Do solutions for catastrophic forgetting degrade model robustness to naturalistic corruption and adversarial attacks in binary neural networks?

TUAN PHAM, The University of Chicago, USA
AMIT PRADHAN, The University of Chicago, USA
SEBASTIAN BARRIOS, The University of Chicago, USA

Additional Key Words and Phrases: neural networks, catastrophic forgetting, continual learning, robustness, natural corruptions, adversarial attacks

1 INTRODUCTION

Developing a deep learning model, for example, in image classification, is not only about achieving the best classification accuracy. Even with increasing computational power, multiple issues also need to be solved, including - to name a few - fast inference time, manageable memory size, continual learning, and robustness to data/model corruption. An attractive solution for fast computation, efficient power consumption, and potentially more straightforward hardware implementation is binary neural networks (BNN) [1], in which only the signs of the model's hidden weights are utilized during inference.

A recent study takes inspiration from biological metaplasticity to solve catastrophic forgetting (CF) problems for continual learning in BNN [8]. More specifically, by creating a form of multiplicative gating during learning for hidden weights to represent weight consolidation (Figure 1), they are able to solve the permuted-MNIST task, sequential learning with CIFAR-10/100 dataset, and stream learning with these datasets. This approach shares certain similarities with another study in regular neural networks, also inspired by neuroscience literature, which takes into account (a) weight stabilization based on task importance, combined with (b) context-dependent gating allowing for more sparse, non-overlapping population activations, to facilitate learning and remembering a large number of tasks [11].

Back to BNNs, due to such quantization, they can be quite sensitive to corruption, for example, adversarial data attacks [9] or potential soft errors in hardware accelerators [5]. Hence, we wish to ask whether previous solutions for CF [8] could further degrade model robustness to data corruption or help ameliorate it.

More specifically, we partially reproduced the results of the metaplasticity training in the permuted MNIST task and the split-class CIFAR task. Then we tested the resulting models under different types of naturalistic corruption to input data are introduced [4] and different types of adversarial attacks [3, 14, 15]. We also included the Lipschitz regularization term as proposed for defensive quantization [9] in the pMNIST task.

While metaplasticity for BNN was helpful in preventing catastrophic forgetting (pMNIST and split-class CIFAR tasks), the resulting models were not as robust. There appears to be a trade-off for continual learning and robustness, as optimized for the latter would degrade the former, though not in a monotonic manner. Additionally, for the pMNIST task, Lipschitz regularization has mixed effects on model robustness for attacks and corruptions for the pMNIST task.

2 METHODS

The repository is at https://github.com/tuanpham96/bnn-cf-vs-robust/

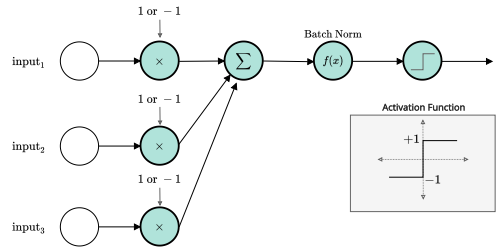
Authors' addresses: Tuan Pham, tuanph18@uchicago.edu, The University of Chicago, Chicago, Illinois, USA; Amit Pradhan, pradhanak@uchicago.edu, The University of Chicago, Chicago, Illinois, USA; Sebastian Barrios, sebastian.barrios@chicagobooth.edu, The University of Chicago, Chicago, Illinois, USA.

2.1 Binarized neural networks

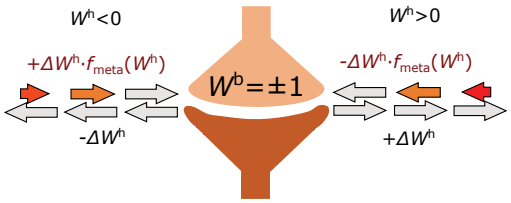
BNN uses binary weights and activations during the inference phase and the training phase binary weights, while the hidden weights are used for training. For the pMNIST task, the BNN is an MLP with 2 hidden layers of varying widths. For the split-class CIFAR, the BNN is a CNN pretrained on ImageNet followed by a metaplastic classifier.

2.2 Metaplasticity

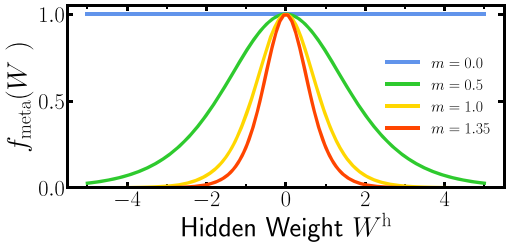
The metaplasticity algorithm is shown in Figure 1d (adapted from [8]), which controls how plastic the hidden weights are. The function f_{meta} controls how much these weights could be trained - higher hidden weights values indicate more consolidation and disallows further changes. The choice of the function is $1 - \tanh(m \cdot W^h)$ similar to the original paper. The values of the metaplasticity variable m (used interchangeably with `meta` throughout the text) are varied.



