CODE DEMO
Capsule Networks

Agenda

- Theory
- Model Architecture
- Layers
- Training
- Testing
- Reconstruction Images

Theory Explained

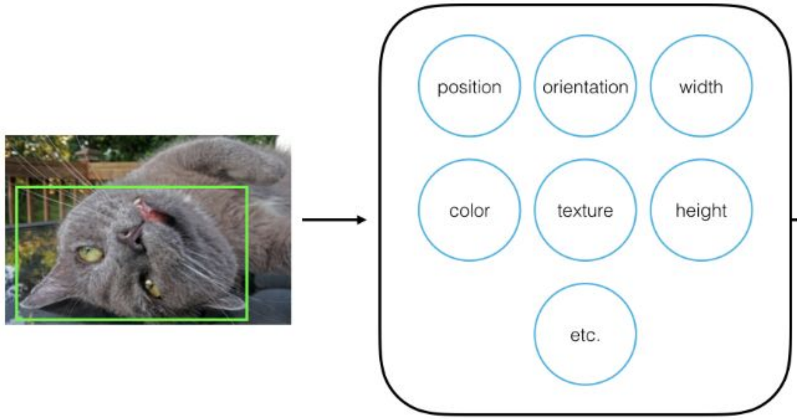
What are Capsules?

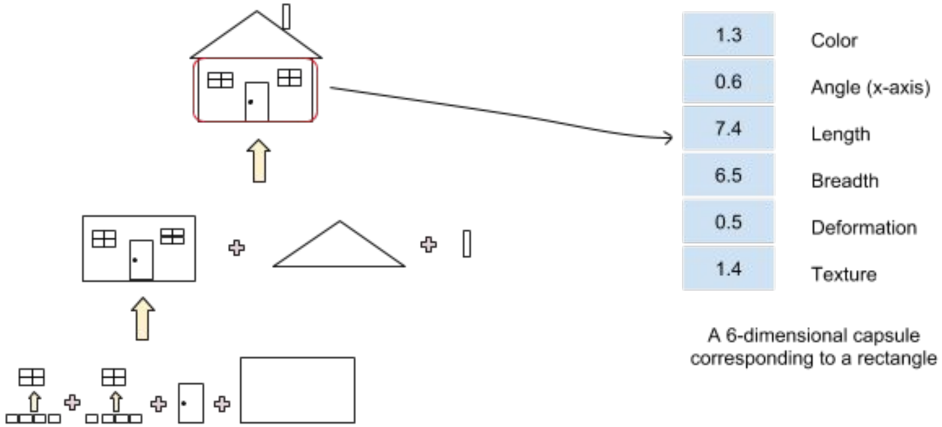
Capsules are a small group of neurons that have a few key traits:

Each neuron in a capsule represents various properties of a particular image part; properties like a parts color, width, etc.

Examples:

- Cat: what is the position, orientation texture etc.
- House: What is the angle, what shapes are present etc.





A 6-dimensional capsule corresponding to a rectangle

Representing Relationships Between Parts

Dynamic Communication:

- Capsules communicate to determine data flow.
- Enables learning of spatial relationships between parts.

Learning Spatial Relationships:

- Capsule networks learn how parts relate to wholes.
- Facilitates object recognition regardless of orientation.

Advantages Over Vanilla CNN:

- Improved object identification in various orientations.
- Better at recognizing multiple, overlapping objects.
- Enhanced learning from smaller training datasets.

