Performance Evaluation on Transformers augmented with Hebbian and Gradient based plasticity

Brianna Grissom **Christina Wang**

1 Background

The strong capability and potential of Transformers has been proven by the extensive computational analysis done by OpenAI. When increased to a very large scale (from 125M to 175B parameters), transformers are capable of generalizing NLP tasks after training on a large and extensive dataset, without extensive task-specific finetuning [1]. Thus, a lot of research has been trying to either investigate the learning abilities of the transformer model, or aiming to improve their architecture so that they learn more efficiently.

One thing Transformers, or all general machine learning models suffer from is the inefficiency to quickly process new information that can be derived from already learned data after post-training.

Inspired by the famous Hebbian Plasticity Rule that updates weights synchronously every time it encounters new data, Duan et al researches on the prospect of implementing plasticity on RNN models, based on both the famous Hebbian plasticity rule and their newly proposed plasticity rule derived from Gradient descent. [3]

The project done by Siddharth Chaudhay then follows up their work by implementing both the Hebbian and Gradient plasticity rule on decode-only Transformers, and does a computational test of several tasks to investigate their performance.[2]

In this report, we will first introduce the Hebbian and Gradient Plasticity rules and its implementation, then follow up the work of Siddharth Chaudhay and design more tests to evaluate and compare the performance of the two models.

2 Hebbian & Gradient Plasticity

The inability of Transformers to permanently store new information encountered post-training does not allow them to learn new things. To solve this problem, decoder-only Transformers are augmented with a fast-weight component that allow for in-sequence adaption, meaning that the model can adjust its behavior quickly as it processes new data.

In the training stage, the model undergoes forward passes that updates its fast weights every time it encounters new data, also known as the "inner loop". Meanwhile, the Transformer's static weights are updated via gradient descent at a much less frequent rate, also known as the "outer loop".

In the testing stage, the static weights remain fixed while the fast weights, initialized to zero at the start of each new sequence, continue to be updated for each new sequence through the model's in-sequence plasticity mechanisms. This allows the model to adapt to new information and learn within the context of that specific sequence while keeping the static weights fixed.

The position-wise feed-forward neural networks have the weight matrix

$$W_l(t) = \tilde{W}_l(t) + w_l(t)$$

for each plastic layer l, where $\tilde{W}_l(t)$ are the static meta-trained weights optimized by an outer loop backpropagation process and $w_l(t)$ are the fast weights initialized to zero at the start of each new

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

sequence and are updated locally during the forward pass. The inner loop does its fast weight updates using either a Hebbian or a gradient-based plasticity rule.

Figure 1 presents pseudocode for the training algorithm [2].

Algorithm 1 Outer–Inner Meta-Training for Plastic Transformers

while not converged **do**
  Sample batch of sequences $\mathcal{T}_i \sim \mathcal{T}$
  for each sequence $\mathcal{T}_i$ **do**
    Initialize fast weights $w_l(0) \leftarrow 0$ for all plastic layers
    for time step $t = 1..T$ **do**
      Compute output $o_t = f(x_t; \tilde{W} + w(t))$
      Compute modulation signal $\eta(t)$
      Update $w_l(t) \leftarrow (1 - \eta(t))w_l(t-1) + \eta(t)\alpha_l \circ \Delta w_l(t)$
    end for
  end for
  Update static parameters $\tilde{W}_l, \alpha_l$ via gradient descent on accumulated meta-loss
end while

Figure 1: Pseudocode for Training Algorithm

α_l is a connection-specific learning rate and $\eta(t)$ represents the modulation signal that controls how strongly the fast weights are updated, allowing for context-dependent plasticity. At each step, the model outputs the token prediction y_t and variables $(\tilde{\eta}_t, \bar{y}_t)$ that control the modulation signal $\eta(t)$. $\tilde{\eta}_t$ acts as a neuromodulation signal and $\bar{y}_t$ is a vector produced by a small head alongside the token prediction.