a



b



c

Input: $\mathbf{W^h}$, $\boldsymbol{\theta}^{\text{BN}}$, $\mathbf{U_W}$, $\mathbf{U}_{\boldsymbol{\theta}}$, $(\mathbf{x}, \mathbf{y})$, m, η.
Output: $\mathbf{W^h}$, $\boldsymbol{\theta}^{\text{BN}}$, $\mathbf{U_W}$, $\mathbf{U}_{\boldsymbol{\theta}}$.
$\mathbf{W^b} \leftarrow \text{Sign}(\mathbf{W^h})$ ▷Computing binary weights
$\hat{\mathbf{y}}$, cache $\leftarrow$ Forward$(\mathbf{x}, \mathbf{W^b}, \boldsymbol{\theta}^{\text{BN}})$ ▷Perform inference
$C \leftarrow$ Cost$(\hat{\mathbf{y}}, \mathbf{y})$ ▷Compute mean loss over the batch
$(\partial_{\mathbf{W}} C, \partial_{\boldsymbol{\theta}} C) \leftarrow$ Backward$(C, \hat{\mathbf{y}}, \mathbf{W^b}, \boldsymbol{\theta}^{\text{BN}}, \text{cache})$
▷Cost gradients
$(\mathbf{U_W}, \mathbf{U}_{\boldsymbol{\theta}}) \leftarrow \text{Adam}(\partial_{\mathbf{W}} C, \partial_{\boldsymbol{\theta}} C, \mathbf{U_W}, \mathbf{U}_{\boldsymbol{\theta}})$
for W^{h} in $\mathbf{W^h}$ **do**
if $U_W \cdot W^{\text{b}} > 0$ **then** ▷If U_W prescribes to decrease$|W^{\text{b}}|$
$W^{\text{h}} \leftarrow W^{\text{h}} - \eta U_W \cdot f_{\text{meta}}(m, W^{\text{h}})$ ▷Metaplastic update
else
$W^{\text{h}} \leftarrow W^{\text{h}} - \eta U_W$
end if
end for
$\boldsymbol{\theta}^{\text{BN}} \leftarrow \boldsymbol{\theta}^{\text{BN}} - \eta \mathbf{U}_{\boldsymbol{\theta}}$
return $\mathbf{W^h}$, $\boldsymbol{\theta}^{\text{BN}}$, $\mathbf{U_W}$, $\mathbf{U}_{\boldsymbol{\theta}}$

d

Fig. 1. Metaplasticity in BNN. (**a**) A typical BNN. (**b**) Visual representation of metaplasticity (adapted from [8]). (**c**) Plot of hidden weights W^h varying by metaplasticity level m (adapted from [8]). (**d**) Metaplasticity algorithm for BNN (adapted from [8]).

2.3 Lipschitz regularization

This was introduced in [9] as an additional regularization term for improving the robustness of quantized networks against adversarial attacks: $\mathcal{L}_{\text{DQ}} = \frac{1}{2}\beta_{\text{DQ}} \sum \|\mathbf{W}^T\mathbf{W} - \mathbf{I}\|^2$. Briefly, this form of regularization keeps the spectral norm of the hidden weights close to 1 to prevent propagated errors to “explode”.

2.4 Tasks

The pMNIST task consists of 6 permuted MNIST tasks in which the pixel locations are varied from one task to another. The split-class CIFAR(10/100) task consists of 2 (for CIFAR 10) and 6 (for CIFAR 100) subsets of class. The goal is to achieve high performance in all tasks in an incremental learning scenario without access to previous trained task.

2.5 Natural corruption

The naturalistic corruption transformations are taken from [4]. The examples are shown in Figure 2 for the pMNIST task and Figure S1 for the CIFAR task. For the pMNIST dataset, we modified the code teHendrycks$_2$018*toallowformonochromenaturalcorruptionson.Becaus generatedbeforetraining, tosavetimeduringtesting.Additionally, onlyonelevelofseverityistestedforallofnaturalcorruptions*.

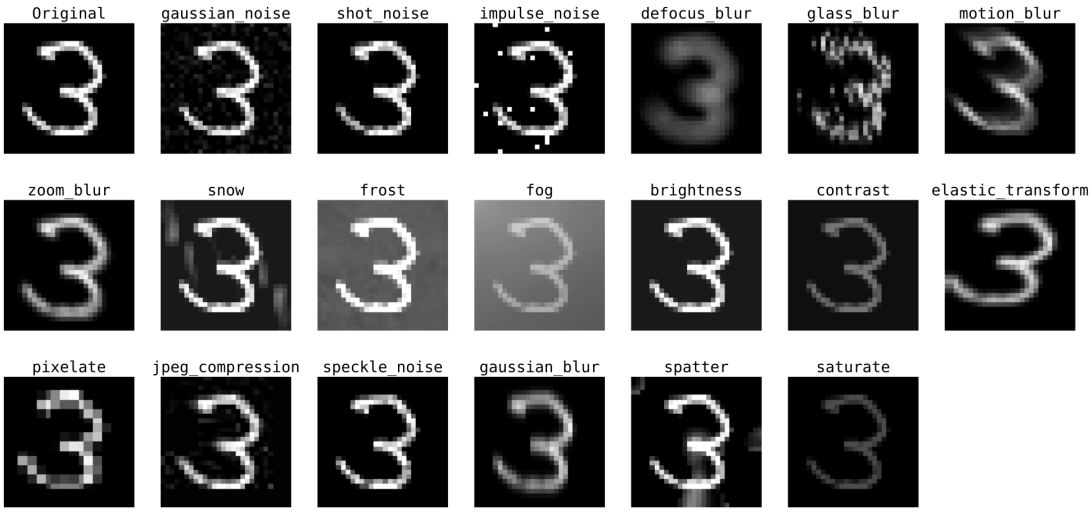


Fig. 2. Natural corruption applied to the MNIST dataset.

