Employing Capsules in Speech Recognition

Comp504 Project

Gürsu Gülcü

***Abstract*—This project is an application of the concepts discussed in the paper titled "Dynamic Routing Between Capsules" by Sabour, Frosst, Hinton to speech recognition. 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 in the context of phoneme recognition on the famous TIMIT dataset.**

I. 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*".

II. THE 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 Fig. 1, lower-level capsule vectors $\mathbf{u}_i$'s are multiplied by linear transformation matrices $\mathbf{W}_{ij}$'s whose elements are to be learnt, to calculate prediction vectors $\hat{\mathbf{u}}_{j|i}$, which iteratively help calculate higher level capsule vectors $\mathbf{v}_j$'s, as will be detailed next.

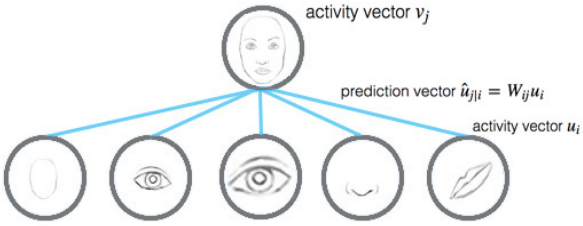

Figure 1: Applying a Transformation Matrix to the Capsule Output of the Previous Layer

First, we define c_{ij} , coupling coefficients between $\mathbf{u}_i$ and $\mathbf{v}_j$, whose sum over j equals 1.0; 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 $\hat{\mathbf{u}}_{j|i}$ 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 Fig. 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 $\mathbf{u}_i$ 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 Fig. 3.

The transformation matrix **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, as explained in Fig. 4. The second component, r*econstruction loss*, is an optional

regularizer, and slightly penalizes deviations from the learnt ideal output shape corresponding to the label.

procedure ROUTING($\hat{\mathbf{u}}_{j|i}$, r, l)

for all capsule i in layer l and capsule j in layer (l+1):

$b_{ij} \leftarrow 0$

for r iterations do:

for all capsule i in layer l: $c_i \leftarrow \text{softmax}(\mathbf{b}_i)$

for all capsule j in layer (l+1): $\mathbf{s}_j \leftarrow \sum_{\{i\}} c_{ij}\, \hat{\mathbf{u}}_{j|i}$

for all capsule j in layer (l+1): $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$

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

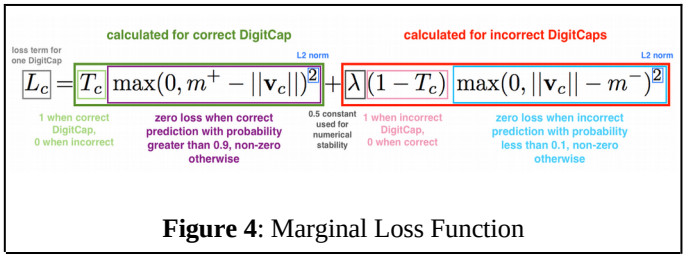

Figure 4: Marginal Loss Function

III. CAPSULE NETWORK ARCHITECTURE

In the original paper [1], the inputs, which are 28x28 images, 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 Fig. 5.

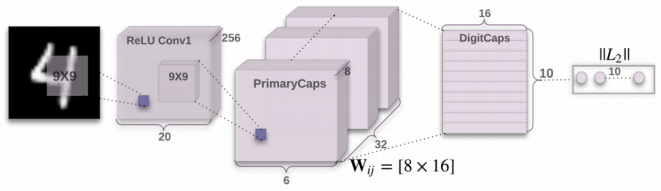

Figure 5: Capsule Network Architecture

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 Fig. 6. Each of these 8-dimensional vectors are to be transformed by a 8x16 matrix **W**, to arrive at class (digit) capsules of dimension 16 using the iterative routing algorithm described above. The norm of the digit capsules are class existence probabilities. They do not add up to 1.0 when summed over all classes, giving the ability to detect multiple classes, albeit without their counts.

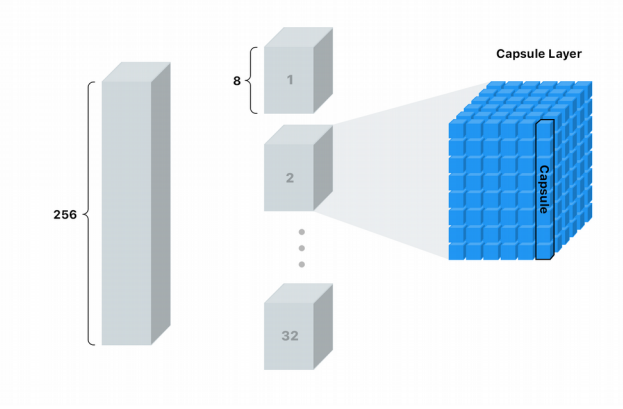

Figure 6: Formation of Primary Capsules

IV. EXPERIMENTS AND RESULTS

I have applied Capsule Network techniques to speech recognition on the TIMIT dataset. First, mel filterbank features of 3696 training and 1344 test utterances sampled at 16 Khz (sound files containing individual sentences) were calculated. SA files were omitted since those contain dialects. 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.

