BAD PREDICTIVE CODING ACTIVATION FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate predictive coding networks (PCNs) by analyzing their performance under different activation function choices. We expand a previous theoretical discussion of a simple toy example of PCN in the training stage. Compared to classic gradient-based empirical risk minimization, we observe differences for the ReLU activation function. This leads us to carry out an empirical evaluation of classification tasks on FashionMNIST, CIFAR-10. We show that while ReLU might be a good baseline for classic machine learning, for predictive coding, it performs worse than other activation functions while also leading to the largest drop in performance compared to gradient-based empirical risk minimization.

1 INTRODUCTION

Consider the following setup: Let $f : \mathbb{R} \to \mathbb{R}$ be a (typically nonlinear) *activation function*, $\alpha, \beta \in \mathbb{R}$ *weights* and s, v, w are its three nodes, where we assume that the s node is fixed to an input value. Using the notation from Frieder & Lukasiewicz (2022), our simple predictive coding network is:

$$\begin{tikzcd} s \arrow[r, "\alpha"'] & v \arrow[r, "\beta"'] & w \end{tikzcd}$$

We focus on this single, illustrative example (rather than the general case of a network composed of an arbitrary number of nodes) to make the paper easily readable.[1] Note that the predictive coding literature defines the network nodes independently of input values and then fixes arbitrary nodes as input nodes (the same holds true for output values when training the network); for brevity of exposition, we directly assume s to be the input. We define the *energy function* associated to this network class by

$$\mathrm{F}(s, v, w, \alpha, \beta) := \frac{1}{2}(v - \alpha f(s))^2 + \frac{1}{2}(w - \beta f(v))^2, \tag{1}$$

which governs both the inference as well as the training of the network, which, in predictive coding, are two distinct stages.

For the inference stage, the weights are fixed to vales $\alpha := \xi, \beta := \tau$, for some $\xi, \tau \in \mathbb{R}$. Predictive coding then aims to find

$$(v^\star, w^\star) = \min_{(v,w)} \mathrm{F}(s, v, w, \xi, \tau)$$

and $w^\star$ is called the *output* of the network.

For the training stage, where $(s, q) \in \mathbb{R}^2$ is our single training datum, predictive coding aims to minimize

$$(v^\star, \alpha^\star, \beta^\star) = \min_{(v,\alpha,\beta)} \mathrm{F}(s, v, q, \alpha, \beta). \tag{2}$$

It was shown in Frieder & Lukasiewicz (2022) that for the inference stage, executing convergence numerically is unnecessary since the fixed point, to which the system would converge, is easy to characterize: $w^\star$ is precisely the output of the network if it were to be interpreted to be a fully-connected neural network, i.e. $w^\star = \tau f(\xi f(s))$. Hence, no further analysis specific to predictive coding is necessary in this case, as results from classic machine learning hold. But investigating the training stage has been left open.

[1]We advocate systematically using this example-first approach, adapted from Frieder & Lukasiewicz (2022), once the general model equations exceed a certain threshold of complexity.

