Comp541 Final Project

Gürsu Gülcü

June 1, 2018

Abstract

This project is an implementation of the paper titled "Dynamic Routing Between Capsules" by Sabour, Frosst, Hinton. A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. Capsules rectify some deficiencies of Convolutional Neural Networks by achieving viewpoint invariance. They bring additional benefits like the ability to detect multiple classes at once. In this report, the mechanism, architecture, implementation and performance of capsule networks will be described, starting with the famous MNIST dataset. The discussion will then be extended to AffNIST, a dataset derived from MNIST using affine transformations, and then to speech recognition on TIMIT dataset.

1 INTRODUCTION

Currently, the most successful deep learning technique for image recognition is Convolutional Neural Networks (CNN). However, CNNs have some drawbacks limiting their success, which this paper attempts to address:

- A CNN merely checks for the existence of some features it has learnt, not for the relationships among those features; jumping to the conclusion of the existence of the whole even when mismatching parts are present.
- Feature detection process is not robust to transformations, like when an image is rotated, accuracy decreases.
- In a CNN, repetitive operations of convolution followed by pooling leads to crude spatial invariance at the expense of information loss. Even when a feature is slightly moved, if it is still within the pooling window, it can still be detected. Nevertheless, this approach keeps only the max feature (the most dominating) and throws away the others.

Ignoring details this way keeps neural activities somewhat constant, but perhaps we should aim to keep some weights constant instead, which would code viewpoint-invariant knowledge like the size and orientation of an object. This is supported by the fact that representation of objects in the brain does not depend on view angle, and relationships between 3D objects can be represented by pose information. Hinton calls this process "*inverse computer graphics*".

2 MECHANISM OF CAPSULE NETWORKS

Hinton introduces the notion of a capsule, which is a group of neurons that captures the likelihood (via vector norm) as well as the parameters (via orientation) of a specific feature. Capsules acting in multiple layers of hierarchy collectively address the part-whole fitting problem by testing the likelihood of a lower-layer capsule being the child of a higher-layer capsule through an iterative procedure called *dynamic routing*.

As depicted in Figure 1, lower-level capsule vectors u_i's are multiplied by linear transformation matrices W_{ij}'s whose elements are to be learnt, to calculate prediction vectors $\hat{u}_{j|i}$, which iteratively help calculate higher level capsule vectors v_j's, as will be detailed next.

First, we define c_{ij}, coupling coefficients between u_i and v_j, whose sum over j equals one; and we reset to zero their log prior probabilities b_{ij}. For r iterations (usually between 1 and 3), we start by calculating the softmax of b_{ij}'s along the second (row) axis so that its element-wise product with

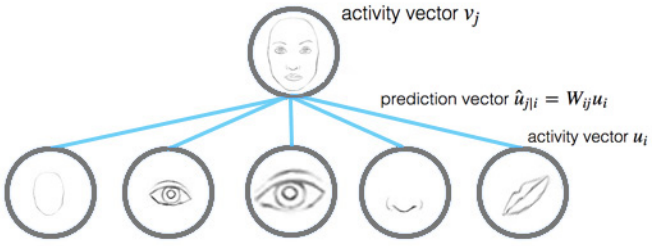

Figure 1: Applying a Transformation Matrix to the Capsule Output of the Previous Layer

$\hat{\mathbf{u}}_{j|i}$ summed along the first (column) axis will output the higher capsule v_j when its magnitude is normalized (squashed). Squashing, whose formula is given in Figure 2, introduces a non-linearity and shrinks smaller vectors much more than large vectors.

$$\mathbf{v}_j = \underbrace{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}_{\text{additional "squashing"}} \underbrace{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}_{\text{unit scaling}}$$

Figure 2: Squashing Function

The coupling coefficients are then updated by adding the dot product of prediction and higher capsule vectors, better reflecting whether the whole-part relationship exists with the pose information u_i is holding. The result of the dot product is called agreement vector; and routing a capsule to the capsule in the layer above based on relevancy is called routing-by-agreement.The steps of the routing algorithm is shown in Figure 3.

