Do solutions for catastrophic forgetting degrade model robustness to naturalistic corruption and adversarial attacks? *CMSC 35200 Project Mid Report*

Tuan Pham Amit Pradhan Sebastian Barrios

[tuanph18@uchicago.edu](mailto:tuanph18@uchicago.edu) [pradhanak@uchicago.edu](mailto:pradhanak@uchicago.edu) [sebastian.barrios@chicagobooth.edu](mailto:sebastian.barrios@chicagobooth.edu)

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, there are multiple issues that 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 easier 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 and continual learning problems for BNN [2]. More specifically, by creating a form of multiplicative gating during learning for hidden weights to represent weight consolidation, 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 [3].

Back to BNNs, due to such quantization, they can be quite sensitive to corruption, for example data adversarial attacks [4] or potential soft errors in hardware accelerators [5]. Hence, we wish to ask whether previous solutions for CF [2] could further degrade model robustness to data corruption or help ameliorate it.

In conclusion, we propose a plan to (i) replicate the selected studies that tackle the topic of catastrophic forgetting, and (ii) observe whether solutions to avoid CF lead to models that have a higher level of degradation when different types of naturalistic corruption to input data are introduced [6], as well as the potential resulting vulnerability of these models to adversarial attacks [4].

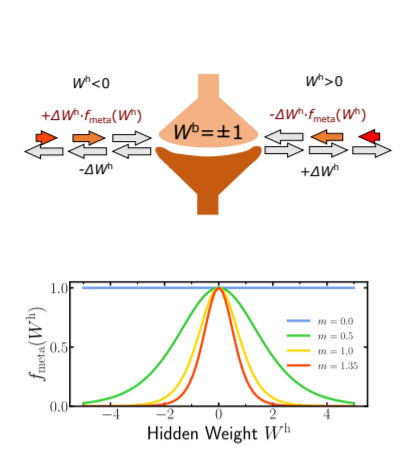
Background

Recent advancements in deep learning made it possible to achieve human level performances in multiple tasks. Techniques like batch normalization, dropouts, residual connections along with suitable hidden weight initialization methods made it possible to train very deep neural networks. However neural networks still don’t perform well when trained on multiple tasks sequentially. They tend to forget the previous tasks as new tasks are trained. This problem occurs due to the changes to the hidden weights as we train a new task. Another important task where catastrophic forgetting is natural is stream learning: the model learns only one task but it has access to one subset at a time. The model cannot access prior datasets at any point of time.

Binarized neural networks use binary weights and activations during inference phase and during training phase binary weights and activations are used for computing the gradients. BNNs present an attractive alternative as they significantly reduce the memory size and accesses and replace arithmetic operations with bitwise operations. So this leads to lower power consumption and faster inference.

BNNs are also prone to catastrophic forgetting when trained on multiple tasks sequentially. In this work, we have used BNNs to study different solutions to catastrophic forgetting. One such solution to catastrophic forgetting is inspired by neuroscience. This approach studies how the human brain is able to learn and retain multiple tasks for a longer period of time. Neuroscience literature suggests that the human brain uses hidden weights within each synapse to control the metaplasticity of each synapse. Higher the value of a synapse, it’s value is more consolidated and less likely to change. Metaplasticity controls the importance of the tasks leant and provides a solution to catastrophic forgetting. Since BNNs have hidden weights along with binary weights, it makes sense to apply the idea of metaplasticity to BNNs.

This solution views the hidden weights of the BNNs as metaplastic variables and changes the training algorithm of BNN slightly. As shown in **Fig. 0**, it introduces a function *f*meta whose behavior depends on the value of meta. The gradient updates are now multiplied by *f*meta which means *f*meta now controls plasticity or rigidity of the neuron. For a larger value of meta, the neuron becomes more rigid (less plastic) even for small weight values. The basic intuition behind this idea is, when weights of neurons become larger that means they are important to specific tasks and their values should not be changed by subsequent tasks.



**Figure 0**: *f*meta is shown for different values of meta[2].

Code repository available at:

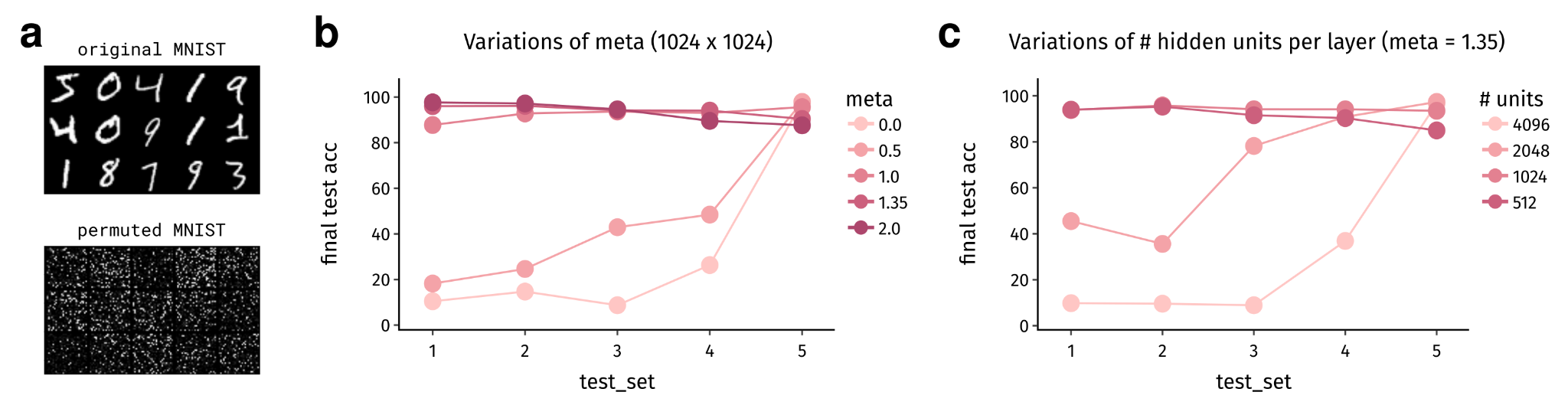
<https://github.com/tuanpham96/bnn-cf-vs-robust>

Results

Experiment 0: *Replication of BNN training with metaplasticity parameter with permuted MNIST*

