# Meta-Learning Deep Energy-based Models

*Group Members:*
Yash Gondhalekar (2019A7PS0481G)
Vedant Dhande (2018B5A40625G)

**Paper Summary**:
The paper focuses on auto-association memory models for restoring images given its partially distorted version. The authors note the limitations in attractor networks like the Hopfield networks which are deemed to be inflexible to store additional patterns and are impractical to use for real-world datasets. To solve this issue, the authors propose the Energy-Based Memory Model (EBMM) in which, apart from storing patterns such that they become attractors of the update dynamics of the network, it allows for faster writing of compressed patterns of input data into the network (write procedure) and faster retrieval (read procedure). In essence, the read and write procedures are meta learned that use very few examples during meta-training. To achieve this, arbitrary neural networks can be used to store patterns of input data which allows for leveraging optimization-based meta-learning.

**Experiments**:
We used Model-agnostic meta-learning (MAML) in our approach to training our model's parameters for fast adaptation. Since any choice of neural networks can be used as memory, optimization-based models were our first choice. We used `learn2learn` library for training and testing splits of the dataset and used its implementation of MAML.

The basic experimental procedure follows the paper: Write a fixed-size batch of images into a memory network, perform random distortion of the original images to form query images, then read the distorted image to retrieve the originally stored image. Due to the RAM limitations in Google Colab, we use 5 epochs, and T = 2 write and K = 3 read iterations in each epoch. The corruption is in the form of randomly distorting a $n \times n$ region of the original image with Gaussian noise with mean = 0 and standard deviation = 1. We used n=7 throughout our experiments.

We experimented with two types of memory: ResNet with Fully-connected memory (RFCM) and ResNet with Convolutional memory (RCM), as described in the paper. Both follow the architecture described in sections C.2 and C.3 respectively in the paper. We used the same EBMM decoder for both the above memory types for a fair comparison. It is a convolutional network and is a slightly modified version of the transpose of the encoder used in RFCM that increases the no. of channels followed by decreasing the channels to match the no. of channels in the original image.

*Note: Since we experiment with two different memory types, figures and tables denoting results explicitly state which memory has been used, fully connected or convolutional.*

For the writing procedure, since the paper states that using only the first two terms in the writing loss also helps for meta-training, we only use the first two terms to devise our writing loss.

$$\mathcal{W}(\mathbf{x}, \boldsymbol{\theta}) = E(\mathbf{x}; \boldsymbol{\theta}) + \alpha ||\nabla_x E(\mathbf{x}; \boldsymbol{\theta})||_2^2 + \beta ||\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}||_2^2.$$

Image Source: Bartunov et al., 2019

We set the initial value of α = 0.5 and decrease it by a factor of 0.95 after each writing iteration. *(This choice was arbitrary that seemed a safe choice. However, this could also be meta-learned)*.

The fast-adaptation learning rate was set to 0.5, and initialize the outer learning rate to 5e-5 as done in the paper. The authors of the original paper used a softplus activation to ensure learning rates never become negative. For our scheme, since we keep the learning rate constant, we did not need to explicitly ensure learning rate positiveness.

Although the read procedure described in the paper iteratively improves the query image towards the original stored image using the energy gradient of the partially reconstructed image at the kth iteration, we experimented with two approaches:

***(1) Using a separate convolutional EBMM decoder module as a learner using the MAML algorithm and optimizing it using reconstruction error.***

Unlike the original paper, we used a combination of the Structural Similarity Index (SSIM) and the Mean-squared error (MSE) as our loss function. Since both the terms are differentiable, it makes it suitable for backpropagation. For calculating the SSIM loss, we use the `piqa` library and normalize each image in the range [0, 1] across the batch. We then calculate the MSE and SSIM of the partially reconstructed image with the originally written image.

Since images are normalized and both SSIM and MSE are important, we simply assign an equal proportion to each term in the loss function, however, further experiments could be done to fine-tune it on some validation data or even meta-learn the proportions.

We observed that this significantly enhances the model's performance since the SSIM term forces the model to produce visually similar reconstructions.

Moreover, using the SSIM term allows getting rid of the random block distortions introduced in the query image whereas using only the MSE loss still shows some traces of the distortions.

