Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

# Capsule Networks – A survey

Mensah Kwabena Patrick [a,*], Adebayo Felix Adekoya [a], Ayidzoe Abra Mighty [b], Baagyire Y. Edward [c]

[a] Department of Computer Science & Informatics, University of Energy and Natural Resources, P.O. 214, Sunyani, Ghana
[b] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China
[c] Department of Computer Science, University for Development Studies, P.O. Box 24, Navrongo, Ghana

ARTICLE INFO

*Article history:*
Received 13 July 2019
Revised 24 September 2019
Accepted 25 September 2019
Available online xxxx

*Keywords:*
Artificial intelligence
Deep learning
Capsule network
Squashing function
Dynamic routing
Expectation maximization
Backpropagation

ABSTRACT

Modern day computer vision tasks requires efficient solution to problems such as image recognition, natural language processing, object detection, object segmentation and language translation. Symbolic Artificial Intelligence with its hard coding rules is incapable of solving these complex problems resulting in the introduction of Deep Learning (DL) models such as Recurrent Neural Networks and Convolutional Neural Networks (CNN). However, CNNs require lots of training data and are incapable of recognizing pose and deformation of objects leading to the introduction of Capsule Networks. Capsule Networks are the new sensation in Deep Learning. They have lived to this expectation as their performance in relation to the above problems has been better than Convolutional Neural Networks. Even with this promise in performance, lack of architectural knowledge and inner workings of Capsules serves as a hindrance for researchers to take full advantage of this breakthrough. In this paper, we provide a comprehensive review of the state of the art architectures, tools and methodologies in existing implementations of capsule networks. We highlight the successes, failures and opportunities for further research to serve as a motivation to researchers and industry players to exploit the full potential of this new field. The main contribution of this survey article is that it explains and summarizes significant current state of the art Capsule Network architectures and implementations.


**Contents**



* Corresponding author.
*E-mail addresses:* patrick.mensah@uenr.edu.gh (M. Kwabena Patrick), adebayo.adekoya@uenr.edu.gh (A. Felix Adekoya), mighty.ayidzoe@uds.edu.gh (A. Abra Mighty), ybaagyere@uds.edu.gh (B.Y. Edward).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2019.09.014
1319-1578/

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

## 1. Introduction

Computer vision applications ranges from real time detection of product defects in manufacturing through to defense and security where thousands of video footages can be analyzed. These are critical applications requiring solutions in real time. Such real time demands cannot be met by a human being. The availability of enormous amounts of data makes it possible for Neural Networks (NN) (Russakovsky et al., 2015) and deep Convolutional Neural Networks (LeCun et al., 2014; LeCun et al., 1998a) to excel in areas of Computer Vision. They have effectively been used for tasks such as plant disease detection (Dey et al., 2016; Golhani et al., 2018; Sladojevic et al., 2016), facial (expression) recognition (Sun et al., 2017; Fasel and Luettin, 2003), image processing and speech recognition (Sukittanon et al., 2004; Abdel-hamid et al., 2014; Lecun et al., 2015; Chen et al., 2017). CNNs are invariant as they can extract features automatically and handle the data in different forms.

CapsNets (Sabour et al., 2017) were introduced with superior performance on the MNIST (LeCun et al., 1998c) dataset. Since their introduction in 2017, there has been an upsurge in deep learning models employing Capsule Networks as their core building blocks. The most popular version of CapsNets uses an algorithm called "routing by agreement" between different layers. This algorithm replaces pooling in CNNs and a vector output replacing scalar outputs in CNNs. The magnitude of the output vector represents the likelihood that a feature being represented by the capsule exist in the input image whereas the orientation of the entity represents the instantiation parameter values. As a result, CapsNet will not recognize an image with nose below the mouth and an eye below the nose as a human face (Sabour et al., 2017). However, CNN will classify this image as a human face so far as it has a nose, an eye and a mouth since they lose the spatial relationship between features. CNNs require huge sets of data for training. The acquisition and labelling of these huge sets of data is labor intensive.

This paper therefore seeks to highlight the weaknesses of CNN and the remarkable performance of CapsNet by reviewing implementations in literature. The paper presents and evaluates state of the art models in this emerging field as well as consider possible future dimensions of CapsNets.

The contributions of this paper are as follows: to

- present state-of-the-art models in this emerging field to motivate researchers and industry players,
- explore possible future areas of research and
- explore current performance evaluation methods.

Since Capsule Networks are relatively new, this paper first attempts to explain into detail the concept behind them and other concepts that serve as their building blocks. Secondly, we review the most relevant CapsNet implementations, their strengths and limitations and suggest possible future directions. To the best of our knowledge, this is the first paper to present a comprehensive review on capsule networks. We take cognizance of the work in (Dombetzki, 2018) and state here that it does not review exhaustive literature on the topic as done in this paper.

This paper is organized as follows: In Section 1, we provide the objectives of the paper together with the background of the field under consideration. Section 2 provides a brief overview of AI concepts that are heavily utilized in DL. CNNs and CapsNets are discussed in Sections 3 and 4 respectively. The structure, implementations and performance evaluation methods are reviewed in Section 5 with Section 6 concluding the paper.

## 2. Artificial Neural Networks and deep learning

Artificial Intelligence (AI) is a popular field in Computer Science that is based on the intelligence of the biological brain's network of neurons. Network of Artificial Neurons contains layers and neurons connected together by synapses which are weighted. The weights are adjusted through backpropagation (Nielson, 2019; LeCun et al., 1998b) enabling the network to learn. The firing strength of a neuron is controlled by an activation function. They are also needed for non-linearity (Pattanayak, 2017). Let $x$ be the value of the weighted sum of the inputs $x_i$ and weights $w_i$, then from $i = 1$ to $n$, then

$$x = \sum_{i=1}^{n} w_i x_i \tag{1}$$

Examples of activation functions include sigmoid function (Eq. (2)), Rectified Linear Unit (ReLU) (Glorot et al., 2011) (Eq. (4)), Soft-Max Activation Function (Engelbrecht, 2007; Gao and Pavel, 2018; Pattanayak, 2017) (Eq. (3)) and the Hyperbolic Tangent function (tanh) (Eqs. (6) and (7)).

$$\varnothing(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

$\varnothing(x)$ is constrained by a pair of horizontal asymptotes as $x \to \pm\infty$.

$$P(\emptyset(x)_i = 1/x) = \frac{e^{x^i}}{\sum_{j=1}^{n} e^{x^j}} \tag{3}$$

where $n$ = *number of output classes*, $i$ and $j$ represents the *ith* and *jth* classes with $P(\emptyset(x)_i)$ being the predicted probability for the *ith* class.

$$\varnothing(x) = \max(x, 0) \tag{4}$$

Other versions of ReLU such as Parametric Rectified Linear Unit (PReLU) (Eq. (5)) and Leaky Rectified Linear Unit (Leaky ReLU) solves the zero-gradient problem.

$$\varnothing(x) = max(0, x) + \beta \min(0, x) \tag{5}$$

where β is the parameter learned during training. When β = -1, then $\varnothing(x) = x$ and we obtain a version of ReLU called Absolute Value

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

ReLU. When β is very small, the activation function is referred to as leaky ReLU.

tanh is not computationally expensive and is given by:

$$\varnothing(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{6}$$

or

$$\emptyset(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7}$$

For further reading on how to select activation functions, readers are encouraged to read the work of Mhaskar and Micchelli (1994).

Another important concept of NNs is the calculation of the cost function. The perceptron learns by calculating a cost function referred to as Mean Square Error (MSE). There are many other types of cost functions that can be used to determine the output error. Gradient descent (GD) is useful in this light. However, it requires the cost function to be convex resulting in the introduction of Stochastic Gradient Descent (SGD).

These concepts are heavily used in deep NNs for tasks such as language translation, plant disease detection (Dey et al., 2016; Golhani et al., 2018; Sladojevic et al., 2016), facial (expression) recognition (Sun et al., 2017; Fasel and Luettin, 2003), image processing and speech recognition (Sukittanon et al., 2004; Abdel-hamid et al., 2014; Lecun et al., 2015; Chen et al., 2017) among others.

## 3. Convolutional Neural Networks (CNNs)

Before we dive fully into CNNs, let's discuss the properties of images that makes it possible for CNNs to automatically detect features from them. Fig. 1 shows how a computer sees different forms of 2 × 2 images.

Assuming we have a black and white image of 4 pixels (shown as ***a*** in Fig. 1). It can be represented by the computer in a two-dimensional array. Each of the pixels is represented by 8-bits; ranging from 0 to 255 (or $2^8 = 256$) in decimal. The range defines the intensity of the colour such that 0 is complete black, 255 is pure white and in-between is a grayscale range of intensities between black and white. Fig. 1 ***b*** shows a coloured version with Blue (B), Green (G) and Red (R) channels. Features in Fig. 1 ***c***, can be represented as 0 or 1 depending on whether a feature is absent or present respectively.

Convolutional Neural Networks (Wu, 2017) have convolutional layers, pooling layers, fully connected layers and does flattening. During convolutions, a *nxm* kernel ($n > 0$ and $m > 0$) scans the input image to automatically extract features. The filter is smaller than the image and is imposed on the image, then moved across the image based on a stride value to generate a feature map. The larger the stride, the smaller the feature map will be. A stride greater than or equal to 2 results in the convolution losing some features of the image. To maintain as many features as possible, several unique kernels are used to obtain several feature maps. To add non-linearity into the model and reduce model computational complexity, ReLU (Kuo, 2016; He et al., 2015a) is applied just after the convolutional step.

