

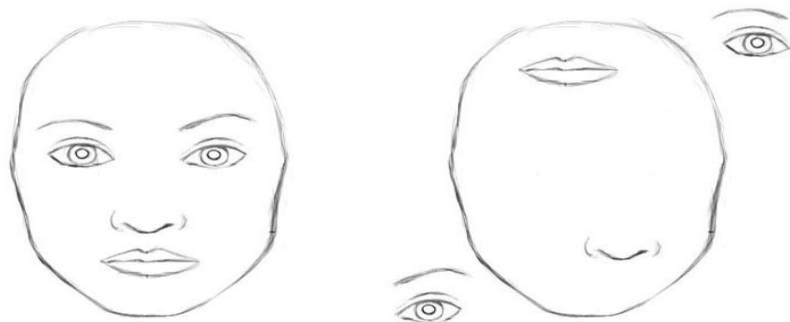
Capsule Networks

COMPUTER VISION PROJECT 03

Ojas Barve (200213803)
Sharvari Deshpande (200206230)
Anushka Gupta (200206727)

Motivation

CNNs are Translational invariant, means they are unable to identify the position of one object relative to another. For example, CNN's predicts a face to a bunch of randomly assembled face parts because all the key features are there. But Capsule Networks have the ability to identify that the face parts are not in correct position relative to another.