Appendix 1 shows a comparison of reconstructed images with and without using the SSIM term. Table 1 shows the mean energy values of the images. Although the decoded images have lower energy than the query image, we believe further gradient updates could help reduce the energy even more.

| Original | Decoded | Query |
|---|---|---|
| -0.1949 | -0.0623 | -0.0421 |

*Table 1: Mean energy values of all images in a batch size of 10 using the Omniglot dataset. The energy of the original images is the lowest, as expected. The energy of the decoded images is lower than the query image denoting that the read procedure could perform updates on the query image to reach closer to the original image without explicitly using the energy gradient information. These results are using fully-connected memory.*

This approach is computationally heavier than option (2) since the decoder module needs to perform a backward pass and update its weight in order to optimize the reconstruction error.

Few examples (*Note: Since we use* `learn2learn` *for sampling batches and tasks, we have kept their internal transforms, such as downsampling all images to size 28×28, intact. All experiments for the Omniglot dataset use 5-way 1-shot learning*):

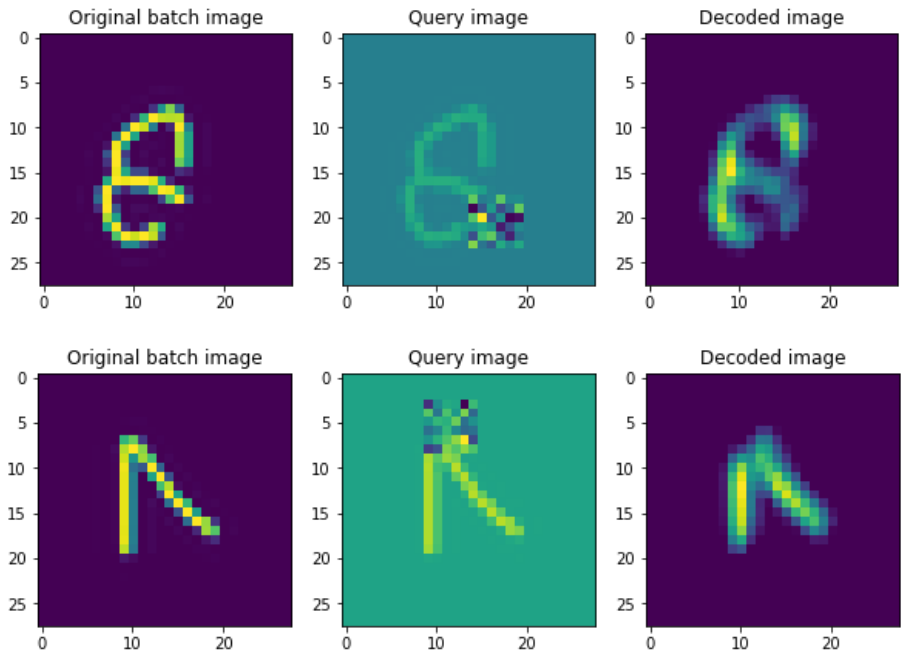


*Fig 1: The first column is the original (written) image, the second column is the query (distorted version) image, and the third column is the retrieved image. The model is successful in identifying and removing the distorted blocks in the query image and retrieving the original pattern. These results are using fully-connected memory.*

### *(2) Using gradient descent to update the query image using the energy gradient*

In this approach, the query image is updated using the equation:

$$\mathbf{read}(\tilde{\mathbf{x}};\boldsymbol{\theta}) = \mathbf{x}^{(K)}, \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \gamma^{(k)}\nabla_{\mathbf{x}}E(\mathbf{x}^{(k)}), \quad \mathbf{x}^{(0)} = \tilde{\mathbf{x}}.$$

Image Source: Bartunov et al., 2019

In this case, the writer itself is the learner that is optimized using the writing loss, as defined in equation 3 from the paper and which is made to efficiently compress patterns from the input image using only a few steps.

Then, using the energy gradients of the partially retrieved image, the read procedure tries to efficiently retrieve the original image.

