Content-based Image Retrieval using Capsule Networks

Davis, Bradley and **Gutierrez-Barragan, Felipe** and **Praveena, Pragathi** and **Zheng, Liu**
{badavis6, fgutierrez3, praveena, zliu577}@wisc.edu

Abstract

Content-based image retrieval (CBIR) uses visual information in images to identify images relevant to a query image. Feature extraction is an essential step for CBIR. Learning-based feature extraction methods are the current state of the art. Nonetheless, these methods are not robust to 3D transformations if these transformations are not extensively represented in the dataset. Capsule networks are a novel neural network architecture that has achieved state of the art results in image recognition tasks. One particular promising characteristic of capsule network models is that they are rotation, translation, and viewpoint equivariant which make them robust to 3D transformations without the need of data augmentation. In this project we study and implement two capsule network architectures and their respective routing algorithms and apply them to the CBIR task on the Google Landmark dataset.

Introduction

Content-based image retrieval (CBIR) uses visual information in images (rather than metadata like captions or geotags) to identify images relevant to a query image. Feature extraction, as seen in the pipeline for image retrieval in Figure 1, is an integral step for good performance. The algorithms used for feature extraction for image retrieval as outlined by a recent review paper (Zhou, Li, and Tian 2017) state that learning-based features generally outperform hand-crafted features.

Why not use CNN-based feature representations for CBIR? In recent years, convolutional neural networks (CNNs) have become the state-of-the-art in many computer vision tasks, in part due to their ability to learn good feature representations. Yet, CNNs are not robust to common transformations such as scaling and rotation. A simple solution to this problem is to perform data augmentation and include more samples with diverse transformations. This is a sub-optimal solution that not only avoids tackling this fundamental limitation of CNNs, but also compels the need for more computational resources to train the model with the larger augmented dataset.

In this project, we explore capsule networks for CBIR. Capsule network is a novel neural network architecture that attempts to solve the limitations in CNNs (Sabour, Frosst, and Hinton 2017; Hinton, Frosst, and Sabour 2018) in three ways. First, capsule networks do not use pooling layers. Despite their usefulness in practice, pooling layers allow the learned features to be positional invariant. However, this also means that the network forgets where the feature was and more importantly where it was with respect to other features. Capsule networks are described to be equivariant as opposed to invariant. Equivariance is the property where a transformation of input image results in an equivalent transformation of the feature representation. Secondly, capsule networks leverage the linearity in pose transformation in 3D space (translation, rotation, viewpoint, scale). Finally, capsule networks use a specialized feedforward algorithm (called routing) that attempts to direct activations from lower level features (neurons/capsules) only to the relevant higher level features.

The described properties of capsule networks in theory should lead to a feature representation that is robust to 3D transformations. In the remainder of this report we will define the CBIR task and provide more background on capsule networks. Then, we present the results for experiments performed on two standard datasets.

Background & Methods

CBIR Task

CBIR is a learning task that uses visual information in images to identify other images similar to the query image. In the past, CBIR has been done in supervised, unsupervised, and semi-supervised settings. The typical CBIR pipeline is outlined in Figure 1.

There are three technical aspects to the CBIR task:

1. **Image representation**: This refers to the feature extraction part of the pipeline. Images are represented as a set of visual features such that they are descriptive (identify similar images), discriminative (distinguish dissimilar images) and robust to image transformations. This is the aspect of the CBIR task that we are focusing on in this project.
2. **Image organization**: This refers to the efficient storage and indexing of the feature representations for each image.

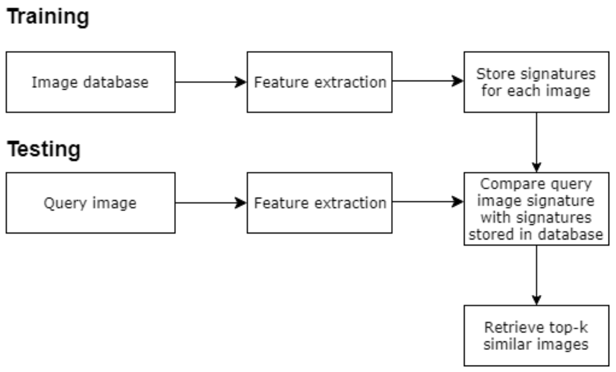
Training
Image database
Feature extraction
Store signatures for each image
Testing
Query image
Feature extraction
Compare query image signature with signatures stored in database
Retrieve top-k similar images

Figure 1: Pipeline for content-based image retrieval

3. **Image similarity measurement**: This refers to the comparison of a query image with the images existing in a database, that results in a score for each image. The scores are ranked and the top-k relevant images are returned to the user. The relevance score is often obtained by measuring distance between the image representations or aggregating number of local visual feature matches.

