Object Classification using Capsule Networks

Dhanaprakaash G Jayabrata Chowdhury Jayateja Kalla Joji Joseph

Abstract

Capsule networks are an improvement on CNNs. A capsule is a group of neurons whose outputs represent different properties of the same entity. The activity vector of the capsule represents the instantiation parameters of a specific type of entity such as an object or object part. Capsnet replaces scalar-output feature detectors with vector-output capsules and max-pooling with routing-by-agreement mechanism: A lower-level capsule will send its output to higher level capsules whose activity vectors have a big similarity with prediction coming from lower level capsule. We have implemented capsule network based handwritten digit recognizer and an image recognizer. We also created a web interface to see our networks in action.[1]

1. Problem Description

Convolutional Neural Networks are now widely used in image recognition. Steps for CNN are as follows:

1. Given an input image, a set of filters scan it and perform convolution operation.
2. This creates a feature map inside the network. These features will next pass via activation (ex. ReLU) and pooling layers. Activation gives nonlinearity. Pooling helps in reducing the training time. Pooling makes summaries of each sub-region.
3. At the end, it will pass via a sigmoid classifier.

Limitation of CNN: CNN does not take account of the orientation of an object in an image, because of its translation invariant property. In some cases, this creates difficulty in recognition (ex: classifying images like faces, where the position of eyes, nose, lips can be different so that we can say it is not a real face but CNN will classify it as a face). To solve these types of problems, we need to identify the spatial relation between features but CNN cannot capture this information. Because of this, Capsnet is introduced with translation equivariance property, to hold spatial information between attributes.

[1] Available at https://capsule-networks.herokuapp.com/

2. Literature Review

Capsule network proved to be promising in many vision related applications in terms of improved performance (Patrick et al., 2019). The Architecture of the capsule network has a property of rotational invariance and spatial awareness. These properties found to be useful in the field of astronomy (Katebi et al., 2019). The capsule network is deployed to understand the morphological types of galaxies. The deployment of the capsule network in Galaxy morphological prediction is proved to be a better alternative for CNN.

The capsule network, being able to capture the details such as pose, orientation can be used for safety critical vision system. The capsule network (Kumar, 2018) is deployed for the classification of traffic signals on the German traffic sign data set. The capsule Network is used to increase the performance of the Sensors in Autonomous vehicles (Pöpperl et al., 2019). The ultrasonic sensor is used for detecting the height of the objects. Such sensors require high hardware required for high performance. The application of capsule networks to such sensor systems increases its performance without requiring a high hardware complexity.

In medical imaging, It is used extensively in applications like lung cancer detection and brain tumour detection. The state-of-the-art in detecting lung cancer (Mobiny & Van Nguyen, 2018) uses the CNN to detect the lung nodule inside the lungs. CNNs are involved in the classification task which detects the brain tumour (Afshar et al., 2018). The deployment of the capsule network in these applications show the increased performance to CNN. Moreover, capsule networks require less number of training data sets to train.

3. Capsule Network

In a capsule network, each capsule is made up of a group of neurons with each neuron's output representing various properties of the particular object part. The output of a capsule is in the vector form (activity vector), where its magnitude indicates the existence of a particular feature and its direction is a function of the feature characteristics (width, texture, height, etc.). This provides the advantage

of recognizing the whole entity by first recognizing its parts and its spatial properties. Capsule network replaces scalar-output feature detectors with vector-output capsules and max-pooling with the routing-by-agreement mechanism.

The following are the important takeaways from the paper (Sabour et al., 2017) which introduced routing-by-agreement mechanism.

3.1. Squashing

We use squashing function to make the output vector represent the probability of entity represented by that vector.

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

where $\mathbf{v}_j$ is the vector output of capsule j and $\mathbf{s}_j$ is its total input.

3.2. Dynamic Routing

The first layer of capsules is called the primary capsule layer and the subsequent capsule layers are called secondary capsule layers. For all but the first layer of capsules, the total input to a capsule $\mathbf{s}_j$ is a weighted sum over all "prediction vectors" $\hat{\mathbf{u}}_{j|i}$ from the capsules in the layer below and is produced by multiplying the output $\mathbf{u}_i$ of a capsule in the layer below by a weight matrix $\mathbf{W}_{ij}$.

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \,, \qquad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i \tag{2}$$

where the c_{ij} are coupling coefficients that are determined by the "routing by agreement" mechanism as given in algorithm 1.