In the paper, the initial value of $\gamma^{(1)}$ is meta-learned such that once a good initial $\gamma$ is learned, it can be used to devise a non-decreasing sequence of $\gamma^{(k)}$'s. In our case, we experimented with different $\gamma^{(1)}$ values and found $\gamma^{(1)} = 500$ to be a sufficiently good choice that allowed for a relatively smoother updation of $x^{(k)}$, and assign $\gamma^{(k)} = 0.5 \times \gamma^{(k-1)}$, for $k \geq 2$.

Since this approach updates the partially retrieved image using gradient descent directly, it is computationally less expensive than using an EBMM decoder. So we use 10 epochs, and use T = 5 write and K = 5 read iterations in each epoch.

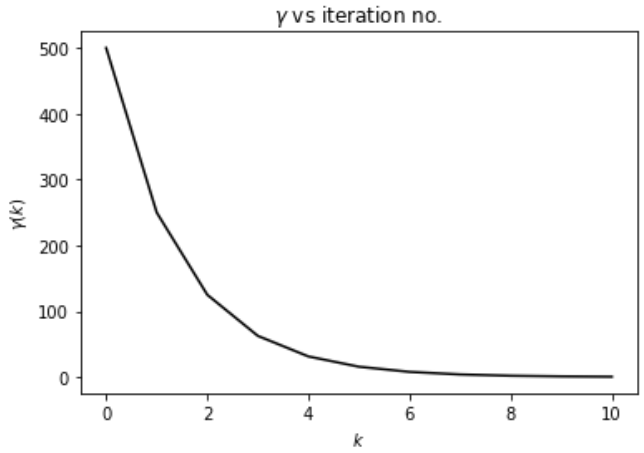


*Fig 2.: Update procedure for γ, the step size, used in gradient descent for updating the partially retrieved image.*

However, this approach did not yield good results as shown in Appendix 2. Refer to Appendix 2 for observations and comments on the results.

***(3) Using approach (1) but using convolutional memory.***

As noted from the above discussion, approach (1) gave us better results and hence we stick with it, as is also done in the paper. In this section, instead of using a fully-connected layer as memory, we leverage convolutional memory which was proved to be more efficient than the fully-connected memory.

Since convolutional layers help reduce the no. of parameters, we could train the model for 50 epochs, and T = 2 write and K = 5 read iterations in each epoch. During our experiments, we found that a fast adaptation learning rate of 0.05 seemed to work better than 0.5, as was done in approach (1). Apart from this, all hyperparameters were kept the same as in approach (1).