Secondly, the sample numbers corresponding to the center of each sliding window was calculated. Then the phoneme label at that window was extracted from the associated PHN file. The duration of a single phoneme may be longer than 100 ms, or 10 windows, so the middle window center of consecutive windows centers belonging to the same phoneme was noted, together with its phoneme label.

Thirdly, the filterbank for the utterance was sliced, starting 7 window centers before and ending 6 window centers after the middle window center, making an image of dimensions (Number of filters x 14). Number of filters was taken as both 28 and 40 in the configurations. The label of the image was taken as the label of the middle window center. A sample output of this process for the phoneme labeled as ‘aa’ (as pronounced in ‘father’) is depicted in Fig. 7.

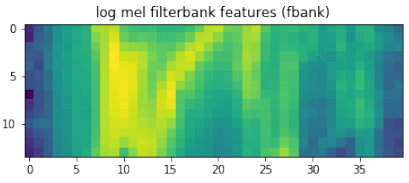

Figure 7: A 40x14 image created for the ‘**aa**’ phoneme

TIMIT has originally 61 different phoneme labels, but it is a common practice to group similar sounds together when the aim is to arrive at word recognition rather than investigate

dialects. The 61 labels were initially reduced to 45, and then further to 38 as in [3].

Generating all images corresponding to each phoneme middle in each utterance resulted in 121583 training and 44125 test images. Different CapsNet configurations were used in comparison to multi-layer perceptrons (MLP's) and CNN's. For the 3-way test error comparison depicted in Fig. 8, the following configurations were used:

MLP: 3 hidden layers of sizes 1024, 256, and 64 with 20% dropout only on input, which was 40x14 images (40 filters). Cross entropy was used as loss metric.

CNN : 3 convolutional layers of 256, 256, 128 channels. Each layer has 5x3 kernels with no pooling and padding=1. The last convolutional layer is followed by two fully connected layers of size 328 and 192. The last fully connected layer is input with 30% dropout into a 45-class softmax function, and cross entropy is used as loss metric. The input was 40x14 images (40 filters).

CapsNet_f28: The input was 28x14 images (28 filters) for this experiment. The two convolutional layers had both 128 channels, with 9x5 filters, no pooling, and with stride=1 and then stride=2. The convolution output was segmented as 6x3 8-D primary capsules per image for 16 vertical stacks, and phoneme capsules of 16-D were classified against 45 classes.

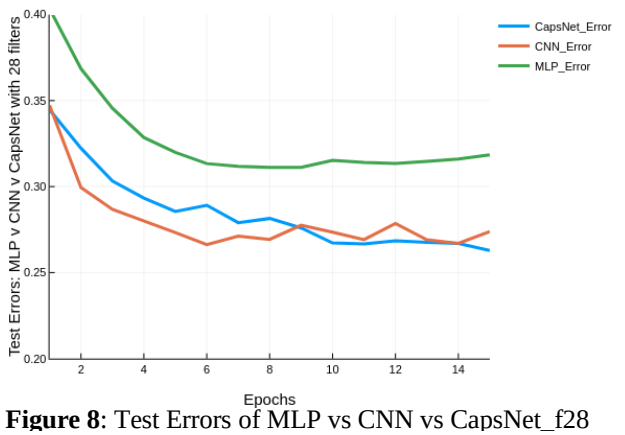

Figure 8: Test Errors of MLP vs CNN vs CapsNet_f28

From Fig. 8, we can see CNN performed on par with CapsNet in the first experiment. The second comparison investigated the effect of using 28 filters in the filterbank vs 40 filters. The CNN and CapsNet_f28 configurations are the same as above, and the additional CapsNet config is like this:

CapsNet_f40: The input was 40x14 images (40 filters). The two convolutional layers had both 128 channels, with 15x1 and 15x3 filters respectively, no pooling, and with stride=1 and then stride=2. The convolution output was segmented as 6x6 8-D primary capsules per image for 16 vertical stacks, and phoneme capsules of 16-D were classified against 45 classes.