Pooling (or down sampling) is carried out on the feature map to ensure that the CNN recognizes the same object in images of different forms and also to reduce the memory requirements of the model. It introduces spatial invariance in CNNs; which eventually turns out to be one of the major weaknesses of CNNs. Several types of pooling exist: max pooling, min pooling, average pooling (also called sub-sampling), and sum pooling (Scherer et al., 2010). Max pooling preserves the best features since the largest numbers in the feature map indicates locations on the image from where the closest similarity of the image features was taken. The pooled feature map is flattened into a column matrix to serve as input to a NN for further computations. The last part of a CNN after flattening is

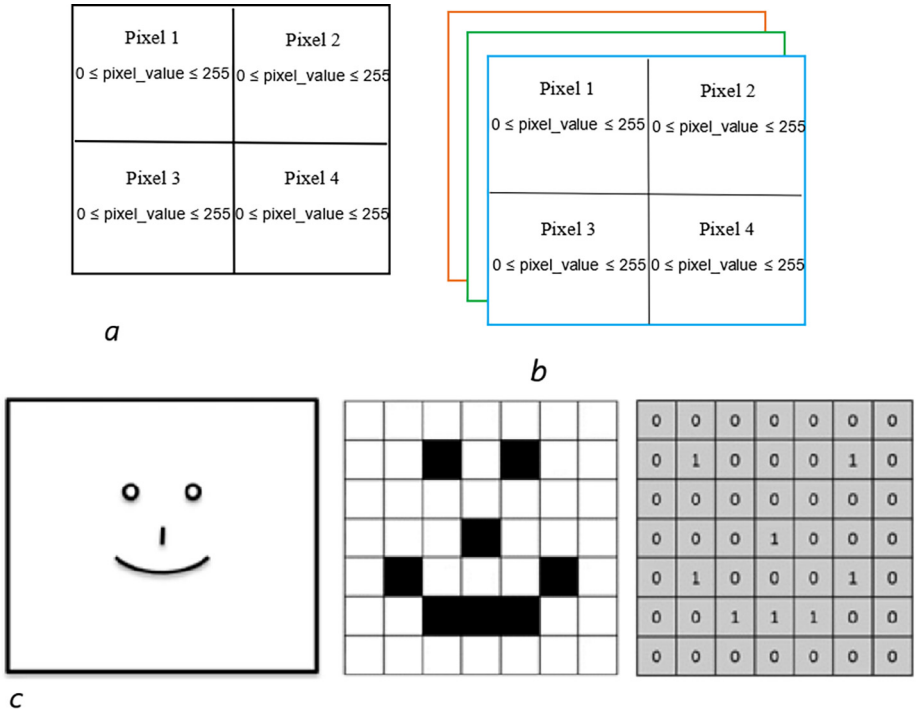


**Fig. 1.** Possible image representations on a computer.

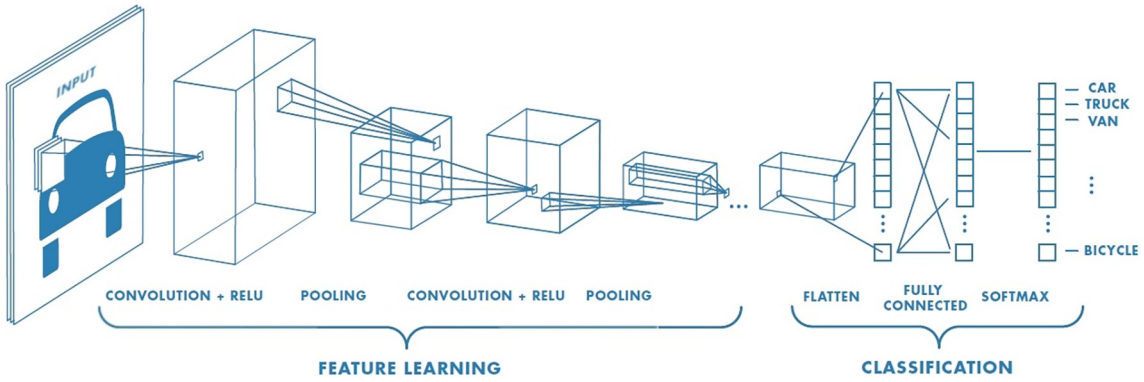


**Fig. 2.** Structure of CNN for Classification (Saha, 2018).

an artificial neural network (ANN) made up of input layers, hidden layers and output layers. The difference between CNNs and ANNs in terms of the fully connected layers is that in CNNs, the hidden layers must be fully connected. However, this is not a strict requirement in ANNs. During backpropagation, both weights and the features detectors are adjusted for better performance. Fig. 2 shows an example structure of a CNN.

In CNNs, the equivalent of the cost function found in ANNs is the Loss function. The most common Loss function for classification tasks is the cross-entropy function (DiPietro, 2016). However, it is only useful for classification. For regression, mean squared error is the best option.

$$L_i = -log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{8}$$

or

$$H(p,q) = -\sum_x p(x)\log q(x) \tag{9}$$

where $q$ = a function of the output from the CNN and $p$ = a function of the target classes.

The SoftMax (Gao and Pavel, 2018) function (Eq. (10), also called Normalized exponential function) is usually introduced at the output layer of CNNs. It is required to restrict the output of the logits between [0,1].

$$f_j(x) = \frac{e^{x_j}}{\sum_k e^{x_k}} \tag{10}$$

### 3.1. Limitations of CNNs

The major challenge of CNNs that led to the introduction of CapsNets is their inability to recognize pose, texture and deformations of an image (Sabour et al., 2017) or parts of the image. In other words, CNNs are invariant as a result of the pooling operation. They are not equivariant and hence lacks equivalence. In addition, the pooling operation in CNN loses some features in the image. They therefore require lots of training data in order to compensate for this loss. They have longer training time compared to CapsNets partially due to the fact that CNNs are deep in depth while CapsNets are deep in width (Shahroudnejad et al., 2018) and at the same time possesses few parameters. CNNs are more prone to adversarial attacks such as pixel perturbations (Su et al., 2019a) resulting in wrong classifications (Su et al., 2019b). As a result of max pooling in CNN, image reconstruction is very demanding as compared to reconstruction in CapsNet.

## 4. Capsule Networks (CapsNet)

Hinton and his colleagues proposed Capsule Networks (Hinton et al., 2011) as an alternative to CNNs. Capsules are equivariant and are made up of a network of neurons which accepts and outputs vectors as opposed to CNNs' scalar values. This property of a capsule allows it to learn the features of an image in addition to its deformations and viewing conditions. In a capsule network, each capsule is made up of a group of neurons with each neuron's output representing a different property of the same feature. This provides the advantage of recognizing the whole entity by first recognizing its parts. The input to a capsule is the output (or features) from a CNN. These features are processed based on the type of capsule employed. A capsule's output is composed of the probability that the feature encoded by the capsule is present and a set of vector values commonly called instantiation parameters. It is the responsibility of the probability of the presence of the capsule's feature to ensure invariance of the network. The instantiation parameters are used to represent equivariance of the network indicating its ability to recognize pose, texture and deformations. Invariance is the property of a model's decision to remain unchanged regardless of any transformations to the inputs. The type called translational invariance is peculiar to CNNs. For instance, if a CNN is to detect a face, irrespective of the position of the eye, it will still detect it as a face. However, equivariance ensures that the spatial location of the features on the face are taken into account. As a result, equivariance does not consider just the presence of an eye in the image, but its location in the image. Equivariance is a desirable property for CapsNets.

Three general methods of capsule implementations exist in literature. These are transforming auto-encoders (Hinton et al., 2011), vector capsules based on dynamic routing (Sabour et al., 2017) and matrix capsules based on expectation-maximization routing (Hinton et al., 2018).

### 4.1. Transforming Auto-encoders

The first capsule network was published with the title Transforming auto-encoders (Hinton et al., 2011). It was built to emphasis the ability of a network to recognize pose. The goal was not to recognize objects in images, but to accept an image and its pose as input and output the same image in the original pose. The output vector of a capsule in this first implementation was made up of output values one of which represented the probability that the feature exists with the rest representing instantiating parameters. Capsules can be arranged in different levels: lower level $l$ called primary capsules and upper level $l + 1$ called secondary capsules. Lower level capsules extracts pose parameters from pixel intensities in order

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

to be able to initiate a part-whole hierarchy. This part-whole hierarchy is an advantage of capsule networks since through recognition of their parts, it can get to recognize the whole entity. To accomplish this, the features represented by the lower level capsules must have the right spatial relationship before they can activate a higher-level capsule at level $l + 1$. For example, let the eyes and mouth be represented by lower level capsules, each of these may make a prediction for the pose of a higher-level capsule representing a face if their predictions agree. To explain the source of the first-level capsules and how an ANN can learn to convert pixel intensities to pose parameters, a 2D image is used as depicted in Fig. 3.