2.6 Adversarial attacks

The adversarial attacks are small perturbations (with varying magnitudes ϵ) applied to these tasks are targeted attacks, taken from [14, 15]. Due to the limited time, we were not able to integrate the various attacks to the CIFAR task (only self-coded FGSM). Briefly, the attacks covered here are:

- `Additive-Uniform-Noise` is not a targeted attack, with only small addtive noise sampled a from uniform distribution. This serves as a reference as this would suffer less damage than the following targeted attacks.
- `FGSM` (Fast Gradient Sign Method) [3] is a white-box attack that applies small perturbations following the sign of the loss gradients for the current input.
- `Basic-Iterative-Attack` [7] and `PGD` (Projected Gradient Descent) [10] are `FGSM` extensions that allow for iterative `FGSM` perturbations with magnitude clipping. The latter starts with additive signs of a sampled Gaussian distribution, in addition to `FGSM`.

- `DeepFool-Attack` [13] approximates the model linearly and iteratively chooses the pertubation to cross model decision boundary.
- `DDN-Attack` (Decoupled Direction and Norm) [16] projects the perturbation directions onto an ϵ sphere and further adjusts the sphere radius depending on whether the generated input is considered adversarial.

For the pMNIST adversarial evaluation, instead of looking at the monotonically decreasing accuracy-perturbation curves directly, we calculated the normalized area under the curve (`auc_log`) of the log-log plot with controled for the minimum accuracy ($a_{\min} = 1e-4$ instead of 0.0) and minimum $\epsilon_{\min}$ (0.9 of the second smallest ϵ instead of 0.0). In other words: $\texttt{auc\_log} = \text{trapz}(\log a - \log a_{\min}, \log \epsilon - \log \epsilon_{\min}) / |\log a_{\min} \times \log \epsilon_{\min}|$

3 RESULTS

3.1 Permuted MNIST task

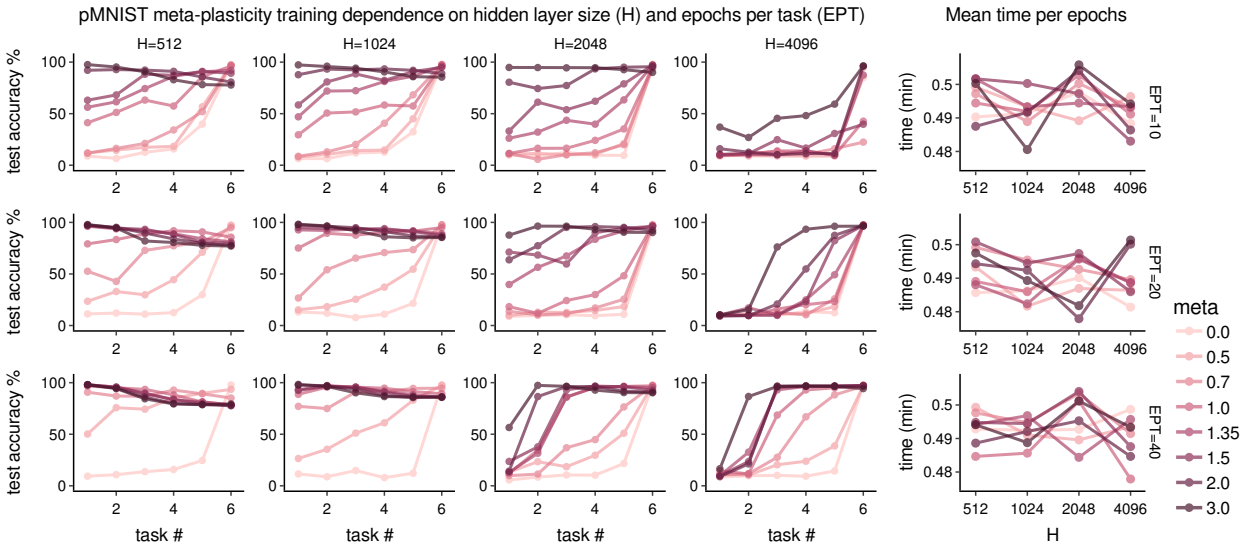


Fig. 3. Results of permuted MNIST tasks for the BNNs with metaplasticity training for different variations of `meta` (colors), network widths H (left to right), number of epochs per task (EPT, top to bottom). The leftmost column shows the average time taken to train the model.

3.1.1 Replication of BNN training with metaplasticity parameter for the pMNIST task. Before applying corruptions to the dataset, we first set out to observe whether the original results from [8] were replicable. So we ran their codes for the multipercepton (MLP) version of the BNN with 2 hidden layers for different values of the `meta` hyperparameters: low `meta` means binary hidden weights are allowed to change easily while high means such weights' own plasticity is constrained by their magnitude. In addition, we also varied the widths of the hidden layers (H) and the number of training epochs per task (EPT), for a total of only 6 tasks. This is shown in Figure 3).

