Unit-level surprise in neural networks

Cian Eastwood[*†] **Ian Mason**[*†] **Christopher K. I. Williams** [†‡]

[†] School of Informatics, University of Edinburgh
[‡] Alan Turing Institute, London

Abstract

To adapt to changes in real-world data distributions, neural networks must update their parameters. We argue that unit-level surprise *should* be useful for: (i) determining which few parameters should update to adapt quickly; and (ii) learning a modularization such that few modules need be adapted to transfer. We empirically validate (i) in simple settings and reflect on the challenges and opportunities of realizing both (i) and (ii) in more general settings.

1 Introduction

Neural networks have achieved remarkable successes on problems with *independent and identically-distributed* (IID) data [23]. However, real-world data is not IID—environmental conditions shift [8], new tasks are encountered [27], and agents alter their behaviour [6]. Ideally, we would like our networks to transfer or adapt *quickly* to such *out-of-distribution* (OOD) data [30].

Fast adaptation to OOD or *shifted* data is often achieved by updating only a few parameters—the final layer of an ImageNet-pretrained network [13, 42], the batch-normalization layers [24, 34, 38], or specific mechanisms or modules [31, 15]. For a given problem or (expected) *shift type*, a human usually decides: (i) which few parameters should be updated; and (ii) what architecture or *modularization* will allow the network to be adapted by updating only a few parameters (e.g. Rebuffi et al. [34] use residual adapters for domain adaptation). However, we often do not know what type of shift will occur and thus how to decide (i) and (ii).

In this work we argue that *unit-level surprise* can help identify *what* has changed, and this in turn can be used to: (i) infer which few parameters should be updated to adapt quickly by transferring past knowledge; and (ii) learn an appropriate modularization such that very few modules need be adapted to transfer. For (i), we propose the use of unit-level surprise—*"This looks the same as before, no need for me to update!"*— along with modulatory *update-in-progress* signals—*"This is being dealt with by someone else, hold tight!"*. These two additional pieces of information at the unit-level allow the network to update only the appropriate parameters to facilitate fast adaptation. This idea is depicted in Figure 1 and further motivated in Appendix A. For (ii), we propose to use the number of surprised units as a *proxy score* for the number of parameters that need to be adapted to a given shift, which can then be optimized to learn a modularization that changes sparsely and thus can be adapted quickly.

As a first step, we empirically validate the usefulness of unit-level surprise by focusing on use case (i) above. Through experiments on shifted EMNIST [7] datasets, we show that unit-level surprise can be used to create shift-dependent fine-tuning strategies that often align with our intuition about which few parameters should update. We then critique these results, identifying several challenges and opportunities for using unit-level surprise in more general settings where its full potential could be unlocked. Despite such challenges, the "beauty" of unit-level surprise—its alignment with our intuitions about how to handle OOD data, its seemingly strong signal for learning how to modularize knowledge, and its relations to neuroscience concepts like metaplasticity [1, 2, 41] and the free-energy principle [12]—convinces us that it *should be better* for realizing (i) and (ii) in more general settings.

[*]Equal contribution.

