CS230

Hierarchical Neural Network

Suresh Sugumar
Department of Computer Science
Stanford University
ssugumar@stanford.edu

Abstract

The state-of-the-art convolutional neural networks works well for image recognition tasks but suffers on translational invariance (an inability to recognize the same object viewed from a different angle), as the pooling layers used ignores the relation between the parts and the whole. We propose an alternate neural net architecture to mitigate this issue, which is a hierarchical capsule network, where 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 and the length of the activity vector is used to represent the probability that the entity exists and its orientation to represent the instantiation parameters. We implemented a hierarchical multi-layer capsule network and compared its performance against a traditional convolution network that are similarly trained. Results show that both networks achieve similar performance on MNIST but capsule network is significantly better than a convolutional network at recognizing highly overlapping, rotated and twisted digits.

1 Introduction

Object recognition is a problem of fundamental importance in visual perception. The ability to extrapolate from raw pixels to the concept of a coherent object persisting through space and time is a crucial link connecting low level sensory processing with higher-level reasoning. A fundamental problem in object recognition is the development of image representations that are invariant to common transformations such as translation, rotation, and small deformations. Intuitively, these are desirable properties for an object recognition algorithm to have: a picture of a cat should be recognizable regardless of the cat's location and orientation within the image. There are multiple hypotheses regarding the source of translation invariance in CNNs. One idea is that translation invariance is due to the increasing receptive field size of neurons in successive convolution layers. Another possibility is that invariance is due to the pooling operation.

The figure 1 below describes the problem of translational invariance and brings up the weakness of CNN. Today workarounds for this problem include training the network with lots of augmented data varying angle, view point, scale, etc., but at the cost of exploding set of train data set, and is very inefficient method. The max-pooling layer is claimed to be responsible for loosing lots of valuable information thereby unable to derive a relation between the parts and whole, and this needs to be fixed to improve translational invariance.

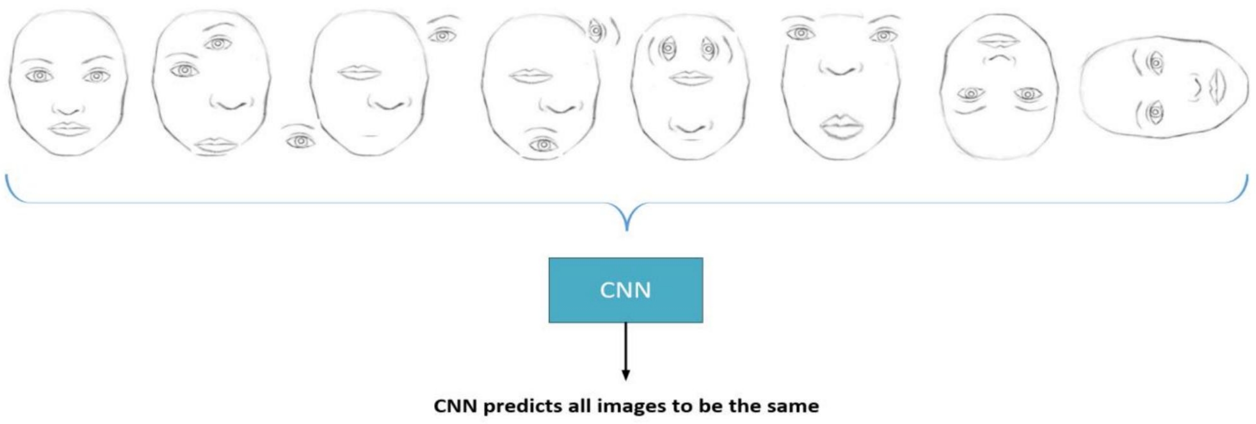


Figure 1: Poor Translational Invariance property of CNN

Based on Hinton et. al., [1] we introduce the concept of hierarchical neural network (HNN) with capsule as a building block for the new neural net architecture that is geared towards high translational invariance by eliminating pooling layers. We assume that our multi-layer visual system creates a parse tree-like structure on each fixation, and we ignore the issue of how these single-fixation parse trees are coordinated over multiple fixations. Parse trees are generally constructed on the fly by dynamically allocating memory. Each layer will be divided into many small groups of neurons called “capsules” and each node in the parse tree will correspond to an active capsule. Using an iterative routing process, each active capsule will choose a capsule in the layer above to be its parent in the tree. For the higher levels of a visual system, this iterative process will be solving the problem of assigning parts to wholes, is more aligned towards our biological process of visual processing in the brain.