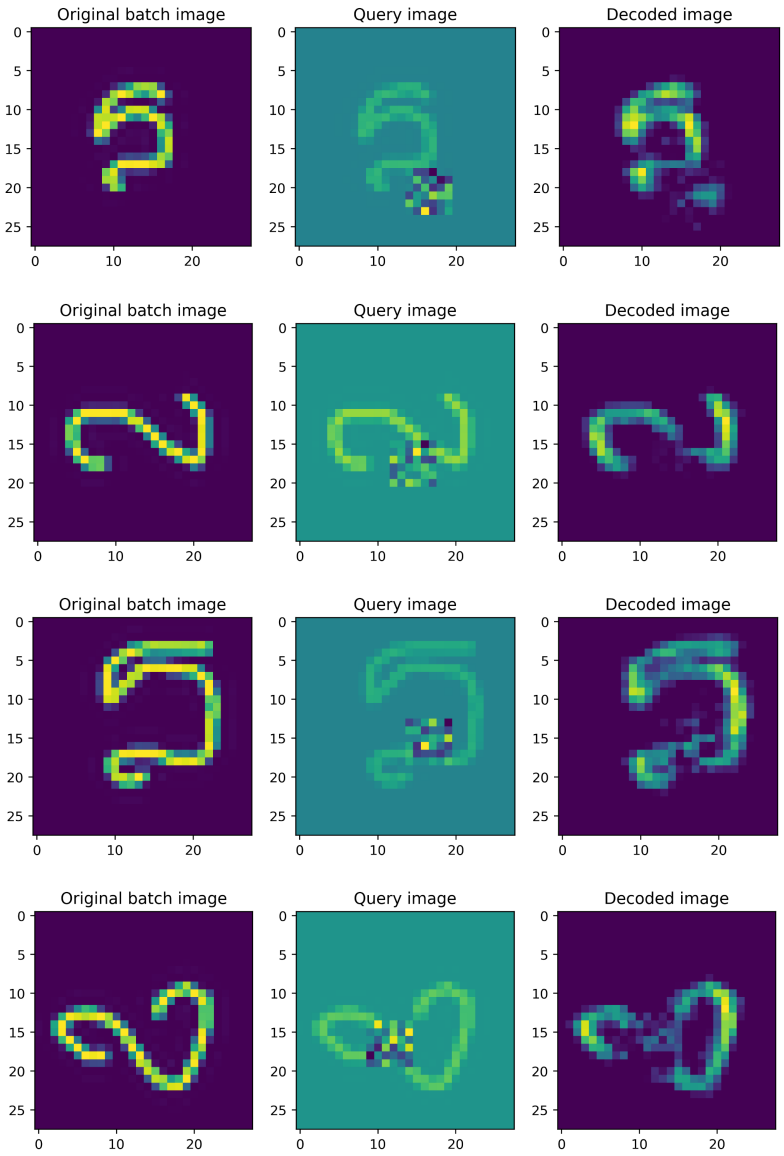


*Fig 3.: Results of the retrieved images using a convolutional memory.*

We observed that in the majority of the cases, convolutional memory outperforms fully-connected memory, which confirms the observation made in the original paper.

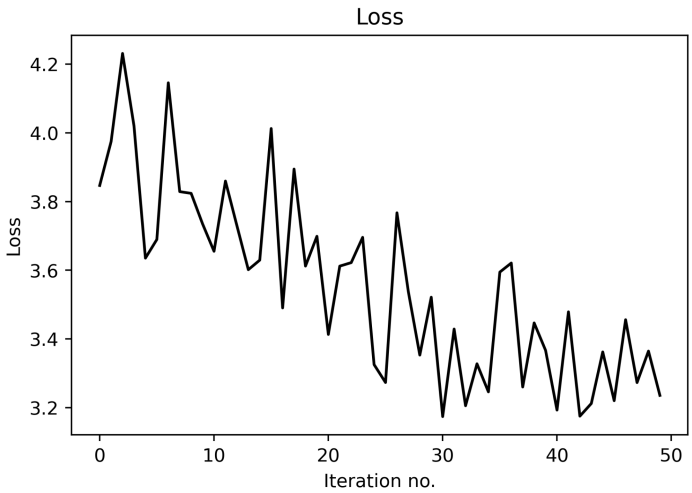


*Fig 4.: Reconstruction loss of the retrieved images as compared to the originally written images vs no. of epochs.*

Finally, we also experimented with our method on the FC100 dataset which has 3-channel images of size 32×32. Since the FC100 dataset contains thrice the number of channels and slightly larger images as compared to the Omniglot images, the training was computationally heavier. Also, to develop random distortion blocks on FC100 images, we selected the first channel and corrupted a random n×n region of that channel (we set n=7). This was done since otherwise each channel would get separately distorted yielding three distortion blocks in the query image. We note that this is a more difficult case than the Omniglot images in that the images of real scenes with great variety. We trained the model for 20 epochs with K = 3 read iterations and T = 2 write iterations in each epoch. Details of the experiments and initial results are mentioned in Appendix 3. As stated in Appendix 3, our model could not efficiently retrieve images as expected. However, we note the following: (1) The retrieved images suggest that the model could retrieve some structure, for example, the edges. (2) As seen below, the loss seems to decrease.