Consistent with the paper, higher `meta` leads to better performance (potentially criticality is around 1.0), in which earlier tasks still perform well. Interestingly, too high `meta` could lead to some degradation (though still good) for more recent tasks - this is possibly because high values would bias most changes happening for earlier tasks and consolidate

many hidden weights from early ones. Regardless, these results confirm the benefits of the metaplasticity to solve the pMNIST task, which is one of the benchmarks for catastrophic forgetting. Surprisingly, the average time taken to train each epoch did not vary significantly between different types of variations shown here

However, contrary to the benefits from increased hidden layer widths shown in the original paper, we found that large networks could be harmful. The peak performance is around $H = 1024$ but further model width increase decreases performance even in comparison to the $H = 512$ network tested here. We hypothesized that maybe larger networks would require more time to balance consolidation and learning. Hence we varied EPT (in the paper EPT=40), but we never could replicate the size effects, for multiple runs on multiple different machine resources. Currently, we do not have any explanation for this.

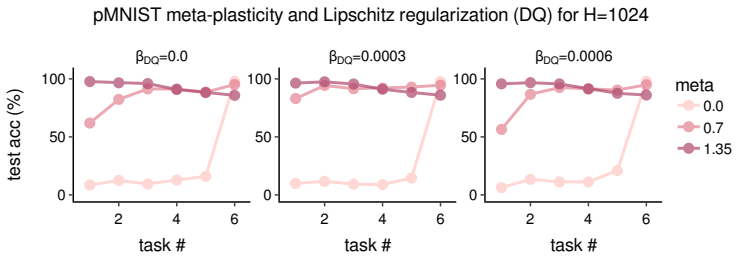


Fig. 4. The effects of Lipschitz regularization importance β_{DQ} (left to right) to the pMNIST task in addition to the metaplasticity training (colors)

3.1.2 Lipschitz regularization. In addition, we also tested the Lipschitz regularization for improving the robustness of quantized neural networks [9] on top of metaplasticity training. We chose the hyperparameter β_{DQ} similar to the original paper on defensive quantization (DQ). Intriguingly, these two processes counteract each other to some extent in Figure 4, as the medium value of β_{DQ} improves slightly catastrophic forgetting, but the largest value here degrades it, especially for medium `meta` hyperparameter.

However, one drawback of the metaplasticity method not explicitly mentioned in the main text is the reliance on task-relevant batchnorm states. According to the authors [8], technically, these states are cheap to store relative to weights, but realistically it is not explicitly clear how to load these weights automatically and effectively. Hence, moving forward with perturbation robustness testing for the pMNIST task, we removed this dependence to fairly assess the robust continual learning.

3.1.3 Natural corruptions to pMNIST tasks. Next, we moved on to applying perturbations onto our MNIST and pMNIST datasets after training the BNN models with metaplasticity optimization and Lipschitz regularization. The examples of these corruption transformations are shown in Figure 2. It must be noted that these types of corruption are possibly more realistic when considered with colored images like CIFAR or ImageNet, and the more appropriate benchmarks for catastrophic forgetting coupled with these types of perturbations would be MNIST-FMNIST or split-class CIFAR. Regardless, for now, we chose permuted MNIST as our first approach to assess catastrophic forgetting with natural corruptions.

The results are shown in Figure 5 for the models at the end of training for different values of `meta`. First, note that removing the task-relevant batchnorm states here reveals the model's accuracy for the original case, though

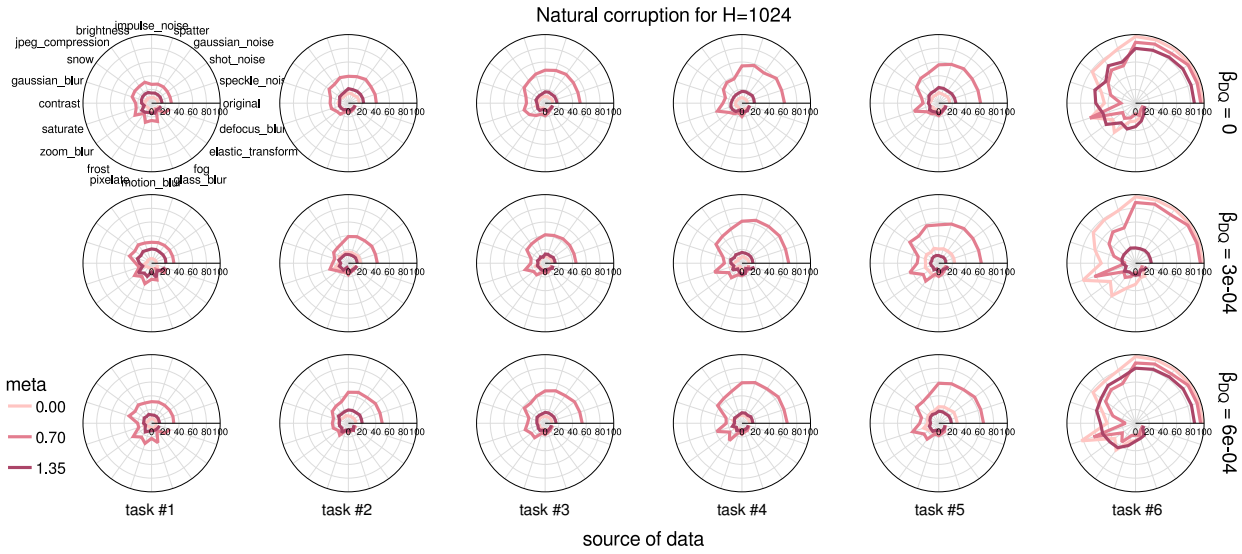


Fig. 5. Natural corruption robustness after metaplasticity training (colors) and Lipschitz regularization (top to bottom) in the pMNIST task. From left to right is the source of the task data. The radius shows the test accuracy due to these different perturbation methods (different angles, with the original at 0°).

the metaplasticity training preserves some nontrivial performance but not qualified for continual training. However, although the effect of magnitude is debatable, the inclusion of metaplasticity improves robustness against many other naturalistic corruptions, but, as expected, not so much for the most current task. The inclusion of the Lipschitz regularization has quite mixed effects, as there appears to be both cases of improvements and degradations for earlier tasks.