Algorithm 1 Dynamic Routing (Sabour et al., 2017)

1: **Input:** $\hat{\boldsymbol{u}}_{j|i}, r, l$
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ {`squash` computes Eq. 1}
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} . \mathbf{v}_j$
8: **end for**
9: **return** $\mathbf{v}_j$

3.3. Margin Loss for Existence of an Object

The margin loss of an object class is given by,

$$\begin{aligned} L_k = {} & T_k \ \max(0, m^+ - ||\mathbf{v}_k||)^2 \\ & + \lambda \ (1 - T_k) \ \max(0, ||\mathbf{v}_k|| - m^-)^2 \end{aligned} \tag{3}$$

where $T_k = 1$ iff an object of class k is present and $m^+ = 0.9$ and $m^- = 0.1$. The λ down-weighting of the loss for absent object classes.

We define total loss as the sum of margin losses. In case if we are using a reconstruction network as a regularizer we define total loss as the sum of reconstruction loss and margin losses.

4. Implementation

We implemented a handwritten digit recognizer and an image recognizer using capsule networks. We used PyTorch (Paszke et al., 2019) machine learning framework to implement our capsule networks.

4.1. Handwritten Digit Recognizer

It is a direct implementation of the model given in the paper (Sabour et al., 2017). We used MNIST dataset (LeCun et al., 1998) for training. Figure 1 shows the architecture of our capsule network (Sabour et al., 2017).

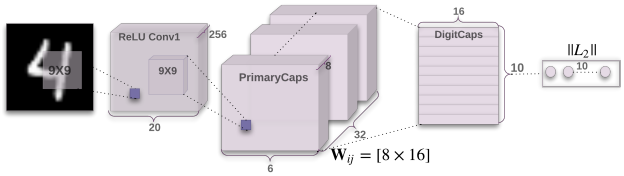

Figure 1. Capsule network architecture for handwritten digit recognition

The first layer is a convolution layer with 256, 9x9 kernels. The second layer is a primary capsule layer. The primary capsule layer has 32 channels of convolutional 8D capsules. The third layer is a secondary capsule layer having one 16D capsule per digit class. The output from the secondary capsule layer is passed to a decoder which is also a neural network. Figure 2 shows the architecture of the decoder (Sabour et al., 2017).

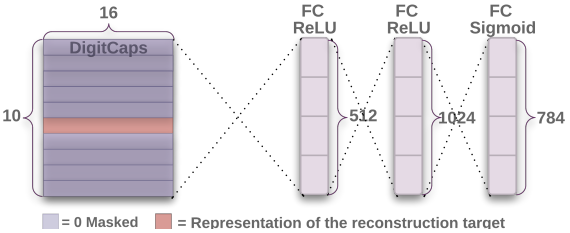

Figure 2. Decoder architecture along with secondary capsule layer

We reshape the 1D output vector from the decoder to produce a reconstructed 28x28 image.

4.2. Image Recognizer

This image recognizer is based on CIFAR-10 dataset (Krizhevsky et al.). We modified the architecture of our handwritten digit recognizer for recognizing images. In the first convolution layer, we took 3 input channels as training data are RGB images. The architecture is shown in figure 3. In the decoder section we used an additional linear layer with $3 \times 1024 = 3072$ dimension output. We then reshaped this output vector to produce reconstructed images.

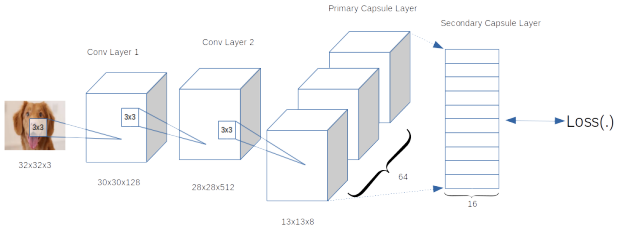

Figure 3. The architecture of image recognizer

5. Results

5.1. Handwritten Digit Recognizer

At the end of the training, we got a validation accuracy of 99.44%. Figure 4 shows few test images and corresponding reconstructed images. Figure 5 shows few screenshots of deployed model. The last screenshot is an example of a wrong prediction.