Hebbian Plasticity Rule

Hebbian Plasticity is a biologically inspired mechanism that updates the fast weights based on pre- and post-synaptic activity correlations. This allows the Transformer to store short-term associations. Let $p_l(t)$ denote the input (the pre-synaptic activations) and $q_l(t)$ denote the layer output (the post-synaptic activations) for layer l of the feed forward neural network. The Hebbian plasticity update for the fast weights is

$$\Delta w_l(t) = (p_l(t)q_l^T(t)),$$

where the neuromodulation factor $\eta(t)$ is

$$\eta(t) = \eta_0 \sigma(\tilde{\eta}_t) \times \min\left(1, \frac{\text{max\_norm}}{\|\delta_t\|_2}\right) \text{ and}$$

$$\delta_t = \text{Concat}\big(\text{Vec}(p_l(t)q_l^\top(t)) \mid l \in S\big).$$

η_0 is a scalar hyperparameter than controls the maximal learning rate, $\sigma(\tilde{\eta}_t)$ is the sigmoid function applied to $\tilde{\eta}_t$, max_norm is a constant used to prevent aggressive or unstable changes to the fast weights, and δ_t encodes the Hebbian weight change pattern over all plastic layers at time t.

Interpretation: Strong correlations between pre-synaptic and post-synaptic neurons captured in the dot product term indicate that their connection strength (fast weight) is modified significantly, making the neuron become more sensitive to that input component in the future.

Gradient Plasticity Rule

Gradient Platicity is a modern framework that makes use of gradient descent on global information to update the pre- and post-synaptic weights [3]. For a customized global loss function L, The update for each weight vector:

$$\Delta w_l(t) \propto \frac{\delta L}{\delta w_l}$$

While the traditional Hebbian rule does not care about a bias term b present in each layer of the network, the Gradient rule by Duan et al also adds plasticity to the bias term:

$$\Delta b_l(t) \propto \frac{\delta L}{\delta b_l}$$

The neuromodulation factor $\eta(t)$ is derived similarly as the the Hebbian Plasticity Rule.

3 Experiments

The following experiments will test two hypotheses:

1. How well the model can remember recent information (short-term memory)
2. How well the model can quickly learn a new rule as it processes a single sequence (rapid, in-sequence learning)

Each experiment is run with $S = 3$ seeds and reported metrics are mean $\pm$ standard deviation across seeds.

During training, the fast weights are updated every step via the Hebbian or gradient rule and the static weights are updated once per sequence using the task loss accumulated across that sequence.

During testing, the models adapt only through the fast weights.

3.1 Reproduced Experiments

3.1.1 Copy Task

In this task, the Transformer model first is presented n tokens, then is shown 20 distractor inputs which are all blank tokens. Finally, it has to predict the original n tokens again. This is an example of a short-term memory task.

The metrics tracked here are validation loss and recall accuracy. Their experiment uses 50 episodes per epoch for 2 epochs with state reset between sequences.

The results are shown together with the next experiment in table 2 and figure 2.

3.1.2 One-Shot Image Classification

This experiment is an image classification task on the CIFAR-FS and Omniglot datasets, representing rapid, in-sequence learning.

There are two phases of this experiment: support and query.

In the Support Phase, the Transformer sees $N \times K$ labeled examples, where $N = 5$ and $K = 1$. The labeled examples take the form [embedding, one-hot-label, 0], where 0 means it's a support example. In this phase, the model uses the data to update its fast (plastic) weights to build an internal representation of the 5 classes.

In the Query Phase, the model sees $N \times Q$ unlabeled images from the same $N = 5$ classes of the form [embedding, 0, 1]. At every query step the Transformer outputs class scores (logits) and cross-entropy loss is computed using the true class of each query image. Validation accuracy is used as a metric.

They repeat this for a number of episodes, where new images are sampled. After an episode, the static weights are updated.

Eighty and 100 classification episodes per epoch are used for Omniglot and CIFAR-FS, respectively, for a total of 5 epochs. The number of inner updates per episode is $N \times K$ (support) + $N \times Q$ (queries).

The results are shown together as the copying experiment in table 2 and figure 2.

Table 1: Performance across tasks (mean ± std over 3 runs) from original paper

Rule	CIFAR-FS Acc.	Omniglot Acc.	Copy Recall
None	0.289 ± 0.028	0.192 ± 0.020	0.721 ± 0.025
Hebbian	$\mathbf{0.319 \pm 0.017}$	$\mathbf{0.237 \pm 0.028}$	0.727 ± 0.007
Gradient	0.274 ± 0.038	0.201 ± 0.012	$\mathbf{0.745 \pm 0.011}$

Table 2: Reproduced Performance across tasks (mean ± std over 3 runs).