3.1.4 Adversarial attacks to pMNIST tasks. Lastly, for the pMNIST task, we used different adversarial attacks and calculated the `auc_log` as a way to assess the model adversarial robustness after metaplasticity and Lipschitz regularization (see Methods). The results are shown in Figure 6. First, in retrospect, this might not be the most effective way to address adversarial robustness, for example, inspecting a specific range of perturbation rather than considering the whole range would have been a better choice. Regardless, the preliminary results shown in Figure 6 reveals the mixed effects of these two processes again. However, for the baseline case without metaplasticity training, the inclusion of this β_{DQ} in the most current task (leftmost column, lightest color) unfortunately does not achieve noticeable benefits. For previous tasks, the inclusion the medium `meta` hyperparameter seems to improve robustness but the effect of β_{DQ} again does not seem clear. Future experiments should take into account adversarial training in addition to the Lipschitz regularization as suggested in the original paper [9].

3.2 Class Incremental Learning on CIFAR-100 Features

We next experimented with an incremental learning setting where the model learns different subsets of classes of CIFAR-10 and CIFAR-100 datasets. CIFAR-10 was divided into two subsets (animal subset and vehicle subset). Since CIFAR-100 consists of more classes (100 classes), we prepared six subsets based on the class labels. We use a BNN

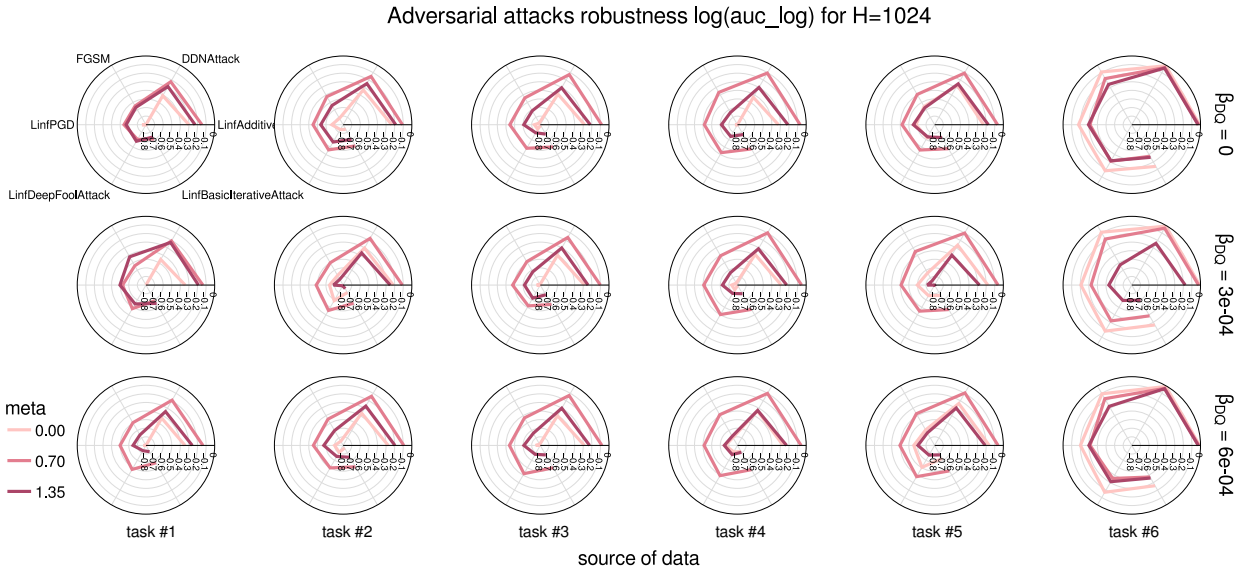


Fig. 6. Adversarial robustness after metaplasticity training (colors) and Lipschitz regularization (top to bottom) in the pMNIST task. From left to right is the source of the task data. The radius shows the calculated log (to see the difference) of auc_log due to these different perturbation methods (different angles, with the original at 0°). See methods for more information on the types of adversarial attacks and the calculation of auc_log.

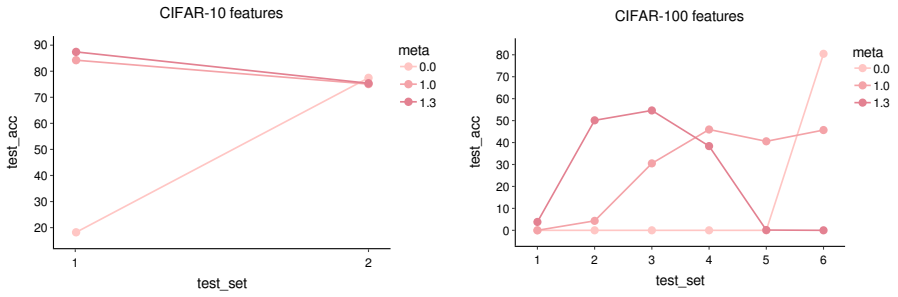


Fig. 7. Metplasticity training with the class incremental learning on CIFAR datasets.

composed of one hidden layer with 2048 neurons for CIFAR-10 and BNN composed of two hidden layers with 2048 neurons each. For the experiment we selected three `meta` values: 0.0, 1.0, and 1.3; for both of the tasks. The key idea behind these tasks is that learning features from images does not change over time.