Dataset

MNIST

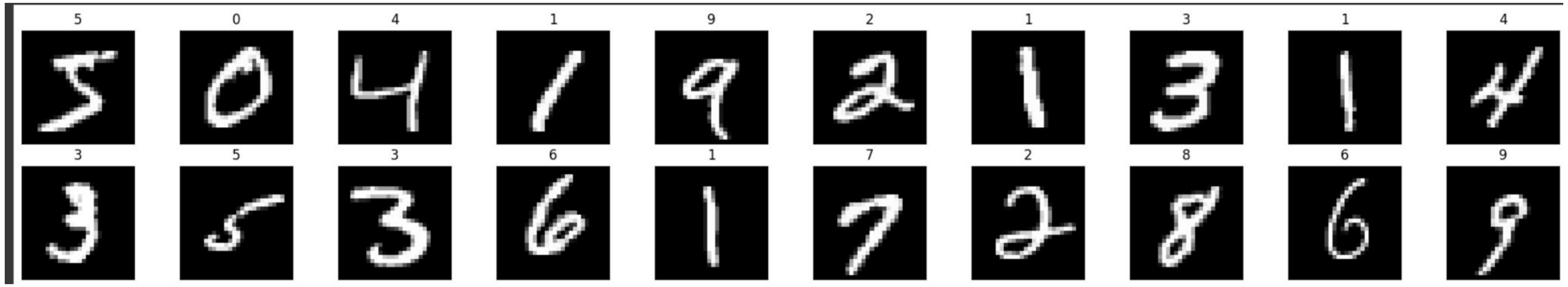
Dataset

For our demo today, we're using the MNIST dataset.

MNIST consists of 70,000 handwritten digits, each a 28x28 pixel grayscale image.

How to Download and Preprocess the Dataset ?

To get started with MNIST, you can easily download it using TensorFlow or PyTorch with built-in functions.



Model Architecture

Conv Layers

Primary Caps

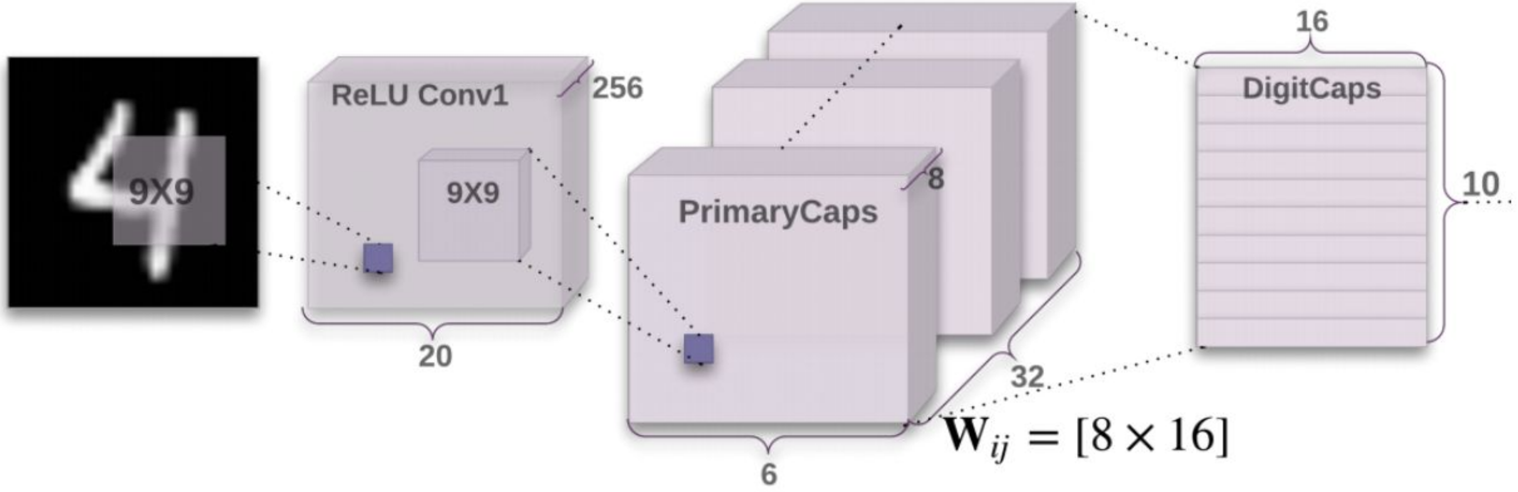
Digit Caps

Decoder

Encoder Layers

Encoder

The encoder is made of a series of layers that are responsible for taking in as input a 28 by 28 MNIST image and learning to encode it into a 16-dimensional output vector.



First Layer: Convolutional Layer

The first layer in our encoder is a convolutional layer that will learn to extract features, like edges, in a given input image.This convolutional layer will create a stack of 256 filtered images, given one input MNIST image.