Capsule Networks

Capsule networks (Hinton, Frosst, and Sabour 2018; Sabour, Frosst, and Hinton 2017) differ from traditional neural networks in two ways. Firstly, the fundamental unit of the network is called a *capsule*. Secondly, signals between layers are propagated in a novel way, termed as *routing*, which results in similar capsule activations (predictions) being grouped together.

Capsules The fundamental unit of a capsule network is a *capsule*. A capsule represents the presence (or absence) and parameters of a multi-dimensional entity (e.g. object, feature, shape) of the type that the capsule detects. Similar to activation units in a neural network, capsules will output the detection probability for such entity. In addition to this probability, each capsule will also output a pose matrix associated with that feature. The pose matrix and activation probabilities of a capsule i are denoted as $\mathbf{M_i}$ and a_i. These parameters are computed during the forward pass through the network.

Connections Between Capsules A lower level capsule, i, is connected to a higher level capsule, j, via a weight matrix, $\mathbf{W_{ij}}$. These matrices are learned during backpropagation. During forward propagation the pose matrices from lower level capsules $\mathbf{M_i}$ are multiplied by $\mathbf{W_{ij}}$. If capsules i and j are related the resulting matrix is a prediction of what $\mathbf{M_j}$ should look like. The relation between capsules i and j is quantified by an assignment probability, r_{ij}, which is calculated during the routing-by-agreement step. In (Sabour, Frosst, and Hinton 2017) they refer to this assignment probability as the coupling coefficients between capsules.

Routing Procedures (Hinton, Frosst, and Sabour 2018; Sabour, Frosst, and Hinton 2017) proposed two different methods to perform routing between layers of capsules. Routing in capsule networks is an alternative to pooling in CNNs, which directs (groups) the outputs of lower level capsules that make similar predictions for the pose matrix of the higher level capsules. This means that the output pose matrix of a higher level capsule will be mainly based on the input lower level capsules that made similar predictions. The other lower level capsules that made different predictions will have very little impact on the prediction of the new pose matrix. Both papers introduce routing algorithms to calculate the next layer pose matrices, activation probabilities, and the assignment probabilities. In this project, we study the performance of *Dynamic Routing* (Sabour, Frosst, and Hinton 2017) and *Expectation-Maximization Routing* (Hinton, Frosst, and Sabour 2018). Hereon, we refer to these two algorithms as CapsNet-DR and CapsNet-EM.

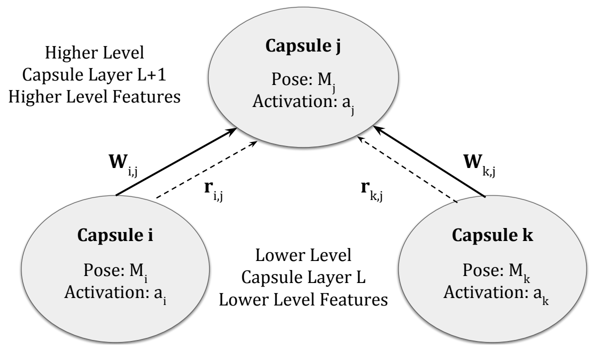

Figure 2: Capsule Network Diagram

CBIR with Capsule Networks

In this project we use a capsule network learned under the supervised setting for image recognition to perform content-based image retrieval for query images. In order to do this, we first train a capsule network for image recognition using supervised learning on the Google Landmark dataset. We then use the features encoded by the last layer of the network on the training data to compare with the features from a query image encoded by the same network. Finally, we use the L^2 norm as the distance metric to retrieve top-5 number of similar images (ideally of the same content).