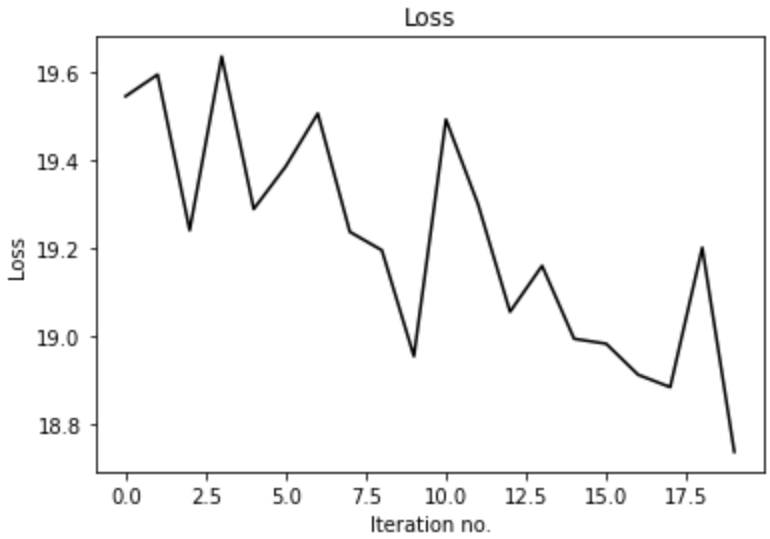


*Fig. 5: Reconstruction loss vs no. of epochs for the FC100 dataset.*

Since FC100 images are more complicated, we feel that the results could be improved by training for more number of epochs. For example, the original paper authors trained the models for ~1 week. Despite the fact that since the distortion models used are stochastic, it is difficult to differentiate through the expectation of reconstruction losses taken over a batch of images, we observed a decrease in the reconstruction loss as the epochs progressed, which was interesting.

## Conclusion

Based on the original paper's idea to use meta-learning in the context of deep energy-based models, we re-implemented the EBMM architecture using different memory types and carried out experiments to test the implementation on the Omniglot and the FC100 datasets. The loss function proposed here, consisting of SSIM and MSE terms, proved to be beneficial for the efficient retrieval of the images. We observed that convolutional memory slightly outperforms the fully-connected memory which matches the original paper's observation. We showed via experiments on the Omniglot dataset that good-quality retrieval is possible without explicitly using the gradient information but instead updating parameters of a separate decoder for fast and efficient associate retrieval. We also observe and verify that for more complicated images with high variation in the types of images, like in the FC100 dataset, it is beneficial for the model to train for more epochs in order for the decoder to learn an efficient way of retrieval.

## Appendix 1

This section shows the importance of incorporating the SSIM term in the loss function since it allows for improving the reconstructed images' quality. Below are the results on the Omniglot dataset.

