Anubhav Jaiswal, Prayash Das, Shahaji Deshmukh

Professor Hong Man

CPE 646 - Pattern Recognition

17 December 2023

CapsuleNets on MNIST Data

ABSTRACT

Capsule Networks, a novel architecture proposed by Geoffrey Hinton and colleagues, revolutionize the field of neural networks, particularly in the domain of computer vision. Departing from conventional convolutional neural networks (CNNs), Capsule Networks introduce capsules, groups of neurons that collaboratively represent entities and encode information about their spatial relationships. The dynamic routing mechanism employed by Capsule Networks in Dynamic Routing Between Capsules [Hinton et al] allows adaptive adjustments of connections between capsules, enhancing the model's ability to capture hierarchical features and handle variations in pose and viewpoint. In this study, we explore the application of Capsule Networks to the MNIST dataset, a well-established benchmark for handwritten digit recognition. Our investigation involves the design and implementation of a Capsule Network architecture, training on the MNIST dataset, and evaluating its performance on unseen data using Python and Pytorch. The results provide insights into the effectiveness of Capsule Networks for image classification tasks, shedding light on their potential advantages and areas for further refinement.

INTRODUCTION

CONVOLUTION NEURAL NETWORKS: DRAWBACKS

Convolutional Neural Networks (CNNs) have demonstrated effectiveness in addressing computer vision tasks, leveraging their capacity to autonomously extract features from unstructured data. While training, The input data, typically an image, is processed through convolutional layers. Convolution involves sliding small filters (kernels) over the input data, performing element-wise multiplications, and summing the results to produce feature maps. After convolution, pooling layers downsample the spatial dimensions of the feature maps. Max pooling, for example, retains the maximum value from a group of neighboring pixels, reducing the spatial resolution while retaining important features.

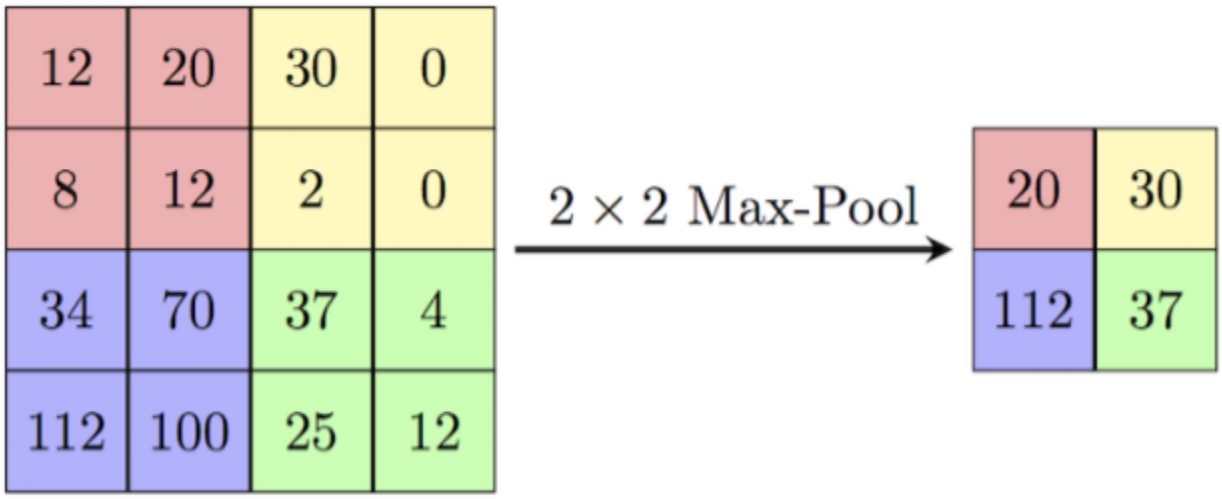


Fig. 1 Pooling operation in CNN

Pooling involves discarding information by selecting the maximum value (max pooling) or average value (average pooling) from a group of neighboring pixels. This can lead to a loss of

fine-grained details, especially in regions where subtle features are important. Pooling operations make the network less sensitive to the precise position and orientation of features. While this can be an advantage in achieving translational invariance, it may not be desirable in tasks where spatial relationships are crucial, such as object localization. Pooling operations also use a fixed grid to sample information, which might not be adaptive to the inherent structure of the data. In cases where features have varying sizes or orientations, a fixed pooling grid is not optimal.