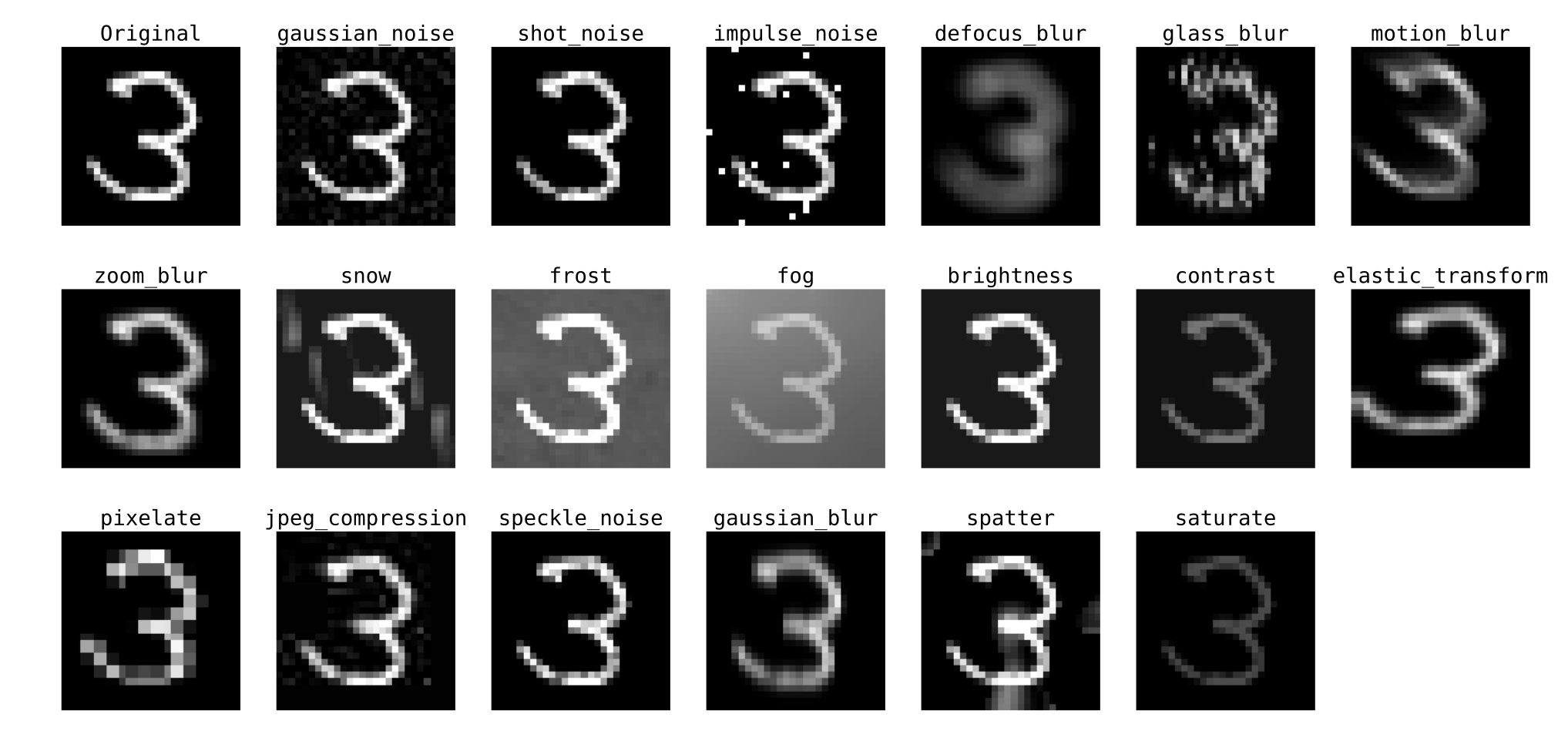
Before applying corruptions to the dataset, we first set out to observe whether the original results from [2] were replicabled. So we ran their codes for the multipercepton (MLP) version of the BNN with 2 hidden layers (1024 units each) for different values of the hyperparameters: low means binary hidden weights are allowed to change easily while high means such weights’ own plasticity is constrained by their magnitude. This is shown in **Fig. 1**b. Consistent with the paper, higher leads to better performance (potentially criticality is around 1.0), in which earlier tasks still perform well. Interestingly, too high value 2.0) might lead to performing not as well (though still good) for the current task - this is basically because high values would bias most changes happening for earlier tasks and would require longer time to learn the current task. Regardless, these results confirm the benefits of the metaplasticity to solve the permuted MNIST task, which is one of the benchmarks for catastrophic forgetting.

Additionally, we also try to replicate the relationship between the sizes of the hidden layers and the performance (**Fig. 1**c). Contrary to the monotonic trend between the trend and the performance shown in the original paper, we found that large networks could be harmful. For now, we are unsure why. One of the reasons might be that we only allowed for 20 epochs per task for this benchmark while the original paper allowed for 40 epochs. Maybe larger networks require a bit more time to consolidate properly. Another possible explanation might be, in very large networks there will be a large number of unconsolidated neurons and they might degrade an earlier task performance.



**Figure 1** *Results of permuted MNIST tasks for the BNNs with metaplasticity training.* (**a**) Example of original (top, task 1) and generated permuted (bottom, after task 1) MNIST dataset. (**b**) Performance of the BNN (MLP, 2 hidden layers, 1024 units each) with different values of metaplasticity hyperparameter (**c**) Performance with variations in the number of hidden units per layer with