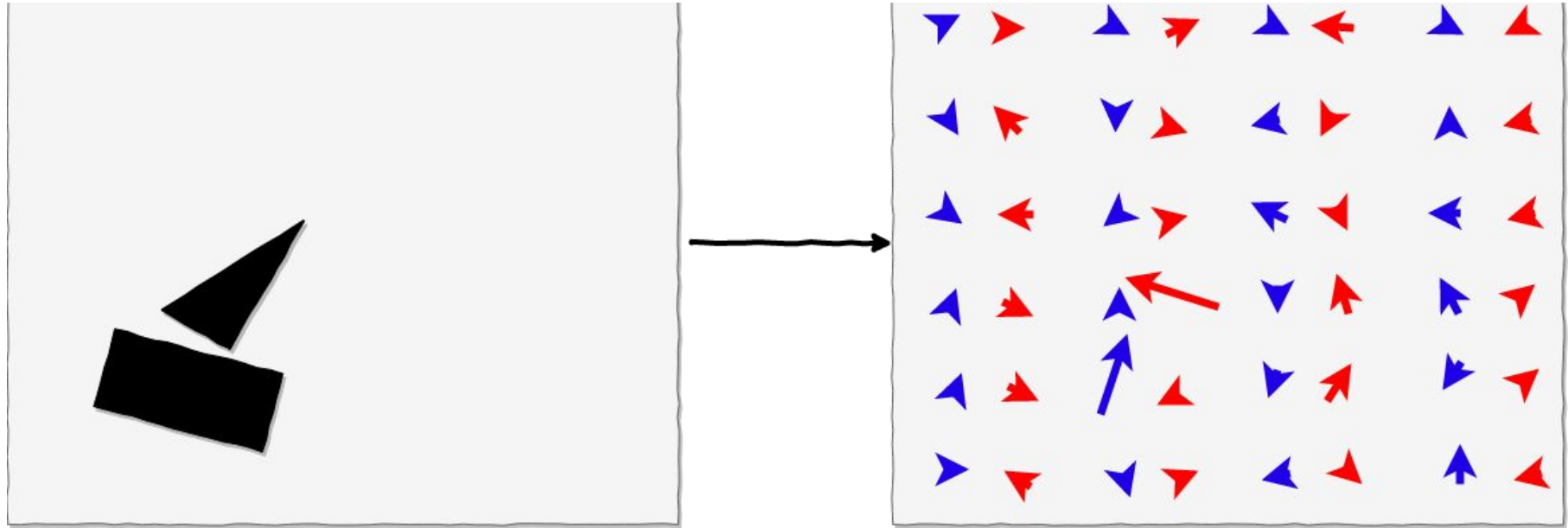


Objective

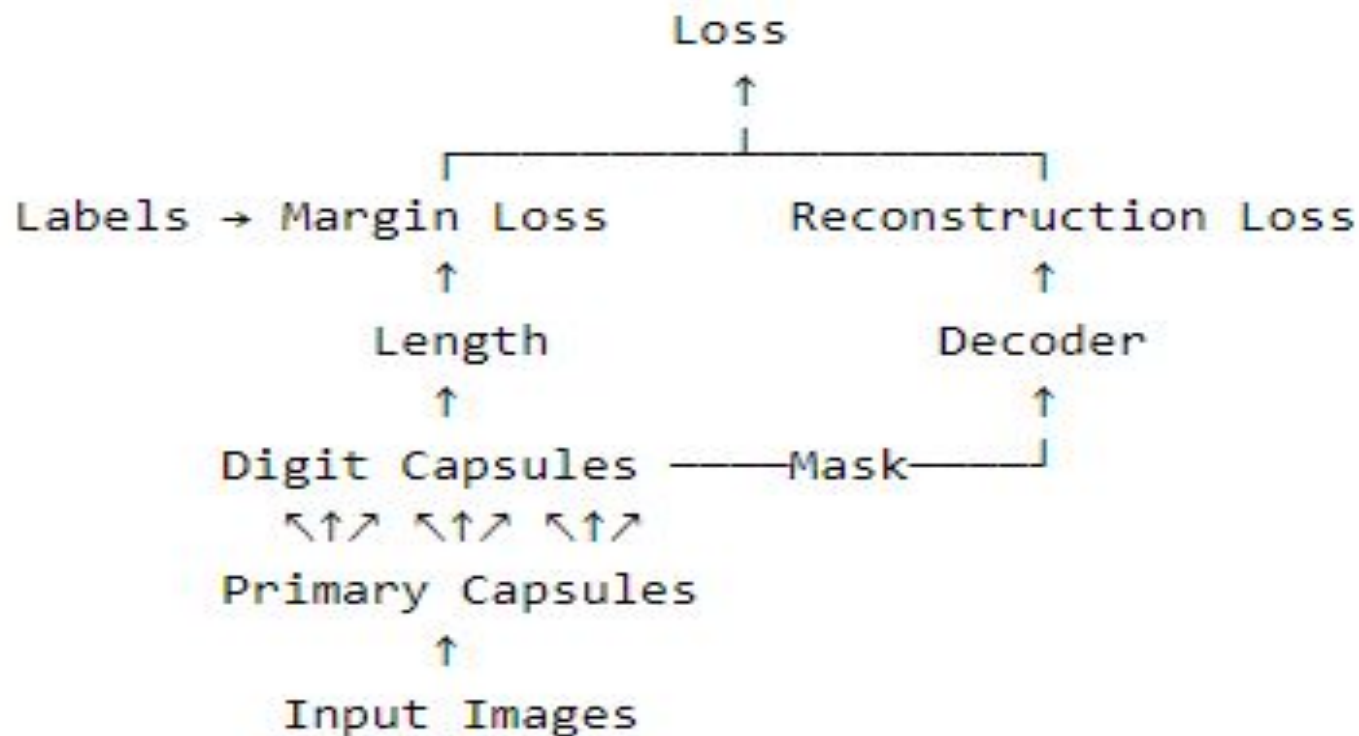
The key idea behind the project is to explore the capsule network space on the Street View Housing Numbers (SVHN) dataset using the TensorFlow framework based on the paper.



Overview of CapsNet – Inverse graphics Approach



CapsNet Flowchart



Squashing Function Used:

$$(\mathbf{s}) = \frac{\|\mathbf{s}\|^2}{1 + \|\mathbf{s}\|^2} \frac{\mathbf{s}}{\|\mathbf{s}\|}$$

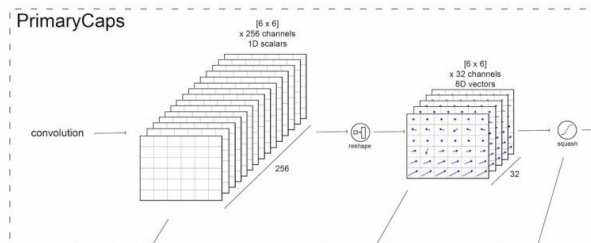
Margin Loss:

$$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda(1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2$$

- T_k is equal to 1 if the digit of class k is present, or 0 otherwise.
- In the paper, $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

Method overview

A Visual Representation of Capsule Connections in *Dynamic Routing Between Capsules*



Standard convolution outputs 256 channels of scalars in 6x6 arrays.

