# NEW ARCHITECTURE IN DEEP LEARNING: CAPSULE NETWORKS (CapsNet)

Sumeyra Busra SENGUL[1], Ilker Ali OZKAN[2]

## INTRODUCTION

Deep learning is a machine learning method that has gained a lot of popularity and is still developing, especially in its ability to accurately analyze images. It is a learning method that can be used for both supervised and unsupervised learning and can predict outputs based on the dataset used as input. The depth it has, namely the hidden layers, distinguishes it from machine learning. Using non-linear operations for feature extraction and transformation, it extracts different features from each layer of the image it gets as input [1].

Although there are various definitions of deep learning, the common feature of all of them is that it has multiple layers of nonlinear operations and can extract features between layers using supervised or unsupervised learning [2]. The most common deep learning architectures are Convolutional Neural Networks, Autoencoders, Generative Adversarial Networks, and Boltzman Machines.

Convolutional neural networks (CNN) are feedforward neural networks that use convolution in at least one of their layers to recognize two-dimensional images such as images and videos [1]. Many problems have been successfully solved using CNN algorithms [3]. Even though convolutional neural networks (CNN) have been successfully used in computer vision for many years, there are some constraints when extracting image features.

Translation invariance is one of these constraints [2]. The ability of an object to be recognized as an object even if its location changes is referred to as translation invariance.

1 Selcuk University, Technology Faculty, Department of Computer Engineering, Konya, Türkiye
2 Selcuk University, Technology Faculty, Department of Computer Engineering, Konya, Türkiye

Although CNN correctly identifies a normal human face in Figure 1, the image with the mouth and eyes misplaced is also identified as a face. CNN recognizes objects in the image with high accuracy. It cannot, however, define the part-whole relationship because it cannot store spatial information. It recognizes the parts but may not recognize the whole correctly because it cannot define the part-whole relationship. As has been shown in this example, CNN needs two eyes, a nose, and a mouth to classify an image as a face. The positions of the objects are insignificant.

Point-of-view invariance is another constraint. Point-of-view invariance refers to an object's ability to be recognized regardless of its direction. When faced with different angles of the same image, CNN may misunderstand and fail to classify it correctly. Because it does not evaluate the hierarchical relationship between objects and the whole, it may produce incorrect results. Because of the angle difference, the inverted human face in Figure 2 cannot be correctly classified.

The pooling process is one of CNN's most significant constraints. In CNN's size reduction is accomplished through pooling in order to reduce computational complexity. The spatial information of the features is lost during this process due to pixel loss, and the method's success is decreased. In order for the network to increase its performance, more training data is needed. This increases the number of parameters in the network and, as a result, increases the training time [4]. Figure 3 shows an example of the pooling process.

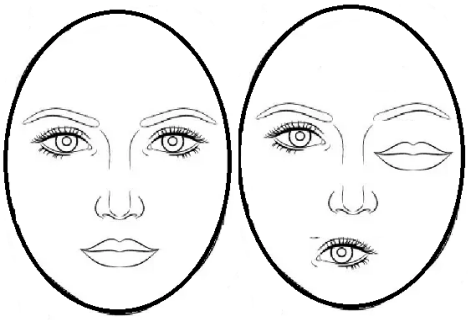

**Figure 1.** Translation Invariance Example

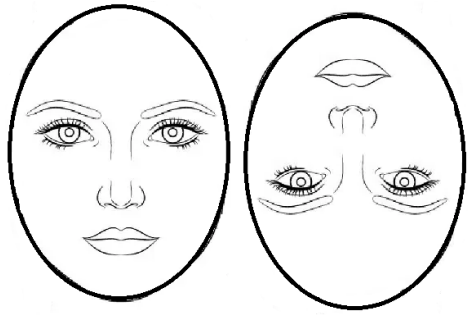

**Figure 2.** Point of View Invariance Example

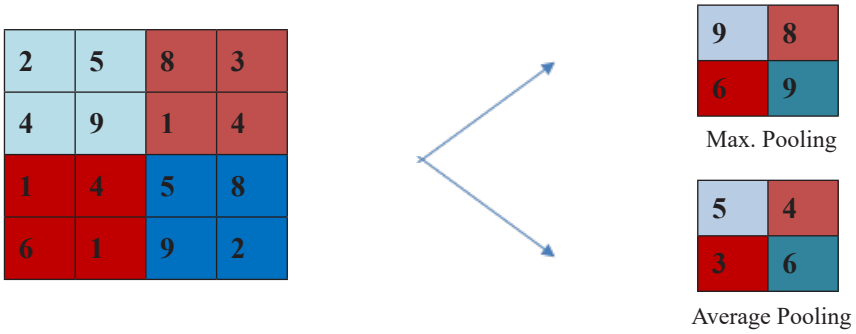


**Figure 3.** Pooling Example

Sabour et al. [5] proposed Capsule Network Architecture (CapsNet) and a dynamic routing algorithm to overcome these constraints of convolutional neural networks and correctly analyze data. CapsNet, unlike CNN, considers not only objects but also the relationships between them. It examines object relationships as part-whole relationships. It also stores these relationships via vectors. Instead of CNN's scalar output feature maps, CapsNet uses vector output capsules, and the dynamic routing algorithm is used in place of pooling. They used the squash function as the activation function. They tested the proposed architecture on the MNIST dataset and discovered that it performed better than CNN while using fewer parameters.

Table 1 shows the error rates of CapsNet in training different data sets.

**Table 1.** Error rates of CapsNet in different datasets[5]

| Dataset | Error Rate (%) |
|---|---|
| MNIST | 0.25 |
| MultiMNIST | 5.2 |
| CIFAR10 | 10.6 |
| smallNORB | 2.7 |

## WHAT IS A CAPSULE?

The capsule is the basic unit of CapsNet and is composed of many neurons. These neurons store not only the objects in the image, but also information about their physical properties, such as pose (position, direction, and size), deformation, color, and texture [6].