Implementation Details

With our ultimate goal of exploring novel solutions for CBIR on a large dataset, we focus our efforts on the nascent capsule network architecture. To this end, we define 4 key stages of our project. The goal of the first 3 stages is to improve our understanding of capsule networks and perform image recognition in the supervised setting. The final stage attempts to implement and evaluate a capsule network based image retrieval pipeline.

1. MNIST for Implementation Validation: In this stage we validated the capsule network architectures and routing algorithms introduced in (Sabour, Frosst, and Hinton 2017; Hinton, Frosst, and Sabour 2018).

2. Capsule Network vs. CNN: In this stage we took the network learned in stage 1 and benchmarked it against a state-of-the-art CNN (Convolutional-Neural-Network). In particular, we tested the claim that capsule networks need less data than CNNs. We created a learning curve of the performance of these two models on the MNIST dataset.
3. Recognition on a Complex Dataset: In this stage we evaluated the capsule network models on the labeled Google Landmark dataset. We chose the more promising model for the image retrieval task.
4. Supervised Image Retrieval: In this stage we extended the learned network from stage 3 to build an image retrieval pipeline that takes a query image and returns the top-5 most similar images as determined by the learned features of our network.

Datasets

We trained three different neural network models on the MNIST dataset and a post-processed Google Landmark dataset (Kaggle-Landmark-Recognition 2018; Kaggle-Landmark-Retrieval 2018; LeCun et al. 1998).

MNIST dataset MNIST is a popular dataset for computer vision based machine learning research due to its small size and ease of use. It consists of 60,000 training and 10,000 test images of size 28x28 pixels. The images are normalized black and white digits drawn from the same distribution. We use MNIST for stages 1 (to validate the capsule network architecture on a simple dataset) and 2 (to compare capsule network with the state-of-the-art CNN).

Google Landmark Recognition and Retrieval dataset Google recently released the *Google Landmark* dataset and two Kaggle challenges for landmark recognition and landmark retrieval (Google-Landmark 2018; Kaggle-Landmark-Retrieval 2018; Kaggle-Landmark-Recognition 2018). The dataset contains images from more than 30,000 landmarks (i.e. it has around 30,000 classes). The full dataset contains more than 2 million images. There are a few particular characteristics of this dataset. Firstly, popular landmarks such as the Rialto bridge in Venice (Figure 9, first row) will have many more images than less popular ones. In fact, through some initial analysis we find that 50-100 classes compose around 30 percent of the full dataset. Secondly, as opposed to other datasets that try to recognize object categories such as lamps and chairs, landmarks will have very little intra-class variations. Most of the differences will come from different viewpoints, illumination changes, occlusions, weather, and camera. Finally, in many of these images the particular landmark might not be the primary focus of the photo. For instance, the main character in the image might be a person or group of people and the landmark will be in the background (see images u, v, w, y in Figure 9). Overall, the landmark dataset is one of the largest datasets available to date making it a good candidate to evaluate capsule networks on a larger scale problem.

Landmark Dataset Post-processing: Due to time constraints and limited compute and storage resources available, we work with 50 classes from the landmark dataset. Furthermore, due to the imbalance in the number of samples available for each class we specifically choose the 50 classes with the most images available. This guaranteed that we did not encounter a situation where a class only had a single image. The final dataset with 50 classes had a total of around 300,000 images. Additionally, the images in the dataset all had different scales so we performed the appropriate amount of down-sampling for the different neural networks we implemented. We use 90% of the images for training and 10% for testing and perform retrieval on the test set.

Software & Hardware

We configured Tensorflow using Python 3 on multiple machines with NVIDIA GPUs (1070Ti, 1080Ti, Titan XP). The code and implementation documentation for all models and data postprocessing can be found in this repository (760-Project-Repository 2018). The code was modified from available implementations of capsule networks (CapsNet-DR-Tensorflow 2018; CapsNet-EM-Tensorflow 2018).

Architectures

We briefly describe the implementations used for the three models used in our experiments: Convolutional neural network (CNN), capsule network with dynamic routing (CapsNet-DR) and capsule network with routing by Expectation-Maximization (CapsNet-EM).