In the simplest case, 2D images and capsules with x and y positions as their only pose outputs are used. Once learned, the network will take an image and the required shifts $\Delta x$ and $\Delta y$ after which it can output the image with the specified shift in pose. The network is made up of separate capsules each of which has its own "recognition units" (red or lower level capsules in Fig. 3) meant to compute three numbers $x$, $y$ and $p$ which the capsules will send to higher level layer capsules. Output $p$ represents the likelihood that the capsule's feature is present in the image. To compute the capsule's contribution to the transformed image, its "generation unit" (green or upper level capsules in Fig. 3) is fed with $x + \Delta x$ and $y + \Delta y$ as input. To prevent inactive capsules from affecting the output, the contribution of the "generation unit" to the capsule's output is multiplied by the probability $p$. A major drawback of this first implementation was the need to supply as input the shifts (pose) of the entity under consideration.

### 4.2. Dynamic routing between capsules

The next modification of capsule networks (Sabour et al., 2017) defined a capsule to be a group of neurons with instantiation parameters being represented by activity vectors and the length of the vector representing the likelihood that the feature exist. The improvement over the previous implementation was the fact that the pose data is not required as an input in this case. The network is made up of Convolutional layer, Primary capsule (PC) layer and Class capsule layer. The Primary capsule layer is the first capsule layer followed by unspecified number of capsule layers until the last capsule layer also called the Class capsule layer. Feature extraction from the image is done by the convolutional layer and the output fed into the Primary capsule layer. Each capsule $i$ (where $1 \leq i \leq N$) in layer $l$ has an activity vector $u_i \in \mathbb{R}$ to encode the spatial information in the form of instantiation parameters. The output vector $u_i$ of the $i^{th}$ lower level capsule is fed into all capsules in the next layer $l + 1$. The $j^{th}$ capsule at layer $l + 1$ will receive $u_i$ and find its product with the corresponding weight matrix $W_{ij}$. The resulting vector $\hat{u}_{j|i}$ is capsule $i$ at level $l$'s transformation of the entity represented by capsule $j$ at level $l + 1$. The prediction vector of a PC, $\hat{u}_{j|i}$ indicates how much the primary capsule $i$ contributes to the class capsule $j$.

$$\hat{u}_{j|i} = W_{ij} u_i \tag{11}$$

A product of the prediction vector and a coupling coefficient representing the agreement between these capsules is carried out to obtain a single primary capsule $i$'s prediction to the class capsule $j$. If the agreement is high, then the two capsules are relevant to each other. As a result, the coupling coefficient will be increased, otherwise it will be reduced. A weighted sum ($s_j$) of all these individual primary capsule predictions for the class capsule j is calculated to obtain the candidates for the squashing function ($v_j$). From Fig. 4, the following can be found:

$$s_j = \sum_{i=1}^{N} c_{ij} \hat{u}_{j|i} \tag{12}$$

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

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{14}$$

The squashing function ensures that the length of the output from the capsule lies between 0 and 1 just like a likelihood. $v_j$ from one capsule layer is routed to the next layer capsule and treated in the same way as discussed. The coupling coefficient $c_{ij}$, ensures that the prediction of $i$ in level $l$ is linked to that of $j$ in layer $l + 1$. During each iteration, $c_{ij}$ is updated by finding the dot product of $\hat{u}_{j|i}$ and $v_j$. Specifically, vector values associated with each capsule can be

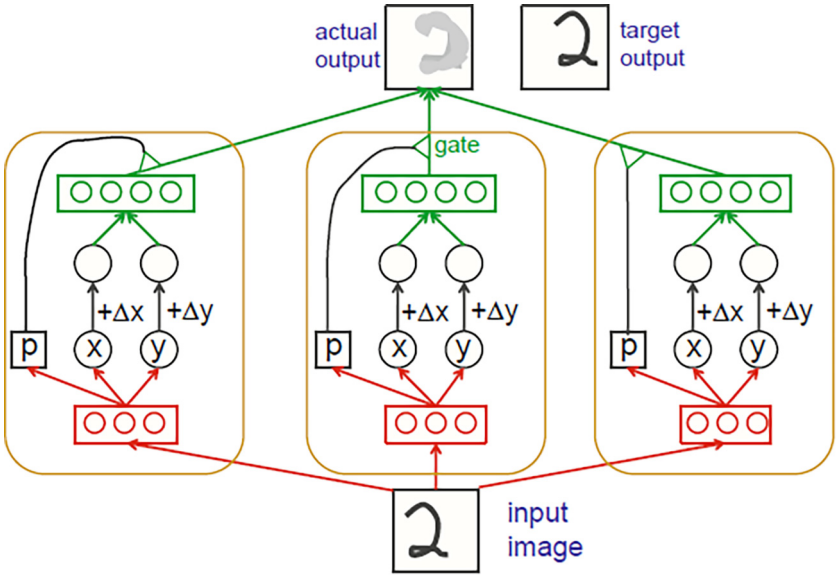


**Fig. 3.** Auto-encoder Capsule structure (Hinton et al., 2011).

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

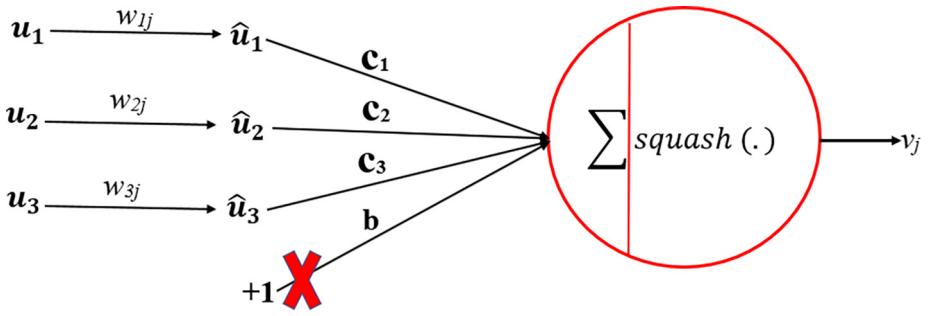


**Fig. 4.** Representation of a capsule in Sabour et al. (2017).

viewed as a segment of two numbers (Shahroudnejad et al., 2018); the probability which represents the existence of the feature that a capsule encapsulates and a set of instantiation parameters which can serve as an explanation of consistency between the layers. So relevant path by agreement stems from the fact that, when lower level capsules agree on a higher-level layer capsule, they "construct a part whole" relationship indicating relevance of the path. This algorithm is called dynamic routing-by-agreement and is shown in Algorithm 1.

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

1. **procedure** ROUTING $(\hat{u}_{j|i}, r, l)$
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 $l$ : $c_i \leftarrow softmax(b_i)$ ▷*softmax computes* $c_{ij}$
5. **for** all capsule $j$ in layer$(l+1)$ : $s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$
6. **for** all capsule $j$ in layer $(l+1)$ : $v_j \leftarrow squash(s_j)$ ▷ *squash computes* $v_j$
7. **for** all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$ : $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}.\ v_j$
8. **return** $v_j$

### 4.3. Matrix capsules with EM routing

As opposed to using vector outputs, Hinton et al., (2018) proposed the representation of a capsule's input and output as matrices. This was necessary in order to reduce the size of the transformation matrices between capsules since with matrices they could be made with $n$ elements instead of $n^2$ when using vectors. The dynamic routing by agreement was also replaced with expectation maximization algorithm (Engelin, 2018). Dynamic routing was a *cosine* between two pose vectors which didn't work to perfection. In addition, the probability of the presence of an entity being represented by a capsule was replaced with a parameter $a$ instead of the length of a vector. This aided in avoiding the squashing function which was considered "not objective and sensible". EM routing algorithm leverages on capsule networks having several layers of capsules to work effectively. Let the set of capsules in the primary layer be denoted by $\Omega_L$, M representing each capsule's pose matrix and $a$ its activation probability. Between capsule $i$ at level $L$ and capsule $j$ at level $L+1$ is a trainable transformation weight matrix $W_{ij}$. EM works by ensuring that the pose matrix of capsule $i$ is transformed by the transformation weight matrix $W_{ij}$ to cast a vote for the pose matrix of capsule $j$ at level $L+1$. The vote is a product of the output matrix $M_i$ and the transformation matrix $W_{ij}$.

$$V_{ij} = M_i W_{ij} \tag{15}$$

The poses and activations of all $L+1$ level layers are found by inputting $V_{ij}$ and $a_i$ into the non-linear EM routing algorithm. Iteratively, EM updates the means, variances, and activation probabilities of layer $L+1$ capsule and also the assignment probabilities between lower and higher-level capsules. Using EM for routing by agreement ensures that each capsule in layer $L+1$ corresponds to a Gaussian distribution and the pose of each active capsule in layer $L$ corresponds to a data point.

## 5. Survey of CapsNets structure and implementations

The basic structure of the first successful capsule network was that of Sabour and Hinton (2017), which consisted of two convolutional layers. Conv1 had 256 channels each made up of $9 \times 9$ filters with a stride of 1 and a ReLU activation function applied to a $28 \times 28 \times 1$ MNIST image.

The second layer was designed as a convolutional capsule layer with $6 \times 6 \times 32$ capsules each outputting an 8D vector. At a stride of 2, each primary capsule has 8 convolutional units operating with a $9 \times 9$ kernel. Non-linearity is obtained by applying a squashing function (as activation function) to create 10, 16D capsules. This layer is also the primary capsule layer receiving features as scaler outputs from Conv1 layer after which all layers will have to deal with 8D vector values. The layer consists of 32 channels each with $6 \times 6$ grid of primary capsules. The third layer (DigitCaps) is a fully connected layer with ten 16D capsules, each receiving input from all capsules from the layer below to perform classification based on 10 classes. The last layer determines the length of each capsule in the previous layer necessary to obtain the probability that the entity is present. Reconstruction of the image is done using the decoder made up of fully connected (FC) layers (as shown in Fig. 5).