CAPSULE NETWORKS

Capsule Nets are inspired by Inverse graphics, a theoretical framework in computer vision that seeks to understand and reverse the process of image formation. It aims to infer the underlying 3D scene or structure from a 2D image, essentially reversing the steps of computer graphics [Tran, Minh, et al]. While inverse graphics is a concept rooted in computational models of vision, it also has connections to how human vision might operate. The brain is organized into modules, which can be considered capsules. Capsule networks incorporate dynamic routing algorithms to estimate features of objects like pose (position, size, orientation, deformation, velocity, albedo, hue, texture, and so on).

Each layer in a capsule net is divided into many small groups of neurons called “capsules”. Using an iterative routing process, each active capsule will choose a capsule in the layer above to be its output node. For the higher levels of a visual system, this iterative process will be solving the problem of assigning parts to wholes. The activities of the neurons within an active capsule represent the various properties of a particular entity that is present in the image. These properties can include many different types of instantiation parameters such as pose

(position, size, orientation), deformation, velocity, albedo, hue, texture, etc. One very special property is the existence of the instantiated entity in the image.[Hinton et al.]

The fact that the output of a capsule is a vector makes it possible to use a powerful dynamic routing mechanism to ensure that the output of the capsule gets sent to an appropriate parent in the layer above. Initially, the output is routed to all possible parents but is scaled down by coupling coefficients that sum to 1, using a squashing function. For each possible parent, the capsule computes a "prediction vector" by multiplying its own output by a weight matrix. If this prediction vector has a large scalar product with the output of a possible parent, there is top-down feedback which increases the coupling coefficient for that parent and decreases it for other parents. This increases the contribution that the capsule makes to that parent thus further increasing the scalar product of the capsule's prediction with the parent's output.

There are four major steps that are performed in capsule nets

1. Affine Transformation - Input vectors that our capsule receives. Lengths of these vectors encode probabilities that lower-level capsules detected their corresponding objects and directions of the vectors encode some internal state of the detected objects. Let us assume that lower level capsules detect straight lines and curves. These vectors then are multiplied by corresponding weight matrices that encode important spatial and other relationships between lower level features (straight and curved lines) and higher level features (digits). After multiplication by these matrices, what we get is the predicted position of the higher level feature. This means the vector output of this multiplication gives us the position of the digit based on the location of straight and curved lines.

2. Weighting - The scalar weights for each vector are determined by how the input from lower capsules maps to capsules in the layer above. Every capsule in the layer above will get input vectors from the layer below. Weights will be assigned upon the best clustering of vectors from lower layers in the upper layer. This technique was introduced by Hinton et al and is called Dynamic Routing by Agreement.
3. Sum - This step is similar to the regular artificial neuron and represents a combination of inputs.
4. Non-Linear Activation – Squashing - Another innovation that Capsnet HintonEt al introduced is the novel nonlinear activation function that takes a vector, and then “squashes” it to have length of no more than 1, but does not change its direction.