Rule	CIFAR-FS Acc.	Omniglot Acc.	Copy Recall
None	$\mathbf{0.274 \pm 0.034}$	0.234 ± 0.055	0.774 ± 0.002
Hebbian	0.269 ± 0.028	$\mathbf{0.246 \pm 0.054}$	0.771 ± 0.005
Gradient	0.262 ± 0.034	0.236 ± 0.066	$\mathbf{0.774 \pm 0.004}$

3.1.3 Results

The results from the original paper are shown in Figure 2b and Table 1: The accuracy for the copying task is almost uniform across models, and it appears that the Hebbian Plasticity rule performs the best for image classification.

The reproduced accuracy as of Figure 2a and Table 2 exhibits the same behavior for the copying task. For image classification, Hebbian is still the best for the Omniglot Dataset. In our CIFAR-FS case, however, perhaps due to the inherent randomness of testing, the baseline Transformer without any plasticity rule seems to perform the best.

The neuromodulation and the plastic weight norm are of similar scale compared to the results of the original paper.

3.2 Extended Reproduction Experiments

From the original paper, we also extended their experiments in two ways.

1) We increased the training epochs from 2 to 5 in the copying task, and increased the training epochs for image classification from 5 to 20 and look at the results. The results are shown combined in Figure 3 and 3.

2) We tested the copying task for the three models with varying signal length, in order to see the memory capacity for different models while fixing the same model size and parameters. We train 5 epochs over each copying length and look at the models' final accuracy. The results are shown in Figure 4.

Here, We limit the model to be of 2 layers, with 2 transformer blocks and 4 attention heads, 128 hidden parameters, and 0.1 dropout probability. This is consistent with their original test model for their copying experiments.

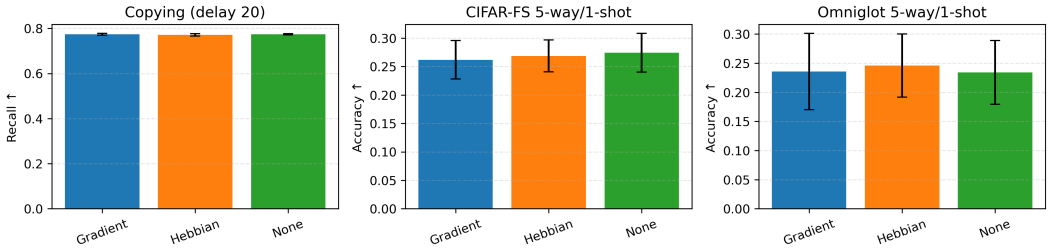
3.2.1 Increased Training Epochs

By simply increasing the number of epochs, we clearly see a discrepancy from the original paper's results. Here, the gradient model becomes the best one, with only slight win over in the copying task, and becomes significantly better than the other two in image classification.

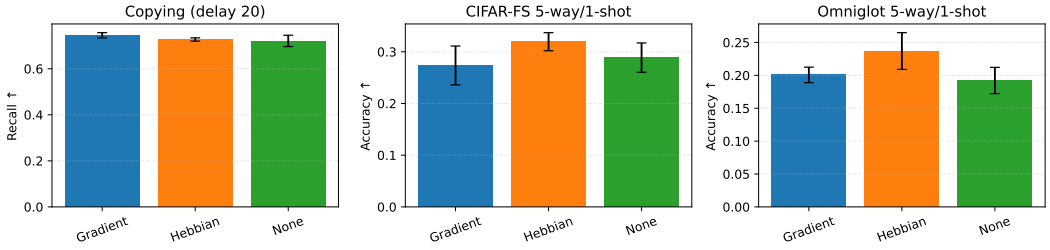
Neuromodulation and Plastic Weight Norms are on the same scale as shown before, indicating that the model is actively adapting to new information.

The original paper only ran 5 epochs using a very small model for the tasks, and especially for image classification, they stopped with a final classification accuracy only up to around 0.3. In real world scenarios, significantly larger models are used and being trained with significantly larger number of epochs.