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

Figure 3: Steps of the Routing Algorithm

The transformation matrix $\mathbf{W}$ is trained by backpropagation using a cost function which is the sum of two components: The first component, *marginal loss*, forces an object of class k (Digit Capsules for MNIST) to exceed m^+=0.9 if it is present, or fall below m^-=0.1 otherwise. The second component, *reconstruction loss*, is an optional regularizer, and slightly penalizes deviations from the learnt ideal output shape corresponding to the label.

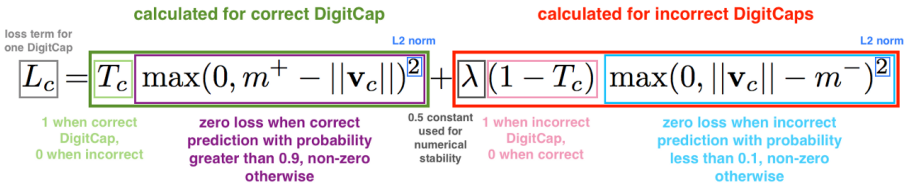

Figure 4: Marginal Loss Function

3 CAPSULE NETWORK ARCHITECTURE

The input, 28x28 MNIST images in this case, are fed into two convolutional layers in tandem: The first with 256 9x9 kernels and stride=1, and the second again with 256 9x9 kernels but stride=2 this time. There is no pooling operation in either of the layers. A block diagram is given in Figure 5.

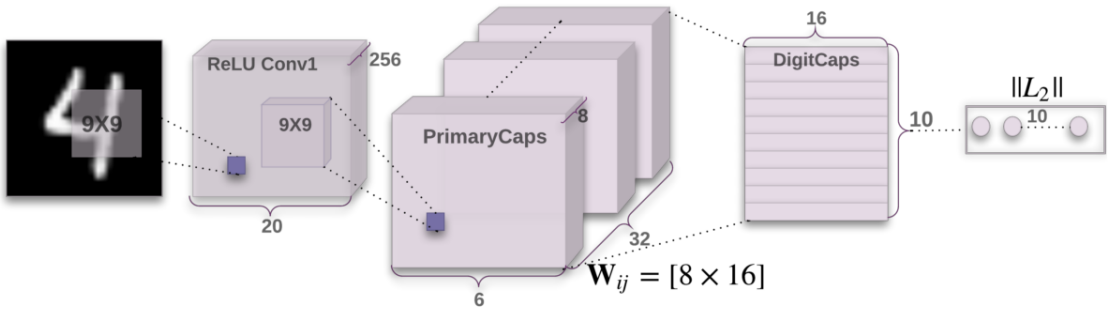

Figure 5: Capsule Network Architecture

The 6x6x256 output is then organized as primary capsules into 32 groups of 6x6 vectors of dimension 8, as depicted block diagram is given in Figure 6. Each of these 8-dimensional vectors are to be transformed by a 8x16 matrix $\mathbf{W}$, to arrive at class (digit) capsules of dimension 16 using the iterative routing algorithm described above. The norm of digit capsules are class existence probabilities. They do not add up to 1.0 when summed over all classes, hence the ability to detect multiple classes, albeit without their counts.

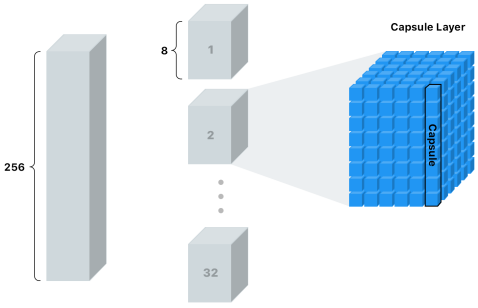

Figure 6: Formation of Primary Capsules

3.1 RESULTS

As a first step, Hinton's baseline model was implemented as described in Section 5 of the paper. The model is a standard CNN with three convolutional layers of 256, 256, 128 channels. Each has 5x5 kernels and a stride of 1. The last convolutional layer is followed by two fully connected layers of size 328, 192. The last fully connected layer is input with dropout into a 10-class softmax function, and cross entropy is used as loss metric.