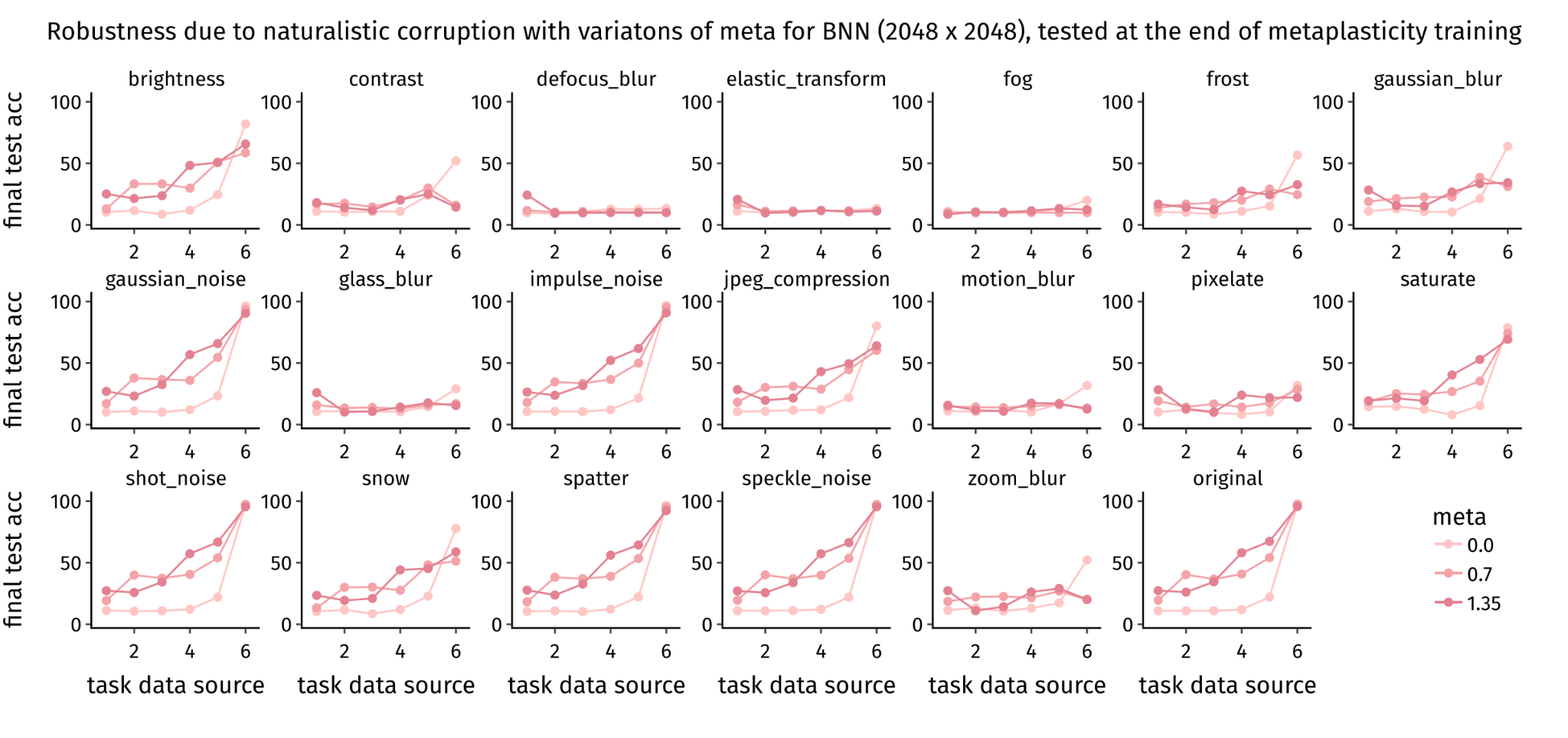
Experiment 1.1: *Metaplasticity-trained BNN in response to natural corruptions to pMNIST tasks*



**Figure 2** *Examples of the natural corruptions for MNIST data, plotted with similar color limits*

Next we moved on to applying perturbations onto our MNIST and pMNIST datasets after training the BNN models with metaplasticity optimization. First, we modified the code from [6] to allow for monochrome natural corruptions on our (p)MNIST datasets. Because some of these transformations are not easily standardized for tensor transformation, these corrupted data sets are pre-generated before training, to save time during testing. The examples of these corruption transformations are shown in **Fig. 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-FashMNIST 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 **Fig. 3** for the models at the end of training for different values of , tested for the original (last panel) test data sets of all tasks (i.e. *task data source*) and the natural corruptions shown in **Fig. 2**. For many different perturbations (for example *brightness* or *shot\_noise, impulse\_noise*), at least the inclusion of the metaplasticity training seems to help safeguard the model against the corruption for earlier tasks, though there’s not a consistent trend for the actual magnitude of . At the same time, for many of these transformations, no improvement is observed regardless. It must be noted that these models are 2048 x 2048, in which **Fig. 1**c already shows that this particular setting is not optimal.