To extract features from CIFAR-10 and CIFAR-100 datasets, we used the convolutional layers of the ResNet-18 network (removing the last layer), which is pretrained on the ImageNet dataset. CIFAR-10 and CIFAR-100 images were reshaped from 32 x 32 pixels to 220 x 220 pixels, randomly cropped using 200 x 200 windows, and randomly flipped (along the vertical axis). We made ten passes through the training dataset to extract 500,000 image features, but only performed one pass through the test dataset to extract 10,000 images. The extracted features were then trained using

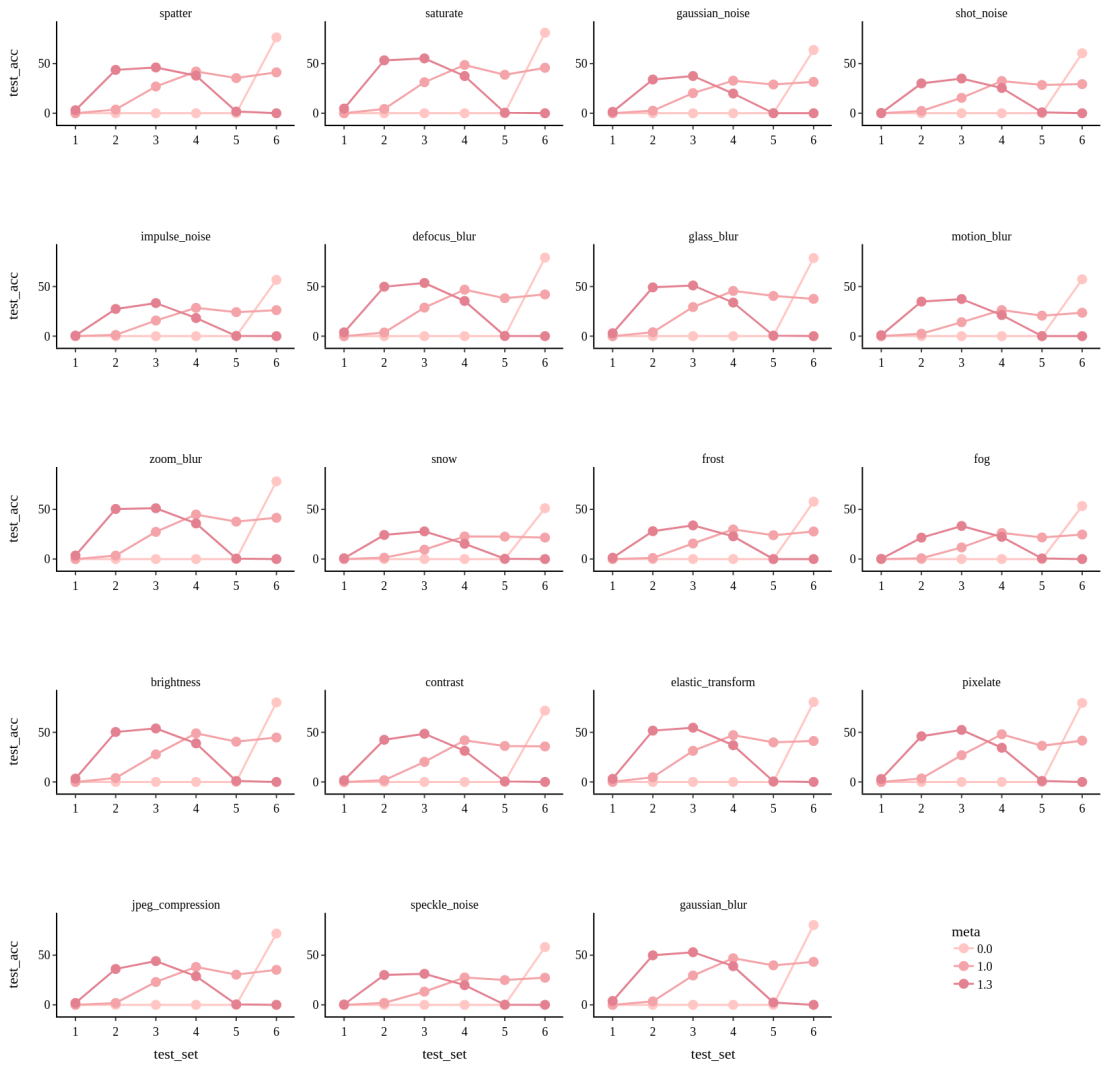


Fig. 8. Effects of applying various natural corruptions to model accuracy in case of CIFAR-100 dataset, across different `meta` parameter

binarized neural networks, with shapes 512 x 2048 x 10 and 512 x 2048 x 2048 x 100 for CIFAR-10 and CIFAR-100, respectively.