CNN Figure 3 shows the architecture for the baseline model for our experiments with the MNIST dataset.

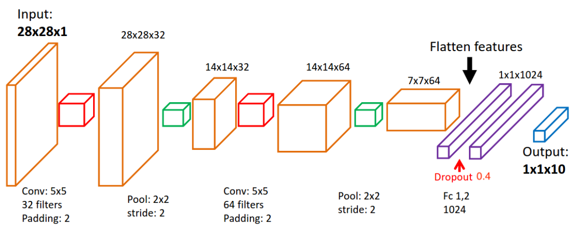

Figure 3: CNN: Convolutional neural network.

CapsNet-DR As shown in Figure 4, the first layer is a 9x9 convolutional layer with 256 channels and a stride of 1 with a ReLU non-linearity. This is followed by a convolutional capsule layer with 32 capsules each with 8 convolutional units with a 9x9 kernel and a stride of 2. The final capsule layer has one capsule per output class, 10 in the case of MNIST dataset and 50 in the case of Landmark dataset, that is fully connected to the layer below.

CapsNet-EM As shown in Figure 5, the first layer is a 5x5 convolutional layer with 32 channels and a stride of 2 with a ReLU non-linearity. This is followed by three capsule layers, the primary capsule layer with B=8 capsule types and the

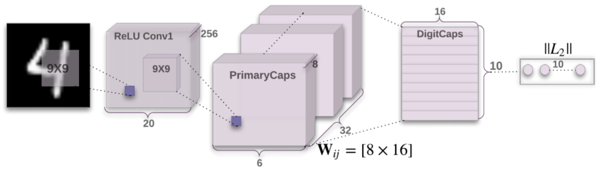

Figure 4: CapsNet-DR: Capsule network with dynamic routing. Diagram obtained from (Sabour, Frosst, and Hinton 2017)

two 3x3 convolutional capsule layers (K=3) with C=D=16 capsule types and strides of 1. The final capsule layer has one capsule per output class, E=10 in the case of MNIST dataset and E=50 in the case of Landmark dataset.

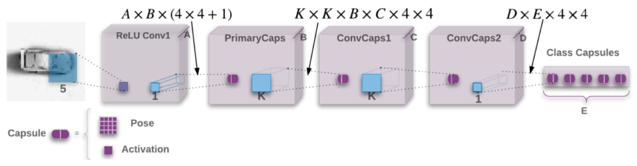
$A \times B \times (4 \times 4 + 1)$
$K \times K \times B \times C \times 4 \times 4$
$D \times E \times 4 \times 4$
ReLU Conv1
PrimaryCaps
ConvCaps1
ConvCaps2
Class Capsules
Capsule
Pose
Activation
E

Figure 5: CapsNet-EM: Capsule network with routing by Expectation-Maximization. Diagram obtained from (Hinton, Frosst, and Sabour 2018)

Experiments and Results

In this section we present the different results obtained with all three models we implemented.

MNIST Validation

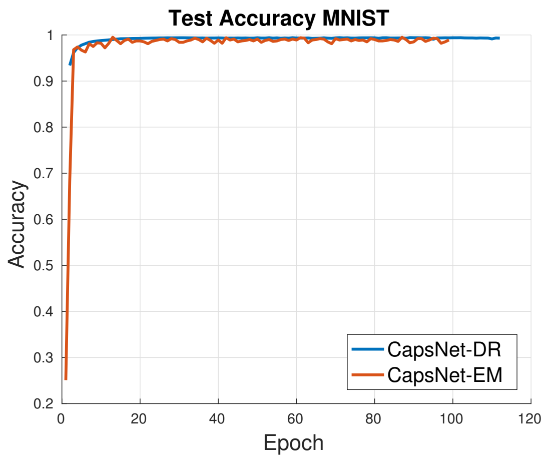

Figure 6: Epoch vs test accuracy results for MNIST dataset.

In order to validate both the capsule network implementations, we trained them on the MNIST dataset. Figure 6 shows the test accuracy the capsule network models would achieve after each training epoch. After roughly 10 epochs, CapsNet-DR achieves 99.42% accuracy and CapsNet-EM achieves 99.5% accuracy. This is comparable, but falls slightly short of the performance reported in the paper.