### 5.1. Factors affecting CapsNet performance

Properties of a dataset are crucial to the performance of an algorithm. The MNIST dataset is simple with only one channel as compared to more complex datasets having varying colour, size, noise, natural scene background, affine transformations, multiple digits in a single sample, etc. Irrespective of these complexities, CapsNet performs better than CNN (Nair et al., 2018) on a variety of more complex datasets (Gordienko et al., 2018). On difficult dataset such as SVHN (Netzer et al., 2011) and CIFAR10 (Krizhevsky and Hinton, 2009) which has high intra-class variation and background noise, CapsNet's performance is short of the state of the art (Xi et al., 2017; Mukhometzianov and Carrillo, 2018; Yang and Wang, 2019) but still better than CNN (AlexNet) (Mukhometzianov and Carrillo, 2018). However, for the MNIST family of datasets; MNIST (LeCun et al., 1998c) and Fashion-MNIST (Xiao et al., 2017), CapsNet can attain state of art performance since they are comparatively

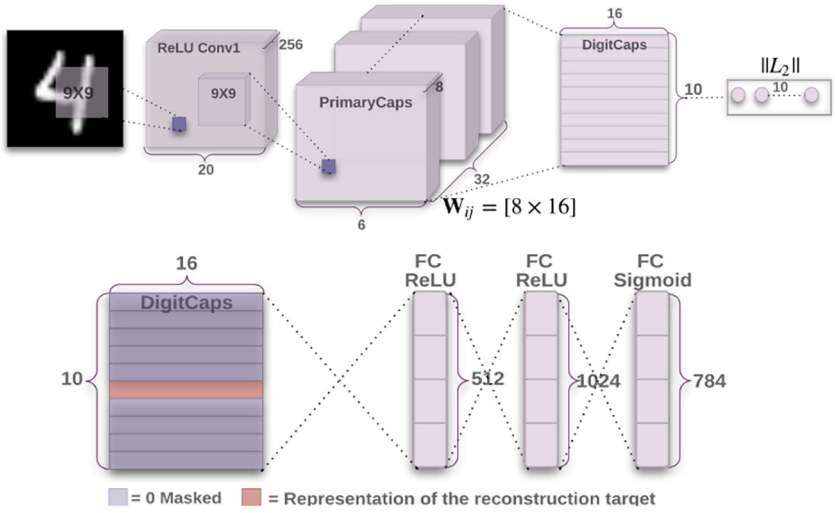


**Fig. 5.** Structure of the capsule network in Sabour et al. (2017).

less complex. Varying hyperparameters such as learning rate, momentum, batch size, drop-out rate and learning rate decay does not significantly affect the performance of CapsNets. However, appropriate routing operations (Lin et al., 2018) and the number of routing iterations of the dynamic routing algorithm does, and it is the most important hyperparameter that significantly affects performance of CapsNets. CapsNets with dynamic routing, CNNs, and CapsNets with EM have been compared for convergence (Chauhan et al., 2018) based on hyperparameters such as Optimizers (*Adam, Adadelta, Adagrad* and *Rmsprop*), number of channels in Conv1 layer, number of capsules in primary capsule layer, number of capsules in convolutional layers following primary layers, number of capsules in class capsule layers and number of routine iterations. Higher values for the above parameters lead to faster training and convergence in Matrix caps with EM routing (Chauhan et al., 2018). It is sensitive to the number of routing iterations and converges faster under Adam or Rmsprop optimization. Parameter sharing (Ren et al., 2019) is used to reduce the number of parameters in a bid to improve the overall generalization ability of a CapsNet. In this case, parameters are shared by sharing a transform matrix (selects features of low-level capsules for each high-level capsule), requiring the need for *M* and not *MxN* transform matrices; where *M* = number of high-level capsules and *N* = number of low-level capsules.

### 5.2. Modifications to the original implementation

Sabour and Hinton (2017) implementation of CapsNet has seen several proposed modifications (Li, 2018; Deliege et al., 2018; Xiang et al., 2018; Amer and Maul, 2019; Neill, 2018; Rawlinson et al., 2018; Bahadori, 2018; Ma and Wu, 2019; Rosario et al., 2019; Zhao et al., 2018b; Li et al., 2018) in a bid to resolve some limitations of the original implementation and improve performance. Critical analysis of the routing-by-agreement shows it does not automatically ensure a higher-level capsule (parent) is coupled with multiple lower level capsules (children) to form a parse tree (Peer et al., 2018). Instead of allowing lower level capsules to send their outputs to all higher-level capsules as is the case in the original routing-by-agreement algorithm, a lower level capsule can select a single parent enabling the network to be deep and resilient to Whitebox adversarial attacks (Peer et al., 2018). In this light, a generative adversarial network (GAN) with a highly performing discriminator made from capsules (Jaiswal et al., 2018) can be helpful in determining whether a given image is natural or artificially created (fake). The better the discriminator (critic) is, the faster the generator learns (Upadhyay and Schrater, 2018). However, modelling the probability distributions of data in deep generative models soon becomes intractable. As a result, CapsNets become better alternatives (Saqur and Vivona, 2018; Sastry, 2018) to CNNs as discriminators in a GAN, ensuring that vital information is not lost through pooling. The dynamic routing algorithm can be seen as an optimization problem that can be formulated as a minimization of an objective function (Wang and Liu, 2018). This modification prevents the activation probability from becoming highly unbalanced as the number of iterations increases. To stabilize the training process, Sabour and Hinton (2017) performed regularization on the weight matrix by using a margin loss. A more general solution is to rescale the weight matrix and ensure that the inner product between the input and the weighted sum ($s_j$) of all the individual primary capsule predictions for capsule $j$ is set below 1 for each iteration (Wang and Liu, 2018). Equal weight initialization routing scheme in the original CapsNet has the tendency to slow convergence and result in poor accuracy. A better option for faster convergence and improved accuracy is to model the initial routing weights as trainable parameters to be trained by backpropagation (Ramasinghe et al., 2018). We can leverage on the realization (and using the fact) that primary capsule predictions are not independent to improve the performance of CapsNets on multi-label classification tasks (Ramasinghe et al., 2018). Focusing on the length of a capsule rather than individual capsule outputs proves to be a better entity detection approach (Zhang et al., 2018b). In this case, the lengths of capsules rank an entity's presence and their orientation instantiates its pose (position, size, orientation), deformation, velocity, albedo, hue, texture and so on.

In (Sabour et al., 2017), the Softmax function is used to normalize routing coefficients representing assignment probabilities between capsules of adjacent layers. However, Softmax constraints the possible set of values the routing coefficients can assume, leading to uniform probabilities after several routing iterations. The loss in performance from this problem can be addressed by using Max-Min function in place of Softmax (Zhao et al., 2019). Max-Min performs scale invariant normalization allowing lower level capsules to take on independent values as opposed to what Softmax does.

To learn discriminative feature maps better for subsequent use by CapsNets, convolutional layers can be replaced with densely connected convolutional layers (Phaye et al., 2018). However,

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

increasing network depth results in vanishing gradient problem. Fortunately, it can be resolved by feeding subsequent layers with signals from previous layers of ResNet (He et al., 2015b; Larsson et al., 2017) or by adding dense connections between every layer in a feedforward manner (Huang et al., 2017) with feature concatenations. It can also be argued that the routing procedure in CapsNets is not properly integrated into the training procedure since the routing procedures are not embedded into the optimization procedures and the choice of the optimal number of routing must be found manually (Chen and Crandall, 2018). This manual selection does not guarantee convergence. Capsules can perform comparably better than CNN Transfer learning models (e.g. InceptionNet and DenseNet) on other datasets aside MNIST or smallNORB (LeCun and Huang, 2004) with additional configurations such as increased number of Conv layers and FC layers (Phong and Ribeiro, 2019). Dynamic routing based CapsNets do not guarantee equivariance or invariance (Lenssen et al., 2018). This is due to the fact that space of a pose is a manifold whiles votes are averaged in a vector space failing to produce equivariance mean estimates in the manifold. In addition, trainable transformation kernels in the capsule layers are defined over local receptive fields found in the spatial vector domain which has receptive field coordinates that are ignorant of the pose. Lenssen et al. (2018) proposed group equivariant capsules layers with pose vectors restricted as elements of a group. Under a general routing by agreement algorithm and under certain conditions, equivariance and invariance can be guaranteed for such groups.

### 5.3. Applications of CapsNets

CapsNets are promising in terms of improving our socio-economic activities as they can be deployed to solve real life problems in astronomy (Katebi et al., 2018), autonomous cars (Kumar et al., 2018), machine translation (Wang et al., 2018a), handwritten and text recognition (Mandal et al., 2019; Zhao et al., 2018a; Kim et al., 2018a) intent detection (Xia et al., 2018; Zhang et al., 2018a), mood and emotion detection (Wang et al., 2018c; Rathnayaka et al., 2018; Guo et al., 2018; Chao et al., 2019) and so on.