Figure 4 shows an example of a capsule containing a neuron array. The house capsule is consisting of neurons that represent window, door, and roof objects. The information in the capsule is saved in vector format. The length of the vectors shows the probability that the object exists in the image, while the direction indicates where its placement in the image. Figure 5 shows the vector's direction change in relation to the direction of the roof object.

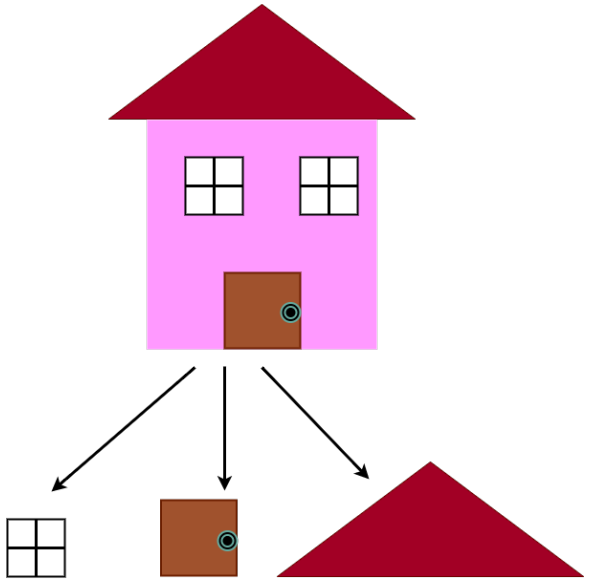

**Figure 4.** Capsule Example

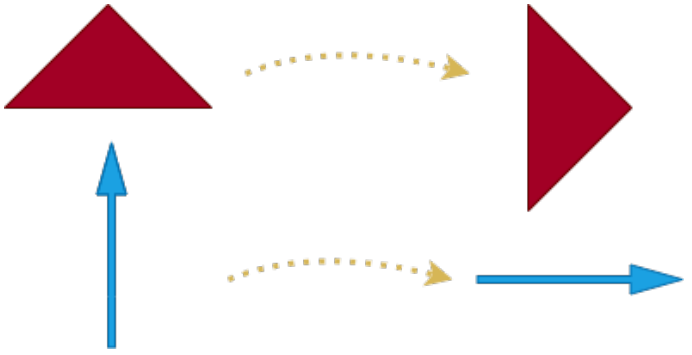

**Figure 5.** Change of Vector According To The Change of Direction of The Object

### How The Capsule Works?

The lower-level capsules from the previous layer are represented by the $U_1$, $U_2$ ,, and $U_3$ capsules. Weight matrices containing their relationships with higher-level capsules are multiplied by these capsules. Weight matrices store the part-whole relationship between high-level capsules and low-level capsules. The estimated output vectors $\hat{U}_1$ , $\hat{U}_2$ , and $\hat{U}_3$ are obtained as a result of the multiplication. The weighted sum is passed through the squash function to determine the capsule's output value, $V_j$. Figure 6 shows the capsule's procedure.

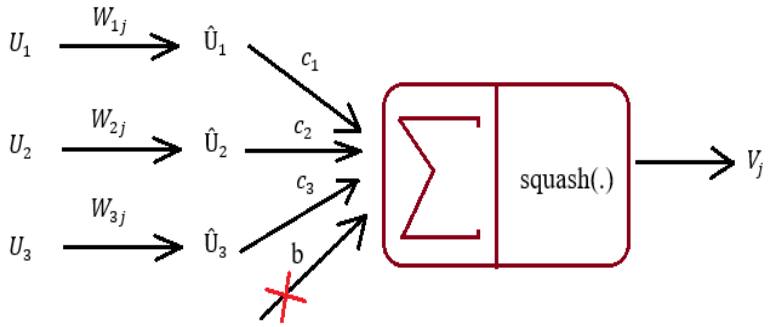


**Figure 6.** The Capsule's Procedure

### Squash Function

A structure called the squash function in capsule networks is created to replace the place of activation functions like ReLU, Sigmoid, and Tangent used in deep neural networks. Squash is a vector-outputting activation function. It makes sure that long vectors are near 1, and that short vectors are near 0. It is a nonlinear function that restricts the neuron's output value to a range of 0 to 1 [5]. The equation of the squash function is given in Equation 1. The first part of the equation restricts the output value to the range of 0 to 1, and the second part of the equation transforms the value into a unit vector.

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

### Dynamic Routing Algorithm

The dynamic routing algorithm is used to connect the capsules in the lower layer with the capsules in the upper layer. This algorithm determines the calculation of the routing coefficients that show which capsule will be connected to which capsule. The algorithm is shown in Algorithm1.

**Algorithm1.** Dynamic Routing Algorithm [5]

**procedure** ROUTING ($\hat{u}_{j|i}$, *r*,*l*)
  for all capsules *i* in layer *l* and capsule *j* in layer (*l*+1): $b_{ij} \leftarrow 0$.
  **for** r iterations **do**
    for all capsule *i* in layer *l*: $c_i \leftarrow$ **softmax** ($b_i$)
    for all capsule *j* in layer (*l*+1): $s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$
    for all capsule *j* in layer (*l*+1): $v_i \leftarrow$ **squash** ($s_j$)
      for all capsule *i* in layer *l* and capsule *j* in layer (*l*+1): $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}v_{ij}$
**return** $v_j$

In this algorithm, $b_{ij}$ is the temporary similarity variable with an initial value of 0. Its value is updated during each iteration. By passing $b_{ij}$ through the softmax function, the $c_{ij}$ matching coefficients are calculated (Equation 2).