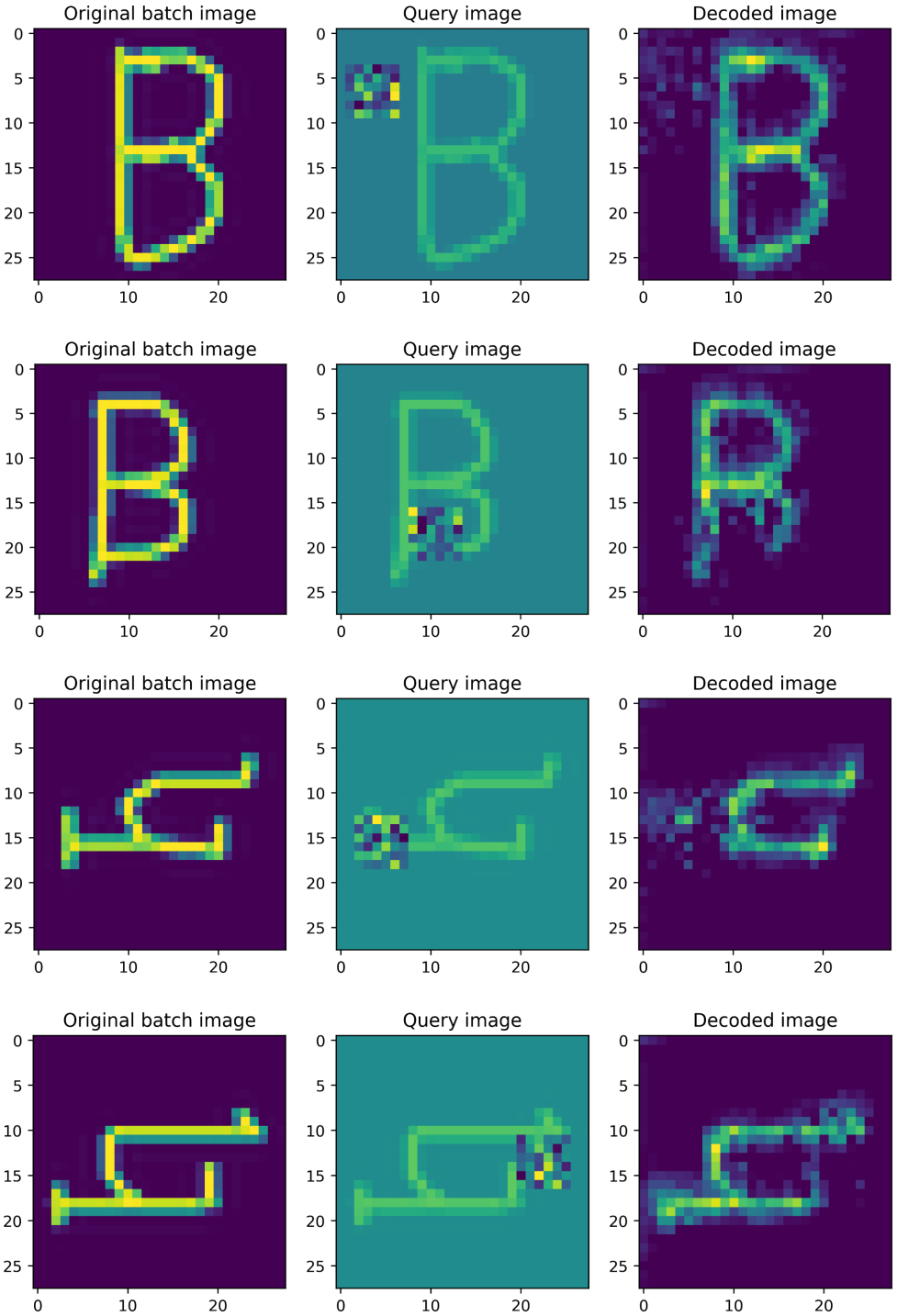


*Fig 1.: Original, query, and decoded images from the test set using only the MSE loss. These results are using fully-connected memory.*