The spatio-temporal nature of traffic data expressed in images lends itself to the application of CapsNets for predicting traffic speed (Kim et al., 2018b; Ma et al., 2018) and abnormal driving (Kim and Chi, 2018) on a complex road network. With the widespread use of social media, users' images and video can be forged. Existing detection methods cannot detect many forms of forgery in video and images. The combination of VGG-16 (Simonyan and Zisserman, 2015) and CapsNets have excelled in this area (Nguyen et al., 2018) by significantly reducing overfitting and improving detection accuracy. The challenge with forgery detection methods has been their inability to detect a wide range of attacks such as computer-generated video/image, replay attacks and so on.

For Natural Language Processing (NLP) task (Renkens and Van, 2018; Zhang et al., 2018a) such as knowledge graph (KG) completion and saving model parameters of word embeddings (Ren and Lu, 2018), CapsNets have shown better performance than CNNs (Nguyen et al., 2019; Zhang et al., 2018c). They better predict the validity of a triple not found in a KG as the appropriate triple that should have been there in a (subject, relation, object) relationship triples. Triple modeling problem has important applications in KG completion and search personalization (Nguyen et al., 2019). On the task of relation extraction, performance of existing models depends on the quality of how instances are represented and they also fail to take into account structural relationships such as the position of the instance in the sentence. This can be modelled as a multi-label classification problem executed by a capsule network as multi-binary classification (Deng et al., 2018). In the case of Ren and Lu, (2018), K-means clustering replaces dynamic routing and avoids the use of the squashing function during iteration. Squashing is done after iteration, $b_{ij}$ is replaced by new $b_{ij}$, instead of $b_{ij}$ being replaced by new $b_{ij}$ plus old $b_{ij}$ in (Sabour et al., 2017).

CapsNets have found important applications in health (Pal et al., 2018; Iesmantas and Alzbutas, 2018; Mobiny and Nguyen, 2018; Zhang and Zhao, 2018) and other important areas (Prakash and Gu, 2018; Annabi and Ortiz, 2018; Garg et al., 2017; Duarte et al., 2018). Extraction of semantic relations between entities in health records by means of recurrent CapsNet can effectively be used as a basis to detect health problems (Afshary et al., 2018) and their level of severity (Wang et al., 2018b). The small number of annotated medical images and the class-imbalance (one category of the data far outweighs the other categories) problem are challenges to effective classification in the health domain. Even though they can partially be alleviated by data augmentation, CapsNets perform better in the case of class-imbalance (Jimenez-Sanchez et al., 2018) and in different cases too (Berman, 2019). They perform well on object segmentation (Lalonde and Bagci, 2018); a very useful technique for examining medical images due to its excellent pixel level classification capabilities. CapsNets have also produced good results in other areas such as Hyperspectral Image Classification (Zhang et al., 2019a; Zhang et al., 2019b) where availability of labelled data is a challenge.

Protein family structure classification (Jesus et al., 2018; Fang et al., 2018; Tobing et al., 2018; Zhang, 2019) is another area where CapsNets have been applied. CapsNet's attention to hierarchical relationships gives them an edge over CNNs in graph (Mallea et al., 2019; Verma and Zhang, 2018) or protein structure classification; a process which is necessary to infer the function and aid in drug design.

To enable classifiers learn from few examples, a combination of generative and non-generative capsule network can be injected with missing data (Gritsevskiy and Korablyov, 2018) after reaching its maximum performance to produce good classification results. This is so if the network has more channels (pathways) than data since unseen data will not activate any capsules in existing routing pathways but would prefer an unused routing pathway.

Autonomous cars will benefit hugely from computer vision applications such as CapsNets. Sensor data will need to be processed with flash speeds in order to allow the automobile to take decisions in split seconds. However, automobile sensors such as ultrasonic sensors are not optimized for high performance (Popperl et al., 2019), but to be less expensive in order not to contribute to raising the cost of the automobile. To compensate for the low hardware performance of automotive ultrasonic sensors, CapsNets have been used to obtain outstanding classification performance on ultrasonic data for assistive driving purposes such as object height determination (Popperl et al., 2019). This could be essential during parking.

Another challenging problem solved by CapsNets is (environmental) sound detection (Iqbal et al., 2018; Jain, 2019; Vesperini et al., 2018). As compared to music or speech recognition, no domain specific knowledge is known a priori for this problem. Notwithstanding, CNNs have chopped some successes in this field aside the fact that they have not been able to avoid overfitting.

Often algorithms with high processing demands can be accelerated in hardware. The systolic array (Kung, 1982) has proved to be an excellent architecture for improving the running time of many algorithms including CapsNets (Marchisio et al., 2018b). For example, in hardware the initial steps of the routing by agreement algorithm can be skipped and then directly initialize the coupling

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

coefficients to increase the running time of the algorithm compared to its GPU software counterpart. CapsNets are vulnerable to adversarial attacks (Marchisio et al., 2019). These types of attacks seek to deceive the network to perform misclassification by introducing perturbations into their inputs. This is very important when it comes to critical applications such as self-driving cars where identifying road signs and pedestrians is crucial.

### 5.4. Data sources

Currently, the performance of Capsule Networks varies depending on the dataset on which they are evaluated. In this section, we provide a brief overview of datasets used in the implementations covered in this paper. The type of datasets used with CapsNets can largely be categorized into natural language processing type, audio/video, speech processing and image processing. The advantage of having these datasets is to help researchers spend little time in pre processing data.

MNIST (LeCun et al., 1998c) is the most popular dataset used to train CapsNets. This is partly due to the fact that it is used as a benchmark since the original paper (Sabour et al., 2017) used it for experimental evaluation. It is made up of handwritten digits, a subset of the NIST dataset and contains 60,000 training images plus 10,000 testing digits. Just like MNIST, Fashion-MNIST (Xiao et al., 2017) contains same number of training and test sets. It is a $28 \times 28$ greyscale image from 70,000 fashion products. This dataset is also popular with some of the papers we reviewed. The small-NORB dataset (LeCun and Huang, 2004) is intended for machine vision task that involves the use of 3D objects. It contains images of toys, cars, humans, trucks, etc. Some papers (Hinton et al., 2018) also evaluated their algorithms on this dataset. The CIFAR10 dataset (Krizhevsky and Hinton, 2009) on the other hand is complex than the MNIST family since is a collection of colour images from the internet with high background noise and inter class variability. As a result, the original CapsNet (Sabour et al., 2017) does not perform well on this dataset. However, Rajasegaran et al., (2019) have shown that with a deeper network and 3D convolution based routing algorithm, CapsNet performance on this dataset and its kind can significantly be improved. It is believed that if a model performs well on this dataset, then its performance is superior. Some papers also tested their models on the SVHN dataset (Netzer et al., 2011) which is made up of 600,000 training images of street view house numbers. It is in a sense similar in characteristics to the MNIST dataset. A number of researchers (Jimenez-Sanchez et al., 2018); (Rathnayaka et al., 2018) used relatively less popular datasets. Based on the application of CapsNet at hand, some researchers used their own datasets (Popperl et al., 2019) whiles others made use of local datasets (Wang et al., 2018b). Table A1 in Appendix A provides a list of the most popular datasets used in CapsNet implementations in literature.

### 5.5. Data preprocessing

MNIST is by far the most popular dataset used to train capsule networks (Sabour and Hinton, 2017; Peer et al., 2018). Data augmentation is a technique used to increase the volume of data by performing transformations on existing samples necessary for machine learning classification tasks. A type of augmentation called deformation (Wong et al., 2016) can be carried out on MNIST images by shifting them up to some number of pixels in any required direction (Sabour et al., 2017). This method produces different versions of the image with the advantage of improving the performance of a classifier. The good thing about capsule networks is that, they don't need extensive data augmentation like CNNs (Nair et al., 2018) since they perform creditable well on smaller datasets.

### 5.6. Methods of visualization

A common visualization technique is to show the reconstructed images to enable visual comparison with the original ones (Sabour et al., 2017). These reconstructions can either be done at the end of selected epochs (Nair et al., 2018), at the end of the training or during testing. A plot of accuracy or loss or mean average precision (*maP*) against the number of epochs is also common in literature. There are also rare cases of precision against recall plots especially when using capsules for Natural Language Processing (NLP) tasks and loss against number of batches. Heat maps are also common in literature. Statistical forms of visualization such as histograms, curve plots, tables and their likes are powerful tools that are sometimes used to report the performance of a model. The objectives of these visualizations are twofold: first to help us use our human intuition based on vision in evaluating how good the performance of the algorithm is. Secondly, to provide us with some level of understanding of the internal operations of the model.

### 5.7. Performance evaluation methods

The correct selection of performance evaluation method is critical since a model can perform well on one evaluation metric and perform badly on another metric (Liu et al., 2014). By far the most common performance evaluation metric for classifiers is accuracy. Others include root mean square error (RMS), mean absolute error (MAE), F-measure, area under the receiver operating curve (AUC), kappa statistic and area under the precision-recall curve (AUPRC). To determine the percentage of correct classified instances of a test class, accuracy is the preferred choice with precision and recall as options for data mining tasks. The above metrics can be grouped as probability, threshold or rank metrics. When selecting an evaluation metric for a model, one must consider whether the classes are balanced or otherwise. For balanced training set, more than one metric should be used for evaluation whereas one metric will suffice for an imbalanced training set (Liu et al., 2014). Threshold measures such as kappa statistic, F-measure and accuracy do not consider whether the predicted value is the true value or not. They only measure how close the predicted value is to a threshold value. ROC (AUC), PR (AUPRC), AUC and AUPRC curves are referred to as rank metrics since they measure the degree to which the model ranks the positive instances over the negative ones (Zhou and Liu, 2018). Finally, RMS and MAE are probability based and reports on the deviation between predicted and true values.