The results for metaplasticity training with CIFAR datasets are shown in Figure 7. As expected, without synaptic metaplasticity in the CIFAR-10 dataset, the model "forgot" task 1 after learning task 2. Similarly, only the current subset was "remembered" for the CIFAR-100 task. For the CIFAR-10 case, `meta` = 1.3 performed the best, but `meta` = 1.0 for CIFAR-100, giving reasonable accuracy values for the last four tasks. One strange observation arose in the case when `meta` = 1.3 for the latter case, the model could not learn the last two tasks at all. We hypothesize that, since a larger

`meta` value makes the model more rigid to changes in the binary weights, tasks 5 and 6 could not be learned. This was qualitatively similar to why high `meta` values decrease performance slightly as a function of task sequence in Figure 3.

3.2.1 Natural corruption to CIFAR split-class task. We then applied various types of natural corruption to CIFAR-10 and CIFAR-100 datasets, a more realistic approach as natural corruption is most common in the case of colored images. From Figure S2 we can observe that the model accuracy is less affected by the various natural corruptions to CIFAR-10 images. Similar behavior is also observed in Figure 8 for the CIFAR-100 dataset. One possible explanation is that the image features remain intact even after adding a small amount of noise. In fact, we introduced noise into the images in the data preparation step as we scaled up the number of pixels from 32 x 32 to 220 x 220, randomly cropped using a 200 x 200 window, and finally applied random horizontal flips.

3.2.2 Adversarial attacks on CIFAR. We finally tested the performance of BNNs with synaptic metaplasticity models under FGSM attacks (Figure 9). We can observe the model accuracy rapidly degraded as we increased the amount of epsilon (noise). Synaptic metaplasticity does not seem to provide any resistance to FGSM attack. Unfortunately, we did not have time to test a wide range of adversarial attacks except for the FGSM method for the CIFAR dataset.

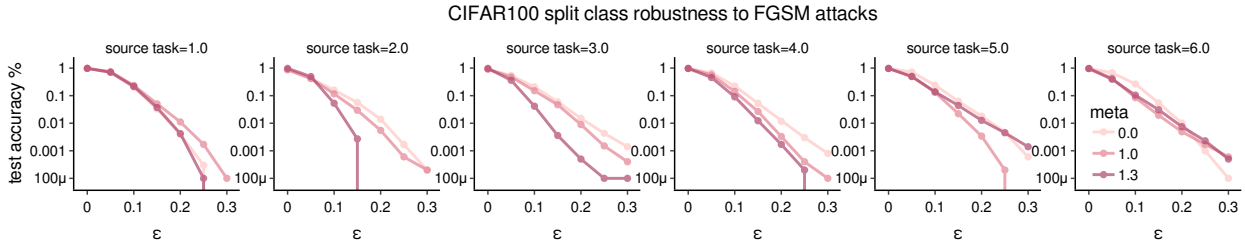


Fig. 9. FGSM adversarial attack on CIFAR-100 data of the BNN trained with metaplasticity (colors) across the different subsets (left to right), showing the test accuracy as a function of perturbation strengths ϵ.

4 CONCLUSION

While metaplasticity for BNN helped prevent catastrophic forgetting (pMNIST and split-class CIFAR tasks), the results were not robust. There appears to be a trade-off for continual learning and robustness - only medium values of `meta` would result in more robust models. Additionally, for the pMNIST task, Lipschitz regularization has mixed effects on model robustness for attacks and corruptions for the pMNIST task.

However, there were also issues within the metaplasticity training: (1) dependence on the task-relevant batchnorm states, (2) irreproducible network size benefits as seen in the paper [8]. Plus, we did not have time to assess model compression benefits and speed in comparison with real-valued networks - one reason was that metaplasticity was not effective in such networks. Hence, future experiments should involve

(1) more continual learning tasks (e.g., stream learning, MNIST-FMNIST) [2, 8, 17]
(2) more systematic comparisons (performance, time, inference, memory cost, robustness) between different known solutions for catastrophic forgetting (e.g. elastic weight consolidation [6], contextual gating [12], synaptic intelligence [18]) for neural networks across different quantization levels, as well as the continuous-valued networks

(3) combination of robustness improvement training including revisiting Lipschitz regularization [9], data augmentation for natural corruptions and adversarial training [10].

REFERENCES

[1] Matthieu Courbariaux and Yoshua Bengio. 2016. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *CoRR* abs/1602.02830 (2016). arXiv:1602.02830 http://arxiv.org/abs/1602.02830

[2] Sebastian Farquhar and Yarin Gal. 2018. Towards Robust Evaluations of Continual Learning. (May 2018). arXiv:1805.09733 [stat.ML]

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. (Dec. 2014). arXiv:1412.6572 [stat.ML]

[4] Dan Hendrycks and Thomas G. Dietterich. 2018. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *CoRR* abs/1807.01697 (2018). arXiv:1807.01697 http://arxiv.org/abs/1807.01697

[5] Navid Khoshavi, Connor Broyles, and Yu Bi. 2020. Compression or Corruption? A Study on the Effects of Transient Faults on BNN Inference Accelerators. 99–104. https://doi.org/10.1109/ISQED48828.2020.9137006

[6] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U. S. A.* 13 (March 2017), 3521–3526. https://doi.org/10.1073/pnas.1611835114 arXiv:1612.00796 [cs.LG]