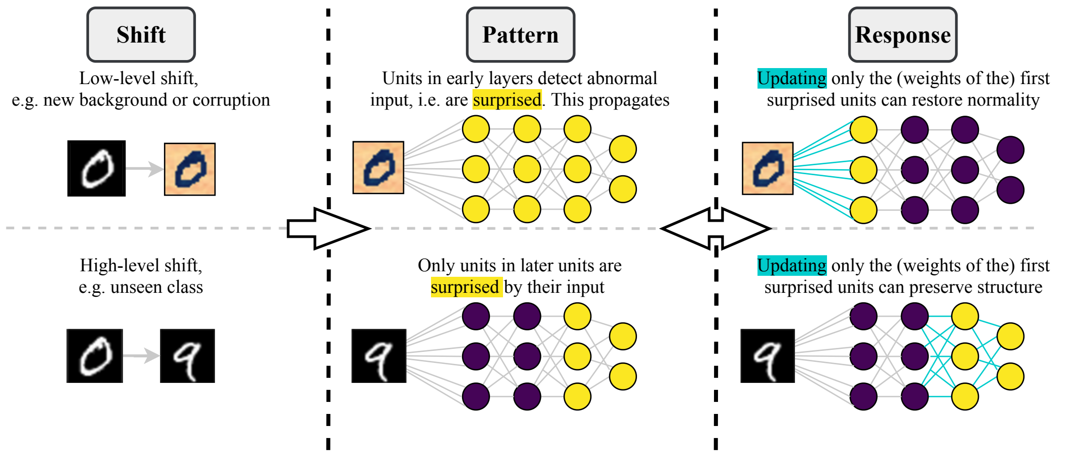

Shift
Low-level shift, e.g. new background or corruption
High-level shift, e.g. unseen class
Pattern
Units in early layers detect abnormal input, i.e. are surprised. This propagates
Only units in later units are surprised by their input
Response
Updating only the (weights of the) first surprised units can restore normality
Updating only the (weights of the) first surprised units can preserve structure

Figure 1: Unit-level surprise can help determine which few parameters should be updated. Purple units are unsurprised, yellow surprised. Blue indicates the weights to be updated. *Top row:* A low-level shift is noticed by units in the first layer (and all those that follow). By blocking those that follow, units in the first layer prevent unnecessary updates in later layers. *Bottom row:* A high-level shift is only noticed by units in later layers, so those in earlier layers don't need to update.

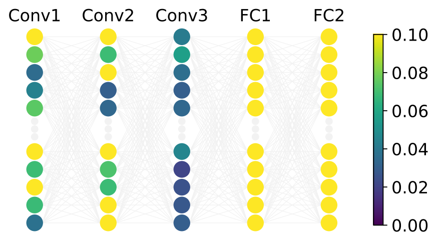
2 Initial validation

In this section we provide an initial empirical validation of the usefulness of unit-level surprise by showing how it can be used to determine which few parameters should be adapted to a given shift.

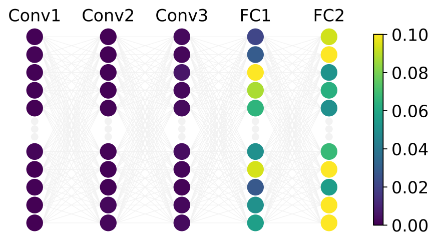
Experimental setup. We train a simple 5-layer network with 3 convolutional (conv) and 2 fully-connected (FC) layers on 37 of the 47 classes in EMNIST [7]. We then adapt this network to new data from 1 of 10 distribution shifts (see Appendix B.1) for which we expect it to be optimal to update different parts of the network. In particular, we use 7 low-level shifts where we expect only the early layers to need to adapt (e.g. new background) and 3 high-level shifts where we expect only the later layers to need to adapt (e.g. held-out/unseen classes). Full experimental details are in Appendix B.2, and code is available at `https://github.com/cianeastwood/unit-level-surprise`.

Calculating unit-level surprise. For each unit in a network we store the 1D distribution of its activations under the training data, $P(A)$, and compare this with the distribution of activations under a shifted data distribution, $Q(A)$. More specifically, we parameterize $P(A)$ and $Q(A)$ as softly-binned histograms [40] and infer the parameters of these distributions from the training and shifted data respectively. We then calculate the (Bayesian) surprise of a unit as $s(A) = D_{KL}(Q(A)||P(A))$ [19]. We discuss this quantity in Appendix B.3 and how to calculate it from bin counts in Appendix B.4.

Surprise patterns. Together, a network's surprise values yield shift-dependent patterns. These patterns, shown in Fig. 2, help us to understand the level of abstraction at which a shift occurs. For example, when the colour of the EMNIST characters is changed under the *crystals* shift (see Fig. 4 in Appendix B.1), we intuitively expect the low-level filters (e.g. black-and-white edge detectors) to be surprised. Fig. 2a shows this to indeed be the case with units in the first layer exhibiting high surprise. Naturally, the abnormal activations which surprised these units propagate through the rest of the network causing the succeeding units to also be surprised. In contrast, when the network is presented with unseen classes, we intuitively expect the low-level features to be unsurprised (similar distribution of black-and-white edges) but the later layers to be surprised (new combinations of low-level features). Fig. 2b shows this to indeed be the case with units in the later layers surprised while those in earlier layers are not.



