# 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 (rather than metadata like captions or geotags) to identify images relevant to a query image.

## 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 task, in part due to their power in learning good feature representations. Nonetheless, 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 because not only they are not tackling one fundamental limitation of CNNs, but also more data means more computational resources to train such a model.

In this project, we explore Capsule Networks (CapsNets) for CBIR. CapsNets are 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, CapsNets do not use pooling layers. Despite their usefulness in practice, pooling layers allow positional invariance to the learnt features. However, this also means that the network forgets where the feature was, more importantly where it was with respect to other features. Capsule nets 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, CapsNets leverage the linearity in pose transformation in 3D space (translation, rotation, viewpoint, scale). Finally, capsule nets specialized feedforward algorithm (dynamic routing) tries to direct activations from lower level features (neurons/capsules) to be directed only to the higher level features that are have strong relationships.

The described properties of CapsNets in theory should lead to a feature representation in the network that is robust to 3D transformations. In the remainder of this report we will define the CBIR task and give more background on capsule networks. We then present the experiments and results we obtained from

## Background & Methods

### CBIR Task

Content-based Image Retrieval is an unsupervised learning task that uses visual information in images to identify other images similar to the query image. The typical CBIR pipeline is outlined in Figure 1.

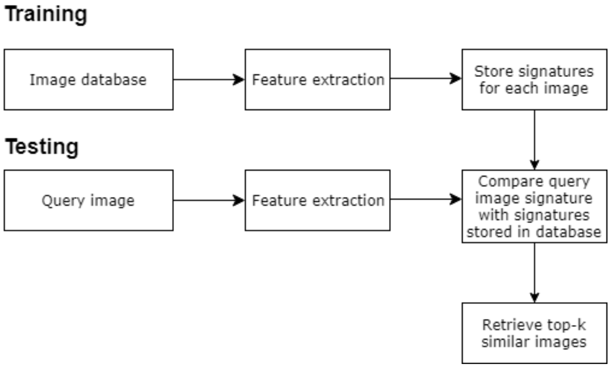

Training
Image database
Feature extraction
Store signatures for each image
Testing
Query image
Feature extraction
Compare query image signature with signatures stored in database
Retrieve top-k similar images

Figure 1: Pipeline for content-based image retrieval

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.
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 are a new type of neural network architecture recently introduced (Hinton, Frosst, and Sabour 2018; Sabour, Frosst, and Hinton 2017). Capsule Networks have two fundamental differences with traditional neural networks. The first difference is the fundamental unit that composes the network called a *capsule*. The second, is a novel way to propagate signals between layers they call *dynamic routing* which group similar capsule activations (predictions) 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 to 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** The connections between a lower level capsule $i$ and a higher level capsule $j$ will have a weight matrix, $\mathbf{W_{ij}}$, associated to them. The matrices are learned in the 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 Procedure** (Hinton, Frosst, and Sabour 2018; Sabour, Frosst, and Hinton 2017) proposed two different methods to perform routing between layers of capsules. Routing in CapsNets is an alternative to pooling in CNNs, which directs (groups) the output of lower level capsules that make similar predictions for the pose matrix of 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 studied the performance of these two models.

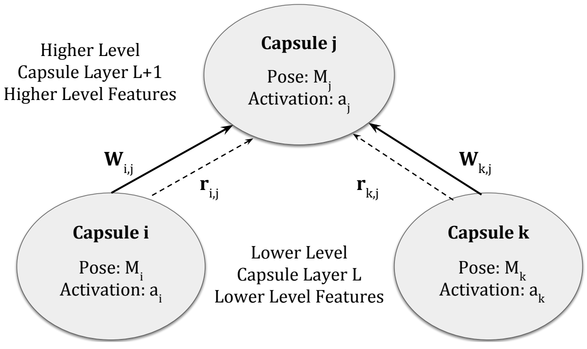


Figure 2: Capsule Network Diagram

## CBIR with Capsule Networks

In this project we plan to use a CapsNet learned under the supervised setting for image recognition to perform content-based image retrieval for query images. In order to do this, we will first train a CapsNet for image recognition using supervised learning on the 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 some top-k number of similar images (ideally of the same content).