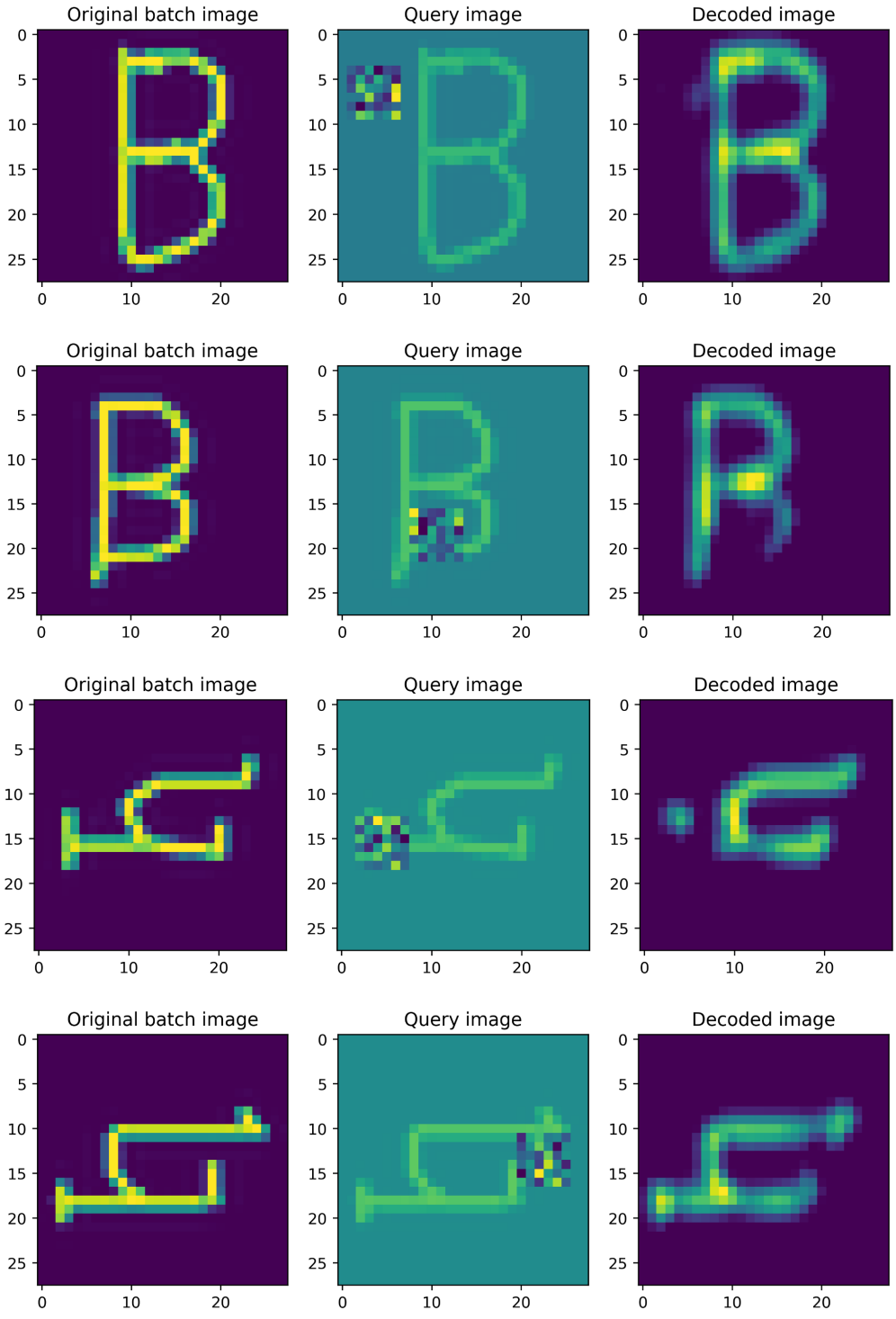


*Fig 2.: Original, query, and decoded images from the test set using SSIM + MSE loss. These results are using fully-connected memory.*

# Appendix 2

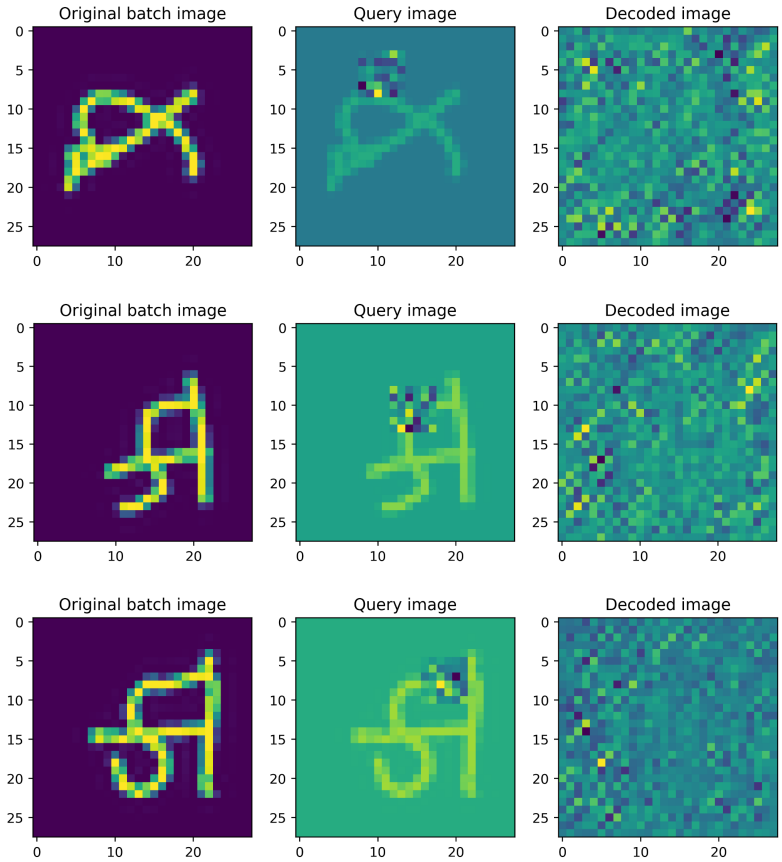


*Fig 3.: Retrieved images using approach 2 from the testing set during meta-training. These results are using fully-connected memory.*

We note the following reasons for this behavior: During meta-training, we observed that the image retrieval quality is significantly dependent on the choice of initial γ, $\gamma^{(1)}$. Due to this, there is no prior guarantee that $\gamma^{(1)} = 500$ would work well for any domain-shifted distribution batch of images. This was verified when we observed the retrieved images on the train set during meta-training where it could at least preserve the original structure of the image, whereas as seen in Fig. 3, the structure itself is lost. This indicates $\gamma^{(1)} = 500$ was not a good choice for the test set during meta-training but was a relatively good choice for the train set during meta-training. As done in the paper, meta-training $\gamma^{(1)}$ could help find good $\gamma^{(1)}$ for any given different batch distribution.