(a) Low-level shift (crystals bckgr.)



(b) High-level shift (unseen classes)

Figure 2: Surprise patterns.

Creating an update rule. Once units have been equipped with the ability to calculate their surprise, how can we best

Table 1: 5-shot accuracy when training single layers, all layers (SGD), using FlexTune [35], or using a surprise-based update rule. *L:* low-level shifts (avg), *H:* high-level shifts (avg), *All:* all shifts (avg).

	Conv1	Conv2	Conv3	FC1	FC2	SGD	FlexTune	Upd. Rule
L	82.7 ± 0.4	70.2 ± 1.2	61.4 ± 0.2	56.3 ± 0.2	51.5 ± 0.3	71.3 ± 2.1	82.7 ± 0.4	$\mathbf{82.9 \pm 0.5}$
H	0.0 ± 0.0	0.3 ± 0.5	25.4 ± 3.8	80.6 ± 4.6	$\mathbf{94.5 \pm 0.9}$	74.4 ± 5.2	$\mathbf{94.5 \pm 0.9}$	79.2 ± 4.2
All	57.9 ± 0.3	49.3 ± 0.8	50.6 ± 1.2	63.6 ± 1.2	64.4 ± 0.3	72.2 ± 2.9	$\mathbf{86.3 \pm 0.5}$	81.8 ± 1.2

Table 2: Accuracy for a varying number of samples-per-class, averaged over different shifts.

	2	5	10	20	50	2000
SGD	60.0 ± 1.9	72.2 ± 2.9	78.9 ± 0.5	82.6 ± 0.2	86.1 ± 0.3	92.1 ± 0.1
FlexTune	$\mathbf{82.6 \pm 0.7}$	$\mathbf{86.3 \pm 0.5}$	$\mathbf{87.8 \pm 0.4}$	$\mathbf{89.2 \pm 0.2}$	$\mathbf{90.4 \pm 0.2}$	$\mathbf{92.5 \pm 0.0}$
Upd. Rule	66.3 ± 2.5	81.8 ± 1.2	86.2 ± 0.6	88.2 ± 0.4	89.7 ± 0.1	92.4 ± 0.0

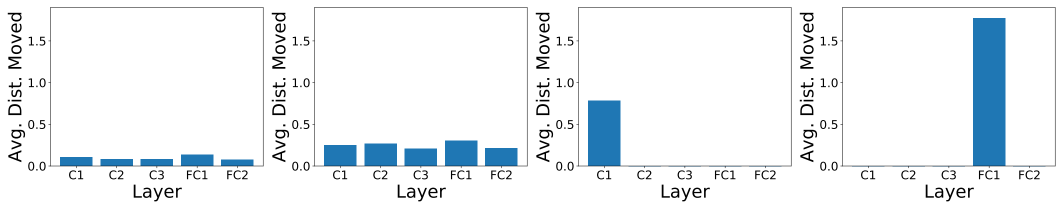
make use of this? We investigate whether or not surprise can give us some indication of the specific few units to update in order to adapt quickly to new data. In particular, we focus on few-shot learning as updating (the parameters of) fewer units should improve sample-efficiency. We consider three possible adaptation strategies or shift *responses*:

1. **SGD.** Ignore surprise patterns and use *stochastic gradient descent* (SGD) to update *all* parameters.
2. **FlexTune [35]**. Ignore surprise patterns and select the best individual layer to update using a test-domain validation set. This serves as an upper-bound on single-layer performance.
3. **Update rule based on unit-level surprise.** Use a rule that determines whether or not a unit should update based on surprise values.We design an update rule with the goal of leveraging unit-level surprise to preserve structure by preventing unnecessary updates. This rule can be summarised as *"update only if: (i) you are surprised; and (ii) your parents are not"*, with the key insight being that surprised parents (units in the preceding layer that are connected to it) indicate that the surprise-causing shift may be resolved in earlier layers. Here, (i) ensures that only affected units update—*"This looks the same as before, no need for me to update!"*—while (ii) acts as a modulatory *update-in-progress* signal from a unit's parents—*"This is being dealt with by someone else, hold tight!"*. We formally define this update rule in Appendix B.5.