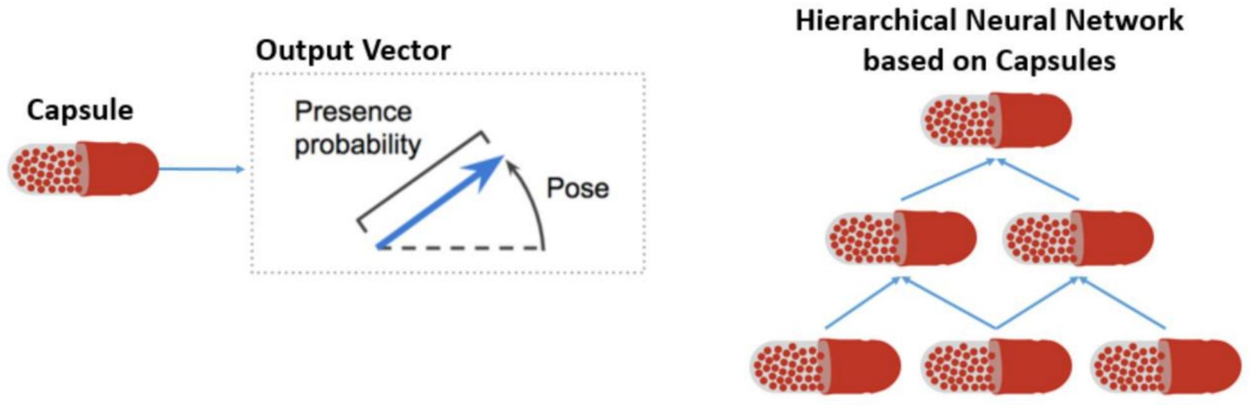
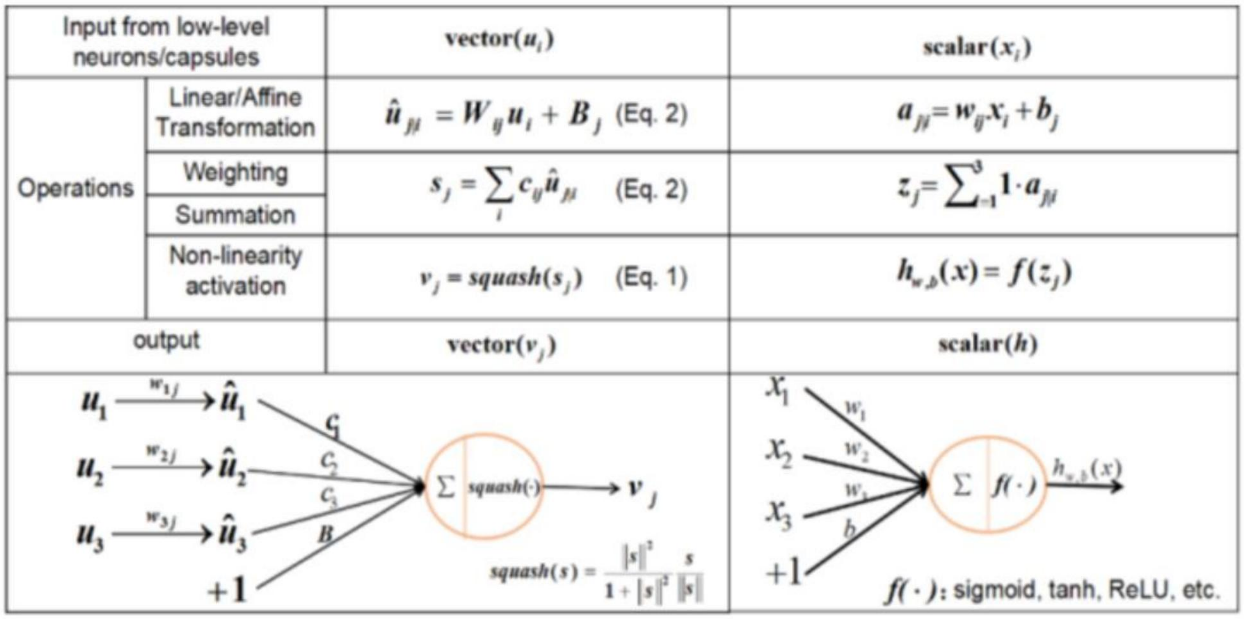


Figure 2: Capsule and Hierarchical Neural Network

The length of the output vector of a capsule to represent the probability that the entity represented by the capsule is present in the current input. We therefore use a non-linear "squashing" function to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1.