Do capsule networks need less training data?

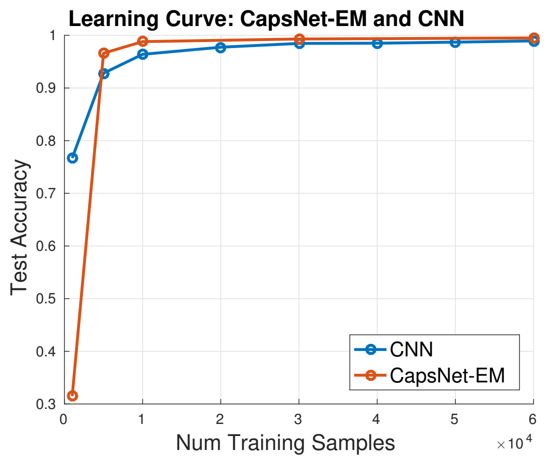

Figure 7: Learning curve of a CNN and a CapsNet with EM routing. Both models are trained on the MNIST dataset using various training set sizes.

Figure 7 shows the learning curve (training set size vs. test accuracy) for a regular CNN and the CapsNet-EM models. The CapsNet-EM model needs less data than the CNN model to achieve comparable test accuracy.

Landmark recognition with capsule network

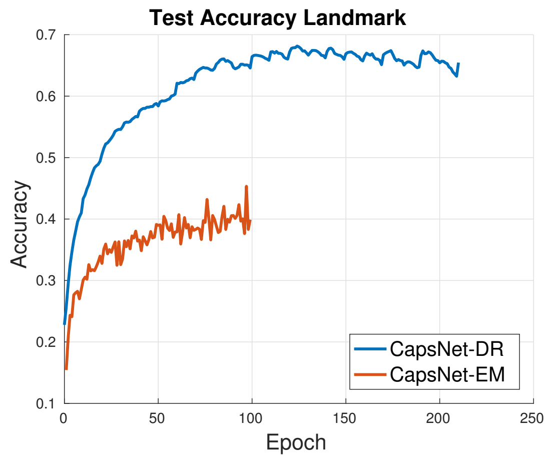

Figure 8: Epoch vs test accuracy results for the Google Landmark dataset.

Figure 8 shows the achieved test accuracy by each of the capsule network models that were implemented. We used the landmark dataset with only 50 classes. Both models significantly outperform random guessing (accuracy of 2% for 50 classes). Due to time constraints (because of the time-consuming iterative nature of EM), it was only possible to train CapsNet-EM with 44% of the training data and for