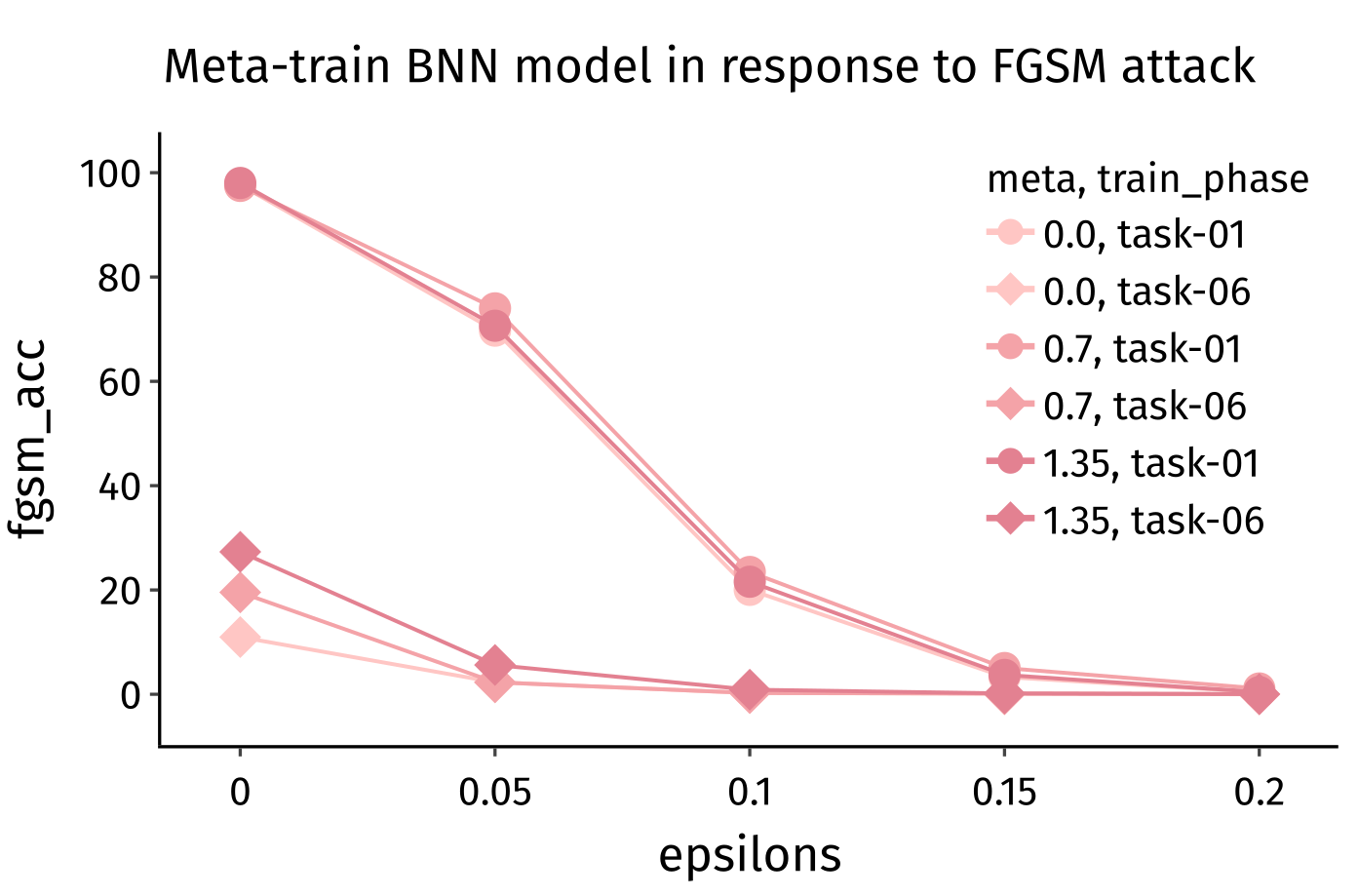


**Figure 3** *Test performance of the BNN models at the end of metaplasticity (colors) training in response to different monochrome natural corruptions (panels, original is last) generated from different tasks.* The model here is MLP BNN with 2 hidden layers of 2048 units each. See **Fig 2.** for demonstrations of natural corruption. The x-axis for each panel represents where the original test data come from, e.g. *task-2* in the *shot\_noise* panel means that the test dataset is the *task-2* (permuted MNIST) that gets corrupted with *shot\_noise* corruption.

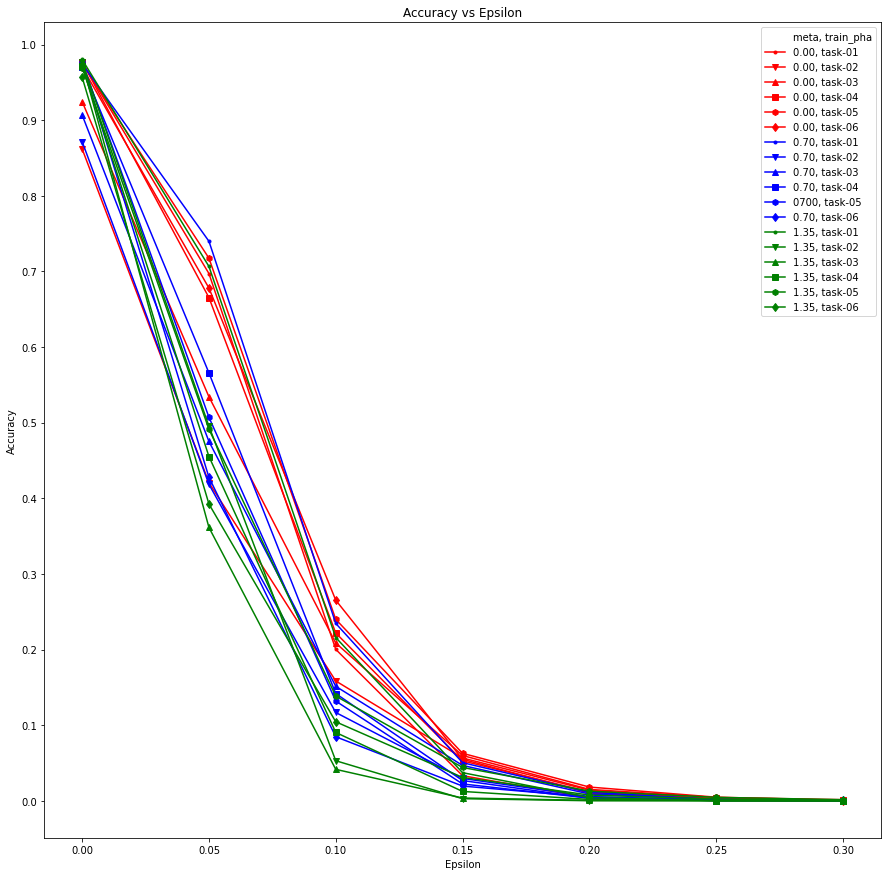
Experiment 1.2: *Metaplasticity-trained BNN in response to adversarial attacks to original MNIST and PMNIST data*

We have reproduced the effectiveness of a white-box adversarial attack ‘*Fast Gradient Sign Attack* (FGSM)’ with the goal of misclassification on BNN. Adversarial attacks in general try to add the least amount of perturbations to the input data to cause the desired misclassification. FGSM is a very powerful yet intuitive attack. For a given input, FGSM calculates the gradients of the loss function with respect to the input and adjusts the input by adding a small perturbation in the directions of the gradients to cause the misclassification.

We have examined the attacks on the original MNIST dataset (i.e. *task-01*) with the models trained like above. The results are shown in **Fig. 4** (some examples are shown in **Fig. 5**), illustrating that BNN is quite sensitive to adversarial attacks and continual learning worsens the performance significantly. The metaplasticity process does not seem to lead to much improvement to guard against FGSM attacks, potentially only very little for quite small perturbation strengths. However, we are unclear yet how accuracy could reach below chance here. **Fig. 6** shows the effects on FGSM attack on BNN trained with six PMNIST tasks. Similar to the MNIST dataset, we observe, BNN is quite sensitive to FGSM attacks and accuracy of the model degrades significantly even for small epsilon values. For larger meta values, the model performance seems to degrade fast.



**Figure 4** *FGSM adversarial attack on MNIST data of the BNN trained with metaplasticity during permuted MNIST task for 2 different training phases of the model*. The test accuracy due to FGSM attacks is shown as a function of the perturbation strengths *epsilons*. Trained at *task-01* refers to the end of the training phase in which the model was trained at MNIST dataset, while *task-06* means that the model was the end of the entire metaplasticity training process. Only attacks on the original MNIST were tested here.