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}$$

where vj is the vector output of capsule j and sj is its total input.



Input from low-level neurons/capsules		vector(u_i)	scalar(x_i)
Operations	Linear/Affine Transformation	$\hat{u}_{j\mid i} = W_{ij}u_i + B_j$ (Eq. 2)	$a_{ji} = w_{ij}x_i + b_j$
	Weighting / Summation	$s_j = \sum_i c_{ij}\hat{u}_{j\mid i}$ (Eq. 2)	$z_j = \sum_{i=1}^{3} 1 \cdot a_{ji}$
	Non-linearity activation	$v_j = squash(s_j)$ (Eq. 1)	$h_{w,b}(x) = f(z_j)$
output		vector(v_j)	scalar(h)

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

u_1, w_{1j}, $\hat{u}_1$
u_2, w_{2j}, $\hat{u}_2$
u_3, w_{3j}, $\hat{u}_3$
+1
c_1, c_2, c_3, B
Σ squash(·)
v_j
x_1, x_2, x_3, +1
w_1, w_2, w_3, b
Σ f(·)
$h_{w,b}(x)$
f(·): sigmoid, tanh, ReLU, etc.

Figure 3: Capsule vs. Traditional Neuron

The architecture has only two convolutional layers and one fully connected layer. Conv1 has 256, 9 × 9 convolution kernels with a stride of 1 and ReLU activation. This layer converts pixel intensities to the activities of local feature detectors that are then used as inputs to the *primary* capsules. The primary capsules are the lowest level of multi-dimensional entities and, from an inverse graphics perspective, activating the primary capsules corresponds to inverting the rendering process. This is a very different type of computation than piecing instantiated parts together to make familiar wholes, which is what capsules are designed to be good at.