Results. Table 1 gives the 5-shot results for training individual network layers and using each of our 3 responses. We see that low-level shifts are best dealt with by fine-tuning the first layer and high-level shifts the last. This aligns with our intuition about which units should update and also with the surprise patterns in Figure 2. Without *a priori* knowledge of the type of shift that will occur (choosing Conv1 for low-level shifts, FC2 for high-level) or exhaustively searching over all possibilities using an oracle-like test-domain validation set (FlexTune), we achieve the best average performance over low- and high-level shifts with our surprise-based update rule—where units only update if they are sufficiently surprised and their parents are not. These results show that it is not always optimal to fine-tune only the later layers of a network (as also found in [35]) and demonstrate that unit-level surprise can indeed help determine which few parameters should update to adapt quickly (i.e. with few samples) to new data. Results for individual shifts are given in Appendix E.

We also evaluate each of these adaptation strategies or responses as a function of the number of shifted samples that are available. As shown in Table 2, all strategies perform well using large amounts of data (2000 samples-per-class). However, as the number of samples-per-class drops, it becomes more important to update few parameters, and thus to choose the right strategy. With 5-50 samples-per-class, simple SGD (training all layers) is outperformed by both other strategies which seek to maintain structure by selecting few units to update. While FlexTune consistently outperforms the update rule (by a small margin), it requires training 5 different models on labelled test-domain data (one per layer) in order to select the best layer to update.

Which units does SGD update? To visualize which parameters are updated we calculate "how far" each unit's input parameters are moved and use this to compare SGD and our update rule. Specifically, we calculate the Euclidean distance between a parameter's value before and after adaptation, then average these distances layerwise to see which layers are being updated by each adaptation strategy. Figures 3a & 3b show that SGD moves all layers of the network similarly and makes no distinction between different types of shift. This confirms that a simple gradient signal is not sufficient to select



(a) Low, SGD (b) High, SGD (c) Low, Upd. Rule (d) High, Upd. Rule

Figure 3: Mean distance moved for units in each layer of a 5-layer CNN when using SGD and the update rule for low (crystals background shift) and high (H1, 5 unseen classes) level shifts.

which few parameters should update to a given shift, explaining the results in Table 1. In contrast, Figures 3c & 3d show that our update rule exhibits shift-specific behaviour which aligns with the results in Table 1—moving early layers for low-level shifts and later layers for high-level shifts.

3 Critique

While we achieved some promising results in carefully-designed settings, it remains unclear how unit-level surprise should be used in more realistic settings to determine which few parameters to update. Below we discuss the main challenges discovered during the initial idea validation above.

Networks are not sufficiently modular to show sparse surprise patterns. For the various shifts we considered, networks did not exhibit sparse or modular surprise patterns—low-level shifts surprised almost every unit in the network (Fig. 2a), while high-level shifts surprised all units in both FC layers (Fig. 2b). This then affects the updates of our update rule, with single, entire layers being updated rather than a modular selection of units (Fig. 3). We posit that such patterns of surprise occur because, using standard training techniques, simple neural networks do not form modules or mechanisms that change independently (and thus can be surprised independently). Moreover, we find ourselves asking if it is *ever* possible to learn such modules or mechanisms that change sparsely (i.e. align with the shifts in data distribution) without explicitly optimizing for it at training time over multiple shifts.

Parents *and* children may need to update. As discussed above, our update rule tends to update single layers, and this can lead to sub-optimal solutions (e.g. updating only FC1 for high-level shifts). We attribute this to the fact that surprised parent units tend to stay surprised, continually blocking their children from updating. One solution would be to update $P(A)$ with samples from $Q(A)$ as training progresses, for units that are being adapted. Intuitively, this would allow parent units to gradually become less surprised by the new data, eventually freeing their children to also update.

