Improved Generalization for Image Classification Through Memorization*

Mid-term Report

Roger Waleffe and Jason Mohoney
{waleffe, mohoney2}

March 24, 2021

1 Original Project Proposal

We begin by summarizing the project proposal as previously described. Specifically, our work is motivated by the failure of current machine learning models to reliable generalize—especially over long-tailed distributions[1] or minor variations between train and test data [2, 4, 5]. In the former case, even state-of-the-art neural networks often exhibit highly variable performance across examples from rare subpopulations [5]. Thus, in order to improve the true generalization of modern machine learning algorithms, it is likely necessary to consider their behavior on both common instances and instances which appear only rarely in the training data.

Recent theoretical results suggest that memorization of outlier labels, and even whole training examples, is necessary for achieving close-to-optimal generalization error in the context of long-tailed distributions [1, 2]. Intuitively this makes sense (how else can a model generalize from one instance?), but this notion is in conflict with conventional machine learning beliefs, and state-of-the-art models are not designed to explicitly memorize (although they may have the capacity to do so).

In this project, our goal is to test the recent theoretical claims suggesting memorization improves generalization by explicitly encoding this hypothesis into an object recognition model. Concretely, we aim to build and empirically study a novel computer vision architecture which memorizes a representation for each training example and then uses this information to help classify future inputs. Our original proposal included an initial design towards this goal, taking inspiration from convolutional neural networks (to extract useful image representations), k-nearest neighbor (k-NN) models (to explicitly memorize representations and labels), and Transformer networks (to contextualize representations) (Figure 1). We plan to evaluate our models by studying the generalization error, worst-group accuracy under imbalanced classes (to model atypical subpopulations), and by comparing to existing object classification models and algorithms which naively memorize the training data.

2 Current Progress

In this section we describe current project progress and highlight early observations and results. Based on our original timeline, the goal was to finish the bulk of the required programmatic development by March 29. This included implementing a CNN architecture (ResNet [3]) and training procedure, nearest neighbor lookup over stored representations of training data, and a Transformer architecture to contextualize test embeddings based on their neighbors. We have successfully met these implementation goals. In a minor deviation from the original project proposal, we have decided to rely on L_2 distance rather than Locality Sensitive Hashing (LSH) to identify neighbors. This change does not affect the functionality of our architecture, but does slightly simplify our 'distance metric' between stored representations.

We have only just begun experiments, primarily to test that each part of the architecture works as expected, but have already noticed some interesting results. We report these 'sub-component' experiments in Table 1. Standard training of our ResNet-20 on the CIFAR-10 dataset results in 91.83% test accuracy. Basic memorization models (k-NN) achieve a slightly lower but comparable accuracy (91.00-91.73%). For these experiments, we memorize the class label and representation after the average pooling layer in ResNet for all training examples. We classify a new test example by computing its representation, and then taking the majority class vote over the nearest stored training embeddings. Note that how the ResNet is trained prior to memorizing the training data is likely to affect the accuracy of the k-NN models. As we have only run experiments to test our code's function-

*https://rogerwaleffe.github.io/cs766/

[1] In long-tailed distributions a few objects occur frequently while many objects occur infrequently.

ality, results in Table 1 use a ResNet pretrained for classification, although pretraining using clustering or self-supervised contrastive losses or end-to-end training of our entire architecture may be more beneficial for memorization.

We also tested our Transformer implementation which aims to extend the functionality of the basic k-NN model. Instead of a simple majority class vote over the nearest neighbors, the Transformer takes as input the test embedding plus the neighboring embeddings/labels and computes a contextualized test embedding which is free to incorporate information from the neighbors and/or their labels. As for the basic k-NN model, results in Table 1 are just preliminary test results and use fixed embeddings from our pretrained ResNet.

At the moment it seems as if the Transformer is not extracting any additional useful information over the basic k-NN memorization model, but we remark that there are a number of hyperparameters and training paradigms we have yet to experiment with. Of specific interest is the loss function used to train the Transformer. For the current test experiment, we just used a linear transformation to classification loss on the contextualized embedding (note that if the Transformer performs an identity transformation then this is equivalent to a linear transformation after the average pooling layer—simply the baseline ResNet), but we would really like a loss which forces the Transformer to utilize information from the neighborhood. End-to-end training of the entire architecture may also help, as the Transformer can influence the distribution of the memorized embeddings.

Finally, to gauge the potential benefit of incorporating memorization into standard object classification models, we also ran an experiment where we evaluated the accuracy of a sudo-model—one which optimally (based on the true class) picks whether to return the output of the 1-NN model or the ResNet. This model achieves 93% accuracy, a 14% relative improvement over the CNN alone with no additional training or parameters. While the optimal choice cannot be made in practice, this experiment points to the possibility that a learned gating function or combination of memorization with the initial ResNet may be beneficial.