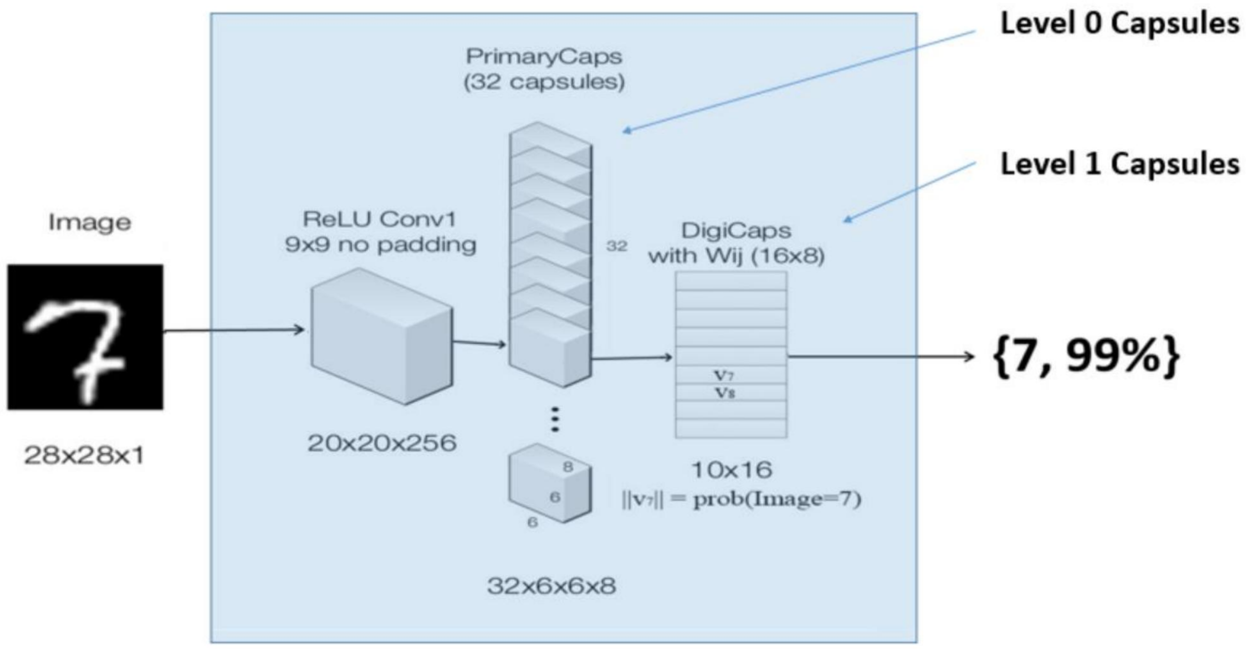


Figure 4: Hierarchical Neural Network Architecture based on Capsules

The second layer (PrimaryCaps) is a convolutional capsule layer with 32 channels of convolutional 8D capsules (*i.e.* each primary capsule contains 8 convolutional units with a 9×9 kernel and a stride of 2). Each primary capsule output sees the outputs of all 256×81 Conv1 units whose receptive fields overlap with the location of the center of the capsule. In total PrimaryCaps has $[32 \times 6 \times 6]$ capsule outputs (each output is an 8D vector) and each capsule in the $[6 \times 6]$ grid is sharing their weights with each other. One can see PrimaryCaps as a Convolution layer with Eq. 1 as its block non-linearity. The final Layer (DigiCaps) has one 16D capsule per digit class and each of these capsules receives input from all the capsules in the layer below.

2 Related Work

This project is very much inspired by the paper by Hinton et. el, [1] where they addressed the same problem of CNN with a proposed Capsule architecture. In this project, we have referenced the original paper and implemented the architecture with a purpose to validate the claims in the original paper.

3 Dataset and Features

Training is strictly performed on MNIST standard train dataset, and no augmented dataset has been trained. Apart from the standard MNIST dataset, we generated additional 500+ test images to test the translational invariance quality of both CNN and Capsule based HNN.

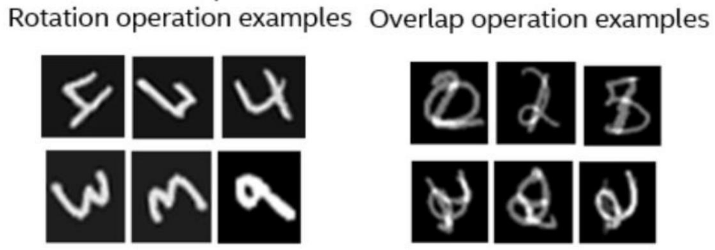
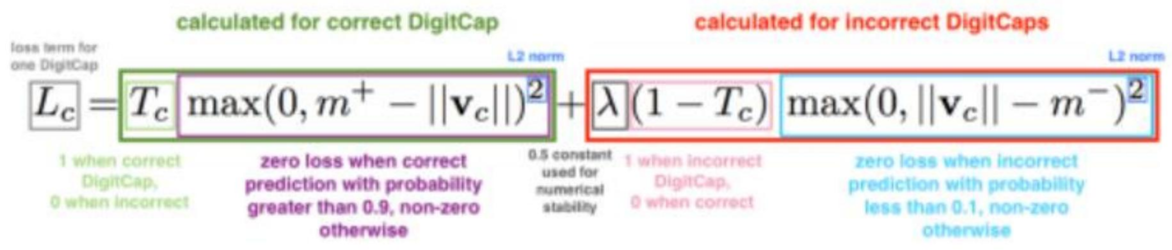


Figure 5: MNIST dataset with Rotation and Overlap operations

4 Methods

Loss Function:


calculated for correct DigitCap
calculated for incorrect DigitCaps
loss term for one DigitCap
L2 norm
L2 norm
$L_c = T_c \max(0, m^+ - ||\mathbf{v}_c||)^2 + \lambda (1 - T_c) \max(0, ||\mathbf{v}_c|| - m^-)^2$
1 when correct DigitCap, 0 when incorrect
zero loss when correct prediction with probability greater than 0.9, non-zero otherwise
0.5 constant used for numerical stability
1 when incorrect DigitCap, 0 when correct
zero loss when incorrect prediction with probability less than 0.1, non-zero otherwise

