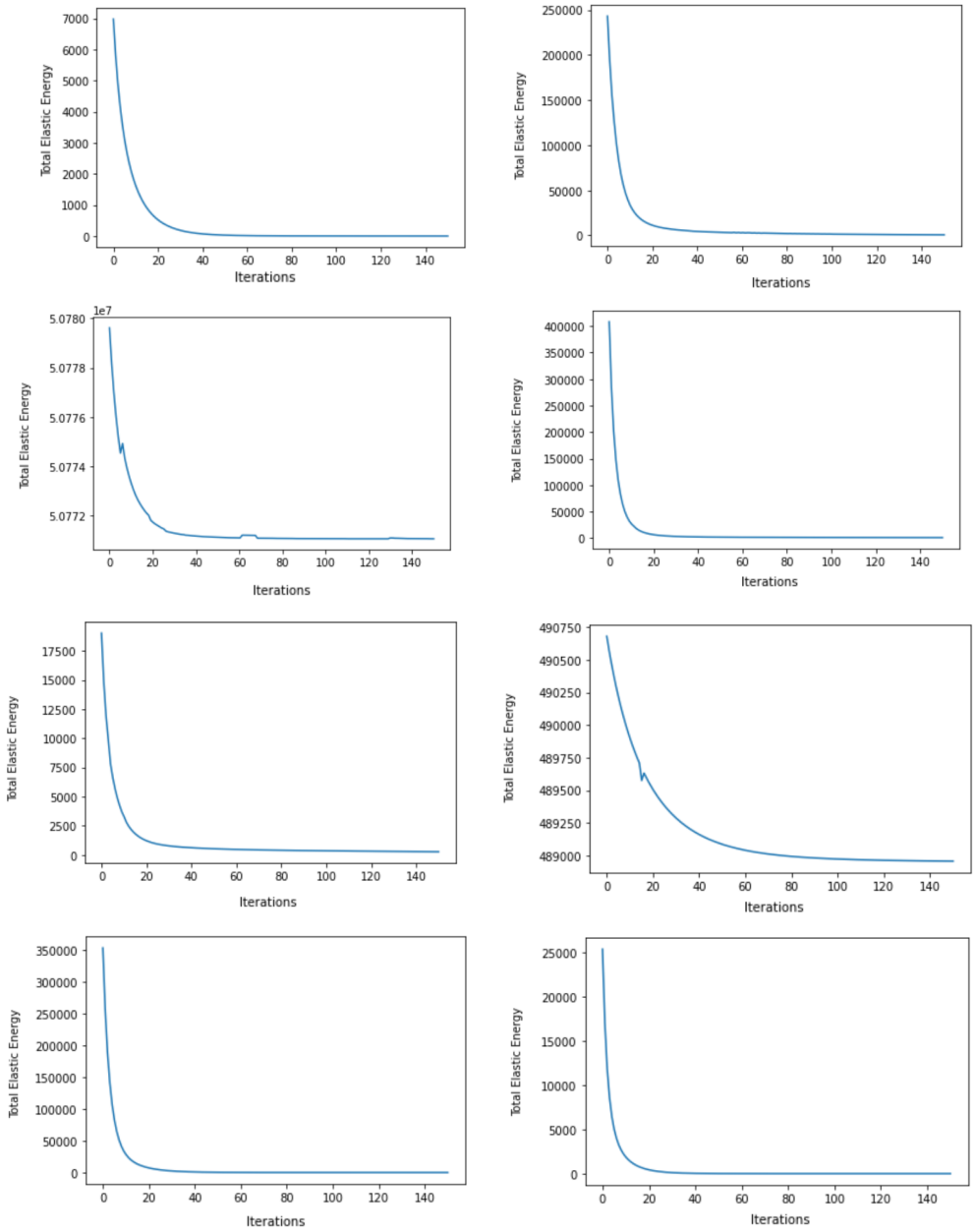


Figure B: Total elastic energy as a function of GPC learning iterations for each independent trial.