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{2}$$

$$\hat{u}_{j|i} = W_{ij} u_i \tag{3}$$

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{4}$$

The estimated output vector of the capsule $\hat{u}_{j|i}$ is obtained by multiplying the input values of the $u_i$ capsule with the weight matrix $W_{ij}$ (Equation 2). Each capsule's prediction vector is multiplied by the matching coefficient, the products are added up, and the weighted sum $s_j$ is found (Equation 3). The value $s_j$ is passed through the squash function, and the capsule's output value, $v_j$, is computed. $b_{ij}$ is updated in the final step of the algorithm, and all steps are repeated *r* from step 4.

### Margin Loss Function

The loss function is calculated after the digit capsule layer. The vector's length represents the possibility that the object is in the image. Margin loss is calculated for each output capsule based on this value. The total margin loss is calculated by adding the margin losses of all capsules.

$$L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k)\max(0, \|v_k\| - m^-)^2 \tag{5}$$

The margin loss is calculated using the formula shown in Equation 5. $T_k$ 1 is used if the prediction is correct; otherwise, $T_k$ 0 is used. If the prediction is correct, the first part of the equation is used (Equation 6); otherwise, the second part is used (Equation 7).

$$L_k = T_k \max(0, m^+ - \|v_k\|)^2 \tag{6}$$

$T_k$ equals 1 if the prediction is correct. The capsule's output value is subtracted from the $m^+$ value and squared. The value of $m^+$ is assumed to be 0.9. If the capsule's output value is greater than 0.9, the error value is zero; otherwise, it is non-zero.

$$L_k = \lambda(1 - T_k)\max(0, \|v_k\| - m^-)^2 \tag{7}$$

$T_k$ is 0 if the prediction is incorrect. It is squared by subtracting the $m^-$ value from the capsule's output value. The value of $m^-$ is assumed to be 0.1. If the capsule's output value is less than 0.1, the error value is zero; otherwise, it is non-zero. The lambda value ($\lambda$) is set to 0.5 for the formula.

### EM (Expectation-Maximization) Routing Algorithm

The expectation-maximization algorithm is used in EM routing. Instead of capsules consist of neurons, the algorithm proposed by Hinton et al. [7] used 4x4

pose matrices. These pose matrices contain the spatial relationships of the object (transformation and translation matrices). The length of the vector in the dynamic routing algorithm indicated the probability of the object's existence. The α parameter is used instead of the vector length in this algorithm.

Each capsule in the lower layer casts one vote for the capsules in the upper layer in the EM algorithm. The capsules ($\Omega_L$) in the lower layer are consists of each capsule's 4x4 pose matrix ($M$) and activation probabilities ($\alpha$). Within the algorithm, the votes from each capsule are weighted by the assignment coefficient and updated iteratively (Equation 8).

$$\mathbf{v}_{ij} = M_j W_{ij} \tag{8}$$

The pose and activation values of the capsules in the upper layer are calculated using the EM algorithm. The values of $\mathbf{v}_{ij}$ and $\alpha_i$ are input into the algorithm. The Gaussian distribution clustering method is used in this algorithm to group sub-level capsules. As a result, each upper-level capsule corresponds to a Gaussian distribution, and each lower-level capsule corresponds to a point in that Gaussian distribution. The algorithm is shown in Algorithm2.

**Algorithm2.** Expectation-Maximization Algorithm [7]

**procedure** EM ROUTING (α, *V*)
$\forall \epsilon \Omega_L, j \epsilon \Omega_{L+1}$: $R_{ij} \leftarrow 1/|\Omega_{L+1}|$
  **for** r iterations **do**
    $\forall i \epsilon \Omega_{L+1}$: M STEP (α, *R, V, j*)
    $\forall$h: $\epsilon \Omega_{L+1}$: M STEP (α, *R, V, j*)
  **return** α, *M*

**procedure** M-STEP (α, *R, V,j*)
  $\forall i \epsilon \Omega_L$: $R_{ij} \leftarrow R_{ij} * a_i$
  $\forall$h: $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$
  $\forall$h: $\left(\sigma_j^h\right)^2 \leftarrow \frac{\sum_i R_{ij}\left(V_{ij}^h - \mu_j^h\right)^2}{\sum_i R_{ij}}$
  $cost^h \leftarrow (\beta_u + \log(\sigma_j^h)) \sum_i R_{ij}$
  $a_j \leftarrow logistic\ (\lambda\ (\beta_a - \sum_h cost^h))$

**procedure** E-STEP ($\mu$, $\sigma$, α,*V, i*)
  $\forall$j: $\epsilon \Omega_{L+1}$: $p_i \leftarrow \frac{1}{\prod_h^H 2\pi\left(\sigma_j^h\right)^2} \exp(-\sum_h^H \frac{\left(V_{ij}^h - \mu_j^h\right)^2}{2\left(\sigma_j^h\right)^2})$
  $\forall$j: $\epsilon \Omega_{L+1}$: $R_{ij} \leftarrow \frac{a_j p_j}{\sum_{k \epsilon \Omega_{L+1}} a_j p_j}$

The EM algorithm uses backpropagation to update the transformation matrices that retain the capsules' information. E-STEP and M-STEP procedures are used to perform this.

## CAPSULE NETWORKS (CAPSNET) ARCHITECTURE

The architecture of the CapsNet is divided into two parts. The encoder, which is the first part, extracts the image's features and performs classification. The image is reconstructed in the second part, the decoder. The architecture is shown in Figure 7. Table 2 shows the input and output dimensions of the architecture's layers, as well as the number of parameters used in each layer.