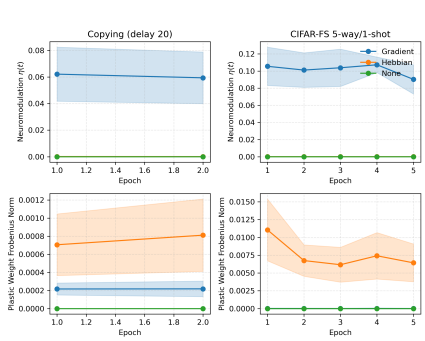
For consistency with the original paper's results, and due to limitation in time and resources, this extended performance test only increased the number of epochs. Even so, we see that the result looks



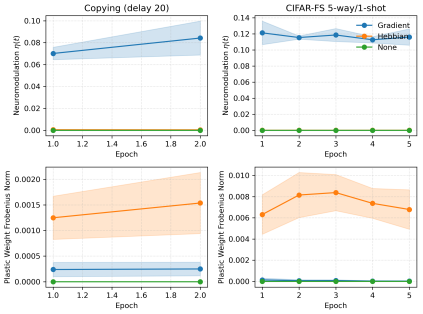
(a) Reproduced Final Accuracy



(b) Final Accuracy in original paper



(c) Reproduced $\eta(t)$ and plastic weight norm



(d) $\eta(t)$ and plastic weight norm in original paper

Figure 2: Results for Copy and Image Classification Experiment

Table 3: Extended Performance on image classification and copying tasks (mean $\pm$ std over 3 runs).

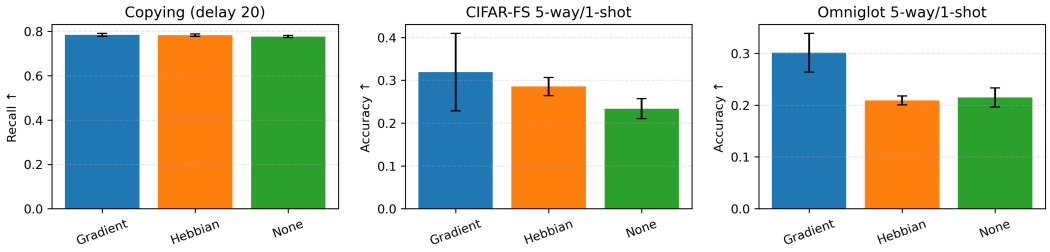
Rule	CIFAR-FS Acc.	Omniglot Acc.	Copy Recall
None	0.234 ± 0.024	0.215 ± 0.018	0.777 ± 0.005
Hebbian	0.286 ± 0.021	0.209 ± 0.009	0.783 ± 0.005
Gradient	$\mathbf{0.319 \pm 0.091}$	$\mathbf{0.301 \pm 0.038}$	$\mathbf{0.784 \pm 0.007}$

significantly different. It is safe to say that the results obtained from the original paper would need more testing to hold better significance in real applications.

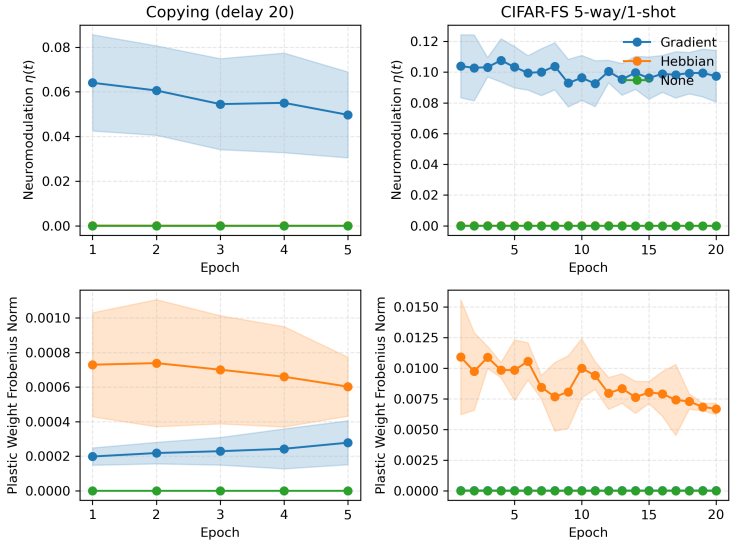
3.2.2 Copying Capacity testing

By testing the copying task on a wide range of Signal Lengths, we see that due to the limitation of parameter sizes, the three models show similar decrease trends.