Surprise may not be sufficient. As shown in Table 1 and Figure 3, the update rule does not always select the "optimal" units to update (updating FC1 rather than FC2 for high-level shifts). This raises some questions—is surprise alone sufficient to determine which units should update? What other unit-level information may be required? Here we list some examples: (a) *gradient magnitude*—may help us to update FC2 rather than FC1 to adapt to high-level shifts; (b) *task importance* [21, 43]—may help in continual learning settings to best preserve performance on past tasks when adapting to new ones; (c) *parameter uncertainty*—may help speed-up learning by only updating the parameters with high uncertainty [3], akin to Bayesian optimization. Additionally, one can imagine shifts that are best resolved at a mid- or high-level but cause changes to the low-level feature distributions—e.g. if we occlude part of a digit, the low-level features (edges) should still be valid, we just need to update how they combine to form the class label. However, the changed low-level feature distributions (some edges are occluded) will surprise units in the early layers, causing them to update. Another similar challenge is that of changing feature frequency. In particular, if we take the view of units as feature detectors, where a specific unit is sensitive to a specific feature, then the activation distributions of units will change if the features they detect occur with a different frequency (i.e. appear more or less often). In this situation, a unit can show high surprise but we may not wish to update its parameters if we still wish to detect this feature. One potential solution to these problems is to explore alternative measures that distinguish between being more or less surprised than expected (see Appendix C).

Lack of interpretability makes validation difficult. Finally, it is difficult to come up with experimental setups where we know what the "right" units to update are. This makes the design and validation of update rules quite challenging. While we may hope for interpretable edges-parts-wholes feature hierarchies (see Figure 9 in Appendix E.1), this is often not the case. In fact, it can take as little as a change in random seed to change the semantic meaning of features, and with it, the optimal units to update. Work on unit-level interpretability [14, 29] may eventually help, but it is not currently at a level to decide which units should update for a given shift.

4 Future work

As discussed above, creating a general update rule using unit-level surprise is challenging as: (i) neural networks are not very modular by default; and (ii) surprise may not be sufficient to determine "who" should update. Below we discuss two potential avenues to overcome these two challenges.

Learning modular structure with unit-level surprise. Recent work in causal discovery assumes that the ground-truth data-generative process consists of independent mechanisms or modules, with many modules expected to behave similarly across different tasks and environments [5, 31, 32, 36, 37]. Thus, if an appropriate modular representation of the world has been learned, *very few modules should need to be adapted in order to transfer* [5, 15, 32, 37]. Bengio et al. [5] exploit this through a meta-learning objective that uses *adaptation speed* to optimize the way in which knowledge is represented or modularized. Here, adaptation speed is a proxy score for the number of modules and parameters that need be adapted and ultimately for how well the learned modularization fits the underlying causal dependencies. We believe surprise could provide an *alternative proxy score* (e.g. the number[2] of surprised units or groups of units) that is easier to meta-optimize (no inner-loop steps) and arguably a more direct measure of the number of modules and parameters that need be adapted.

Learning optimizers to better utilize unit-level surprise. As discussed in the previous section, unit-level surprise may not be sufficient for some shifts—further information may be required to determine "who" should update. This makes it difficult to handcraft a general update rule as one must specify how surprise should be used with this other information. In fact, even without additional information, there is still an enormous space to be explored surrounding how best to use unit-level surprise, e.g. setting the learning rate based on the surprise by using *soft* thresholding. One solution is to *learn* an optimizer [4, 33], or the parameters of an update rule [25], across multiple shifts. However, learned optimizers are notoriously difficult to train [28] and it can be difficult to beat the heavily-tuned few-shot benchmarks against which they are often compared [16].

5 Conclusion

In this work we provided a preliminary validation of the usefulness of unit-level surprise in neural networks. In particular, we showed that it can be useful for analysing *where* dataset shifts are noticed in a network and for devising *shift-dependent* fine-tuning strategies. We also discussed some of the "beauty" that makes us believe that unit-level surprise *should* be useful in more general settings—it often aligns with our intuitions about how to handle OOD data, it seems a strong signal for learning how to modularize knowledge such that only a few modules are surprised (and thus need be adapted), and it has several neuroscience relations such as metaplasticity and the free-energy principle. However, in formally specifying how to use unit-level surprise in simple settings, we encountered several challenges that made us wary of tackling the more complicated settings in which its full potential could be realized.