The motivation behind this baseline is to achieve the best performance on MNIST while keeping the computation cost similar to CapsNet. In terms of number of parameters, the baseline has 35.4M, while CapsNet has 8.2M parameters.

The model was trained with Adam optimizer as indicated in the paper, over 40 epochs and mini-batches of 100. The dropout rate for the fully connected layer was not specified, so I used 25%. The article obtains a test error of 0.39%, and initially I got 0.56%. This may be due to a suboptimal dropout rate. The error settles down after about 20 epochs as can be seen from the graph below, so I decided to try lower and higher dropout rates and ran for fewer epochs.

Increasing the dropout to 40% resulted in an error of exceeding 0.6%, so did decreasing it to 20%. Possibly these differences are not very significant, and I conclude that the baseline CNN model test error is around half a percent.

Next, the CapsNet architecture was implemented both with and without reconstruction, in addition to taking the number of iterations r in the routing algorithm 1 and 3; which yields 4 configurations. Table 1 shows my best test error results on the MNIST dataset, compared to the reported results by Hinton et al.

METHOD	**ROUTING ITER.**	**RECONSTRUCT**	**REPORTED**	**MY RESULT** (*@Epoch*)
Baseline CNN	-	-	0.39	0.56 (*40*)
CapsNet	1	NO	0.34	0.74 (*5*)
CapsNet	1	NO	0.29	0.61 (*16*)
CapsNet	3	YES	0.35	0.86 (*9*)
CapsNet	3	YES	0.25	0.65 (*12*)

Table 1: MNIST Test Errors - Reported vs My Results (%)

Figure 7 displays the dynamic details of the table above. We can possibly infer that reconstruction plays its role as regularizer well since without it models quickly overfit and stop improving. We can also say the number of routing iterations do not seem to play a major role, possibly wasting CPU/GPU time.