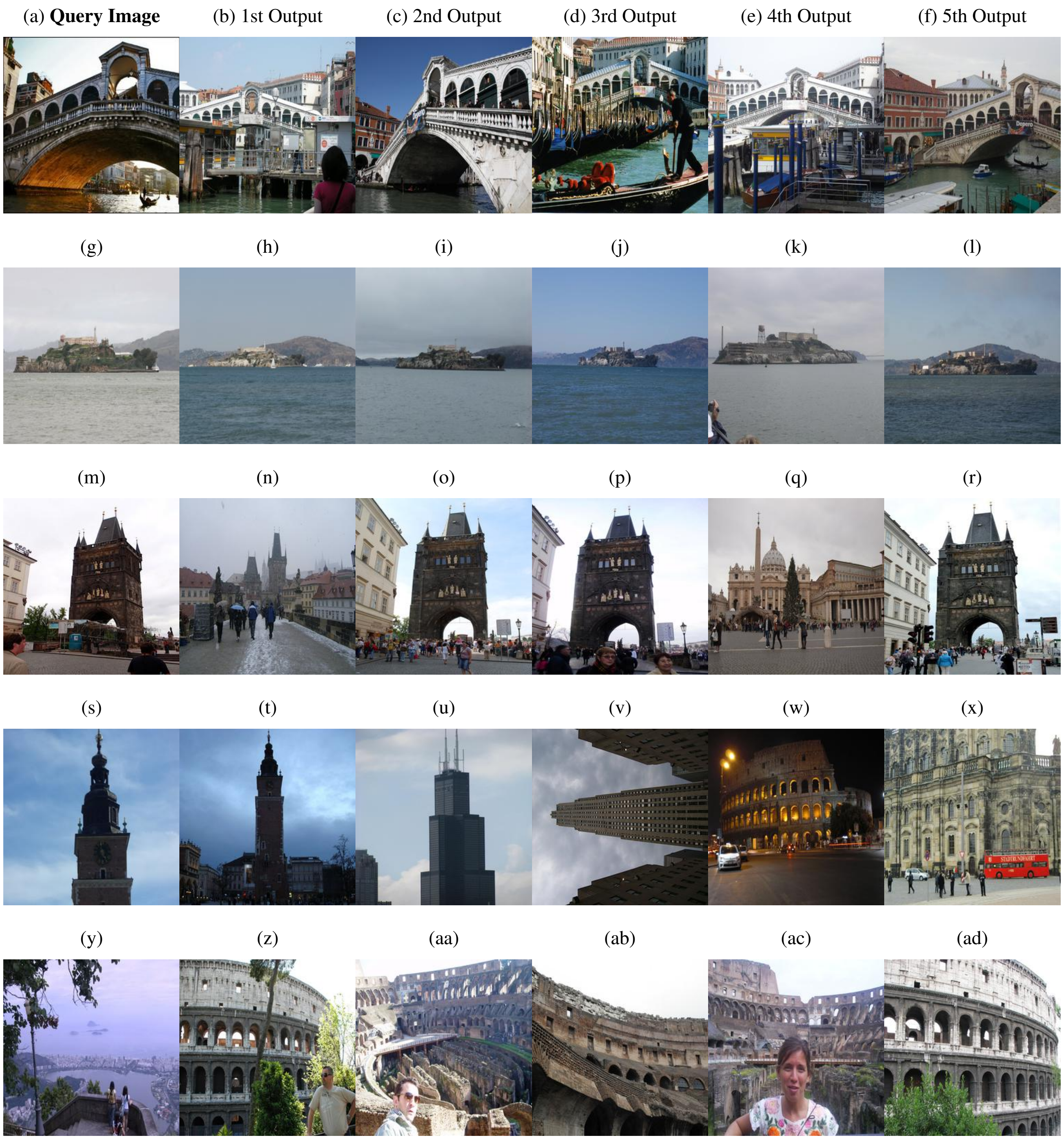

Figure 9: Resulting image retrieval using the CapsNet-DR. The left most image is the query image. The subsequent columns contain the resulting output images from the best to the worst match.

less than half the epochs. This performs better than random guessing but worse than the CapsNet-DR, which is trained on the entire training dataset. Out best performance for landmark recognition was 67.8% accuracy.

CBIR with capsule networks

We use **Mean Average Precision** (MAP) as a metric to measure the performance of the retrieval task. This metric is the mean of average precision across all classes. In a retrieval task, if the top-k results are returned to a user, all k should ideally be relevant. However, if only some are relevant, then it is better if the relevant results are shown first. The MAP score reflects this. For one query from each of the 50 classes, we calculate the average precision and then find the mean across all classes. We obtain a **MAP = 0.683** for the landmark dataset.

Figure 9 shows the image retrieval results using the CapsNet-DR. We only evaluate this model on the retrieval task for two reasons. As seen in Figure 8 it outperforms the CapsNet-EM model. More importantly, the CapsNet-DR model was trained on the full 50-class landmark dataset. The image in the leftmost column is the query image. The second through the fifth column are the images in the test set whose output feature vectors were the most similar in terms of the Frobenius norm. The columns are ranked in decreasing order of similarity to the query image. Additional retrieval results can be found in the project repository: https://github.com/bad884/760-project/tree/master/landmark_capsnetDR/retrieval (760-Project-Repository 2018).

Discussion

The MNIST results shown in Figures 6 and 7 give us a degree of confidence that the implemented models are correct. In particular, they also demonstrate that both CapsNet-EM and CapsNet-DR are already at the accuracy saturation point (similar to CNNs). This means that in order to obtain better metrics of the performance of capsule networks, these models should be benchmarked on larger and more complicated datasets such as CIFAR100 and ImageNet.