## Appendix 3

Few results on using the developed EBMM on the FC100 dataset.

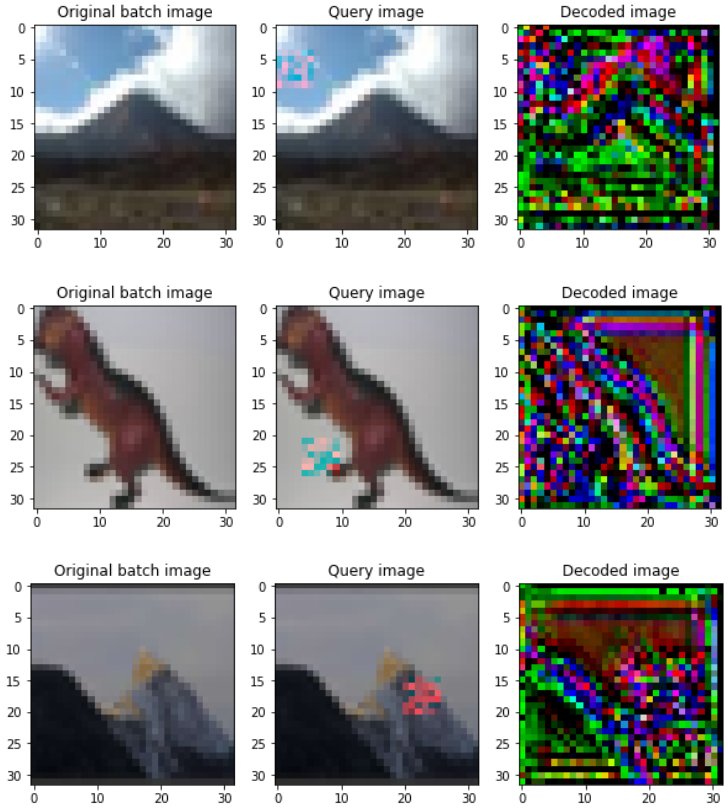


*Fig. 4: Results of retrieved images from the FC100 dataset. The retrieved images do not match with the originally written images, but as stated in the main text, further training of the model could improve the performance. In the retrieved images, the overall structure seems to be preserved, and the reason for that might be the SSIM term in our loss function.*

## Code Notebooks Description

(1) `EBMM_FC_memory_decoder_Omniglot.ipynb`
(2) `EBMM_FC_memory_GD_Omniglot.ipynb`
(3) `EBMM_CNN_memory_decoder_Omniglot.ipynb`
(4) `EBMM_CNN_memory_decoder_FC100.ipynb`

(1) uses fully-connected memory using an EBMM decoder, (2) uses fully-connected memory but updating the image via simple gradient updates, (3) uses convolutional memory and EBMM decoder for retrieval, and (4) uses convolutional memory and EBMM decoder for the FC100 dataset. All (1), (2), and (3) use the Omniglot dataset.

**Since we use standard datasets for all experiments as done in the paper, the required data is downloaded in the notebooks themselves.**

## Team Members and Contributions

(1) Yash Gondhalekar

- Idea of the loss function (SSIM + MSE) and writing code for it.
- Writing code for EBMM with fully-connected memory, convolutional memory, and model training.
- Hyperparameter tuning.
- Writing custom code to generate random distortions for creating query images.
- Analysis of results and comparison.
- Report writing.

(2) Vedant Dhande

- Help in report writing.
- Proofreading.

## References

(1) Bartunov, S., Rae, J.W., Osindero, S. and Lillicrap, T.P., 2019. Meta-learning deep energy-based memory models. *arXiv preprint arXiv:1910.02720*.