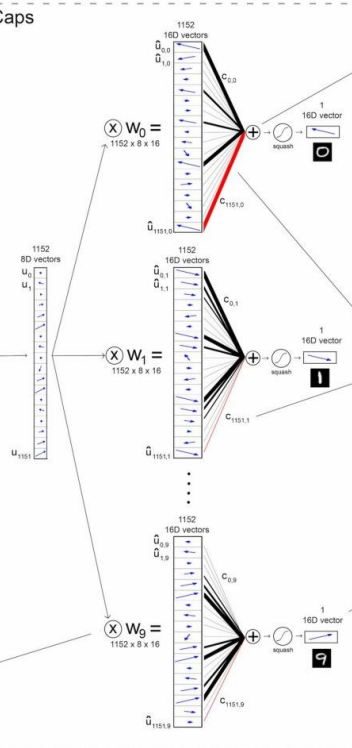
The 256 channels may be grouped by 8, and reinterpreted as 32 channels of 8D vectors in 6x6 arrays.

The squash non-linearity is a vector analogue of the sigmoid. Just as sigmoid remaps scalars onto (0, 1), squash scales vector lengths onto (0, 1), without changing their orientation.

Each of the 1152 8D vectors (\mathbf{u} in the paper) is transformed into a 16D vector via matrix multiplication with \mathbf{W} . The result is $\hat{\mathbf{u}}$, consisting of 1152 16D vectors for each output class capsule (11520 in total).

Mike Ross
mikenon@gmail.com

DigitCaps



Each capsule computes a weighted average of the transformed input vectors in $\hat{\mathbf{u}}$. The "coupling" weights, \mathbf{c} , are determined on each forward pass via an iterative routing algorithm that acts as a sort of orientation-popularity filter. If multiple large vectors point in the same direction, they will get a large weight. Shorter vectors pointing in scattered directions will get a small weight.

The routing algorithm also uses softmax so that each of the 1152 input vectors in \mathbf{u} , sends most of its activity to just one of the ten outputs. The routing for \mathbf{u}_{1151} is shown in red. Although both $\hat{\mathbf{u}}_{1151,0}$ and $\hat{\mathbf{u}}_{1151,1}$ have popular orientations in their respective digit capsules, only one has a large weight, $\mathbf{c}_{1151,0}$, so most of the activity flows to the digit 0 output.

The output vector is squashed so that its length can model the probability that the capsule's digit is present.