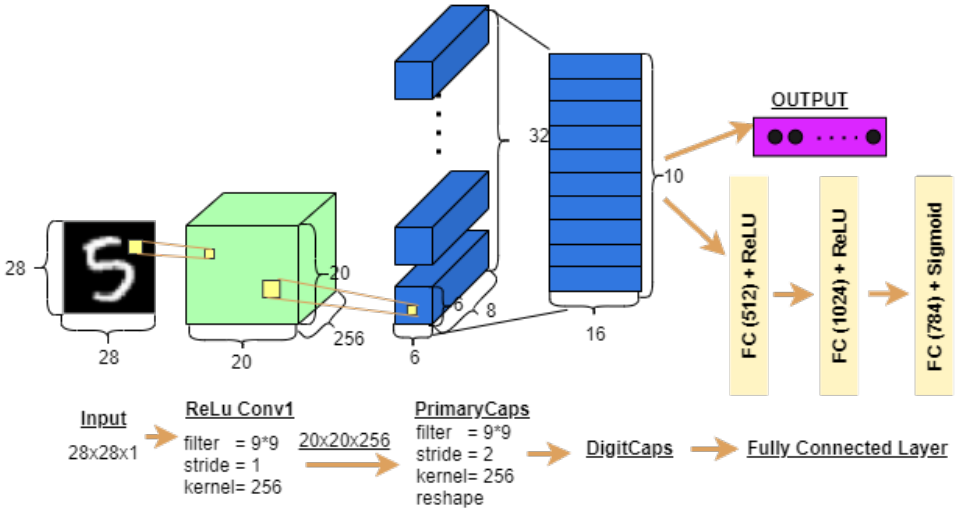


**Figure 7.** CapsNet Architecture

**Table 2**. Summary Of Capsule Network Architecture And Number Of Parameters Used

| Layer | Input Dimension | Output Dimension | The Number Of Parameters |
|---|---|---|---|
| **Convolution Layer** | 28*28 | 20*20*256 | 20992 |
| **Primary Capsule Layer** | 20*20*256 | 6*6*8*32 | 5308672 |
| **Digit Capsule Layer** | 6*6*8*32 | 16*10 | 1497600 |
| **Fully Connected Layer 1** | 16*10 | 512 | 82432 |
| **Fully Connected Layer 2** | 512 | 1024 | 525312 |
| **Fully Connected Layer 3** | 1024 | 784 | 803600 |
| **Total Number of Parameters** | | | 8238608 |

### Encoder

The encoder is consisting of three layers: a convolution layer, a primary capsule layer, and a digit capsule layer.

### Convolution Layer

This layer extracts the image's basic features. It is the first layer. Input is a 28*28 single-channel image. Applying the convolution operation with a 256 kernel size, 9*9 filter, and 1 stride to the input image obtains a 20*20*256 output tensor.

### Primary Capsule Layer

The primary capsule layer is the architecture's second layer. It takes the previous layer's 20*20*256 tensor as input. Applying the convolution operation with 256 kernel size, 9*9 filter, and 1 stride results in a tensor with a size of 6*6*256. The reshaping procedure produces 32 8-dimensional capsules. At the layer output, the squash function is used as the activation function.

### Digit Capsule Layer

This layer takes a tensor of size 6*6*8*32 as input. 10 capsules with 16 dimensions each are created as a result of the dynamic routing procedure. The output is determined by the capsule with the highest probability value among the capsules. The loss calculation is performed after this layer, and the image is reconstructed using fully connected layers.

### Decoder

To reconstruct the input image, the decoder consists of interconnected layers. It is made up of three fully connected layers. The first layer contains 512 neurons, the second layer contains 1024 neurons, and the third layer contains 784 (28*28) neurons. In the first two layers, ReLU activation function is used, and Sigmoid activation function is used in the final layer. It takes 16-dimensional vectors as input and reconstructs a 28*28 image.

### Example of CapsNet

The CapsNet architecture is trained on the PneumoniaMNIST dataset, which is part of the MedMNIST dataset [8], in this section. The PneumoniaMNIST dataset contains 5.856 28*28 pediatric chest X-ray images. The data set is split into training and test data in a 9:1 ratio. The data is divided into two classifications: Pneumonia and Normal. Figure 8 shows images from the data set.

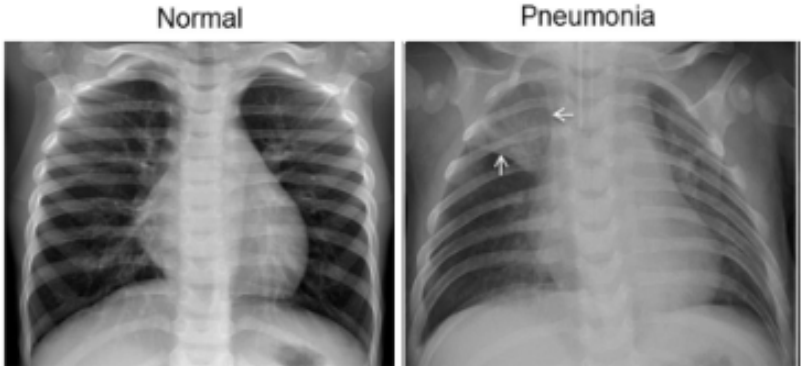


**Figure 8.** Images from the Pneumonia Dataset

The case study was performed on a device equipped with an Intel Xeon Gold 6226R processor and an Nvidia Grid RTX8000-12Q graphics card, as well as Python 3.8, Tensorflow2.3, CUDA10.1.243, and cuDNN7.6.5 technologies.

The study available at "https://github.com/XifengGuo/CapsNet-Keras" was used in the development of the CapsNet model [9].

First, the complexity matrix for the performance evaluation of the CapsNet architecture on the Pneumonia dataset was obtained. Table 3 contains explanations for the complexity matrix.