From Liu et al. (2014) the following expressions can be stated:

$$Acc = \frac{TP + TN}{TS} \tag{16}$$

where TS = total number of negative and positive samples with TN (true negative) and TP (true positive) depicted in the confusion matrix shown in Table 1.

$$RMSE = \sqrt{\frac{1}{M}\sum_{1}^{M}(y - \bar{y})^2} \tag{17}$$

and the absolute version of the RMSE

**Table 1**
Structure of Confusion Matrix.

| Actual label | Predicted label | |
|---|---|---|
| | + | – |
| + | TP | FN |
| – | FP | TN |

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

$$MAE = \frac{1}{M}\sum_{1}^{M}|(y-\bar{y})| \tag{18}$$

The challenges with these metrics are that: first they are not that different from each other since they are all derived from the same confusion matrix and secondly, interpretation of performance becomes a problem when two metrics evaluating a classifier produce conflicting results.

Reconstruction loss is a popular metric used to evaluate the performance of standard autoencoders. It penalizes the network for producing outputs that are different from the actual outputs. It is basically the loss function; either cross entropy or mean square error. This penalization concept encourages the network to be sensitive to inputs and together with a regularizer to discourage overfitting. When reconstruction loss is used without regularizer, the modeler has to implement a good dropout scheme in order to reduce memorization or overfitting. Reconstruction loss encourages every entity in a capsule to encode its instantiation parameter to obtain a final activity vector that will be used to reconstruct the input image in a FC layer. It will minimize the sum of squared differences between outputs of the network and pixel intensities such that it must not be allowed to dominate the margin loss and cause overfitting during training.

Margin loss on the other hand is computed to allow for multiple entities to exist in a top level capsule. Since the length of a vector represents the presence of a capsule's entity, top level capsules with their entities present have long vectors. Margin loss can be computed for each capsule allowing the loss for absent entities to be downweighted. This ensures that multiple entities are accommodated and learning prevented from shrinking activity vector lengths for entities. Sabour et al. (2017) utilized the loss function in equation (19) as follows:

$$L_e = D_e \max(0, n^+ = ||v_e||)^2 + \lambda(1-D_e)max(0, ||v_e|| - n^-)^2 \tag{19}$$

where $D_e \max(0, n^+ = ||v_e||)^2$represents the case when the digit of the class in question is present with $D_e = 1$ and $\lambda(1-D_e)max(0, ||v_e|| - n^-)^2$ represents the opposite case with $D_e = 0$. $n^+ = 0.9$ and $n^- = 0.1$ are set to avoid the length increasing or reducing respectively the loss function unreasonably. $\lambda$ was set to 0.5 to control down weighting of initial weights for absent classes from affecting the decisions of the model.

The aggregate of the losses of all entities forms what is called the total loss. Total loss is also common in literature as a performance evaluation metric for capsules.

For now, there is no consensus on which metric is better for evaluating capsules. The choice depends on the researcher and the objective of the design.

### 5.8. Discussion

Capsules are a new sensation in Deep Learning and are producing amazing results compared to CNNs and traditional NNs. A CapsNet is activated after comparing multiple incoming pose vectors (Hinton et al., 2018) whereas in NNs activation is achieved by comparing a learned weight vector to a single incoming activity vector. This increases the certainty that the CapsNet will recognize the correct pose of the object.

CNN classifiers are not robust against adversarial attacks. However, CapsNets are shown to be relatively resilient against such attacks (Hinton et al., 2018). They are also resilient to affine transformations to the input data (Sabour et al., 2017). In other words, changes made to the input with the desire to trick the network must not succeed in fooling the network to make wrong predictions. This is very crucial when the network is used for critical tasks such as in autonomous self-driving cars where life depends on the accuracy of model outputs.

CapsNets have so far shown promising results in terms of reduced training time (Phong and Ribeiro, 2019). They have minimal number of parameters compared to CNNs as a result of the fact that connections in capsules exist between group of neurons instead of individual neurons. These suggest that, training CapsNets online would be tolerable even though this important area is yet to be fully explored. Currently, online training of DL models is prohibitive in terms of computational time and resources.

The deficiency of CNNs in identifying deformation in objects due to the pooling operation is corrected in capsules allowing them to identify objects from whichever angle they are viewed in addition to being able to classify unseen objects better than CNNs. These are huge accomplishments in computer vision necessary for the realization of safety critical tasks such as face identification for security purposes.

It is known that the inner workings of deep learning models are at times seen as a black-boxes by the modeler. Inputs are fed into the box and some results produced. However, it is very challenging to analyze and understand the internal operations of such models; an ultimate requirement for model performance, resulting in the practice of hyperparameter tuning in the bid to obtain better results for a given dataset. The introduction of capsules has lessened this challenge as the internal operations of capsules are explainable (Shahroudnejad et al., 2018) than CNNs.

CapsNets ability to generalize well on smaller datasets makes them conducive for use in a wide variety of areas as opposed to CNNs which must be trained and deployed on huge datasets. Although one can compensate for possible loss of performance resulting from lack of data by using transfer learning and data augmentation, it will make a huge difference in terms of performance and user's confidence if a model is assured to train and test on the little domain data available.

Even though capsules are shown to outperform CNNs, there are areas where their operations are not optimal. These grey areas have received the attention of researchers as they require further research to improve the capsule even further. For instance, the absence of pooling operation means that the capsule will try to account for every aspect of the input image including background noise. Removal of this extra work of modelling irrelevant parts (Nair et al., 2018) of the input should lessen the running time further and improve overall performance.

The capsule does not perform in the same way across different datasets. Datasets such as CIFAR10 (Krizhevsky and Hinton, 2009) and ImageNet (Nair et al., 2018) still remain a challenge to CapsNets. Unfortunately, these images and their kind may be the data required as inputs to models required to perform critical tasks.

Since its inception, researchers have proposed several modifications (Li, 2018; Deliege et al., 2018; Xiang et al., 2018; Amer and Maul, 2019; Neill, 2018; Rawlinson et al., 2018; Bahadori, 2018; Ma and Wu, 2019; Rosario et al., 2019; Zhao et al., 2018b) showing that we are yet to arrive at an optimal version. Unfortunately, research efforts for now are concentrating on the work in (Sabour et al., 2017) instead of its improved version in (Hinton et al., 2018) which has seen only few modifications.

For dynamic algorithms such as those used in capsules, the difficulty in parallelizing them can be compensated for by implementing them as hardware systolic arrays. However, the number of capsules and parameters cannot be accommodated on FPGA platforms in terms of hardware resource usage, leading to the implementation of only activation functions in hardware. This area is very promising requiring much more research efforts.

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

Other difficulties such as the strict concept of entities for capsules may render the concept unsuitable for other applications that are not computer vision related. The vector/matrix output of capsules presents a challenge to existing loss functions and also, the presence of more than one instance of an entity in a location in the input image gives some difficulty for CapsNets to distinguish. Even though their training time is better than CNNs, it is still not acceptable for time critical operations and highly unsuitable for online training. Research is currently ongoing in this area (Marchisio et al., 2018a) and is showing promising results. These limitations are all open questions that the research community can seek to address in the near future.

We believe that there is the need to thoroughly interrogate CapsNets with EM routing (Hinton et al., 2018) as it was an improvement over the one with dynamic routing algorithm (Sabour et al., 2017). Only after this is done that, we can realize the full potential of capsules.

## 6. Conclusion

Processing unstructured data for machine vision purposes remains one of the most important tasks in AI. The introduction of DL has eliminated the challenging feature engineering task which eventually lead to high dimensionality. DL models such as CNNs have performed well in this field, however they require lots of data and computational power to execute. Capsules were therefore introduced to eliminate the challenges faced by CNNs and have so far performed creditably well. However, the relative newness of the field requires further understanding before its full potential can be realized. This paper therefore examined the models in literature that are significantly impactful in the field. It sets the tone by discussing and reviewing the implementations of Capsule Networks in literature. The paper surveyed the state of the art in Capsule networks and shed more light on existing architectures and implementations. Though the concept is powerful, there is more to learn and improve. This is evident in the fact that most of the existing implementations do not use the matrix capsule algorithm which is an improvement over the capsules with dynamic routing algorithm. The computer vision community's hope will be to leverage on the successes and failures of CapsNets to build robust machine vision algorithms through further research in the field.

## Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix

Table A1.