## Implementation Details and Experiments

With our ultimate goal of exploring novel solutions for CBIR on a large dataset, we decided to focus our efforts on the nascent CapsNet architecture to improve upon the state-of-the-art. To this end, we defined 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 implemented the capsule network architectures and routing algorithms introduced in (Sabour, Frosst, and Hinton 2017; Hinton, Frosst, and Sabour 2018) in order to validate and test our code.

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 (cnn ). 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 continued to build upon the capsule network implementation. In particular, we explored architectures with a larger number of capsules and layers and evaluated them on the labeled Google Landmark Recognition Dataset. We were able to compare our model to others on the Kaggle leaderboard.

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 4 most similar images as determined by the learned features of our network.

## Datasets

We trained the different neural network models we implemented on the MNIST dataset and a post-processed Google Landmark Recognition/Retrieval datasets (lan a; b; mni ).

**MNIST Dataset** The 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 (28x28 binary pixels) of size normalized black and white digits drawn from the same distribution. We use MNIST for steps 1 (to validate the CapsNet architecture on a simple dataset) and 2 (to compare CapsNets with the state-of-the-art CNN).

**Google Landmark Recognition and Retrieval Dataset** Google recently release the *Google Landmarks* dataset and two Kaggle challenges for landmark recognition and landmark retrieval (lan c; b; a). 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. There are a couple particular characteristics of this dataset. The first one is that popular landmarks such as The Colosseum in Rome (third row Figure 8) or the Rialto bridge in Venice (first row Figure 8) 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. Furthermore, 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. 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 decided to only work with 50 classes from the landmark dataset. Furthermore, due to the imbalance in the number of samples available for each class we decided to specifically choose the 50 classes with the most images available. Choosing the classes with the most images guaranteed that we did not run into a situation where some of the classes only had a single image for a given class. We found that this situation is quite common on this dataset. The final 50 class dataset 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 on multiple machines with NVIDIA GPUs. The code and implementation documentation for all models and data postprocessing can be found in this repository (cod ).

## Architectures

We briefly describes 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**:

**CapsNet-DR**: **CapsNet-EM**: As shown in Figure 4, 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 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, which is E=10 in the case of MNIST dataset and E=50 in the case of Landmark dataset.

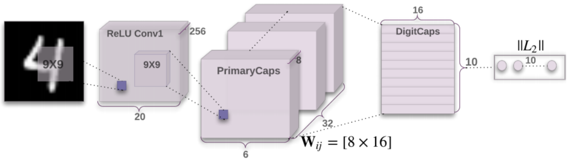


Figure 3: CapsNet-EM: Capsule network with dynamic routing

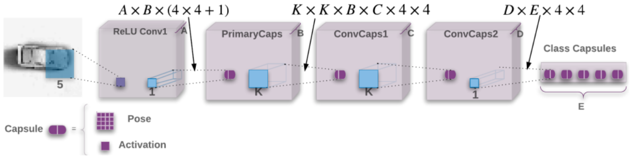

$A \times B \times (4 \times 4 + 1)$
$K \times K \times B \times C \times 4 \times 4$
$D \times E \times 4 \times 4$
5
ReLU Conv1
A
1
PrimaryCaps
B
K
ConvCaps1
C
K
ConvCaps2
D
1
Class Capsules
E
Capsule
=
Pose
Activation

Figure 4: CapsNet-EM: Capsule network with routing by Expectation-Maximization

## Experiments

## Results

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

## MNIST Validation

In order to validate both CapsNet implementations, we trained them on the MNIST dataset. Figure 5 shows comparable results to what was achieved