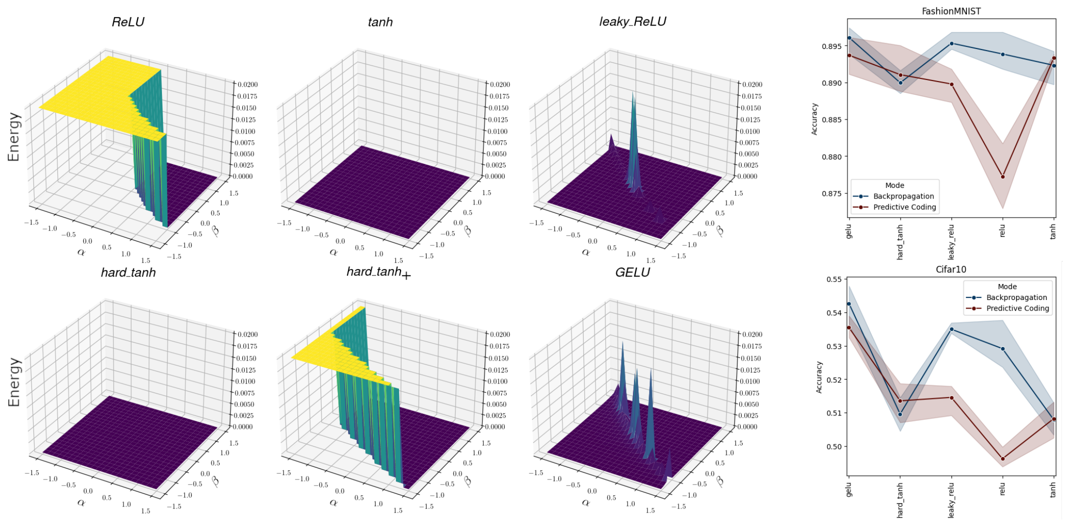


Figure 1: (Left) The energy landscape at convergence for different choices of the activation function over all the possible combinations of initial weights α and β in the chosen intervals. The plot shows wide regions in the cases of ReLU and $\mathrm{hard\_tanh}_{+}$ in which the energy does not converge. For GELU and $\mathrm{leaky\_relu}$ we only see some isolated picks that are due to numerical approximation errors during the training process. (Right) Average best performance of predictive coding and backpropagation over different activation functions. Confirming our observation, choosing ReLU as activation function results in the lowest accuracy for the predictive coding network. The shaded area around each line indicates the standard deviation of the accuracy across the different training seeds.

2 Analysis

From Frieder & Lukasiewicz (2022), we know that under sufficient differentiability conditions of F (and thus on f) the critical points of F admit a characterization using derivatives. Now, those conditions on f also mean we can characterize the critical points of the empirical risk, E, used as the objective function in classic machine learning. In our toy-example framework of a single datapoint $(s, q) \in \mathbb{R}^2$, $\mathrm{E}(s, \alpha, \beta) := \frac{1}{2}(\beta f(\alpha f(s)) - q)^2$, where, for consistency to the definition of F, we also use the squared error loss per datapoint. Compare the resulting two systems of equations whose solutions are the critical point:

$$\begin{cases} v^\star - \alpha^\star f(s) - f'(v^\star)\beta^\star[q - \beta^\star f(v^\star)] = 0, \\ f(s)(v^\star - \alpha^\star f(s)) = 0, \\ f(v^\star)(q - \beta^\star f(v^\star)) = 0, \end{cases} \quad \text{and} \quad \begin{cases} [\beta^\star f(\alpha^\star f(s)) - q]\beta^\star f'(\alpha^\star f(s)) f(s) = 0, \\ [\beta^\star f(\alpha^\star f(s)) - q] f(\alpha^\star f(s)) = 0. \end{cases}$$

Note that the solutions to 2 are among the solutions of the left system, whereas to minima that one obtains in classic machine learning are among the solutions of the right system, which has one coordinate less, as the variable $v^\star$ is missing.

We make the surprising observation that for some activation functions, such as $\mathrm{hard\_tanh}$, the critical points of both systems coincide, which leads us to believe that (modulo difference of the optimizer used to attain, and the local geometry around the minima), the training could result in similar outcomes. For commonly used activation functions in classic machine learning on the other hand, such as ReLU the critical points differ, which shows that there may be a principal barrier in terms of achieving identical results whether we train the network using PC or using classic gradient descent. This observation prompts us to investigate empirically the properties of networks trained with predictive coding and classical gradient descent for varying activation functions, see Figure 1 and Appendix B.

3 Conclusion

We shown how we cannot rely on pre-existing knowledge developed for classical machine learning, where we use backpropagation-based gradient descent to minimize empirical risk. When it comes to predictive coding, new analyses are necessary as known baselines (regarding the efficacy of ReLU) do not seem to carry over. We call for further study of this phenomenon to investigate whether our preliminary observation is confirmed in more systematic investigations.

URM Statement

The authors acknowledge that at least one of the key authors (first/last) of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

References

Nick Alonso, Jeff Krichmar, and Emre Neftci. Understanding and improving optimization in predictive coding networks. *arXiv preprint arXiv:2305.13562*, 2023.

Billy Byiringiro, Tommaso Salvatori, and Thomas Lukasiewicz. Robust graph representation learning via predictive coding. *arXiv:2212.04656*, 2022.

Simon Frieder and Thomas Lukasiewicz. (Non-)Convergence results for predictive coding networks. In *International Conference on Machine Learning*, pp. 6793–6810. PMLR, 2022.

Kuan Han, Haiguang Wen, Yizhen Zhang, Di Fu, Eugenio Culurciello, and Zhongming Liu. Deep predictive coding network with local recurrent processing for object recognition. *Advances in Neural Information Processing Systems*, 31, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv:2006.04182*, 2020.

Luca Pinchetti, Tommaso Salvatori, Yordan Yordanov, Beren Millidge, Yuhang Song, and Thomas Lukasiewicz. Predictive coding beyond Gaussian distributions. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *arXiv:2201.13180*, 2022.

Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870*, 2023.

Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation?—Exact implementation of backpropagation in predictive coding networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22566–22579, 2020.

Yuhang Song, Beren Millidge, Tommaso Salvatori, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature Neuroscience*, pp. 1–11, 2024.

James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, 29(5): 1229–1262, 2017.