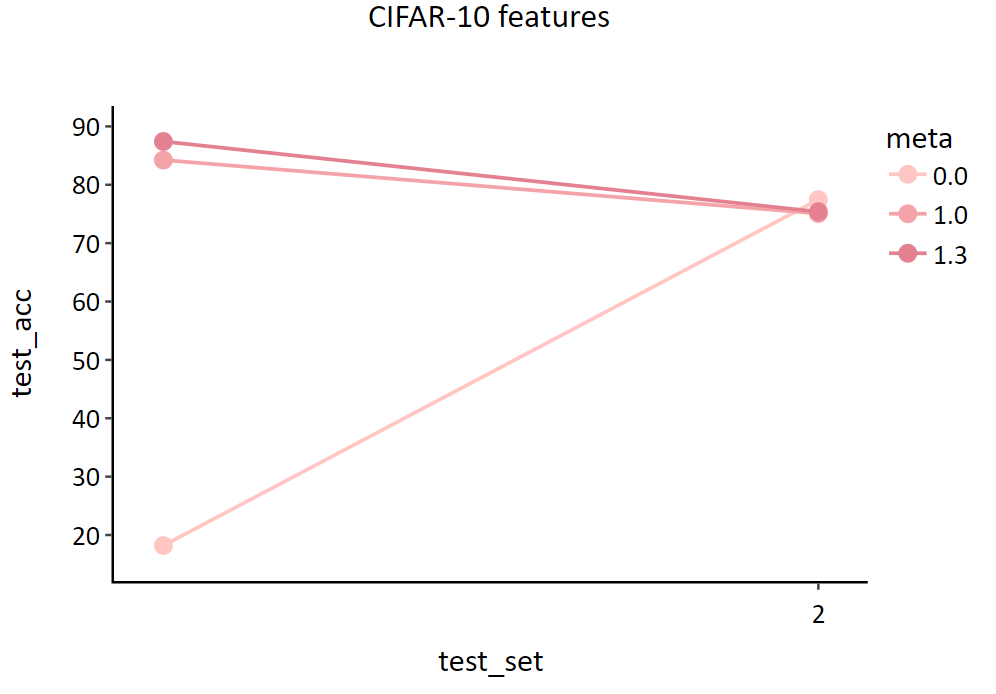


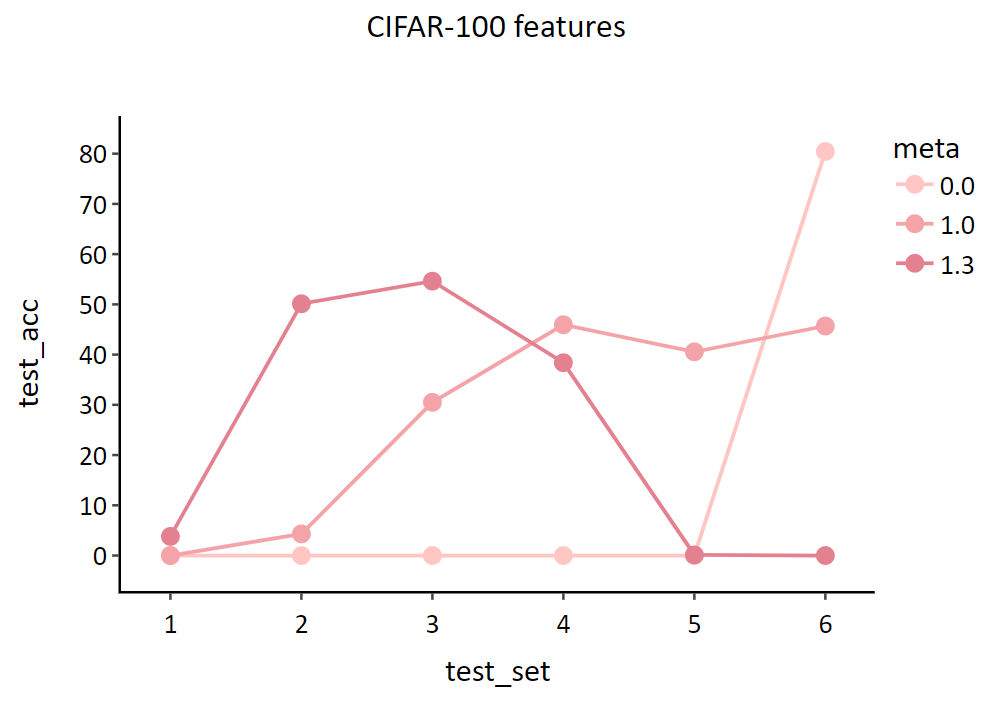
**Figure 6:** FGSM adversarial attack on BNN trained with metaplasticity on PMNIST tasks.

Experiment 1.3: *Class Incremental Learning on CIFAR-10 and CIFAR-100 Features*

We now experiment with an incremental learning setting where the model learns different subsets of classes of CIFAR-10 and CIFAR-100 datasets. CIFAR-10 is divided into two subsets i.e. 5 classes are grouped under animals and vehicles each. Since CIFAR-100 consists of more classes (100 classes), we prepare 6 subsets based on the class labels. We use a binarized neural network (BNN) with one hidden layer with 2048 neurons for CIFAR-10 and BNN with two hidden layers with 2048 neurons each. We have used three meta values i.e 0.0, 1.0 and 1.3 for both of the tasks. The key idea behind these tasks is, the ability to learn features from images does not change over time. For example, one does not forget how to recognize the shapes but they might forget what those shapes are (the abstract concepts).

To extract features from CIFAR-10 and CIFAR-100 datasets, we use the convolutional layers of the ResNet-18 network (remove the last layer) which is pretrained on the Imagenet dataset and readily available in pytorch. CIFAR-10 and CIFAR-100 images are reshaped from 32 \* 32 pixels to 220 \* 220 pixels, randomly cropped using a 200 \* 200 windows and random horizontal flips are applied. We make 10 passes through the train dataset to extract 500,000 images features but we only perform one pass through the test dataset to extract 10,000 images. The extracted features are trained using binarized neural networks (BNNs), 512 \* 2048 \* 10 and 512 \* 2048 \* 2048 \* 100 for CIFAR-10 and COFAR-100 respectively.