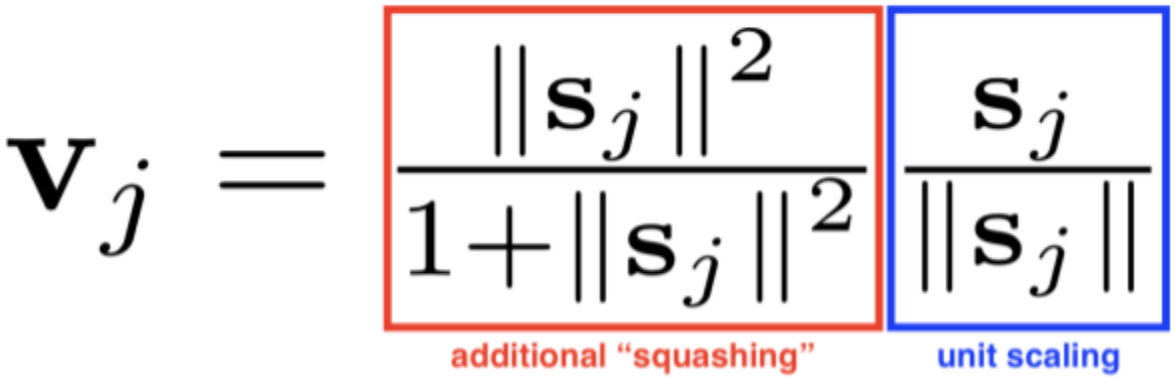


Fig. 2 Squashing

IMPLEMENTATION

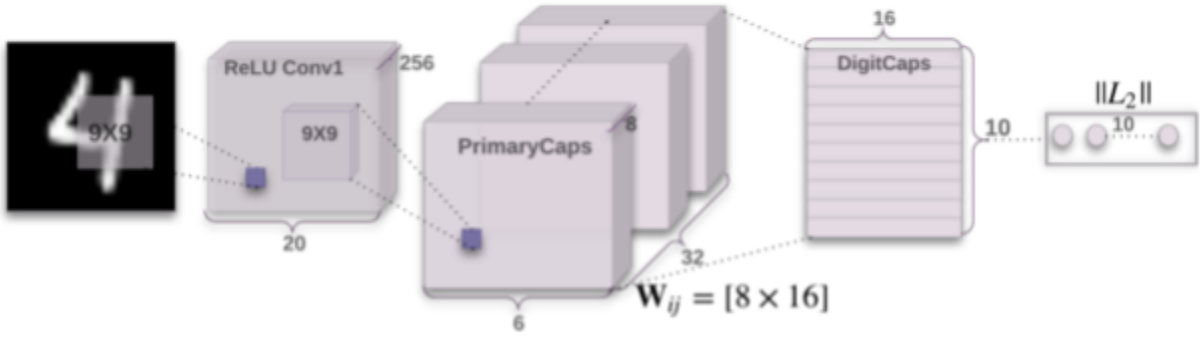


Fig. 3 Capsule Net architecture [Hinton et al.]

Capsule Net implementation can be broken down into two major parts, Encoding and Decoding.

ENCODING

Encoding steps consist of 3 layers, A Convolutional layer, a Primary Capsule Layer and Digit Capsule Layer.

1. Convolutional Layer
    a. Input - (1, 28, 28), Output - (20, 20, 256)
    b. Convolutional layer's job is to detect basic features in the 2D image. In the CapsNet, the convolutional layer has 256 kernels with size of 9x9x1 and stride 1, followed by ReLU activation.
    c. This layer has a separate class in our code, `ConvEncoder`.
2. Primary Capsule

a. Input - (20, 20, 256), Output - (6, 6, 8, 32)

b. This layer has 32 primary capsules whose job is to take basic features detected by the convolutional layer and produce combinations of the features.

c. This layer is represented as class, `PrimaryCaps` in our code.

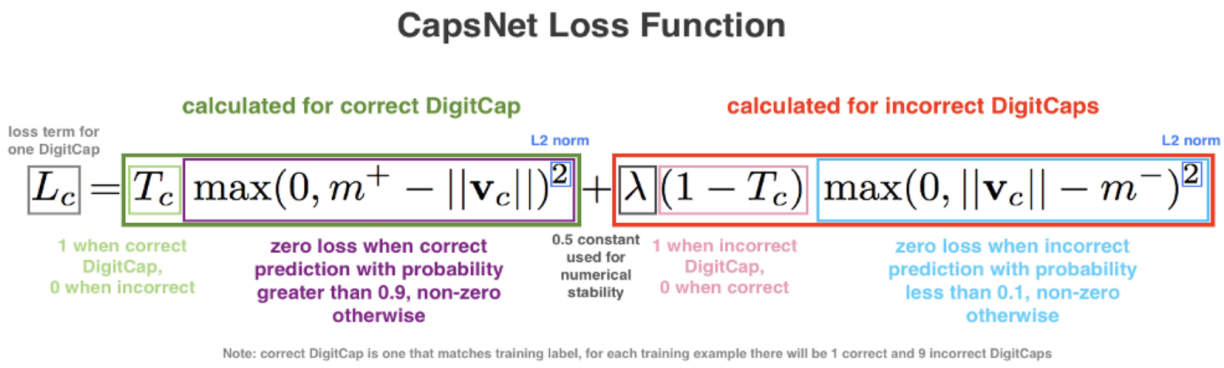
3. Digit Capsule

a. Input - (6, 6, 8, 32), Output - (16, 10)

b. The digit capsule layer contains 10 capsules (one for each digit) of 16 dimensions each. This layer also computes the Weight matrix needed for dynamic routing by agreement. Then it calculates the sum delineated in the sum step above.

c. Digit Capsule also calculates the margin loss. Margin loss to make it possible to detect two or more different digits in each image.

d. This class is represented as `ObjectCaps` in our code.



DECODING

The Decoder takes a 16-dimensional vector from the correct DigitCap and learns to decode it into an image of a digit. Decoder is used as a regularizer, it takes the output of the

correct DigitCap as input and learns to recreate an 28 by 28 pixels image, with the loss function being Euclidean distance between the reconstructed image and the input image. Decoder forces capsules to learn features that are useful for reconstructing the original image, much like the human brain. The decoder is represented as `Decoder` in our code and it consists of 3 fully connected sequential layers with a ReLU activation function.