A Related Work

In recent years, predictive coding has become an influential biologically inspired learning algorithm, proving that alternatives to gradient descent and backpropagation can exist. In particular, predictive coding has been shown to have competitive performances in feed-forward models, convolutional networks, graph neural networks, and transformer models (Whittington & Bogacz, 2017; Han et al., 2018; Byiringiro et al., 2022; Pinchetti et al., 2022). This can be attributed to its similarities with gradient descent and backpropagation Song et al. (2020); Millidge et al. (2020). Another "branch" of works has been focusing on the differences between the two algorithms, highlighting, for example, different architectures obtainable with predictive coding (Salvatori et al., 2022), exciting novel applications of neural networks (Salvatori et al., 2021), and the need for new optimizers (Alonso et al., 2023). Most of the research is focused on empirical methods, while few works (e.g., Frieder & Lukasiewicz (2022)) take a theoretical approach to the analyses of these networks (Salvatori et al., 2023).

B Numerical Experiments

Energy landscape: We perform a grid search over the possible initial value of parameters α and β over the interval $[-1.5, 1.5]$ and show how the energy function might not converge to a value that minimizes the empirical risk. We used a node learning rate $\gamma = 0.01$ and a weight learning rate $\bar{\gamma} = 0.001$ to simulate the dynamical system described in Eq. (10) in Frieder & Lukasiewicz (2022), on which the current work also focuses. As initial values for s and q, we randomly sampled from the uniform distribution within the intervals of, respectively, $(0, 1)$ and $(-1, 1)$ (we avoided negative values for s as it would trivially prevent the network from learning when using the ReLU activation function. Furthermore, v was initialized with a forward pass (i.e., $v = \alpha f(s)$), which is the most common initialization for predictive coding networks and allows us to plot the energy in a 3D space, having only two free variables, α and β (Whittington & Bogacz (2017), Alonso et al. (2023)). To guarantee convergence, we used a high value of steps $T = 100000$. In Fig. 3, we report two more sets of energy landscapes for new random input/output (s, q) pairs.

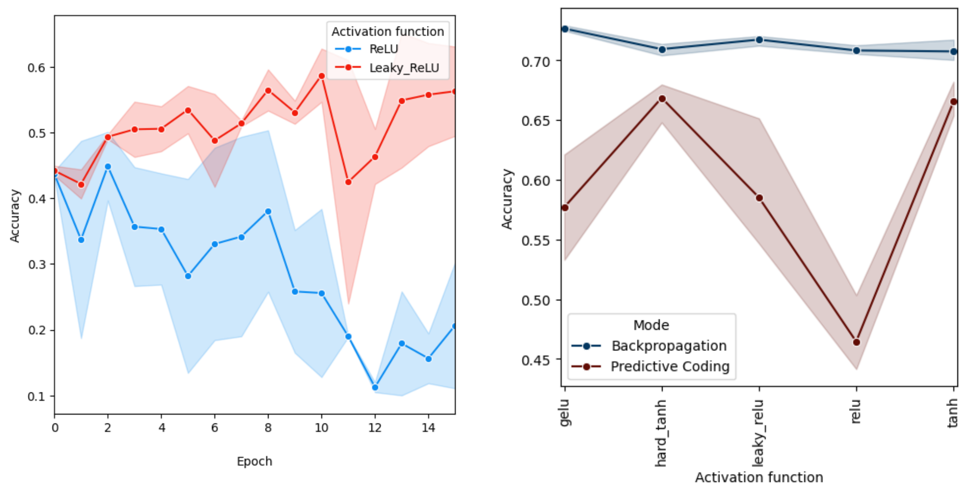


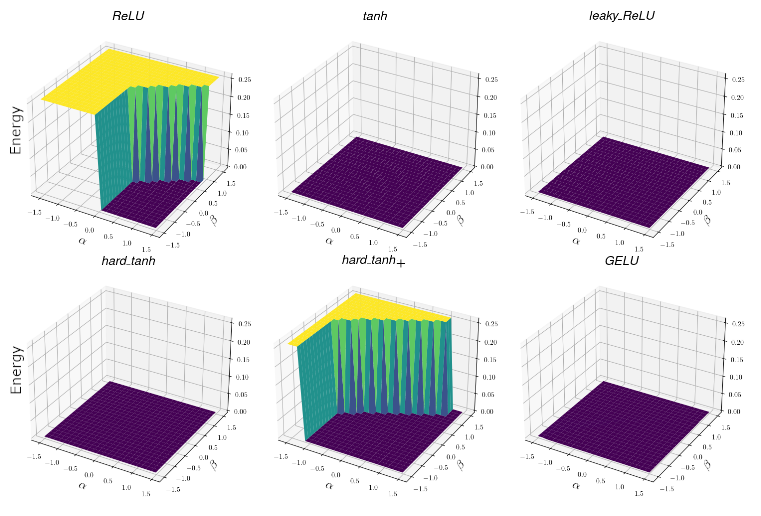
Figure 2: (left) Test error over the training epochs of AlexNet trained with predictive coding when using $\mathrm{leaky\_relu}$ against ReLU. (right) Average best performance of predictive coding and backpropagation over different activation functions with AlexNet. The two graphs highlight the performance impact caused by using the ReLU activation function.