One particular disadvantage of current capsule network models, CapsNet-EM in particular, is the speed of training and the amount of computational resources (memory and GPU) required for training. Nonetheless, further research may show that these disadvantages are offset by advantages such as requiring less training data (Figure 7), faster convergence, and fewer model parameters. Exploring these tradeoffs between current state of the art CNNs and capsule networks might be a promising research direction.

We use the Google landmark dataset to test the performance of capsule networks in a more challenging setting. In Figure 8, we see that CapsNet-DR performs better than CapsNet-EM, contrary to what the authors show (Hinton, Frosst, and Sabour 2018). We posit that this is due to training the CapsNet-EM model with lesser data and for fewer epochs than the CapsNet-DR model. More importantly, the EM routing procedure is an iterative process and we use fewer number of iterations than that reported in the paper. We believe that engineering the CapsNet-EM model to be more efficient in terms of time and memory usage along with better computational resources can improve the results presented in Figure 8.

State-of-art content-based image retrieval for landmarks achieve an MAP > 0.9 on smaller datasets with fewer classes. In Figure 9, for query images (a) and (g), all teh retrieval results belong to the same class. The retrieved images vary in terms of viewpoints, illumination changes, occlusions, weather and camera. The query image (m) has one wrong retrieval in (q). The performance for query images (s) and (y) is poor, possibly because only a small portion of the landmark is seen in the query image. Even though our MAP of 0.683 does not compare with state-of-art deep retrieval models, the retrieval of (n) and (ab) for their respective query images shows promise of equivariant feature representations.

Conclusion

In conclusion, capsule networks are a promising network architecture for content-based image retrieval. In order to establish the robustness of capsule networks to 3D transformations for image retrieval tasks, future work should benchmark against state-of-art CNNs with and without data augmentation. Additionally, we evaluate the performance of capsule networks in a 50-class recognition and retrieval task and achieve encouraging results. Previous work have evaluated capsule networks only for datasets with less than 10 classes and with images obtained in controlled settings.

References

[760-Project-Repository 2018] 760-Project-Repository. 2018. 760 project repository. https://github.com/bad884/760-project. Accessed: 2018.

[CapsNet-DR-Tensorflow 2018] CapsNet-DR-Tensorflow. 2018. Capsnet-tensorflow. https://github.com/naturomics/CapsNet-Tensorflow. Accessed: 2018.

[CapsNet-EM-Tensorflow 2018] CapsNet-EM-Tensorflow. 2018. Matrix-capsules-em-tensorflow. https://github.com/www0wwwjs1/Matrix-Capsules-EM-Tensorflow. Accessed: 2018.

[Convolutional-Neural-Network] Convolutional-Neural-Network. A guide to tf layers: Building a convolutional neural network. https://www.tensorflow.org/tutorials/layers#building_the_cnn_mnist_classifier. Accessed: 2018.

[Google-Landmark 2018] Google-Landmark. 2018. Google landmark dataset. https://research.googleblog.com/2018/03/google-landmarks-new-dataset-and.html. Accessed: 2018.

[Hinton, Frosst, and Sabour 2018] Hinton, G.; Frosst, N.; and Sabour, S. 2018. Matrix capsules with em routing.

[Kaggle-Landmark-Recognition 2018] Kaggle-Landmark-Recognition. 2018. Google landmark recognition challenge. https://www.kaggle.com/c/landmark-recognition-challenge/data. Accessed: 2018.

[Kaggle-Landmark-Retrieval 2018] Kaggle-Landmark-Retrieval. 2018. Google landmark retrieval challenge. https:

//www.kaggle.com/c/landmark-retrieval-challenge/data. Accessed: 2018.

[LeCun et al. 1998] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

[Sabour, Frosst, and Hinton 2017] Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, 3859–3869.

[Zhou, Li, and Tian 2017] Zhou, W.; Li, H.; and Tian, Q. 2017. Recent advance in content-based image retrieval: A literature survey. *arXiv preprint arXiv:1706.06064*.