Modern Hopfield Networks on the CIFAR10 dataset

Haosheng Wang, Edrick Guerrero, Alfonso Gordon Cabello de los Cobos

Introduction

Memory involves the efficient storage and retrieval of information, and it comes in various forms—short-term, long-term, sensory, procedural, among others. Hopfield networks, also known as associative memories, are a class of recurrent neural networks (RNNs) designed to function as content-addressable memory systems. A defining characteristic of these networks is their ability to reconstruct entire patterns from partial or noisy inputs.

The original Hopfield network, introduced by John J. Hopfield in 1982, was based on binary feature representations and binary activation functions. Since then, significant advancements have been made. Modern Hopfield networks generalize the original model to continuous states, dramatically increasing storage capacity and stability. These developments, particularly in networks with continuous dynamics and large memory capacity, have been explored in a series of works since 2016.

In this project, we focus on replicating and analyzing the 2020 paper *"Hopfield Networks is All You Need"*, which presents a modern formulation of Hopfield networks that bridges them with attention mechanisms commonly used in deep learning nowadays.

Methodology

Our implementation exists on a Python Notebook, but can also be run on an external Python file. Given that the original implementation was made in PyTorch, ours is in TensorFlow. Moreover, we used a different dataset of images, specifically CIFAR10.

After a number of images to be analyzed by the model are chosen, these images are preprocessed by being passed through a noise filter (Gaussian) and a mask filter, creating two different and “corrupted” image sets (three total sets of images).

The core of our implementation exists in the hopfield_tf class. A main divergence we included from the original was a different method of comparing patterns in the images: calculating the distance between the pixels rather than the dot product.

To measure the difference between the output of the Hopfield Network and the original images, we implemented two different algorithms: MSE and Vector Similarity. MSE (Mean Standard Error), calculates the pixel-by-pixel difference of every image pair. Vector Similarity is calculated using OpenAI's CLiP model, which vectorizes images and calculates their similarity using the COSINE similarity function.

Results

Our results show a noticeable improvement in performance compared to the original implementation, most likely due to our change in pattern recognition, but also possibly due to change in dataset.

With noisy images, our network is able to retrieve all the images with relatively low MSE compared to the original dataset. Moreover, with 100 gray masked images, our implementation also has a lower MSE than the original and slightly better vector similarity.

Interestingly, with 100 RGB images and a scale of 10e6, our model achieves a near 0 MSE and a near 1 vector similarity, while the original implementation performs only adequately.

Challenges

The main challenge of our work was achieving the results we wanted. Though the premise of our project was to re-implement the original algorithm, we were left unsatisfied with the results upon finishing. The algorithm (both ours and the original are regarded as one since they were the same) didn't perform as well as we expected with our dataset. Upon investigating, we realized that the energy function being used was really volatile, which meant that feeding it a bright image such as a completely white one would have the model always return that.

As such, we set to find an energy function that fit better with out dataset. Moreover, we extended the implementation to also include RGB images.

Reflection

1. **How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?**

   Our project turned out to be better than we thought. We were able to complete the base and target goals, but we made a turn on the stretch by implementing the RGB processing and improving the accuracy of the network retrieval.

2. **Did your model work out the way you expected it to?**

   Our first implementation, which was a complete copy, showed the same outputs as the original implementation. However, these outputs were not satisfying, which was unexpected. Upon making our changes, we were able to achieve surprisingly good results.

3. **How did your approach change over time? What kind of pivots did you make, if any? Would you have done differently if you could do your project over again?**

   We would not make any changes as we are really happy with the results achieved. Initially we were only to use grayscale images but later on we decided to change the original energy function by a distance calculation, which allowed us to use RGB images that can be considered more “challenging” to process as they have more information.

4. **What do you think you can further improve on if you had more time?**

   We could try to add some sort of classification network, like a CNN, as we think this type of networks can make a good use of the HopField network. Moreover, there is also room to implement a custom algorithm that measures the similarity between two images to better measure accuracy.

5. **What are your biggest takeaways from this project/what did you learn?**

   There are a lot more things other than the hyperparameters that can affect the performance of the model, in our case the dataset. Moreover, we learned that such technology is more limited than we had originally thought. Though there is no doubt it’s being continually improved, there are also a lot of things that it isn’t capable of doing. When it comes to technology that is able to output images from noisy inputs, we realized why standalone HopField networks are not enough to get the job done. Instead, it’d be more wise to implement them into already existing models as a layer.