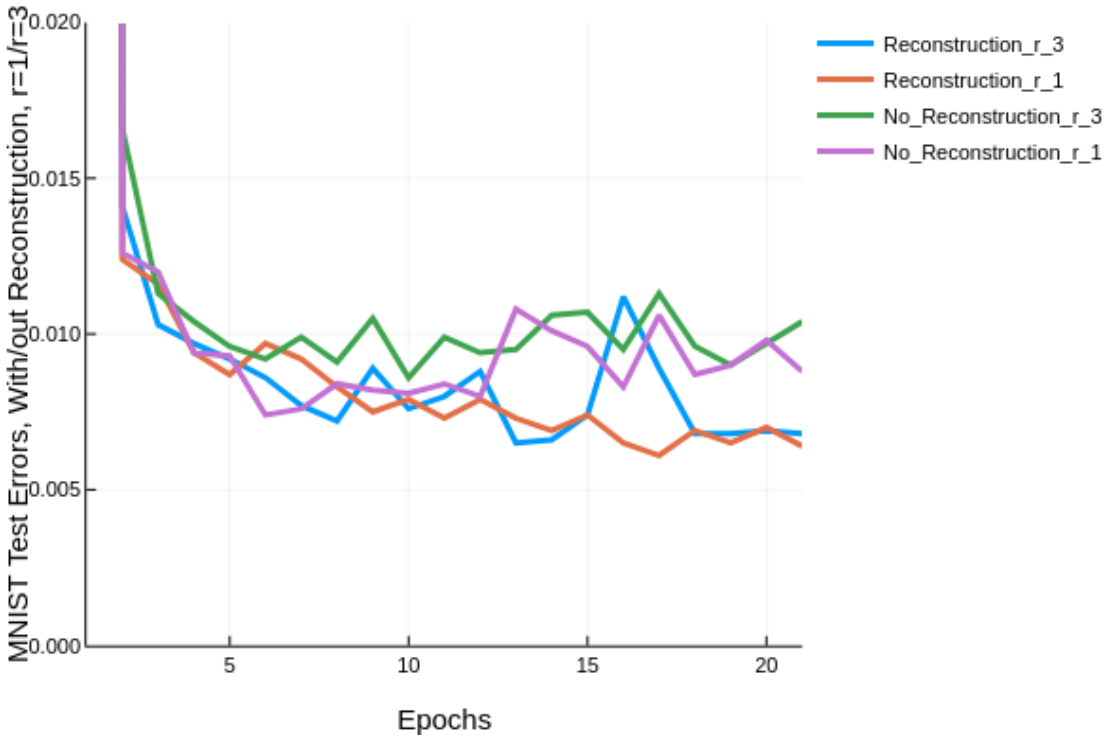

Figure 7: Comparison of MNIST Test Errors by Different Configurations

Moving on to the other dataset called AffNIST, which consists of almost 2M 40x40 images obtained by applying affine trasnformations to MNIST images, I was able to achieve a test error of 2.46% by directly training on 50K images and testing on another 10K images. The paper reports 21% error, but they have used the model trained on MNIST to test AffNIST accuracy. The images from the two datasets are not of the same size however, so the model weight dimensions are different. How they have mapped from one model to another is not detailed in the paper, but the potential of learning viewpoint invariant weights $\mathbf{W}$ has been demonstrated.

MultiMNIST is another dataset mentioned in the paper, but it is not publicly available. Its images are created programmatically by overlaying a digit on top of another digit. I have experimented by superimposing AffNIST images, but my accuracies were not comparable.

While not discussed in the paper, I have further applied capsule network techniques to speech recognition using the TIMIT dataset as a bonus. I calculated mel filterbank features of 3696 training and 1344 test utterances (sound files containing individual sentences). A mel filterbank shows the short-time and frequency characteristics of sound waves, typically in time windows of 25 ms and advancing in 10 ms steps.

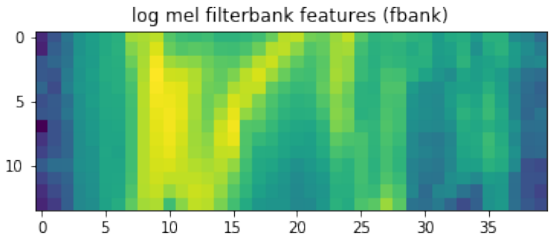

Figure 8: Sample 40x14 image created for the '**aa**' phoneme

Each different sound (phoneme) in each utterence was transformed into an Nfilter x 14 image and was labelled; where Nfilter was tried as 40 and 28. A sample output of the phoneme '**aa**' as pronounced in 'f**a**ther' is depicted in Figure 8.

Phoneme centers have been read from the dataset sample labels (from PHN files), and all corresponding images have been generated. The resulting 121583 training and 44125 test images were used to test the classification power of capsules under different configurations and in relation to simpler CNNs and MLPs. The CNN used is similar to the baseline in MNIST, with filters adjusted to account for the different input image sizes.