**Table 3.** Confusion Matrix

| | | **Predicted Class** | |
|---|---|---|---|
| | | **Normal** | **Pneumonia** |
| **Actual Class** | **Normal** | **TP**<br>Patient does not have pneumonia, correctly identified | **FN**<br>Patient does not have pneumonia, misidentified |
| | **Pneumonia** | **FP**<br>Patient Pneumonia patient, misidentified | **TN**<br>Patient Pneumonia patient, correctly identified |

Accuracy, Precision, Recall, and f1-score metrics were used to evaluate performance based on this confusion matrix. Table 4 shows the formulas used to calculate these values.

**Table 4.** Formulas Of Performance Metrics

| **Metric** | **Formula** |
|---|---|
| **Accuracy** | (TP+TN)/(TP+TN+FP+FN) |
| **Precision** | TP/(TP+FP) |
| **Recall** | TP/(TP+FN) |
| **f1-score** | 2 * (Precision * Recall) / (Precision + Recall) |

Figure 9 shows the confusion matrix formed as a result of training the capsule network architecture with the PneumoniaMNIST dataset. Table 5 shows the performance metrics computed as a result of the confusion matrix.

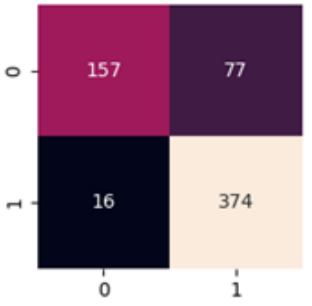


**Figure 9.** Confusion Matrix

**Table 5.** Performance Metrics

| Metric | Formula |
|---|---|
| **Accuracy** | (TP+TN)/(TP+TN+FP+FN) |
| **Precision** | TP/(TP+FP) |
| **Recall** | TP/(TP+FN) |
| **f1-score** | 2 * (Precision * Recall) / (Precision + Recall) |

In addition, the Loss and Accuracy graphics of the study are presented in Figure 10.

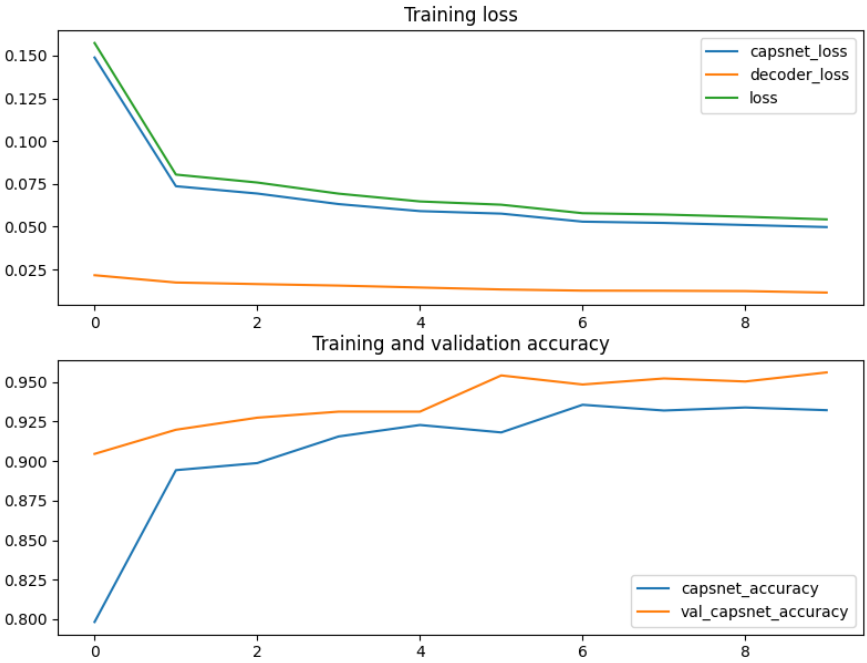


**Figure 10.** Accuracy and Loss Graphics

## CAPSNET IMPLEMENTATIONS

Various studies in the literature aim to improve the performance of Capsule Networks architecture and make it more effective in problem-solving. Various changes have been made by making architectural or dynamic routing algorithm innovations.

To reduce the computational complexity in capsule networks, Zhang et al. [10], used a kernel density estimation approach in a dynamic routing algorithm. They proposed two dynamic routing algorithms based on this. In one, they used the mean shift clustering method, while in the other, they used the expectation maximization method. While the Mean Shift clustering method resulted in 97.40%, 99.58%, 94%, and 84% accuracy on SmallNorb, MNIST, Fashion-MNIST, and CIFAR10 datasets, the other method resulted in 97.80%, 99.58%, 99.68%, and 85.70% accuracy. Zhao

et al. [11], used the max-min normalization method rather than the softmax function, claiming that the softmax function limits the solution range when calculating the routing coefficients that provide communication between the lower and upper layers. They obtained 99.55% accuracy in the MNIST dataset and 75.22% accuracy in the CIFAR10 dataset as a result of the study.

Rezwan et al. [12] aimed to do a study that would reduce the memory requirement. For transformation matrices, they used simple matrix multiplication. In architecture, they used both vector and matrix poses. The Convolutional Capsule layer, named " Single Matrix, was added to the architecture. They achieved 94.72% accuracy in the CIFAR10 dataset and 75.85% accuracy in the CIFAR100 dataset as a result of the study.

Ahmed et al. [13] did another study to reduce computational complexity. They proposed a non-recursive routing algorithm in the study to ensure communication between capsules. They claimed that their proposed mechanism outperforms basic capsule networks on MNIST, SmallNORB, CIFAR10, CIFAR100, and ImageNet datasets. To achieve faster convergence to the solution, Yang et al. [14] proposed a routing algorithm that employs a regularized quadratic programming approach rather than a dynamic routing algorithm. MNIST, FashionMNIST, and CIFAR10 data sets achieved 99.68%, 93.25%, and 85.96% accuracy, respectively. Xiang et al. [15], proposed a two-stage architecture called MS-CapsNet to reduce the computational complexity of capsule networks. The semantic and structural information of the image was extracted in the first stage using feature extraction, and the capsules were divided into three branches of varying sizes based on their feature order and coded in the second stage. In this study, they achieved 92.70% and 75.70% accuracy in the FashionMNIST and CIFAR10.