Table 1: Initial experimental results.

Model	Accuracy
ResNet-20	91.83
1-NN	91.00
10-NN	91.55
20-NN	91.73
30-NN	91.65
10-NN Transformer	91.38
Optimal Gate: 1-NN/ResNet-20	93.00

3 Updated Project Goal

Our primary project goal remains the same: study recent claims suggesting memorization improves generalization by explicitly encoding this hypothesis into an object recognition model. However, we think the project proposal may have focused too much on a specific architecture (Figure 1). We plan to continue to study this design, but may also investigate other ways to incorporate memorization. For example, based on the final experiment above, a learned gate may be worth investigating. Additionally, we would like to consider ways other than nearest neighbor style approaches to memorize training examples and their labels.

We remark that the goal of our project is not to strictly produce a model with higher accuracy, but rather to evaluate whether memorizing training examples can improve performance. We may find that our attempts to include memorization do not succeed—an interesting result in its own right, as this may point to a contradiction with the recently proposed theoretical results.

4 Timeline and Remaining Work

To finish our project in the next month, the remaining work primarily consists of empirically evaluating different configurations and training paradigms. Of specific interest is to evaluate proposed memorization architectures on atypical subgroups where they are explicitly hypothesized to improve generalization (more so than they may improve the average accuracy).

Our project timeline remains the same as outlined in the original proposal:

- ~~March 1: Implement the base CNN architecture.~~
- ~~March 15: Implement the neighborhood lookup and Transformer architecture.~~
- March 29: Finish implementing the end-to-end training.
- April 12: Finish initial generalization experiments and comparisons to baselines.
- April 26: Study finer-grained accuracy (long-tail) and prepare presentation.
- May 5: Finish website.

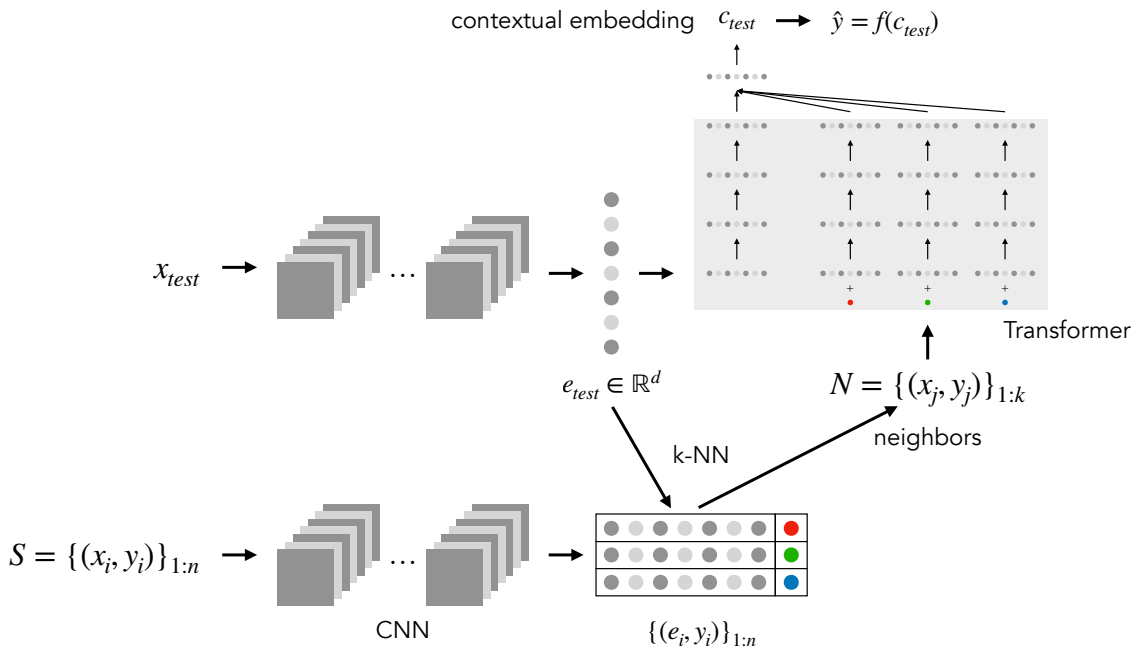

Figure 1: Graphical depiction of the originally proposed architecture to explicitly memorize training examples for image classification.

References

[1] G. Brown, M. Bun, V. Feldman, A. Smith, and K. Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? *arXiv preprint arXiv:2012.06421*, 2020.

[2] V. Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[5] N. S. Sohoni, J. A. Dunnmon, G. Angus, A. Gu, and C. Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *arXiv preprint arXiv:2011.12945*, 2020.