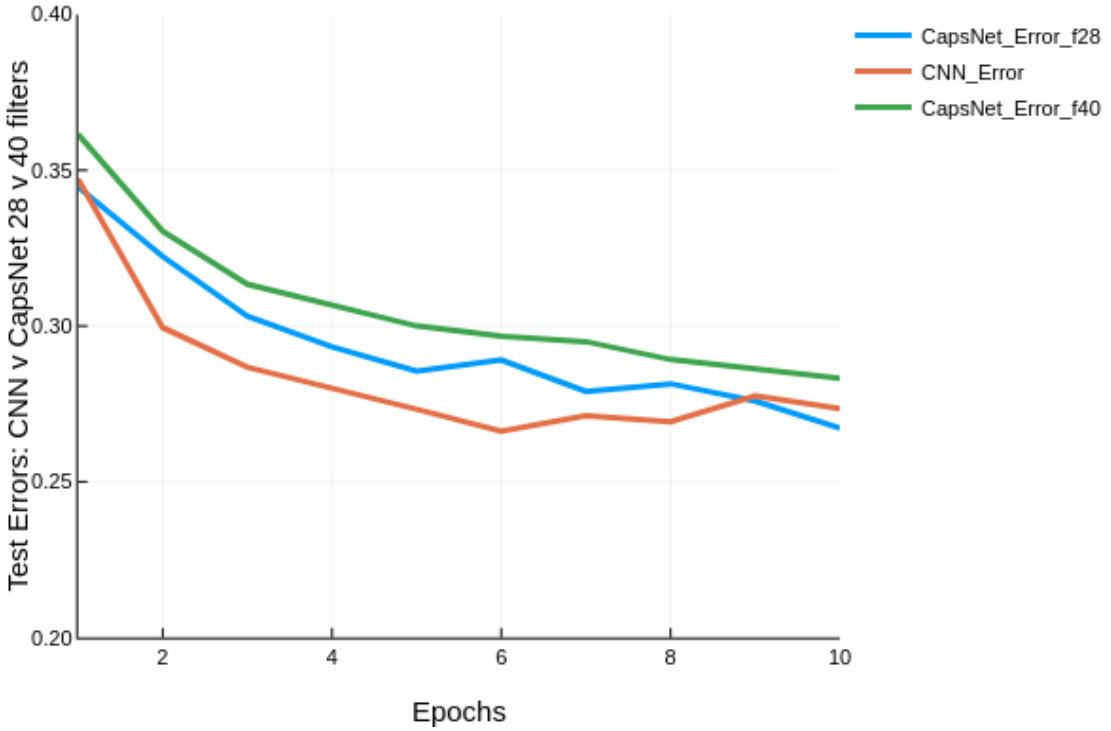

Figure 9: Test Errors of CNN vs CapsNet vs Cap

We can see from Figure 9 that CNN seems to have quickly overfit, while the two CapsNet configurations possibly still retain their learning capacity at the end of epoch 10. The input to CNN (red) is 40x14, as well as one CapsNet model (green), while the input of the second CapsNet (blue) is 28x14. Unfortunately I did not have enough time to continue the experiment beyond epoch 10. The test error rates are above 25%, but a phoneme error rate around 20% is considered normal for TIMIT. In the literature, the CNN architecture is supported with Hidden Markov models, for

instance. [3]

4 CONCLUSION AND FUTURE WORK

During my tests, I have observed Capsule Networks to take an order of magnitude more time to complete one epoch when compared to the baseline CNN model, which has a similar number of parameters. This computational expense may be worth it under the following circumstances:

- Viewpoint invariance is needed, unlike the case for MNIST. Hinton's follow-up paper [2] extends capsules and applies them to 3-D images.
- Multi-class detection is to be performed. The norms of the capsules of the final layer indicate existence probabilities, and do not add up to 1.0, and the loss function is designed to detect multiple classes but not their counts. This capability may especially be useful in speech recognition where phoneme frames are of varibale length, and usually more than one phoneme exists in a typical analysis frame. Moreover, surrounding phonemes influence the shapes of each other, so it is common to analyze multiple phonemes at once.

As a next step, I shall assess the use of Hidden Markov models, sequence to sequence models, and attention mechanisms to increase the accuracy of phoneme detection. I plan to further incorporate a language model to arrive at detecting words after phonemes.

5 REFERENCES

[1] Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton, "Dynamic Routing Between Capsules," in NIPS 2017 Proceedings.

[2] Geoffrey Hinton, Sara Sabour, Nicholas Frosst. "Matrix Capsules with EM Routing," published as a conference paper at ICLR 2018.

[3] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pp. 1533–1545, Oct 2014.