Rajasegaran et al. [16] proposed DeepCaps, a model that deepens capsule networks to increase success in complex images. They used a 3D convolution-based deep learning algorithm and a class-independent decoder network. In the CIFAR 10, SVHN, Fashion MNIST, and MNIST datasets, they achieved 92.74%, 97.56%, 94.73%, and 99.72%. Cheng et al. [17] proposed Cv-CapsNet and Cv-CapsNet++ architectures for the analysis of complex and dense data. These three-stage architectures use a densely connected complex-valued network for feature extraction. They achieved 94% and 85.64% accuracy in the Fashion MNIST and CIFAR 10 datasets.

The study of Xiong et al. [18] is another study for the analysis of complex data. They deepened CapsNet with the Conv-Caps layer by increasing the convolution layers for complex data analysis. To reduce the number of parameters, they developed the Caps-Pool layer, which allows features to be transferred without deterioration. On the CIFAR10 dataset, they achieved 81.29% accuracy using the proposed model.

Rosario et al. [19] performed a different study in which they divided the CapsNet

into lanes. Each lane of the multi-lane capsule network, MLCN, extracts different image features. In the MLCN, FashionMNIST, and CIFAR10 data sets, accuracy was 92.63% and 75.18%, respectively. Chang et al. [20] proposed a multi-lane capsule network with strict-squash based on this research (MLSCN). They aimed to highlight important features in the study by changing the squash function and limiting noisy and unnecessary features. They obtained 99.73%, 81.71%, and 76.79% accuracy in MNIST, affNIST, and CIFAR10 datasets, respectively.

Hoogi et al. [21] proposed the Self-Attention-Capsule-Networks (SACN) architecture, which combined the self-attention module with capsule networks. The Self Attention module extracts important features by separating the region of interest from the image. While they achieved 99.5% accuracy on the MNIST dataset, they improved classification accuracy by 3.5% on the CIFAR10 dataset. They have also achieved high performance on a variety of medical images. Huang et al. [22] developed the Diverse Enhanced Capsule Network (DE-CapsNet) architecture for complex data analysis. In the architecture, they used a two-level primary capsule and a position-wise dot product. Within the dynamic routing function of the DE-CapsNet architecture, which is a hierarchical architecture, they used sigmoid activation function rather than softmax activation function. This architecture achieved 92.96% accuracy in the CIFAR0 dataset and 94.25%.

To reduce computational complexity, Singh et al. [23] proposed the DeepFear-Caps architecture with minimal computational overhead. In the architecture, they used feature extraction blocks that included 3x3 convolution, batch normalization, and 1x1 convolution. They obtained 81.80%, 93.20%, 95.60%, and 99.60% accuracy on CIFAR10, FashionMNIST, SVHN, and MNIST data sets.

Another study is Zeng et al. [24]'s study on increasing the speed of capsule networks. Dense blocks were used in the study to reduce computational complexity and improve performance. They extracted features by using dense blocks. Furthermore, instead of using softmax in the routing algorithm, they calculated the variance and mean of the low-level capsules to find the pose vectors. To increase speed, they did not iterate over the routing algorithm. They observed that the number of parameters in the MNIST data set decreased as a result of the study, as did the error rate. Mazzia et al. [25] proposed a non-recursive routing algorithm. In the primary capsule layer, they used depth wise separable convolution. They utilized a squash function that is more sensitive to minor variations. After the primary capsule layer, they used self-attention. They reduced the error rate on the MNIST dataset to 0.16% using this architecture.

## APPLICATION AREAS OF CAPSNET

The Capsule Networks architecture has produced promising results in a variety of fields, including handwriting and text recognition [26, 27], emotion detection [28], medical image analysis [29], and cancer type detection [30,31].

Kim et al. [32] done a study to estimate traffic speed on complex roads. They gathered information by strategically placing speed sensors throughout the city. In their comparison of CNN and CapsNet, CapsNet outperformed CNN by 13.1% in. Pari et al. [33] developed a traffic sign detection system in order to reduce traffic accidents and improve road safety.

Capsule nets have also been used in the detection and classification of various diseases. It is critical to identify the patient area and extract information from medical images. The evaluation of images by capsule networks using the part-whole relationship has aided in the correct determination of this area. Afshar et al. [30] classified brain tumors with 86.56% accuracy using this feature of capsule networks. Kumar et al. [34] used CapsNet to detect diabetic retinopathy in retinal images. They performed separate image classification studies with five classes: No DR, Mild, Moderate, Severe, and Proliferative, and two classes: NoDR and DR. They had the most achievement when they used the VGG16 architecture before the Digit capsule layer. By increasing the number of convolution layers, they aimed to extract more features. They obtained 82.06% accuracy in 5-class classification and 96.24% accuracy in 2-class classification as a result of the study. Another study in the field of health was conducted by Kavitha et al. [35] Detection of breast cancer from mammography images. Following segmentation in the study, the features of the patient area were extracted using capsule networks and classified using a Back-Propagation Neural Network. They obtained high accuracy on various data sets as a result of the study.

Capsule networks have also been used in biometric recognition research. Linda et al. [36] developed a walking gait recognition system using sensor data. The research by Xu et al. [37] on gait recognition is another study. They used convolution layers before the fundamental CapsNet architecture in their study, CapsGaitNet. The study made use of the CASIA-B dataset, which contains silhouettes of various walking styles, and the OU-SIR Treadmill dataset, which contains walking images made while wearing various outfits. Good results were obtained in both data sets as a result of the study.