1. Layer 4 - Input - (16, 10), Output - (512)
2. Layer 5 - Input - (512) , Output - (1024)
3. Layer 6 Input - Input (1024), Output - (784)

TRAINING

Training capsule networks is a straightforward task. Hinton et al utilizes Pytorch's Adam optimizer with default parameters to train the network. The network does not need any hyperparameter tuning. We just keep printing the loss for every epoch and iteration.

EVALUATION

We reach an accuracy of 99.7% on the MNIST data. Capsule Networks (CapsuleNets) are particularly adept at capturing perturbations in images, which refers to variations or changes in the position, orientation, scale, and other properties of objects within an image. This is particularly evident in their handling of variations in digit representations, such as differences in rotation, position, and skew.

Fig. 4 Perturbations captured by Capsule Nets

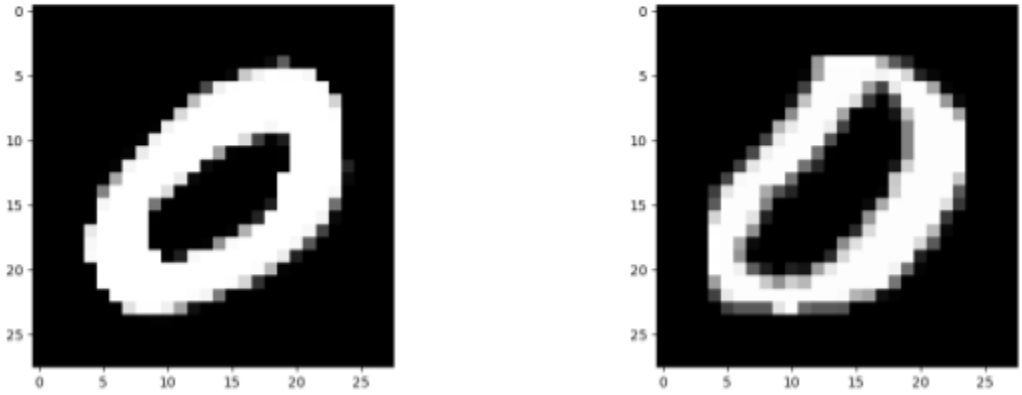


Fig. 5 Original Image against Capsule Net reconstructed image

CONCLUSION

In conclusion, this report demonstrates the robust capabilities of Capsule Networks in processing MNIST data, marking a significant advancement in image classification tasks. The study highlights the unique architecture of CapsuleNets, which offers enhanced feature detection compared to traditional Convolutional Neural Networks (CNNs). The experiments conducted reveal that CapsuleNets achieve high accuracy rates in digit recognition on the MNIST dataset, underlining their effectiveness in capturing spatial hierarchies and dynamic routing. CapsuleNets are better at recognizing spatial hierarchies between features, which is crucial for understanding

complex images with overlapping objects. They also reduce the loss of spatial information, a common issue in CNNs due to pooling layers. Furthermore, CapsuleNets can recognize objects in various orientations and poses without needing extensive data augmentation. This enhanced feature detection capability makes them potentially more effective in tasks requiring a detailed understanding of object relationships and dynamics. These findings suggest that CapsuleNets hold great promise for future applications in computer vision, potentially outperforming existing models in complex image processing tasks.

CapsuleNets tend to require more computational resources and longer training times due to their complex dynamic routing process. This can make them less efficient for large-scale or real-time applications. Additionally, they may not perform as well on certain types of datasets, like those with less structured or more diverse image types, where the advantages of spatial hierarchy recognition are less pronounced. These limitations highlight the need for further research and optimization in CapsuleNet technology. Despite these challenges, CapsuleNets remain a promising development in the realm of computer vision, offering a novel approach to image classification tasks.

Contributions

1. Report - Shahaji Deshmukh, Prayash Das, Anubhav Jaiswal
2. Code - Anubhav Jaiswal, Shahaji Deshmukh, Prayash Das
3. Research findings - Prayash Das, Anubhav Jaiswal, Shahaji Deshmukh

Works Cited

Tran, Minh, et al. "CapsNet for Medical Image Segmentation." arXiv preprint arXiv:2203.08948v1 [eess.IV], 16 Mar 2022.

Sabour, Sara, Nicholas Frosst, Geoffrey E. Hinton. "Dynamic Routing Between Capsules." arXiv preprint arXiv:1710.09829v2 [cs.CV], 7 Nov 2017.

Geron, Aurélien. "Extra Capsule Networks Jupyter Notebook." GitHub, https://github.com/ageron/handson-ml/blob/master/extra_capsnets.ipynb.