Second Layer: Primary Capsules

This layer has 8 primary capsules. Essentially, each capsule is responsible for producing weighted combinations of the features detected in the previous convolutional layer.

Squashing

We define a nonlinear function squash that calculates a certain capsule's normalized, vector output using the following equation.

$$v_j = \frac{|| s_j^2 || s_j}{1 + || s_j^2 || s_j}$$

v_j is the value we want to calculate, the normalized vector output of a capsule j. And s_j is that capsule's total input; a weighted sum over all the output vectors from the capsules in the layer below capsule j .

Third Layer: Digit Capsules

This layer is composed of 10 "digit" capsules, one for each of our digit classes 0-9. Each capsule takes, as input, a batch of 1152-dimensional vectors produced by our 8 primary capsules, above.

Dynamic Routing

- Facilitates finding the optimal connections between capsules in different layers.
- Enables capsules to communicate and determine the flow of data.
- Ensures that relevant information is routed to appropriate parent capsules.

Coupling Coefficients

- Represent the probability that a child capsule's output should be routed to a parent capsule.
- Initially uncertain, coupling coefficients are determined through dynamic routing.
- Sum of coupling coefficients across all possible parent capsules equals 1.

Third Layer: Digit Capsules

Routing by Agreement

- Iterative process updating coupling coefficients between capsules.
- Determines the flow of information between capsule layers.
- Child capsule outputs a vector indicating part existence and pose.
- Computes prediction vector for each possible parent capsule.
- Agreement between prediction and parent vectors increases coupling coefficients.
- High coupling coefficient enhances the child's influence on the parent.
- Capsules iteratively refine their connections based on agreement.

$$c_{ij} = \frac{e^{b_{ij}}}{\sum_k e^{b_{ik}}}$$

$$\hat{u} = Wu$$

$$a = v \cdot u$$

$$b_{ij} = b_{ij} + a$$

Third Layer: Digit Capsules

Digit Capsules

- DigitCaps layer consists of 10 capsules, each representing a digit class (0-9).
- Each capsule takes 1152-dimensional vectors from the primary capsules as input.
- Output of each capsule is a 16-dimensional vector.

Decoder

Decoder Explained

- The decoder takes 16-dimensional vectors from the DigitCaps layer as input.
- Identifies the "correct" capsule output vector, which corresponds to the digit capsule with the largest vector magnitude.
- Measures the difference between the input image and the decoder reconstruction using Euclidean distance.

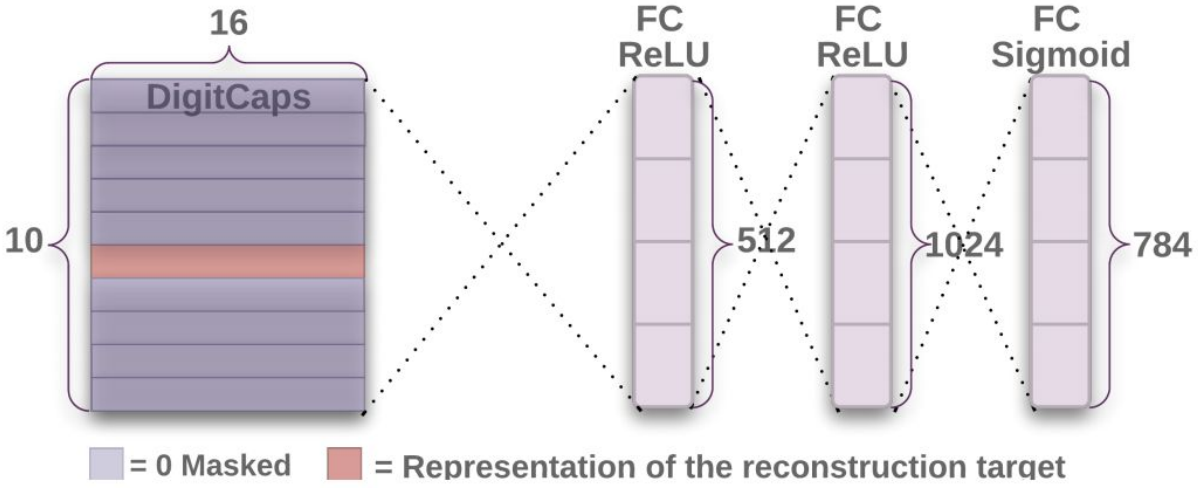


Image from the original Capsule Network paper

Create the Complete Model

- Includes convolutional layers, primary and digit capsule layers, and a decoder.