However, we also see that the baseline Transformer has a significantly larger variance compared to the other two. By adding plasticity rules, in this scenario, the model seems to better adapt to randomness and exhibits stable behavior across different seeds.



(a) Final Accuracy after extended training



(b) $\eta(t)$ and plastic weight norm after extended training

Figure 3: Results for Copy and Image Classification Experiment after extended training

3.3 New Experiment: Three-Shot Text Category Classification

We created a text category classification task as another way to test rapid, in-sequence learning. Similar to the above design, we test a 3-way, 3-shot classification task, but this time with textual data. The model will be presented with words belonging to one of three categories: animals, furniture and clothing. Its goal is to predict which category a given word belongs to. Below are the ten words for each category:

- Animals: "dog", "cat", "fish", "mouse", "deer", "bear", "wolf", "lion", "tiger", "elephant"
- Furniture: "table", "chair", "carpet", "couch", "bed", "desk", "shelf", "cabinet", "wardrobe", "dresser"
- Clothing: "shirt", "skirt", "jacket", "dress", "pants", "shorts", "shoes", "hat", "coat", "gloves"

For training, we provide 3 support words per class and test on 2 query words for each of the 3 classes for each episode. For word embeddings, semantic embeddings are used to ensure that words in each class are similar in embedding space and differ significantly from embeddings in other classes. An embedding for a word was calculated as

category_base + word_offset(word), where *category_base* is the base embedding vector for that category and *word_offset(word)* adds word-specific variation to that category.

We train for 200 episodes and evaluate on 100 episodes for a total of 5 epochs. Our metrics are validation classification accuracy averaged across the 3 seeds (reported as mean $\pm$ standard deviation) and the plastic weight norm and $\eta(t)$ for the plastic model.

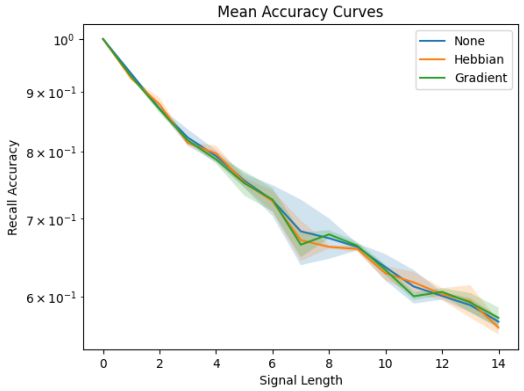
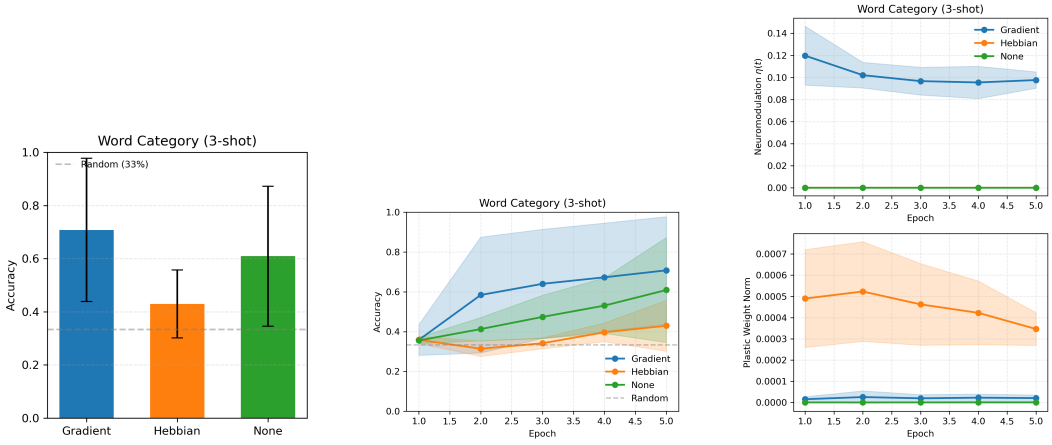


Figure 4: Recall Accuracy for copying tasks graphed respect to signal length. the filled area is the 95% Confidence Interval

3.3.1 Results

Table 4 displays the accuracy results and Figure 5 displays an accuracy bar plot, accuracy across epochs, plot of the neuromoduluation factor, and plot of the plastic weight norm for this task. The values and associated error bars for each plot are calculated using the mean and standard deviation of the three different seeds.