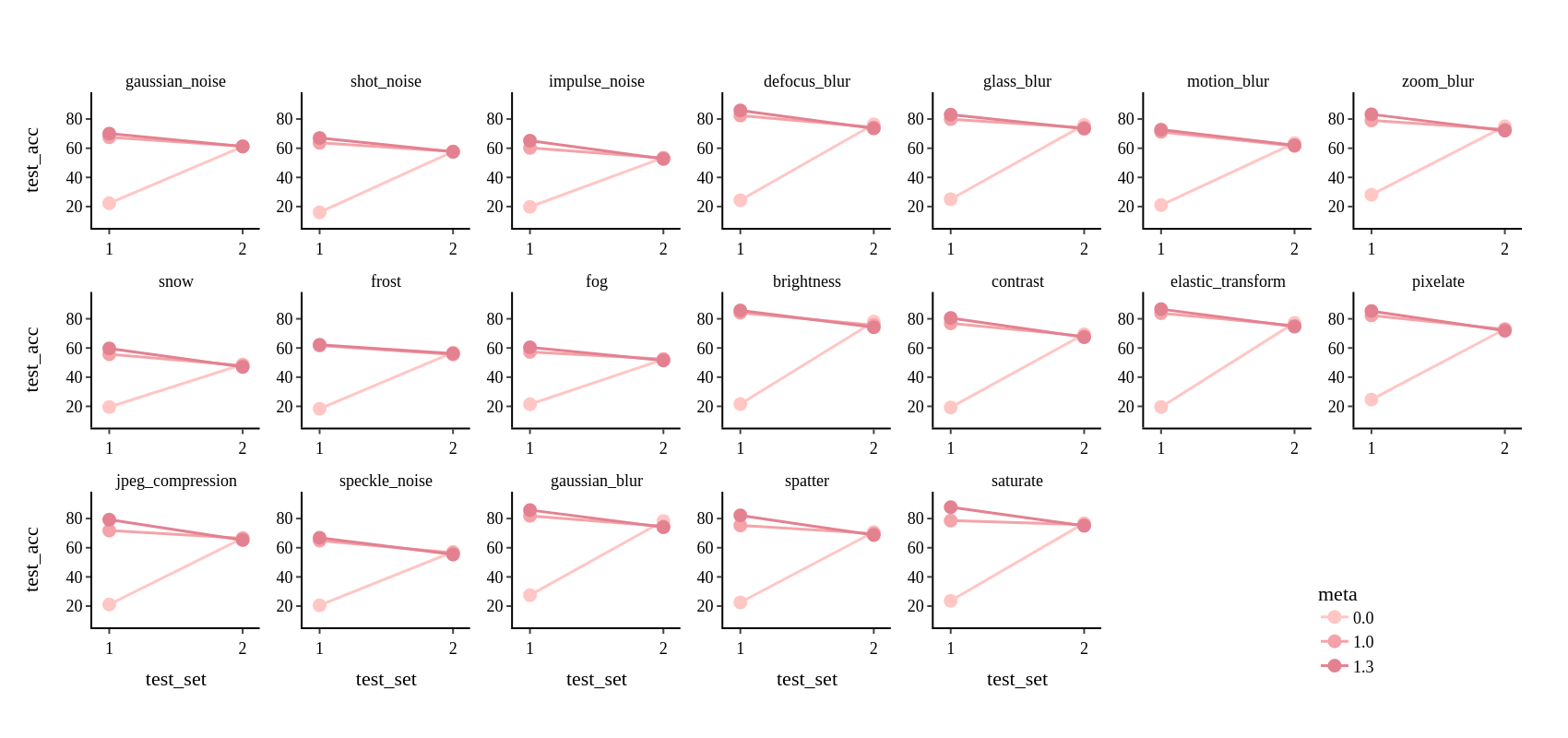




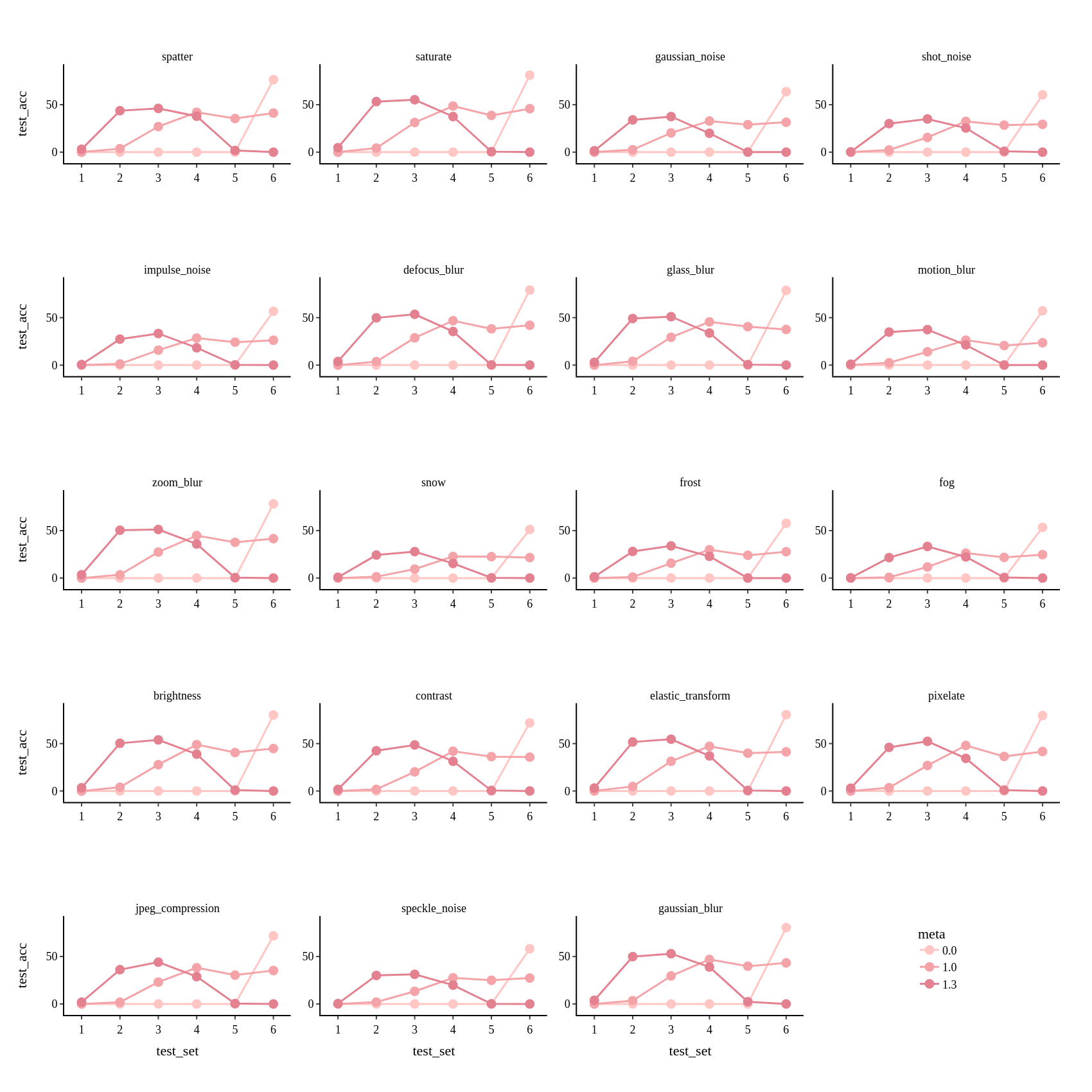
**Figure 7**: Accuracy of each task at the end of training 2 and 6 tasks for CIFAR-10 and CIFAR-100 respectively. Three different meta values are used for this experiment.