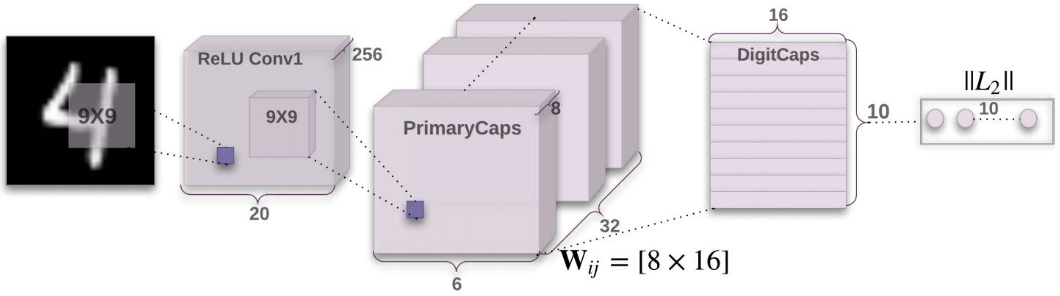


Image from the original Capsule Network paper

```
CapsuleNetwork(
  (conv_layer): ConvLayer(
    (conv): Conv2d(1, 256, kernel_size=(9, 9), stride=(1, 1))
  )
  (primary_capsules): PrimaryCaps(
    (capsules): ModuleList(
      (0-7): 8 x Conv2d(256, 32, kernel_size=(9, 9), stride=(2, 2))
    )
  )
  (digit_capsules): DigitCaps()
  (decoder): Decoder(
    (linear_layers): Sequential(
      (0): Linear(in_features=160, out_features=512, bias=True)
      (1): ReLU(inplace=True)
      (2): Linear(in_features=512, out_features=1024, bias=True)
      (3): ReLU(inplace=True)
      (4): Linear(in_features=1024, out_features=784, bias=True)
      (5): Sigmoid()
    )
  )
)
```

Custom Loss

Margin Loss & Reconstruction Loss

- Custom loss function for training the capsule network.
- Incorporates two key components: margin loss and reconstruction loss.
- Margin loss penalizes deviations from ideal capsule vector magnitudes based on class labels.
- Reconstruction loss measures the dissimilarity between original input images and reconstructed images.
- Combined loss function guides the training process to optimize both classification performance and reconstruction quality.

$$L_c = T_c(\max[0, 0.9 - v_c]) + \lambda(1 - T_c)(\max[0, v_c - 0.1])$$

Loss Function and Optimizer

- Utilize a custom loss function, defined as the **CapsuleLoss**, to optimize the capsule network.

- The CapsuleLoss incorporates both margin loss and reconstruction loss to train the network effectively.

- Implement the Adam optimizer with default parameters for efficient parameter updates during training.