Classification: We tested our findings in classification tasks on the FashionMNIST and CIFAR10 datasets. We have used squared loss for classification, since traditionally, in predictive coding, square loss is used for all tasks, including classification (see, e.g., Whittington & Bogacz (2017), equation (2.2) and algorithm 1, or Song et al. (2024), equation (6)).

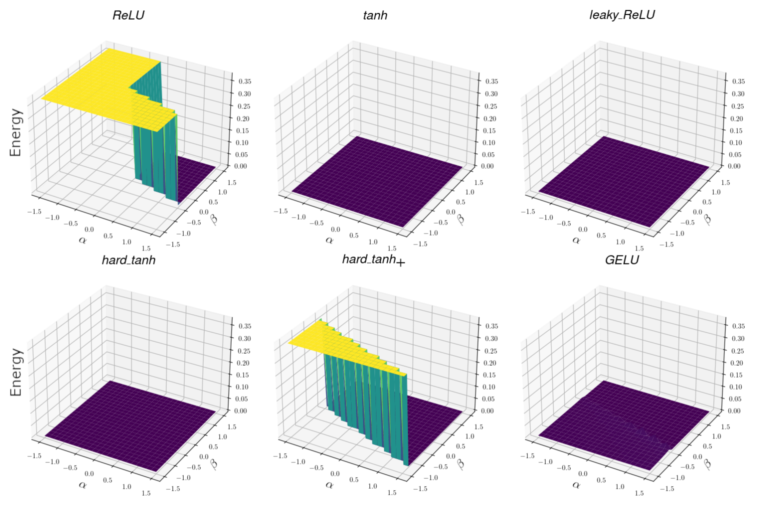
The classification experiments reported in this work were conducted using the variation of predictive coding called Inference Learning (IL) algorithm as described in Song et al. (2020). IL requires a fixed amount of inference steps T to approximate the convergence of the node values. Here, we used $T = 16$, which is a common value found in literature (e.g., Whittington & Bogacz (2017)). In order to guarantee the validity of our results, we perform a hyperparameter search on the node learning rate $\gamma \in \{0.005, 0.01, 0.05, 0.1\}$ (only for predictive coding) and weight learning rate $\bar{\gamma} \in \{10^{-4}, 5 \cdot 10^{-4}, 101{-}3\}$ (for both backpropagation and predictive coding). We reported the best average test accuracy (over 5 seeds, the light area represents the standard deviation) for each activation function and training algorithm. We use *Stochastic Gradient Descent* as an optimizer for the nodes and *Adam* for the weights. We trained for $e = 16$ epochs using a batch size of $b = 128$ and early stopping. Results in Fig. 1 are obtained with a feed-forward fully-connected network composed of $L = 3$ hidden layers of $w = 128$ hidden neurons each. In Fig. 2, we report the accuracy achieved with AlexNet (Krizhevsky et al., 2012) on the CIFAR10 dataset. The performance difference between backpropagation and predictive coding reflects our previous findings. Furthermore, we report the test accuracy for the predictive coding network over the training epochs to highlight the difference between the leaky_relu and ReLU activation functions: with ReLU, not only is the best accuracy is lower, but the training also results unstable, and the network consistently diverges to almost random choice. Due to computing limitations, experiments on AlexNet were conducted on three different random seeds.

We find that 1) for predictive coding, the ReLU activation function performs the worst among the five possibilities (i.e., ReLU, $\tanh$, hard_tanh, GELU, and leaky_relu), and 2) the ReLU activation function results in a much larger drop in performance compared to classic gradient-descent based machine learning. This confirms the expected behavior derived from our theoretical observation regarding differences of critical points. leaky_relu highlights how a small modification of the used activation function can result in significant accuracy gains. We report the average best-achieved test accuracy in Fig. 1, which shows how gradient descent does not incur the same performance loss as predictive coding when using ReLU.

To strengthen our claims by providing a direct comparison, we define hard_tanh_+, a scaled version of $\text{hard\_tanh} := \frac{1}{2}(\text{hard\_tanh} + 1)$ with values in the interval $[0, 1]$, which, similarly to ReLU has a line of zeros (none of the other activation functions have such a property). Fig. 1 shows how, as expected, ReLU and hard_tanh_+ prevent the network to find the energy minimum for the training task.



(a)



(b)

Figure 3: More energy landscapes for new input/output pairs (s, q) that confirm the undesirable network behavior when using activation functions with many zeros.