**Fig 7** shows accuracy values for each task at the end of training 2 and 6 tasks for CIFAR-10 and CIFAR-100 respectively. For the CIFAR-10 case, we can see that a meta value of 1.3 gives us the best results. When we use a meta value of 0.0 (without synaptic metaplasticity), the model forgets task 1 after learning task 2. In the case of CIFAR-100, we can see the model forget all the tasks except the last one when trained without synaptic metaplasticity. Meta value of 1.0 gives the best results in the CIFAR-100 case, giving reasonable accuracy values for the last four tasks. One strange observation in the case meta = 1.3, the model could not learn the last two tasks at all. Since a larger meta value makes the model more rigid to changes in the binary weights, task 5 and 6 could not be learnt. We need to train more epochs to learn tasks 5 and 6 for large meta values or we can increase the number of neurons in the model.

We now apply various types of natural corruption to CIFAR-10 and CIFAR-100 datasets as natural corruption is most common in the case of colored images. From **Fig 8** we can observe that the model accuracy is less affected by the various natural corruptions to CIFAR-10 images. Similar behaviour is also observed in **Fig 9** in the case of CIFAR-100 dataset. One possible explanation is, the image features remain intact even after addition of a small amount of noise. In fact, we introduce noise into the images in the data preparation step as we scale up the number of pixels from 32 \* 32 to 220 \* 220, crop randomly using a 200 \* 200 window and finally apply random horizontal flips.



**Figure 8**: Effects of applying various natural corruptions to model accuracy in case of CIFAR-10 dataset

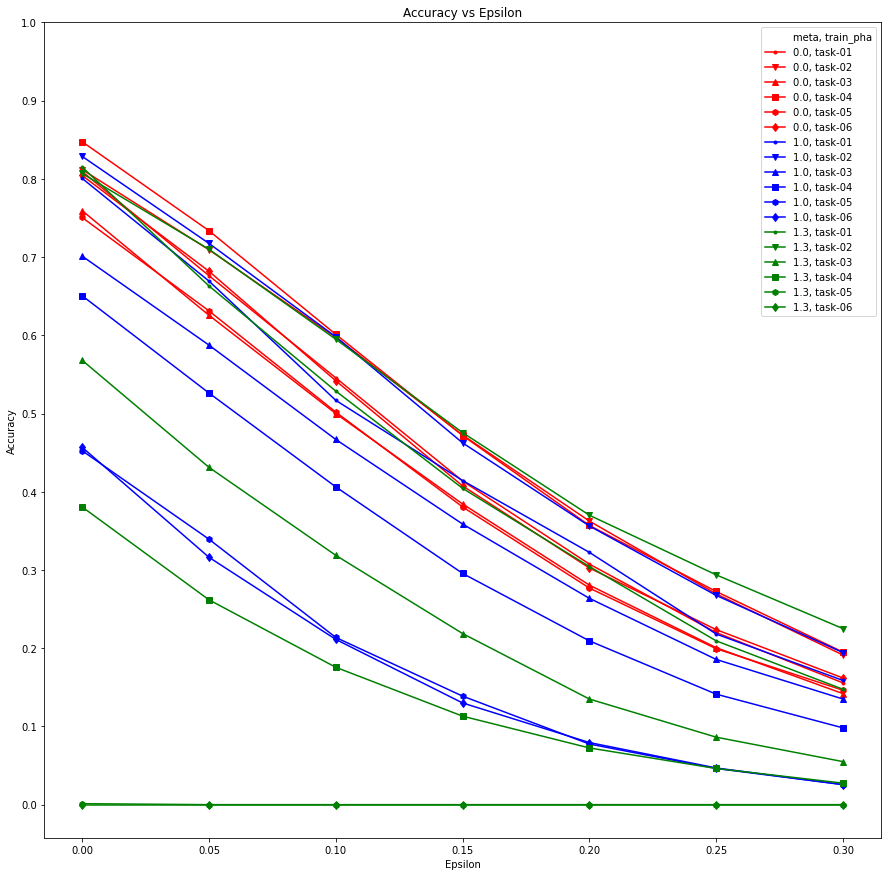


**Figure 8**: Effects of applying various natural corruptions to model accuracy in case of CIFAR-100 dataset

We now observe performance of BNN with synaptic metaplasticity models under FGSM attacks. We observe the model accuracy rapidly degrades as we increase the amount epsilon (noise). Synaptic metaplasticity does not seem to provide any resistance to FGSM attack.



**Figure 9**: Performance of BNN models under FGSM attacks in the case of CIFAR-10



**Figure 10**: Performance of BNN models under FGSM attacks in the case of CIFAR-100

Conclusion

From the above experiments, we can conclude synaptic metaplasticity safeguards BNNs from catastrophic forgetting, but the exact amount of performance gain depends on the type of the task and the architecture of the network. Very high values of meta make the model too rigid to the changes in binary weights and the model can not learn a large number of tasks. In such cases, we might have to increase the model parameters by adding more layers to the network. This technique is very useful especially in the case of BNNs as BNNs already have hidden weights along with the quantized weights used in the inference phase. However this approach does not provide any significant gain against natural corruption and FGSM adversarial attacks. Another drawback of the synaptic metaplasticity method is that batch normalization layer weights need to be stored for each task. Without saving and loading corresponding batch normalization layer weights for the task, the BNN performs very badly in the inference phase.