CapsNet has also been used successfully in biometric recognition studies such as iris recognition [38], biometric recognition from ECG signals [39, 40], and fingerprint recognition[41].

A different study using capsule networks is a classification study in smart cities and IoT networks to ensure network security and service quality [42]. All data traffic used in smart city applications (smart homes, smart traffic management, smart shopping, smartphones, and so on) is classified as benign or malignant. A high accuracy rate was obtained as a result of the study.

## RESULTS

Despite the fact that CapsNet is a new architecture in the field of deep learning, it has produced successful results in a variety of areas. It has a more resistant structure to data transformations, especially because it hides the part-whole relationship [2]. This shows that network security is superior to that of other networks. Furthermore, by using capsules, the lack of information caused by the pooling process was eliminated, and the objects were better identified. Instead of the pooling process, the dynamic routing algorithm was used to evaluate and transmit all of the data in the image to the higher layers.

CapsNet is used in a variety of applications, including medical image analysis, biometric recognition, traffic systems, disease detection, and text recognition.

Although it has advantages over convolutional neural networks, it does not perform as well as CNN's in complex data sets with dense backgrounds, such as CIFAR10, because it considers every detail on the image. The success rate on complex data sets can be increased by making changes such as adding pre-architectural convolution layers and changing the dynamic routing algorithm [43,20,15,44,45]. Furthermore, despite having fewer parameters than CNN, the training time may be longer due to the high computational complexity.

CapsNet is an architecture that can be researched and developed because it is a relatively new design. It has the potential to make significant contributions to the field of computer vision by resolving challenges such as computation complexity and poor performance on complex data sets.

## REFERENCES


[1]. Hao, X., ve ark., 2016, "Technical Survey Deep Learning", International Journal of Semantic Computing, 10(3), 417-439.

[2]. Deng, L. and Yu, D., 2014, "Deep learning: Methods and applications", Foundations And Trends® In Signal Processing, 7 (3–4),197–387.

[3]. Khan, A., Sohail, A., Zahoora, U. et al. "A survey of the recent architectures of deep convolutional neural networks." Artif Intell Rev 53, 5455–5516, 2020. https://doi.org/10.1007/s10462-020-09825-6

[4]. Kwabena Patrick, M., et al., "Capsule Networks – A survey", Journal of King Saud University - Computer and Information Sciences,vol. 34(1), pp. 1295-1310, 2022.

[5]. Sabour, S., N. Frosst, and G. Hinton, "Dynamic Routing Between Capsules", 31st Conference on Neural Information Processing Systems (NIPS 2017), 2017.

[6]. Zhang, X. and S.-G. Zhao, "Fluorescence microscopy image classification of 2D HeLa cells based on the CapsNet neural network", Medical & Biological Engineering & Computing, vol. 57(6), pp. 1187-1198, 2019.

[7]. G. Hinton, S. Sabour and N. Frosst, "Matrix Capsules With EM Routing", ICLR (2018).

[8]. J. Yang, R. Shi and B. Ni, "MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis," 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), 2021, pp. 191-195, doi: 10.1109/ISBI48211.2021.9434062.

[9]. "https://github.com/XifengGuo/CapsNet-Keras", (accessed 23.10.2022)

[10]. S. Zhang, Q. Zhou and X. Wu, "Fast Dynamic Routing Based on Weighted Kernel Density Estimation. In: Lu, H. (eds) Cognitive Internet of Things: Frameworks", Tools and Applications. Studies in Computational Intelligence, vol 810. Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-04946-1_30.

[11]. Z. Zhao et al., "Capsule Networks with Max-Min Normalization", arXiv:1903.09662v1 [cs.CV], pp. 1-15, 2019.

[12]. I. M. Rezwan, M. B. Ahmed, S. S. Sourav, E. Quader, A. Hossain and N. Mohammed, "MixCaps: Capsules With Iteration Free Routing," 2020 Digital Image Computing: Techniques and Applications (DICTA), pp. 1-8, 2020, doi: 10.1109/DICTA51227.2020.9363386.

[13]. K. Ahmed, L. Torresani. "Star-caps: Capsule Networks With Straight-Through Attentive Routing", In: Advances in Neural Information Processing Systems, pp. 9098–9107, 2019.

[14]. H. Yang, L. Shuhe and Y. Bei, "Routing Towards Discriminative Power of Class Capsules", In: arXiv preprint arXiv:2103.04278, 2021.

[15]. C. Xiang, L. Zhang, Y. Tang, W. Zou and C. Xu, "MS-CapsNet: A Novel Multi-Scale Capsule Network," in IEEE Signal Processing Letters, vol. 25, no. 12, pp. 1850-1854, 2018, doi: 10.1109/LSP.2018.2873892.

[16]. J. Rajasegaran et al., "DeepCaps: Going Deeper with Capsule Networks", ArXiv Prepr. arXiv1904.09546, pp. 1-9, 2019.

[17]. X. Cheng et al., "Cv-CapsNet: Complex-valued capsule network", IEEE Access, 7, pp. 85492-85499, 2019.

[18]. Y. Xiong, G. Su, S. Ye, Y. Sun and Y. Sun, "Deeper Capsule Network For Complex Data", International Joint Conference on Neural Networks (IJCNN), pp. 1-8, 2019, doi: 10.1109/IJCNN.2019.8852020.

[19]. V.M. Rosario, E. Borin andM. Breternitz Jr, "The Multi-Lane Capsule Network (MLCN)" arXiv:1902.08431v1 [cs.CV], pp. 1-5, 2019.

[20]. S. Chang and J. Liu, "Multi-Lane Capsule Network for Classifying Images With Complex Background", in IEEE Access, vol. 8, pp. 79876-79886, 2020, doi: 10.1109/ACCESS.2020.2990700.