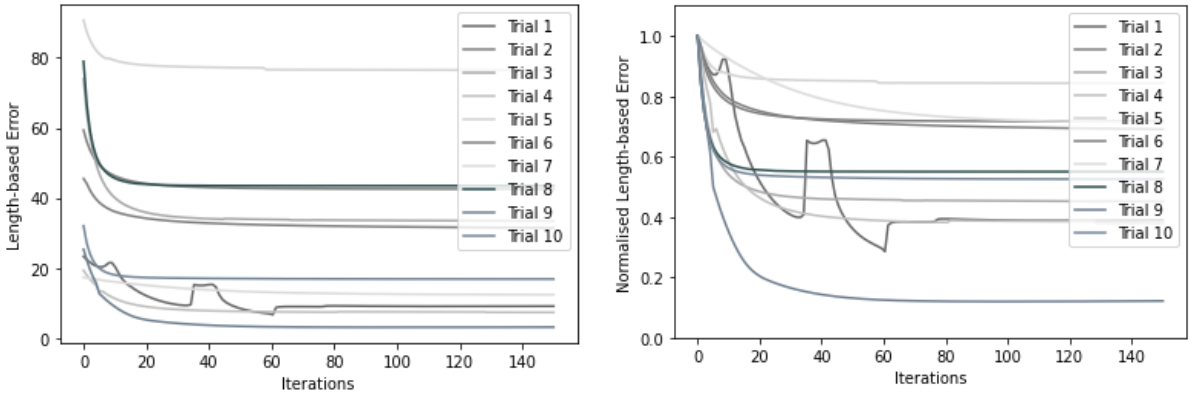


Figure C. a) Unscaled length-based error (distance of target spring length from clamped length) as a function of learning rule iterations for 10 independent trials using GPC learning and b) scaled in relation to initial error.