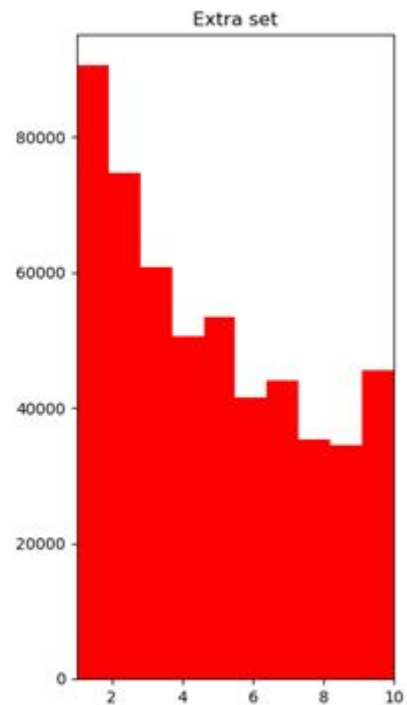
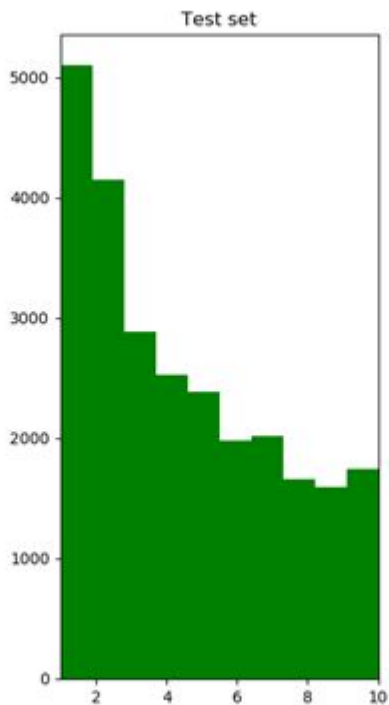
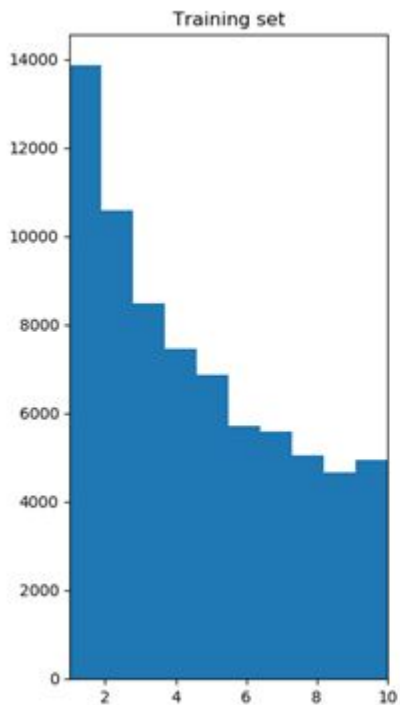
Experimental results

Dataset creation:

- We used the Street View Housing Numbers (SVHN) Dataset for experimentation.
- They are divided as 73, 257 for training, 26, 032 for testing and it has an additional 531, 131 as extra images
- We decided to create a balanced validation set to ensure, the model has an ability to recognize digits with each number having an equal importance.

```
Training set (69257, 32, 32, 1) (69257, 10)
Validation set (6000, 32, 32, 1) (6000, 10)
Test set (26032, 32, 32, 1) (26032, 10)
```


Data View



Experimental results

At each epoch:

Epoch: 1 Val accuracy: 75.0167% Loss: 7.686755
Epoch: 2 Val accuracy: 81.3167% Loss: 7.841952
Epoch: 3 Val accuracy: 85.7500% Loss: 7.548036
Epoch: 4 Val accuracy: 87.6500% Loss: 7.603930
Epoch: 5 Val accuracy: 87.8167% Loss: 7.626252
Epoch: 6 Val accuracy: 88.8333% Loss: 7.597592
Epoch: 7 Val accuracy: 88.5000% Loss: 7.485849
Epoch: 8 Val accuracy: 88.6500% Loss: 7.567253
Epoch: 9 Val accuracy: 87.6333% Loss: 7.603738
Epoch: 10 Val accuracy: 90.8500% Loss: 7.582572

Accuracy:

The final averaged out accuracy and loss for the model are:

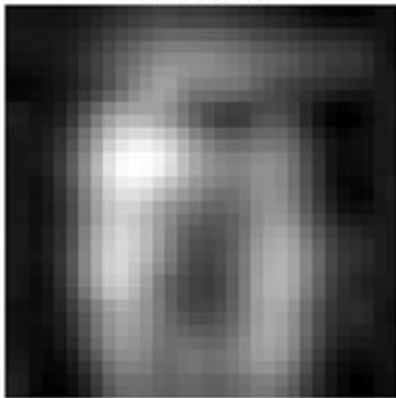
Final test accuracy: 88.4385% Loss: 8.275824

Experimental Results

The model correctly predicts for the following:

- Numbers from images on the downloaded dataset

Pred:6 Label: 6



Pred:5 Label: 5

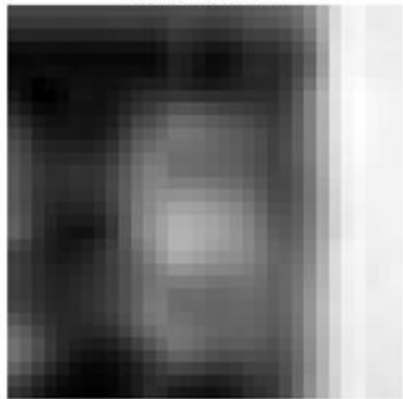


Pred:0 Label: 0



Misclassifications

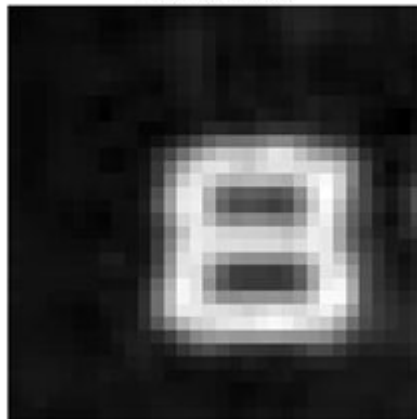
Pred:6 Label: 8



Pred:2 Label: 9



Pred:6 Label: 8



Experimental results

- Predictions on images



Figure : Image Captures

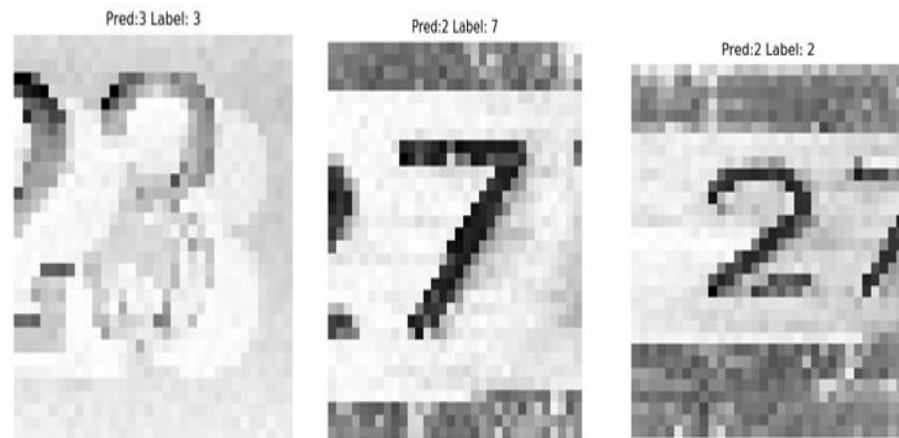


Figure : Image Predictions

Experimental results

- Predictions at different angles, for example 90 degrees.

Pred:1 Label: 2



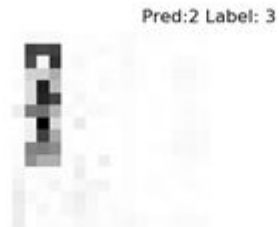
Pred:3 Label: 3



Pred:3 Label: 3



Translated Images



Pred:1 Label: 1



Conclusion

From the results obtained above the capsule network did not perform as well as it performed on the MNIST data set as described in the paper.

MNIST Results:

INFO:tensorflow:Restoring parameters from ./my_capsule_network

Epoch: 1 Val accuracy: 99.4400% Loss: 0.007998 (improved)

Epoch: 2 Val accuracy: 99.3400% Loss: 0.007959 (improved)

Epoch: 3 Val accuracy: 99.4000% Loss: 0.007436 (improved)

Epoch: 4 Val accuracy: 99.4000% Loss: 0.007568

Epoch: 5 Val accuracy: 99.2600% Loss: 0.007464

Epoch: 6 Val accuracy: 99.4800% Loss: 0.006631 (improved)

Epoch: 7 Val accuracy: 99.4000% Loss: 0.006915

Epoch: 8 Val accuracy: 99.4200% Loss: 0.006735

Epoch: 9 Val accuracy: 99.2200% Loss: 0.007709

Epoch: 10 Val accuracy: 99.4000% Loss: 0.007083

Future Work

- We intend to experiment with different face datasets which we used in the previous projects and try inference on translated, skewed, disfigured and rotated images .

References

<https://arxiv.org/abs/1710.09829>

https://github.com/ageron/handson-ml/blob/master/extra__capsnets.ipynb

<https://www.youtube.com/watch?v=pPN8d0E3900&feature=youtu.be>