**Table A1**
Summary of significant Capsule implementations in literature.

| Ref | Problem Definition/Application area | CapsNet algorithm/Architecture and parameters | Dataset | Additional Hyperparameter settings | Performance Metric | Compared with |
|---|---|---|---|---|---|---|
| Nair et al., 2018 | Test CapsNets on datasets that are like MNIST but harder in a specific way. Explore the internal embedding space and sources of error for CapsNets. | Original CapsNet: $28 \times 28 \times 1$ MNIST image. Conv1 with $9 \times 9$, 256 kernels, stride = 1, ReLU. Conv2 with $9 \times 9$ kernels, stride = 2, 8D. Capsules. $6 \times 6 \times 32 \times 8$. $10 \times 16$D output. Decoder part. | MNIST (LeCun et al., 1998c) Fashion-MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky and Hinton, 2009). | Classes depended on Dataset in use | Accuracy | AlexNet (Nair et al., 2018) |
| Kim et al., 2018b | Predict traffic flow in complex road network. | Conv1: $3 \times 3$, 32 channels with ReLU. Conv2: $3 \times 3$, 32 channels with ReLU.Primary Capsule: $3 \times 3$, 128 channels, 8D with ReLU. TrafficCaps: 16D capsules. | SETA EU project ("SETA Project," 2016) dataset. | Learning rate = 0.0005, exponential decay rate = 0.9999. | Mean Relative Error (MRE), Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). | CNN (Ma et al., 2017) |
| Nguyen et al., 2019 | Knowledge graph (KG) completion and Search Personalization | Conv layer: five $1 \times 3$ filters with ReLU, Caps1 layer: 4 caps each with 5 neurons, Caps2 layer: 1 caps with 2 neurons. | WN18RR (Dettmers et al., 2017), FB15k-237 (Toutanova and Chen, 2015), SEARCH17 (Vu et al., 2017). | Adam initial learning rate in $\{1e^{-5}, 5e^{-5}, 1e^{-4}\}$, # of filters N in {50; 100; 200; 400}, batch size = 128, # of neurons in the capsule in the second capsule layer = 10 (d = 10), # of iterations in the routing algorithm in {1; 3; 5; 7}. | mean rank (MR), mean reciprocal rank (MRR) and Hits@10 | TransE (Bordes et al., 2013), ConvKB (Nguyen et al., 2017). |

(continued on next page)

**Table A1** (*continued*)

| Ref | Problem Definition/Application area | CapsNet algorithm/Architecture and parameters | Dataset | Additional Hyperparameter settings | Performance Metric | Compared with |
|---|---|---|---|---|---|---|
| Ramasinghe et al., 2018 | Inability of original CapsNet to perform well on multi-label classifications. Slow convergence and poor accuracy due to equal weight initialization in original CapsNets | Conv1 layer, primary capsule layer, Conditional Random Field (CRF) module: for obtaining semantic relationship among object classes, correlation module: to prioritize predictions for choosing class capsules and Decision capsules. | ADE20K (Zhou et al., 2017) | | Mean average precision (mAP) | CapsNet (Sabour et al., 2017) |
| Phong and Ribeiro, 2019 | CapsNet's difficulty to perform well on other datasets aside MNIST (LeCun et al., 1998c) and SmallNORB (LeCun and Huang, 2004). | 256 14 × 14 filters applied on Conv layer of 64 × 64, 32 × 32, and 16 × 16 sign images. 16D vector P. Caps Layer of 14 × 14 × 16. 29 32D vector Secondary Caps. 26 classes. 2 FC layers. | American Sign Language (ASL) datasets (https://www.kaggle.com/grassknoted/asl-alphabet) | | Speed of classification, Accuuracy | Inception, DenseNet V201, NASNet, MobileNet. |
| Deng et al., 2018 | Single entity-relation extraction and multiple entity pairs relation extraction problem | Bi-LSTM (Zhou et al., 2016) layer, Primary Caps, Class Caps, FC Layer. | NYT dataset (Riedel et al., 2014), Wikidata dataset (Sorokin and Gurevych, 2017) | Learning rate 0.001, batch size 128, LSTMs' unit size 300, dropout rate 0.5, number of routing iteration 3. Caps dimension d=8. | Precision based on Recall. | CNNs (dos Santos et al., 2015) |
| Wang et al., 2018b | Entity-relation extraction, classification of severity of coronary artery disease from medical text. | Embeddings layer, LSTM Layer, P. Caps, C. Caps, FC | Coronary arteriography texts collected from Shanghai Shuguang Hospital. | | precision (P), recall (R) and F1-score (F1) | CNN + MaxPooling (Nguyen and Grishman, 2015), BiLSTM + MaxPooling (Zhang and Wang, 2015), BiLSTM + Attention (Zhou et al., 2016), CRNN + MaxPooling and CRNN + Attention (Raj, 2017). |
| Afshary et al., 2018 | Classification of brain tumor type | Similar architecture as in (Sabour et al., 2017) except the use of 64 feature maps instead of 256 in Conv layer. | 1. Datasets used in (Cheng et al., 2015). 2. Data sets used in (Cheng et al., 2016). | | Prediction Accuracy | CNN (Paul, 2016) |
| Saqur and Vivona, 2018 | 3D image generation and quest to find better discriminators for GANs. | Dynamic routing: Conv1 layer: 32 × 32 input, 9 × 9 kernel of stride 1 or 2, 256 channels each made of 24 × 24. P. Caps Layer: 32 channels each 8 × 8 × 8, 8D. Output Caps: 16 × 1 or one 16D. | MNIST and SmallNORB. | 128 batch size, LR:0.002, epochs:20, Leaky ReL:0.2, adam optim: beta1:0.5, weight init:normal: mean −0, std:0.02. | Training loss and visualization of outputs. | DCGAN (Radford et al., 2015) |
| Jimenez-Sanchez et al., 2018 | Class-imbalance and small dataset size limitations on classification of medical images. | Dynamic routing-by-agreement. Conv1&2:9 × 9, 256 channels. Caps1:1152caps 8D. Caps2: Nk ccaps 16D, where $N_k$ = .FC1:1 × 1, 512 channels and FC2: 1 × 1, 1024 channels | Histological images of the fi rst auxiliary dataset from the TUPAC16 challenge (http://tupac.tue-image.nl/, accessed: 2019-06-19), DIARETDB1 (Kauppi et al., 2007), MNIST and Fashion-MNIST (Xiao et al., 2017) | | F1-score | LeNet (LeCun et al., 1998a) and (Sabour et al., 2017). |
| Jesus et al., 2018 | Protein family structure prediction and classification. | Dynamic routing-by-agreement. Conv1: Primary voxel; 512 × 512. Conv2: 64 channels 9 × 9 each. P. Caps:7 × 256 channels. Caps Voxel: 2 × 64 channels. | KRAS (McCormick, 2015), PSI-BLAST, HRAS. | Loss: categorical hinge and logcosh. Min (filters = 64, kernel size = 5 primarycap dim = 16, voxelcap dim = 32). RMSProp (Learning Rate:0.001, Rho: 0.9, Epsilon: None, Decay: 0.0). | Accuracy | CNN (Ragoza et al., 2017) |
| Popperl et al., 2019 | Non-image classification for Self-driving cars. | Dynamic routing-by-agreement. 14 × 14 image. Conv1:256 kernels each 6 × 6, ReLU. P.Caps: 6 × 6 × 8, 32 channels. Digit Caps: 16 × 4. | Height from artmobile ultrasonic sensor. | ReLU. | Accuracy | CNN-based approaches: |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mobiny and Nguyen, 2018 | Lung Cancer Screening | Dynamic routing by agreement. Encoder: 32 × 32 input to 24 × 24, 256 channels. Primary Caps 8 × 8, 256D vectors. NoduleCaps: each 64 × 256 × 16. Decoder: 8 × 8 × 16, deconv-16 × 16, deconv-32 × 32. | 226 unique Computed Tomography (CT) Chest scans (with or without contrast). | ReLU. | | Precision, recall, error rate | AlexNet and ResNet | |
| Jain, 2019 | Audio Classification | Dynamic routing-by-agreement. Input Layer: Time Step (TS)x59. Batch Norm: Tx59. BiLSTM1: Tx64. BiLSTM2:Tx32. Dropout: Tx32. Caps Layer: Number of Classes (NC)x16. Length of Vec: NCx1. | IEMOCAP (Busso et al., 2008), RAVDESS, VCTK, Urban8K, FSDD, Multi-MUSAN. | | | Accuracy, visualization. | CNN, LSTM (Salamon and Bello, 2016) | |
| Rathnayaka et al., 2018 | Predict emotion of tweets with no emotion terms. | Dynamic routing-by-agreement. Embedding Layer, Bi-GRU layer, Capsule Net, Flattening, Softmax layer. | Data from WASSA 2018 Implicit Emotion Shared Task (Clercq et al., 2018). | Word2Vec (Mikolov et al., 2013) embeddings of 300 dimensions. Gated Recurrent Unit layer: 128 cells. 16 caps each with an output size of 32 and 5 routing iterations. Spatial dropout of 0.3 is applied to the embeddings and dropout of 0.25 is applied to the Capsule network. Gaussian noise of 0.1 is added to both the embedding layer and the Capsule network. | | Precision, Recall, F1-Score. | GRU + Hierarchical Attention. GRU + CNN. GRU + CapsNet. | |
| Zhang and Zhao, 2018 | Cervical image classification | Dynamic routing. Conv1: 256 9 × 9 kernels. Reduces image size to 24 × 24 from 32 × 32. Primary Caps layer:5 × 5 kernel takes 24 × 24 image and reduces it to 10 × 10. 16D vector output produced (ie.16 output neurons grouped into capsules)-10 × 10 × 16 × 16 total vectors produced. DigiCaps layer: 16 × 32 transformation matrix and 16D converted to 32D capsules. | Clinical images of Cervical cancer | Stride 1 in conv1 and 2 in conv2. Padding 0. Used $c_{ij}$ of 1600 × 3. | | Confusion matrix. Accuracy. Margin loss. Reconstruction loss. Total Loss. | ResNet. Inception v2 | |
| Kim and Chi, 2018 | Detection of centerline invasion in abnormal driving. | Dynamic routing. Conv layer: 128 × 12 image into 9 × 9 kernel. Caps Layer: 32 ch, 8D caps. Class caps layer: 32 × 56 × 56 or 100,352 vector output. | YouTube vehicle accident videos | 9 × 9 kernel & stride 1 in conv layer. Conv2/Caps layer: 9 × 9 kernel with stride of 2. | | Accuracy. Confusion Matrix. Model Loss. Validation Accuracy. Train Accuracy. | AlexNet. VGGNet. DenseNet. GoogLeNet. | |
| Zhang, 2019 | Classification of 2D Hela datasets (Protein classification). | Dynamic routing. 382 × 382. Conv1 layer: 250 × 20 × 20 kernels. Image to 9 × 9. P. Caps Layer: 6 × 6 × 8 × 32 vector. DigiCaps layer: 16 × 10D vector. | 2D Hela dataset. | Input Image size: 382 × 382. | | Accuracy. Margin loss. Reconstruction loss. Total Loss. | Five versions of CapsNet differentiated by the number of routing iterations compared. | |
| Chao et al., 2019 | Emotion Recognition | Dynamic routing. Conv1 layer: [16 × 16] × 256 ch. P. Caps layer: [7 × 7] × 256D vector. Class Cap layer: 2 49 vectors each 32D. | DEAP (Koelstra et al., 2011) | Input image size of 18 × 18. | | Accuracy. | Five versions of CapsNets with different parameters compared. | |