5 Experiments/ Results/ Discussion

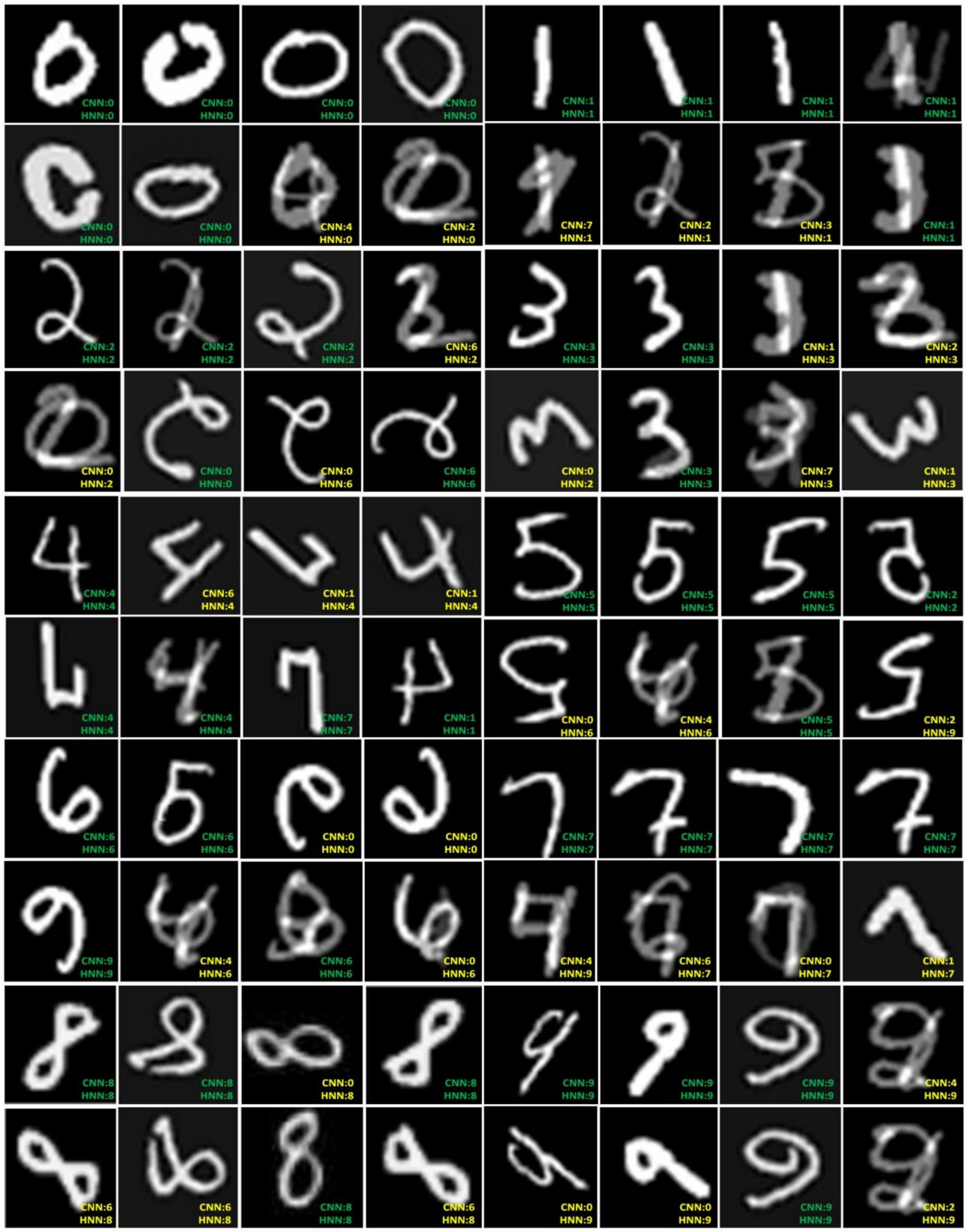


Figure 6: CNN vs. Capsule HNN Results on Handwritten Digits

Above results are from outputs of both a CNN and Capsule based HNN (both trained on standard MNIST train set), with augmented test data set. Each of the output shows the predictions of both network and color coding GREEN means both networks predicted same output, and YELLOW means both networks predicted different values where on most cases HNN predicted correctly.

6 Conclusion/ Future Work

- Capsule based HNN performance matches that of traditional CNN on MNIST dataset
- Capsule based HNN outperforms traditional CNN on highly overlapping, rotated and twisted digits
- Training a capsule based network takes significantly larger time than with a CNN – this needs to be investigated
- Test the capsule network on complex image processing application such as face recognition
- Investigate if capsule concept can be applied to other forms of tasks such as voice recognition, text sentiment analysis, etc.

References

1. Sara Sabour, Nicholas Frosst, Geoffrey E Hinton. Dynamic Routing Between Capsules. arXiv:1710.09829, 2017.
2. Geoffrey E Hinton, Zoubin Ghahramani, and Yee Whye Teh. Learning to parse images. In Advances in neural information processing systems, pages 463–469, 2000.
3. Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for object recognition with invariance to pose and lighting. In Computer Vision and Pattern Recognition, 2004. Proceedings of the 2004 IEEE Computer Society Conference, volume 2, pages II–104. IEEE, 2004.

Appendix

- Source Code: https://github.com/sureshsugumar/HierNet
- Demo Video: https://youtu.be/TL81fI0X6vg