[7] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial Machine Learning at Scale. (Nov. 2016). arXiv:1611.01236 [cs.CV]

[8] Axel Laborieux, Maxence Ernoult, Tifenn Hirtzlin, and Damien Querlioz. 2021. Synaptic metaplasticity in binarized neural networks. *Nature Communications* 12, 1 (Dec 2021), 2549. https://doi.org/10.1038/s41467-021-22768-y

[9] Ji Lin, Chuang Gan, and Song Han. 2019. Defensive Quantization: When Efficiency Meets Robustness. *CoRR* abs/1904.08444 (2019). arXiv:1904.08444 http://arxiv.org/abs/1904.08444

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. (June 2017). arXiv:1706.06083 [stat.ML]

[11] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. 2018. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *CoRR* abs/1802.01569 (2018). arXiv:1802.01569 http://arxiv.org/abs/1802.01569

[12] Nicolas Y Masse, Gregory D Grant, and David J Freedman. 2018. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. (Feb. 2018). arXiv:1802.01569 [cs.LG]

[13] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2574–2582. https://doi.org/10.1109/CVPR.2016.282

[14] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*. http://arxiv.org/abs/1707.04131

[15] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. 2020. Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX. *Journal of Open Source Software* 5, 53 (2020), 2607. https://doi.org/10.21105/joss.02607

[16] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. 2018. Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. (Nov. 2018). arXiv:1811.09600 [cs.CV]

[17] Gido M van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. (April 2019). arXiv:1904.07734 [cs.LG]

[18] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual Learning Through Synaptic Intelligence. (March 2017). arXiv:1703.04200 [cs.LG]

5 SUPPLEMENTAL FIGURES

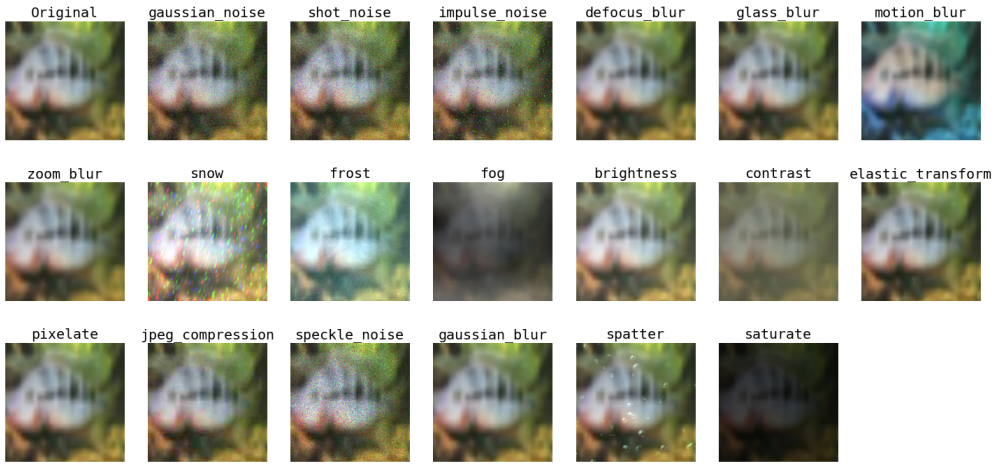


Fig. S1. Natural corruption applied to the CIFAR dataset.

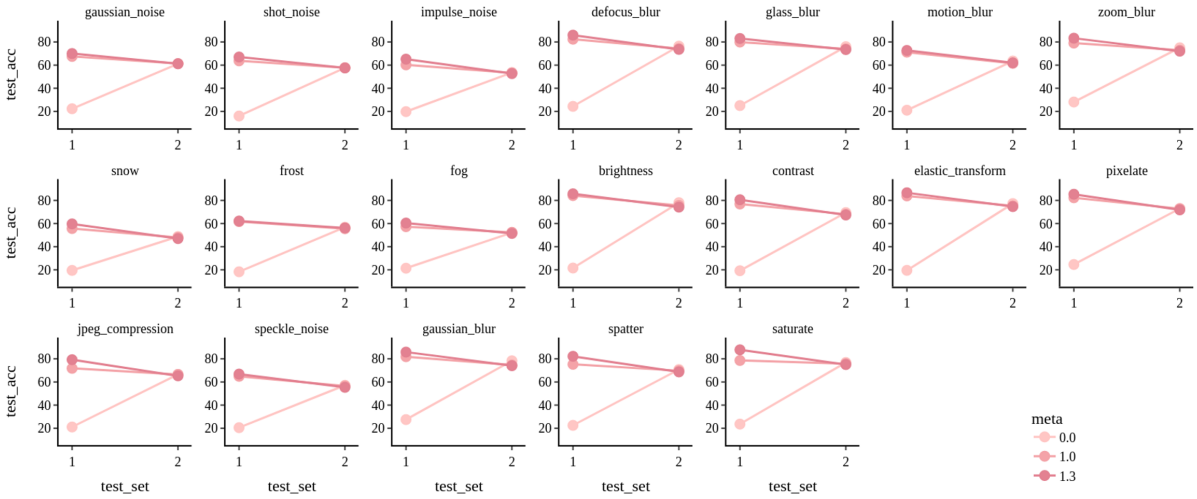


Fig. S2. Effects of applying various natural corruptions to model accuracy in the case of the CIFAR-10 dataset.