Acknowledgements

We thank Matthias Hennig, Tim Hospadales, Amos Storkey, and Sébastian Arnold for helpful discussions and comments. CE acknowledges support from The National University of Ireland via his Travelling Studentship in the Sciences. IM is supported by the Engineering and Physical Sciences Research Council (EPSRC).

[2]A continuous proxy can be used for differentiability.

References

[1] W. C. Abraham. Metaplasticity: tuning synapses and networks for plasticity. *Nature Reviews Neuroscience*, 9(5):387–387, 2008.

[2] W. C. Abraham and M. F. Bear. Metaplasticity: the plasticity of synaptic plasticity. *Trends in Neurosciences*, 19(4):126–130, 1996.

[3] L. Aitchison, J. Jegminat, J. A. Menendez, J.-P. Pfister, A. Pouget, and P. E. Latham. Synaptic plasticity as bayesian inference. *Nature Neuroscience*, 24(4):565–571, 2021.

[4] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

[5] Y. Bengio, T. Deleu, N. Rahaman, N. R. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal, and C. Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In *International Conference on Learning Representations*, 2020.

[6] L. Buesing, T. Weber, Y. Zwols, N. Heess, S. Racaniere, A. Guez, and J.-B. Lespiau. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019.

[7] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[8] D. Dai and L. Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *International Conference on Intelligent Transportation Systems*, pages 3819–3824. IEEE, 2018.

[9] C. Eastwood, I. Mason, C. K. I. Williams, and B. Schölkopf. Source-Free Adaptation to Measurement Shift via Bottom-Up Feature Restoration. *arXiv:2107.05446*, 2021.

[10] S. M. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[11] D. E. Feldman. Synaptic mechanisms for plasticity in neocortex. *Annual Review of Neuroscience*, 32:33–55, 2009.

[12] K. Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[14] G. Goh, N. C. †, C. V. †, S. Carter, M. Petrov, L. Schubert, A. Radford, and C. Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons.

[15] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021.

[16] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[17] S. R. Hulme, O. D. Jones, and W. C. Abraham. Emerging roles of metaplasticity in behaviour and disease. *Trends in Neurosciences*, 36(6):353–362, 2013.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[19] L. Itti and P. Baldi. Bayesian surprise attracts human attention. *Vision research*, 49(10):1295–1306, 2009.

[20] P. S. Katz. *Beyond Neurotransmission: Neuromodulation and its Importance for Information Processing*. Oxford University Press New York:, 1999.

[21] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.

[22] S. Kullback. *Information theory and statistics*. Wiley, 1959.

[23] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. In *International Conference on Learning Representations Workshop*, 2017.

[25] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv:1707.09835*, 2017.

[26] D. J. MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[27] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

[28] L. Metz, N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pages 4556–4565, 2019.

[29] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[30] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[31] G. Parascandolo, N. Kilbertus, M. Rojas-Carulla, and B. Schölkopf. Learning independent causal mechanisms. In *International Conference on Machine Learning*, pages 4036–4044, 2018.

[32] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

[33] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[34] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017.

[35] A. Royer and C. Lampert. A flexible selection scheme for minimum-effort transfer learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2191–2200, 2020.

[36] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. *arXiv:1206.6471*, 2012.

[37] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[38] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. TENT: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[39] S.-Z. Wang and H. W. Tao. History matters: illuminating metaplasticity in the developing brain. *Neuron*, 64(2):155–157, 2009.

[40] Y. Yang, I. G. Morillo, and T. M. Hospedales. Deep neural decision trees. In *ICML Workshop on Human Interpretability in Machine Learning (WHI)*, 2018.

[41] A. X. Yee, Y.-T. Hsu, and L. Chen. A metaplasticity view of the interaction between homeostatic and Hebbian plasticity. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1715), 2017.

[42] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833, 2014.

[43] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *34th International Conference on Machine Learning*, pages 3987–3995, 2017.