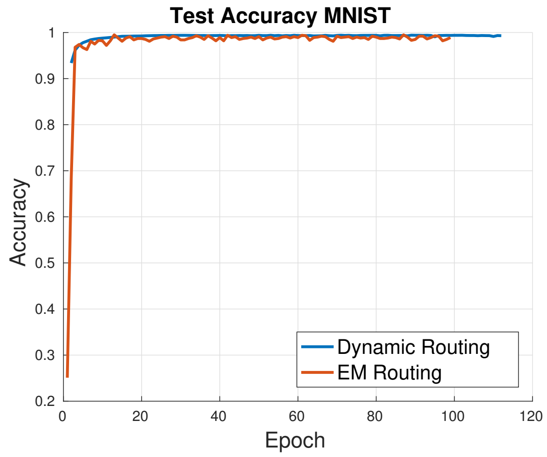


Figure 5: Epoch vs test accuracy results for the MNIST dataset.

### Do CapsNets need less training data?

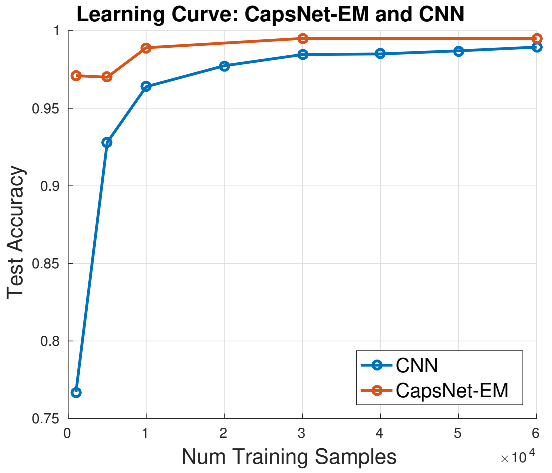


Figure 6: Learning curve of a CNN and a CapsNet with EM routing. Both models are trained on the MNIST dataset using various training set sizes.

### Landmark Recognition with CapsNets

### CBIR with CapsNets

## Discussion

## Conclusion

## References

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

[cod ] 760 project repository. `https://github.com/bad884/760-project`. Accessed: 2018.

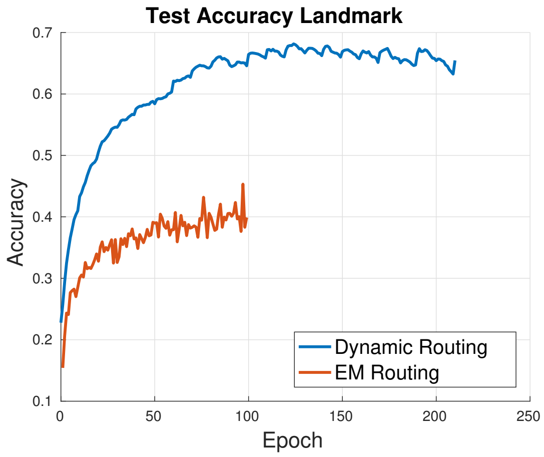


Figure 7: Epoch vs test accuracy results for the Landmark dataset.

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

[lan a] Google landmark recognition challenge. `https://www.kaggle.com/c/landmark-recognition-challenge/data`. Accessed: 2018.