(a) Final Accuracy (b) Accuracy by epoch (c) $\eta(t)$ and plastic weight norm

Figure 5: Results for Text Category Classification Experiment

RULE	ACCURACY
None	0.609 ± 0.26
Hebbian	0.429 ± 0.13
Gradient	$\mathbf{0.708 \pm 0.27}$

Table 4: Three-Shot Text Category Classification Accuracy Results

The Gradient rule performed the best, followed by the standard rule and the Hebbian rule. The Gradient rule was even able to achieve 100% accuracy on one of the seeds. The wide error bars indicate that there was significant variance in accuracy across the seeds. Perhaps increased training metrics such as more epochs would reduce uncertainty in the accuracy. Additionally, the neuromodulation $\eta(t)$ was large for the Gradient rule, suggesting that this rule was able to classify the word categories well and update the fast weights quickly during the evaluation phase.

4 Comparison with Pre-trained Transformers

We also tested how the Plastic transformers performed compared to large standard pre-trained transformers. For the Copying task and and Three-Shot Text Category task, we used the BERT base uncased model trained on English Wikipedia and BookCorpus, where the weights were kept fixed. For The One-Shot Image Classification task, we used the CLIP (Contrastive Language-Image Pre-training) model from OpenAI trained on image-text pairs.

Table 5 displays the results for the experiments, where bold indicates that a pre-trained model performed better than the Plastic transformer for both learning rules.

Copying	Image Classification	Text Classification
0.0946	**0.7843/0.6475**	0.6917

Table 5: Test metrics for pre-trained transformers

The BERT model performed poorly on the Copying Task relative to the previous models because this model does not have any weight updates. Thus, it cannot learn in-context. This means that pre-trained transformers need even further training to learn this task.

The CLIP model performed much better than the Plastic Transformer for the Image Classification task. This is likely due to the fact that it takes significant computational power and training data to train an image classification model, which the Plastic Transformer in this project did not have.

The BERT model performed well on the Text Classification task, though slightly lower than the gradient plasticity model from previously. Below is a confusion matrix during the testing phase for the predicted classes, where the rows are the true category and the columns are the predicted category. The BERT model performed the highest on the animal category and the lowest on the furniture category.

	Animals	Clothing	Furniture
Animals	184	14	2
Clothing	42	153	5
Furniture	58	64	78

Table 6: Confusion Matrix for Experiment 3 (Three-Shot Text Category Classification task)

5 Analysis of Results

The performance tasks in the original paper only used a very small model and ran through very limited epochs. Moreover, the test results stopped at only around 0.3 accuracy for the One-Shot Image Classification task. Our hypothesis is that Hebbian Plasticity seems to dominate because the outer product weight update quickly adapted to the example using local information, while the Gradient Rule works more slowly towards the end goal.

If we increase the training epochs, or make the problem more complex e.g. The Three-Shot Text Category Classification task, we find that the Gradient Method becomes the dominating method.

Other than accuracy boosts, we conducted a capacity test on the Copying task by training the model on varying signal length. We find that in this scenario, the plasticity weights seem to stabilize behavior across random seeds.

By comparing to large Pre-Trained Transformers BERT and CLIP, trained on large amount of text and image data, respectively, we find that these Pre-Trained Transformer performed much better in text classification and image classification, respectively, while they did very poorly on the Copying Task. This is expected, since the Pre-trained Transformers do not have any weight updates during evaluation, they perform poorly on tasks they have not been trained on.

In the results we obtained, the Gradient Plasticity Method seems to perform the best of the three in general cases.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[2] Siddharth Chaudhary. Enabling robust in-context memory and rapid task adaptation in transformers with hebbian and gradient-based plasticity, 2025.

[3] Yu Duan, Zhongfan Jia, Qian Li, Yi Zhong, and Kaisheng Ma. Hebbian and gradient-based plasticity enables robust memory and rapid learning in rnns, 2023.