[21]. A. Hoogi, B. Wilcox, Y. Gupta and D.L. Rubin, "Self-Attention Capsule Networks for Image Classification", arXiv:1904.12483, 2019.

[22]. B. Jia, Q. Huang, "DE-CapsNet: A diverse enhanced capsule network with disperse dynamic routing" Appl. Sci., 10 (3), p. 884, 2020.

[23]. C. K. Singh et al. "A Light-weight Deep Feature based Capsule Network", In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE pp. 1–8, 2020.

[24]. R. Zeng and Y. Song, "A Fast Routing Capsule Network With Improved Dense Blocks," in IEEE Transactions on Industrial Informatics, vol. 18, no. 7, pp. 4383-4392, 2022, doi: 10.1109/TII.2021.3128412.

[25]. V. Mazzia, F. Salvetti and M. Chiaberge, "Efficient-CapsNet: Capsule Network With Self-Attention Routing". Sci Rep 11, 14634 ,2021, https://doi.org/10.1038/s41598-021-93977-0

[26]. T. Iqbal, H. Ali, M. M. Saad, S. Khan and C. Tanougast, "Capsule-Net for Urdu Digits Recognition", 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), pp. 495-499, 2019,doi: 10.1109/IDAACS.2019.8924362.

[27]. Y. Hongge, T. Yuxing, X. Chunqiu, Y. Jun and B. Xiaojun, "Deep Capsule Network For Recognition And Separation of Fully Overlapping Handwritten Digits", Computers & Electrical Engineering, Volume 91, 2021, 107028, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2021.107028.

[28]. A. B. Nassif et al., "Emotional Speaker İdentification Using A Novelcapsule Nets Model", Expert Syst. Appl.,193, 116469, 2022.

[29]. M. A. Ayidzoe at al., "Visual Interpretability Of Capsule Network For Medical Image Analysis", Turkish Journal of Electrical Engineering and Computer Sciences: Vol. 30: No. 3, Article 31, 2022, https://doi.org/10.55730/1300-0632.3822

[30]. P. Afshar, A. Mohammadi and K.N. Plataniotis, "Brain Tumor Type Classification Via Capsule Networks", IEEE International Conference on Image Processing (ICIP), pp. 3129-3133, 2018.

[31]. P. Afshar et al.,"MIXCAPS: A capsule network-based mixture of experts for lung nodule malignancy prediction.", Pattern Recognition 116, 2021, https://doi.org/10.1016/j.patcog.2021.107942NS

[32]. Y. Kim, P. Wang, Y. Zhu and L. Mihaylova, "A Capsule Network for Traffic Speed Prediction in Complex Road Networks", arXiv:1807.10603v2 [cs.CV], 2018.

[33]. P. S. Neelavath,T. Mohana and V. Akshaya, "Real-Time Traffic Sign Detection using Capsule Network", Advanced Computing (ICoAC) 2019 11th International Conference on, pp. 193-196, 2019.

[34]. G. Kumar, S. Chatterjee, and C. Chattopadhyay, DRISTI: a hybrid deep neural network for diabetic retinopathy diagnosis, Signal, Image and Video Processing, 15, 2021.

[35]. T. Kavitha et al., "Deep Learning Based Capsule Neural Network Model for Breast Cancer Diagnosis Using Mammogram Images", Interdiscip. Sci. Comput. Life Sci., 1–17, 2021.

[36]. G. M. Linda et al., "Intelligent recognition system for viewpoint variations on gait andspeech using CNN-CapsNet", International Journal of Intelligent Computing and Cybernetics, Volume 15, Number 3, pp. 363-382(20), 2021.

[37]. X. Zhaopeng et al., "Gait Recognition Based On Capsule Network", Journal of Visual Communication and Image Representation, Volume 59, pp 159-167, ISSN 1047-3203, 2019, https://doi.org/10.1016/j.jvcir.2019.01.023.

[38]. T. Zhao, Y. Liu, G. Huo and X. Zhu, "A Deep Learning Iris Recognition Method Based on Capsule Network Architecture," in IEEE Access, vol. 7, pp. 49691-49701, 2019, doi: 10.1109/ACCESS.2019.2911056.

[39]. B. Imane et al., "A Wavelet-Based Capsule Neural Network For ECG Biometric İdentification", Biomedical Signal Processing and Control, Volume 76, 103692, ISSN 1746-8094, 2022, https://doi.org/10.1016/j.bspc.2022.103692.

[40]. A. J. Prakash, "Capsule Network for the Identification of Individuals Using Quantized ECG Signal Images", IEEE Sensors Letters, vol. 6, no. 8, pp. 1-4, Aug. 2022, Art no. 7003004, doi: 10.1109/LSENS.2022.3195174.

[41]. D. Gumusbas, T. Yildirim, M. Kocakulak and N. Acir, "Capsule Network for Finger-Vein-based Biometric Identification", 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 437-441, 2019, doi: 10.1109/SSCI44817.2019.9003019.

[42]. H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang and Z. Han, “Capsule Network Assisted IoT Traffic Classification Mechanism for Smart Cities”, IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7515-7525, 2019, doi: 10.1109/JIOT.2019.2901348.

[43]. E. Xi, S. Bing and Y. Jin, ”Capsule Network Performance on Complex Data”, arXiv1712.03480v1 [stat.ML], pp. 1-7, 2017.

[44]. S. Yang et al., “RS-CapsNet: An Advanced Capsule Network,” in IEEE Access, vol. 8, pp. 85007-85018, 2020, doi: 10.1109/ACCESS.2020.2992655.

[45]. K. Sun, L. Yuan, H. Xu and X. Wen, “Deep Tensor Capsule Network,” in IEEE Access, vol. 8, pp. 96920-96933, 2020, doi: 10.1109/ACCESS.2020.2996282.