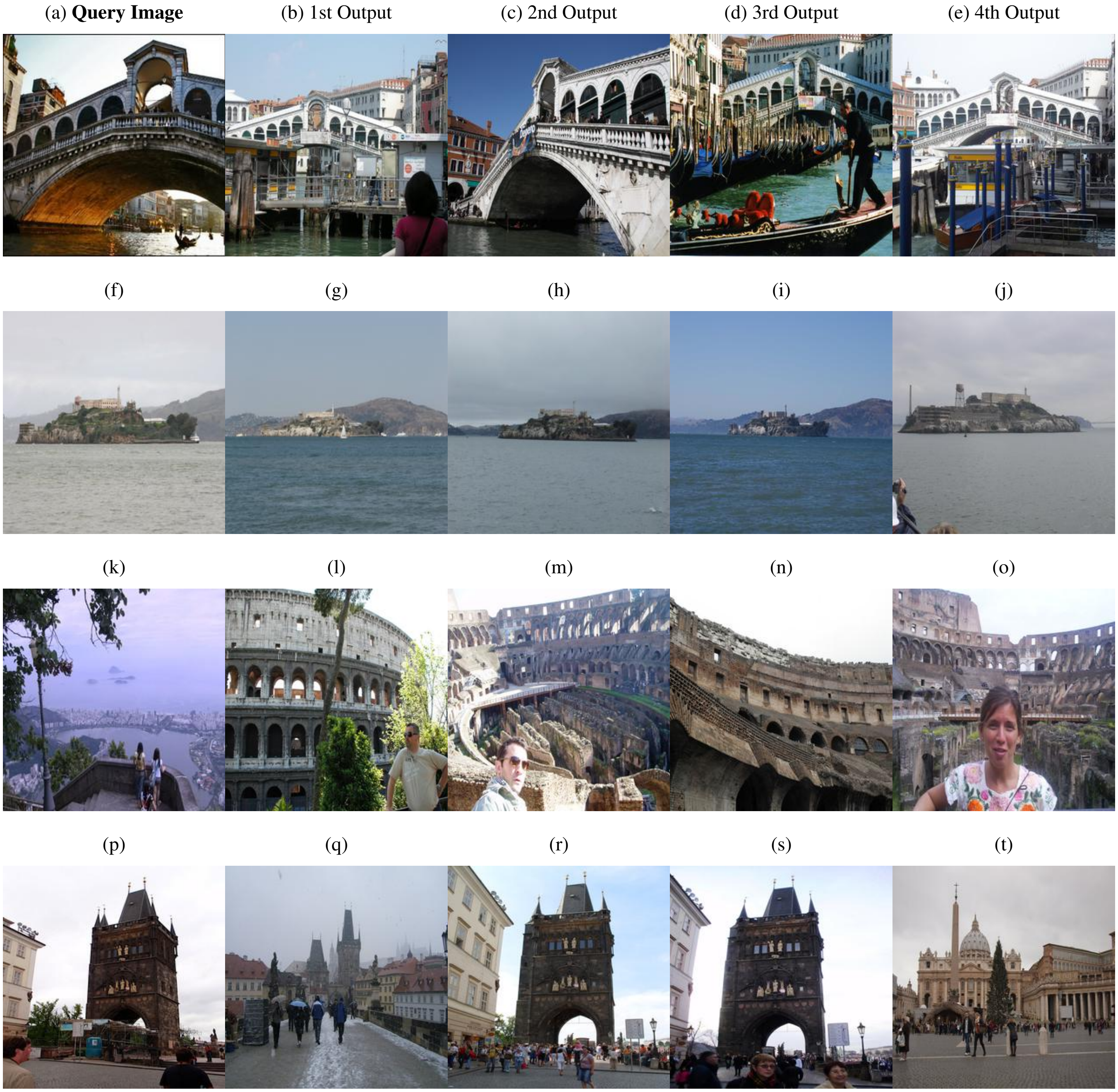
[lan b] Google landmark retrieval challenge. `https://www.kaggle.com/c/landmark-retrieval-challenge/data`. Accessed: 2018.

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

[mni ] Mnist dataset. `http://yann.lecun.com/exdb/mnist/`. Accessed: 2018.

[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*.



(a) **Query Image** (b) 1st Output (c) 2nd Output (d) 3rd Output (e) 4th Output

(f) (g) (h) (i) (j)

(k) (l) (m) (n) (o)

(p) (q) (r) (s) (t)

Figure 8: Resulting image retrieval using the CapsNetDR. The left most image is the query image. The 1st, 2nd, 3rd, and 4th are the resulting output images from the best to the worst match.