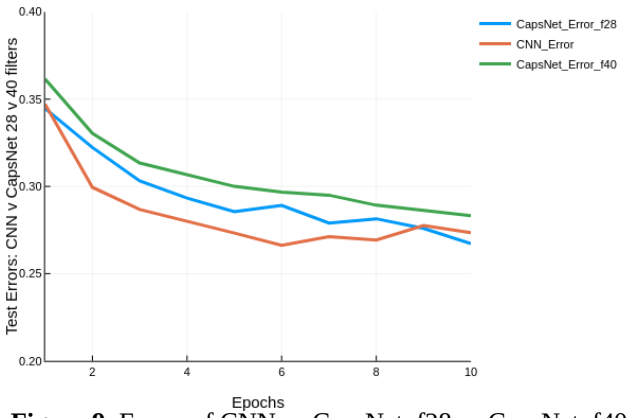

Figure 9: Errors of CNN vs CapsNet_f28 vs CapsNet_f40

Surprisingly the 40-filter CapsNet did not perform better that the 28-filter CapsNet, but both seemed to retain their learning capacity at the end of epoch 10, unlike CNN. Unfortunately I did not have enough time to continue the experiment beyond epoch 10.

Although error rates around 20% are considered normal for this dataset, I wanted to check if increasing the capacity of secondary (phoneme) capsules would increase accuracy. The previous configurations were adopted from the MNIST architecture, but TIMIT has far more number of classes than MNIST (45 vs 10). So, in third comparison the phoneme capsules are 20-D instead of 16. I was able to run the new model for 5 epochs, and the new configuration is as follows.

CapsNet_sec20D: The input was 40x14 images (40 filters). The two convolutional layers had both 64 channels, with 9x3 filters both, no pooling, and with stride=1 and then stride=2. The convolution output was segmented as 12x5 8-D primary capsules per image for 8 vertical stacks, and phoneme capsules of 20-D were classified against 45 classes.

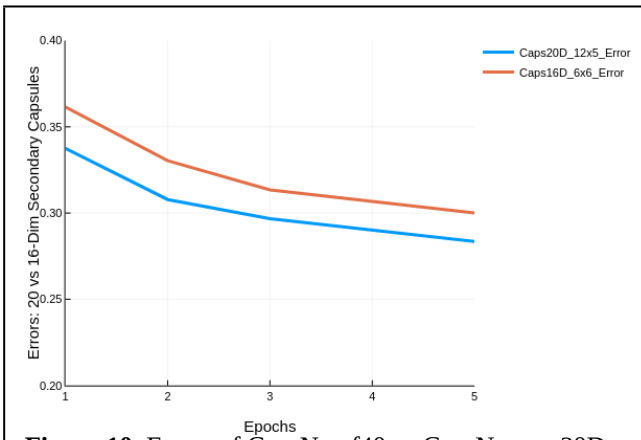

Figure 10: Errors of CapsNet_f40 vs CapsNet_sec20D

The observation of decreased errors of about 2% from Fig. 10 may indicate that secondary capsules need more capacity to hold phoneme information, as suspected.

One final experiment tested the effect of reducing the phoneme labels from 45 to 38. The data for 45 labels is of CapsNet_f40, and the new model configuration is defined below.

CapsNet_phn38: The input was 40x14 images (40 filters). The two convolutional layers had both 128 channels, with 15x5 filters both, no pooling, and with stride=1 and then stride=2. The convolution output was segmented as 6x3 8-D primary capsules per image for 16 vertical stacks, and phoneme capsules of 16-D were classified against 38 classes.

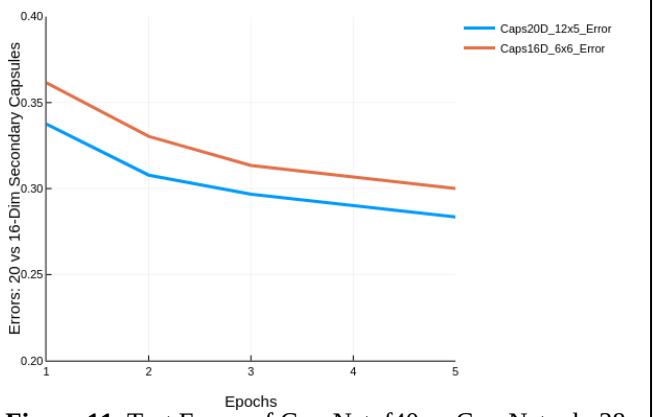

Figure 11: Test Errors of CapsNet_f40 vs CapsNet_phn38

As expected, a lower number of classes decrease the minimum error, aroud 2%. In the literature, the CNN architecture is supported with Hidden Markov models, like in [4] for instance.

V. 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. Same phonemes have different durations and change shape when the speaker is different so some kind of translational invariance may be useful for speech recognition.
- 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. This capability may especially be useful in speech recognition where phoneme frames are of variable 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.
- It is argued that we are nearing the limits of what can be achieved using deep neural networks. Capsule Networks bring a new approach, an additional round of of routing by agreement, to traditional neural network calculations. Further research may lead to refinements of capsules which could enable much higher accuracies than previously possible.

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.

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] Lee, K. & Hon, H. (1989). Speaker-independent phone recognition using hidden Markov models. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol.37 (11), November 1989, pp. 1642-1648, ISSN: 0096-3518.

[4] 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. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.