Train the Network

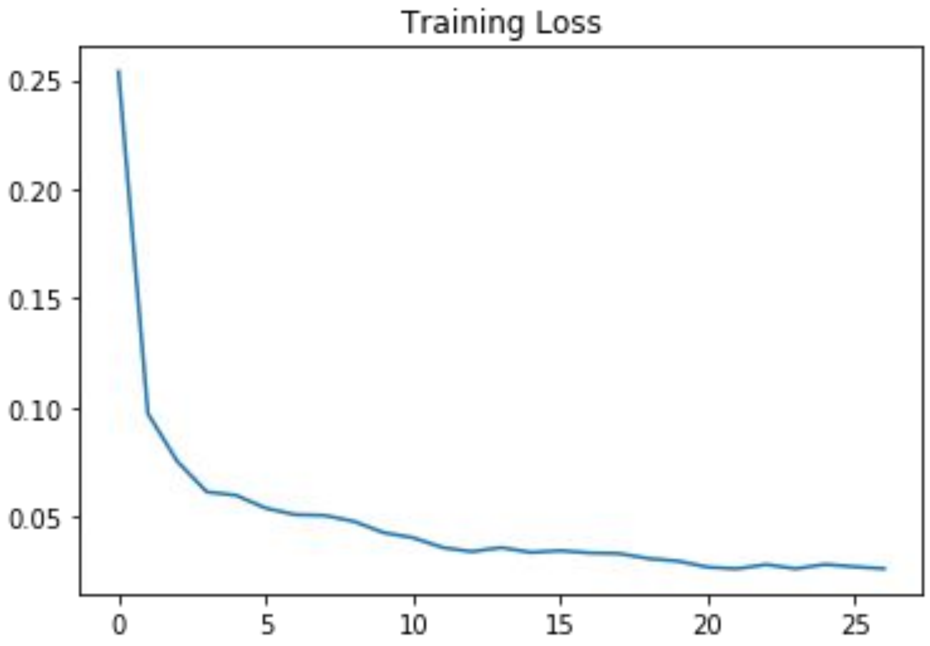
Steps

- Clear gradients, forward pass, calculate loss, backward pass, and optimization steps are performed in each iteration.

- Training loss is recorded and printed intermittently to monitor training progress.

- We are providing insights into how the model learns over time and improves its performance.

- Plot the recorded training loss to visualize how it decreases over the epochs, indicating model improvement.



Test the Trained Network

Testing

- Using the test data to evaluate the performance of the trained model.
- Evaluate classification accuracy and loss metrics on unseen test data.
- Granular analysis of performance on each class.
- Provides insights into how well the model generalizes to new data.

```
Test Loss: 0.03409957

Test Accuracy of Class 0: 99.69% (977/980)
Test Accuracy of Class 1: 99.65% (1131/1135)
Test Accuracy of Class 2: 98.64% (1018/1032)
Test Accuracy of Class 3: 97.92% (989/1010)
Test Accuracy of Class 4: 98.27% (965/982)
Test Accuracy of Class 5: 98.99% (883/892)
Test Accuracy of Class 6: 98.85% (947/958)
Test Accuracy of Class 7: 99.51% (1023/1028)
Test Accuracy of Class 8: 99.08% (965/974)
Test Accuracy of Class 9: 98.32% (992/1009)

Overall Test Accuracy: 98.90% (9890/10000)
```

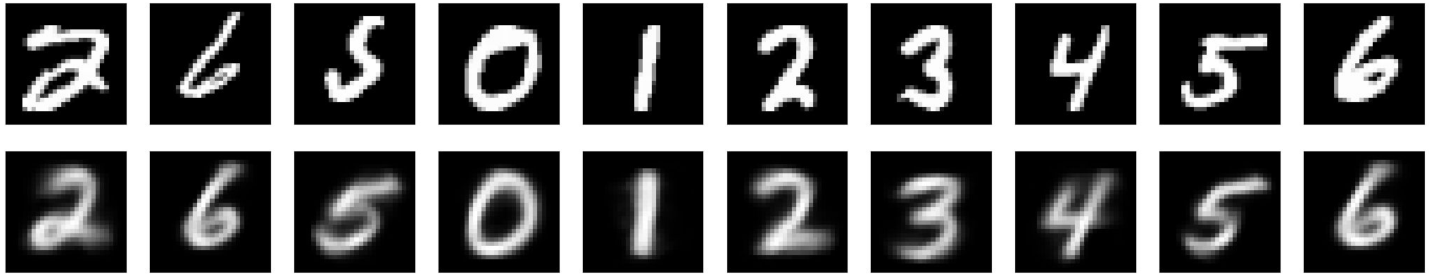
Display Reconstructions

Image Reconstructions

- Display the original MNIST images and their reconstructions to evaluate decoder performance.

- We use a function to plot one row of original images and another row of their corresponding reconstructions.

- Reconstructions appear nearly identical to the original images, with slight blurring/smoothing at the edges.



Task

Implement Capsule Networks on a New Dataset: You will apply the capsule network architecture to alternative datasets such as Fashion-MNIST or CIFAR-10. This exercise will enhance understanding of the model's adaptability across various image data types and allow exploration of its efficacy in more intricate settings.

Thankyou