References

[1] Courbariaux, Matthieu, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. “Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1.” *arXiv [cs.LG]*. arXiv. <http://arxiv.org/abs/1602.02830>.

[2] Laborieux, Axel, Maxence Ernoult, Tifenn Hirtzlin, and Damien Querlioz. 2021. “Synaptic Metaplasticity in Binarized Neural Networks.” *Nature Communications* 12 (1): 2549.

[3] Masse, Nicolas Y., Gregory D. Grant, and David J. Freedman. 2018. “Alleviating Catastrophic Forgetting Using Context-Dependent Gating and Synaptic Stabilization.” *arXiv [cs.LG]*. arXiv. <http://arxiv.org/abs/1802.01569>.

[4] Lin, Ji, Chuang Gan, and Song Han. 2019. “Defensive Quantization: When Efficiency Meets Robustness.” *arXiv [cs.LG]*. arXiv. <http://arxiv.org/abs/1904.08444>.

[5] Khoshavi, Navid, Connor Broyles, and Yu Bi. 2020. “Compression or Corruption? A Study on the Effects of Transient Faults on BNN Inference Accelerators.” In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, 99–104.

[6] Hendrycks, Dan, and Thomas G. Dietterich. 2018. “Benchmarking Neural Network Robustness to Common Corruptions and Surface Variations.” *arXiv [cs.LG]*. arXiv. <http://arxiv.org/abs/1807.01697>.

[7] Leavitt, Matthew L., and Ari S. Morcos. 2020. “On the Relationship between Class Selectivity, Dimensionality, and Robustness.” *arXiv [cs.LG]*. arXiv. <http://arxiv.org/abs/2007.04440>.

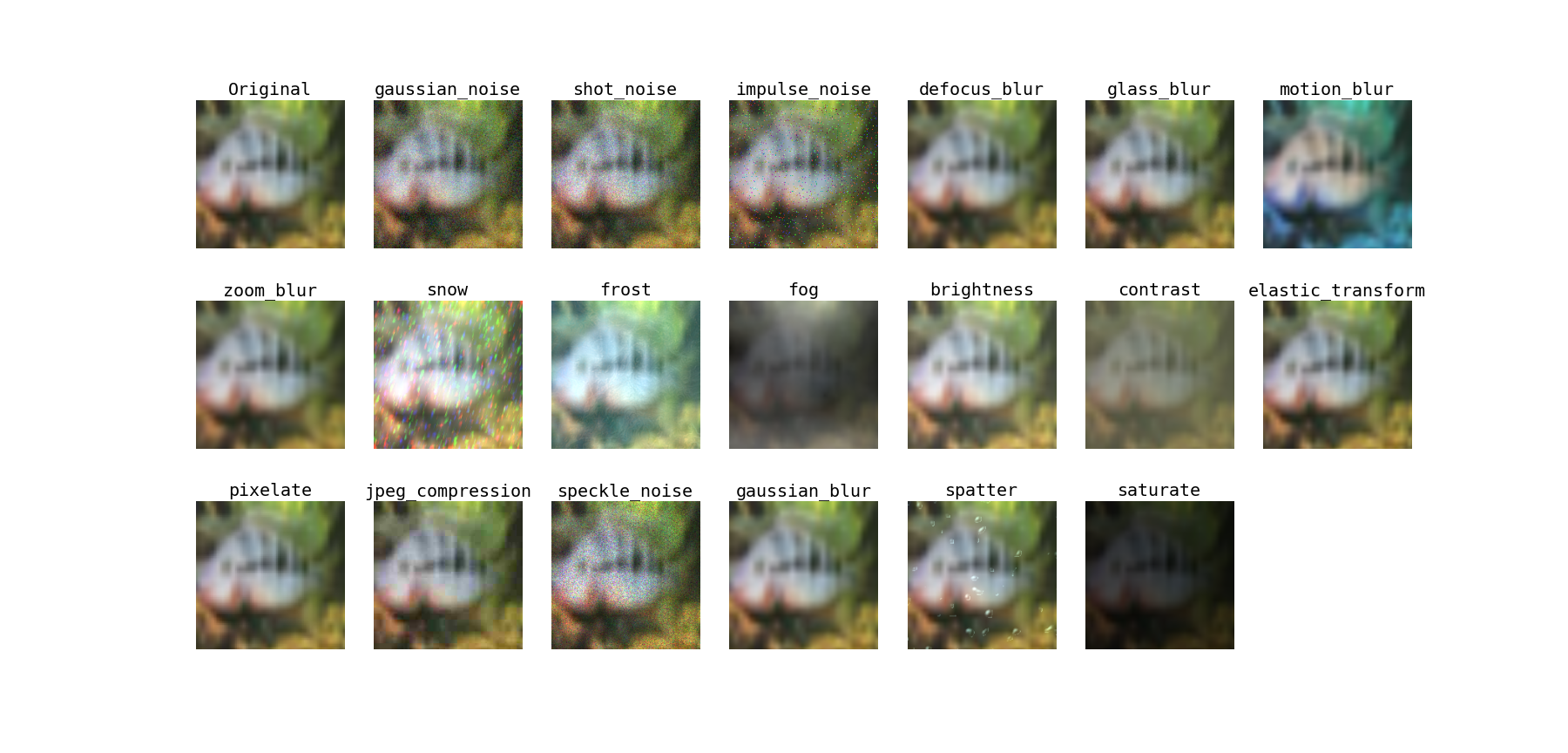
[8] Nathan Inkawhich, ADVERSARIAL EXAMPLE GENERATION, <https://pytorch.org/tutorials/beginner/fgsm_tutorial.html#adversarial-example-generation>



**Figure 5:** *The input images for different values of epsilon*. This is for FGSM attack on MNIST dataset for model trained at the end of the *task-01* (i.e. original MNIST) data without any metaplasticity.



**Figure 11**:*Examples of different types of natural corruptions applied to CIFAR-10 images*



**Figure 12**:*Examples of different types of natural corruptions applied to CIFAR-100 images*