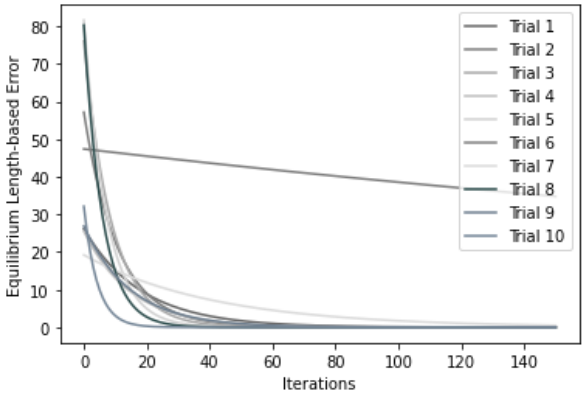
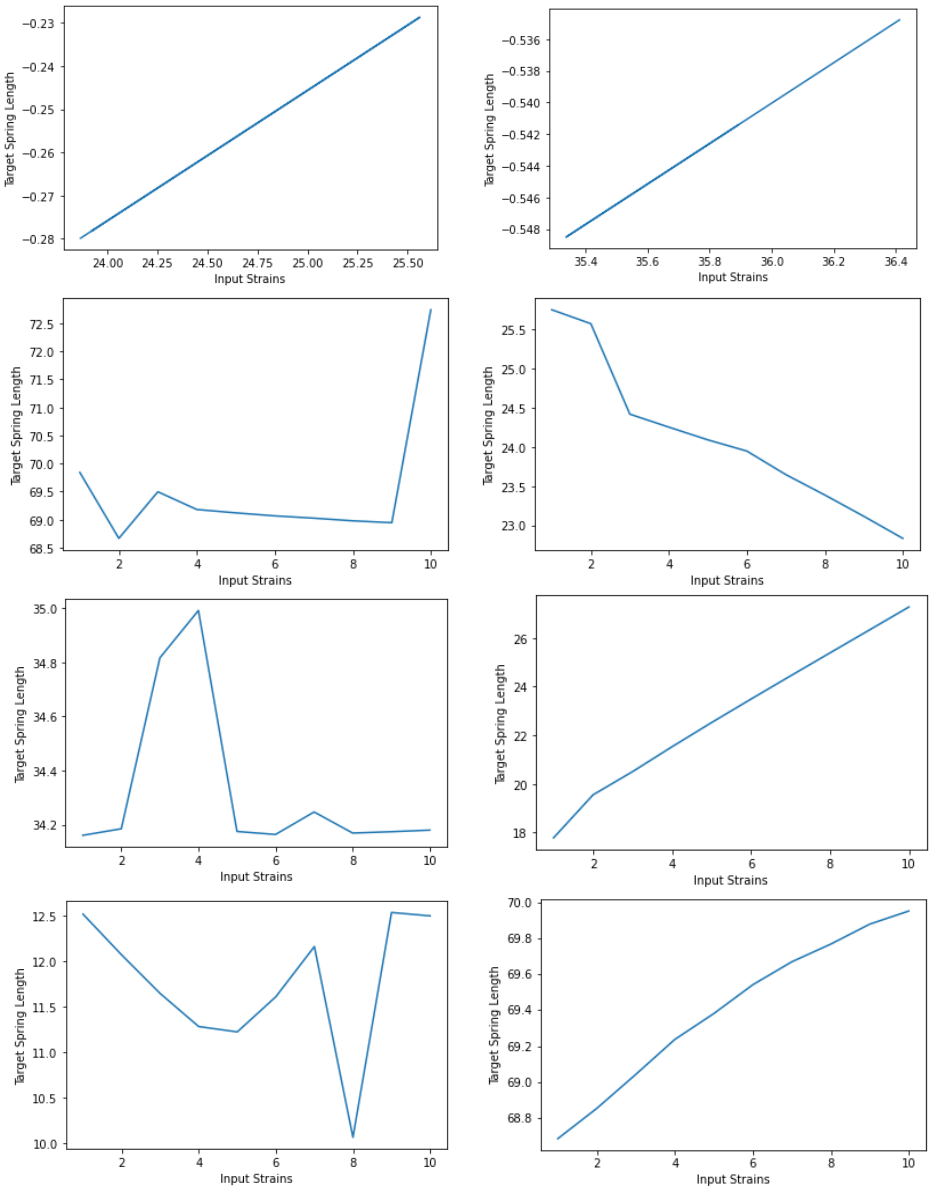


Figure D. Unscaled equilibrium length-based error (distance of target spring equilibrium length from clamped length) as a function of learning rule iterations for 10 independent trials using GPC learning.


Target Spring Length
Input Strains
Target Spring Length
Input Strains
Target Spring Length
Input Strains
Target Spring Length
Input Strains
Target Spring Length
Input Strains
Target Spring Length
Input Strains
Target Spring Length
Input Strains
Target Spring Length
Input Strains

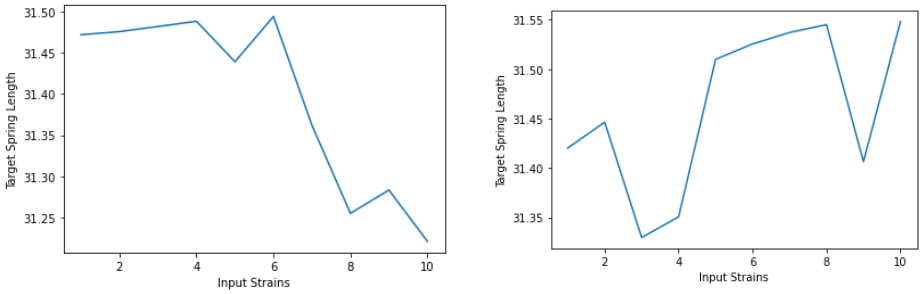


Figure E. Target spring strain in response to application of different source spring strains for 9 independent trained networks from GPC learning. The graphs show the networks' responses (in blue) which if the couplings learned were trivial (i.e. remained the same as the desired allosteric target response regardless of input, would be straight lines. Note the small changes in target spring length likely reflect the fact that input strains are on a scale of 0-10, yet the length of the clamped targets were on average an order of magnitude larger.