(a) Input images

(b) Reconstructed images

Figure 4. Reconstruction of images

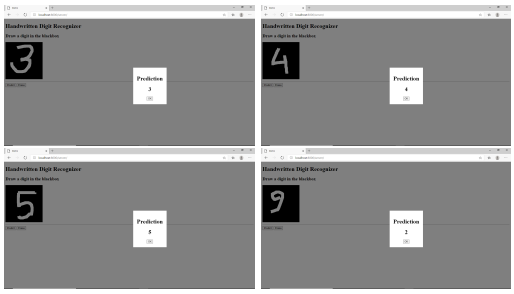

Figure 5. Predictions by the deployed model

5.2. Image Recognizer

At the end of the training we got a validation accuracy of 76.42 % and figure 6 shows the confusion matrix of all class of our model on cifar10 test dataset. The architecture we used is shown in figure 3.

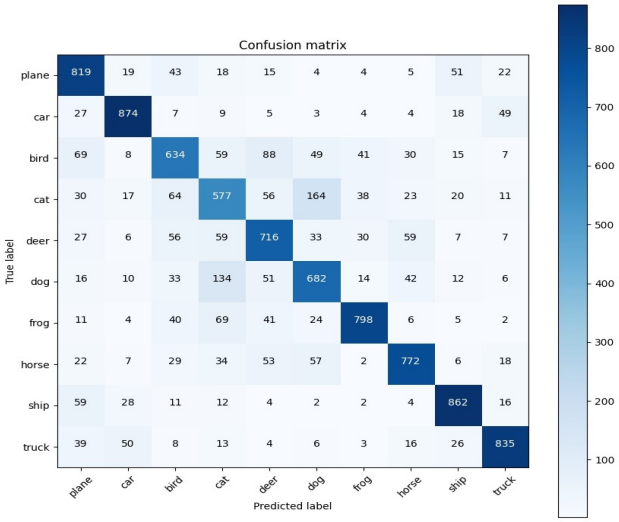

Figure 6. Confusion matrix of all classes on cifar10 test dataset.

Figure 7 shows few screenshots of deployed model. The last screenshot is an example of a wrong prediction.

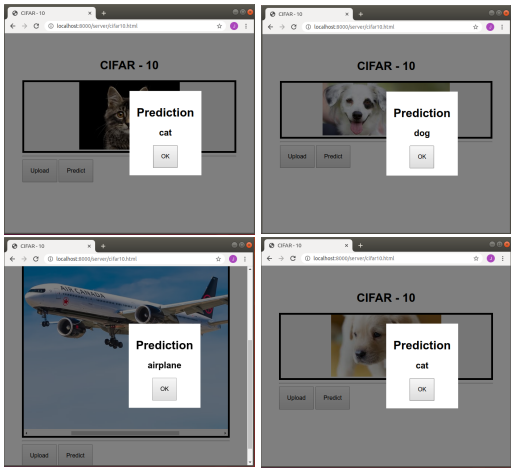

Figure 7. Predictions by the deployed model of image recognizer

Acknowledgement

We adapted the capsule network implementation of Dulat Yerzat. The source code is available at https://github.com/higgsfield/Capsule-Network-Tutorial/blob/master/Capsule%20Network.ipynb.

References

Afshar, P., Mohammadi, A., and Plataniotis, K. N. Brain tumor type classification via capsule networks, 2018.

Katebi, R., Zhou, Y., Chornock, R., and Bunescu, R. Galaxy morphology prediction using Capsule Networks. *Monthly Notices of the Royal Astronomical Society*, 486(2):1539–1547, 04 2019. ISSN 0035-8711. doi: 10.1093/mnras/stz915. URL https://doi.org/10.1093/mnras/stz915.

Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. URL https://www.cs.toronto.edu/~kriz/cifar.html.

Kumar, A. D. Novel deep learning model for traffic sign detection using capsule networks, 2018.

LeCun, Y., Cortes, C., and Burges, C. J. The mnist database of handwritten digits, 1998. URL http://yann.lecun.com/exdb/mnist/.

Mobiny, A. and Van Nguyen, H. Fast capsnet for lung cancer screening. *Lecture Notes in Computer Science*, pp. 741–749, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-00934-2_82. URL http://dx.doi.org/10.1007/978-3-030-00934-2_82.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8026–8037. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Patrick, M. K., Adekoya, A. F., Mighty, A. A., and Edward, B. Y. Capsule networks – a survey. *Journal of King Saud University - Computer and Information Sciences*, 2019. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2019.09.014. URL http://www.sciencedirect.com/science/article/pii/S1319157819309322.

Pöpperl, M., Gulagundi, R., Yogamani, S., and Milz, S. Capsule neural network based height classification using low-cost automotive ultrasonic sensors, 2019.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules, 2017.