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

## References

Abdel-hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., Yu, D., 2014. Convolutional Neural Networks for Speech Recognition. IEEE/ACM Trans. AUDIO, SPEECH. Lang. Process. 22, 1533–1545.

Afshary, P., Mohammadiy, A., Plataniotis, K.N., 2018. Brain tumor type classification via capsule networks. arXiv: 1802.10200v2 [cs.CV].

Amer, M., Maul, T., 2019. Path capsule networks. arXiv: 1902.03760v1 [cs.LG].

Annabi, L., Ortiz, M.G., 2018. State representation learning with recurrent capsule networks. arXiv: 1812.11202v4 [cs.LG].

Bahadori, M.T., 2018. Spectral Capsule Networks. In: ICLR 2018, pp. 1–5.

Berman, D.S., 2019. DGA CapsNet: 1D Application of Capsule Networks to DGA Detection. Information 10, 1–15.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data. Adv. Neural Inf. Process. Syst. 1–9. https://doi.org/10.1109/NAVITEC.2014.7045139.

Busso, C., Bulut, M., Lee, C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J.N., Lee, S., Narayanan, S.S., 2008. IEMOCAP: Interactive emotional dyadic motion capture database. Kluwer Acad. Publ., pp. 1–30.

Chao, H., Dong, L., Liu, Y., Lu, B., 2019. Emotion Recognition from Multiband EEG Signals Using CapsNet. Sensors 19, 1–16.

Chauhan, A., Babu, M., Kandru, N., Lokegaonkar, S., 2018. Empirical Study on convergence of Capsule Networks with various hyperparameters. http://people.cs.vt.edu/~bhuang/courses/opt18/projects/capsule.pdf.

Chen, Q., Xu, J., Koltun, V., 2017. Fast Image Processing with Fully-Convolutional Networks. In: CoRR Abs/1709.00643 (2017), pp. 2497–2506. ArXiv: 1709.00643.

Chen, Z., Crandall, D., 2018. Generalized Capsule Networks with Trainable Routing Procedure. arXiv1808.08692v1 [cs.CV], 1–4.

Cheng, J., Huang, W., Cao, S., Yang, R., Yang, W., Yun, Z., Wang, Z., Feng, Q., 2015. Enhanced performance of brain tumor classification via tumor region augmentation and partition. PLoS One 10, 1–13. https://doi.org/10.1371/journal.pone.0140381.

Cheng, J., Yang, W., Huang, M., Huang, W., Jiang, J., Zhou, Y., Yang, R., Zhao, J., Feng, Y., Feng, Q., Chen, W., 2016. Retrieval of brain tumors by adaptive spatial pooling and fisher vector representation. PLoS One 11, 1–15. https://doi.org/10.1371/journal.pone.0157112.

Clercq, D.O., Klinger, R., Mohammad, S.M., Balahur, A., 2018. IEST: WASSA-2018 Implicit Emotions Shared Task. arXiv:1809.01083v2 [cs.CL], 1.

Deliege, A., Cioppa, A., Droogenbroeck, M Van, 2018. HitNet : a neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules. arXiv:1806.06519v1 [cs.CV], 1–19.

Deng, S., Sun, Z., Chen, X., Zhang, W., Chen, H., 2018. Attention-based capsule networks with dynamic routing for relation extraction. extracarXiv: 1812.11321v1 [cs.IR].

Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S., 2017. Convolutional 2D knowledge graph embeddings. https://arxiv.org/abs/1707.01476.

Dey, A.K., Sharma, M., Meshram, M.R., 2016. Image processing based leaf rot disease, detection of betel vine. Procedia Comput. Sci. 85, 748–754. https://doi.org/10.1016/j.procs.2016.05.262.

DiPietro, R., 2016. A Friendly Introduction to Cross-Entropy Loss [WWW Document]. https://doi.org/10.1109/MDSO.2007.4370101.

Dombetzki, L.A., 2018. An overview over Capsule Networks. In: Seminars FI/IITM SS 18, Network Architectures and Services, pp. 89–95. https://doi.org/10.2313/NET-2018-11-1.

Duarte, K., Rawat, Y.S., Shah, M., 2018. VideoCapsuleNet: A Simplified Network for Action Detection. arXiv:1805.08162v1 [cs.CV].

Engelbrecht, A.P., 2007. Computational Intelligence-An Introduction. Second ed. John Wiley & Sons Ltd, Pretoria.

Engelin, M., 2018. CapsNet Comprehension of Objects in Different Rotational Views A comparative study between capsule and convolutional networks. STOCKHOLM, SWEDEN.

Fang, C., Shang, Y., Xu, D., 2018. Improving protein gamma-turn inception capsule networks prediction using inception capsule networks. Nature.

Fasel, B., Luettin, J., 2003. Automatic facial expression analysis: a survey. Pattern Recognit. 36, 259–275. https://doi.org/10.1016/S0031-3203(02)00052-3.

Gao, B., Pavel, L., 2018. On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning. arXiv:1704.00805v4 [math.OC], 1–10.

Garg, S., Alexander, J., Kothari, T., 2017. Using Capsule Networks with Thermometer Encoding to Defend Against Adversarial Attacks. Proceedings of the CS229 Final Project Session, Stanford, CA.

Glorot, X., Bordes, A., Bengio, Y., 2011. Deep Sparse Recti er Neural Networks. In: 14th International Con- Ference on Arti Cial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA, pp. 315–323. https://doi.org/10.1.1.208.6449.

Golhani, K., Balasundram, S.K., Vadamalai, G., Pradhan, B., 2018. A review of neural networks in plant disease detection using hyperspectral data. Inf. Process. Agric. 5, 354–371. https://doi.org/10.1016/j.inpa.2018.05.002.

Gordienko, N., Gordienko, Y., Taran, V., Gang, P., Kochura, Y., Stirenko, S., 2018. Capsule Deep Neural Network for Recognition of Historical Graffiti Handwriting. IEEE 3.

Gritsevskiy, A., Korablyov, M., 2018. Capsule networks for low-data transfer learning. arXiv1804.10172v1 [cs.CV], 1–11.

Guo, J., Fang, F., Wang, W., Ren, F., 2018. EEG Emotion Recognition Based on Granger Causality and CapsNet Neural Network. In: 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). IEEE, pp. 47–52.

He, K., Zhang, X., Ren, S., Sun, J., 2015a. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. IEEE Int. Conf. Comput. Vis. 2015 Inter, pp. 1026–1034. https://doi.org/10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., Sun, J., 2015b. Deep Residual Learning for Image Recognition. Microsoft Res. arXiv1512.03385v1 [cs.CV].

Hinton, G., Sabour, S., Frosst, N., 2018. Matrix capsules with em routing. In: ICLR, pp. 1–15. https://doi.org/10.2514/6.2003-4412.

Hinton, G.E., Krizhevsky, A., Wang, S.D., 2011. Transforming auto-encoders. In: ICANN 2011, Part I, Lecture Notes in Computer Science 6791. Springer-Verlag Berlin Heidelberg, pp. 44–51. https://doi.org/10.1007/978-3-642-21735-7_6.

Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: Proc. – 30th IEEE Conf. Comput. Vis. Pattern Recognit. 2017-Janua, pp. 2261–2269. https://doi.org/10.1109/CVPR.2017.243.

Iesmantas, T., Alzbutas, R., 2018. Convolutional capsule network for classification of breast cancer histology images. https://arxiv.org/abs/1804.08376, 1–8.

Iqbal, T., Xu, Y., Kong, Q., Wang, W., 2018. Capsule routing for sound event detection. arXiv:1806.04699v1 [cs.SD].

Jain, R., 2019. Improving performance and inference on audio classification tasks using capsule networks. caparXiv: 1902.05069v1 [cs.SD].

Jaiswal, A., Abdalmageed, W., Wu, Y., Natarajan, P., 2018. CapsuleGAN: Generative Adversarial Capsule Network. arXiv:1802.06167v7 [stat.ML], 1–10.

Jesus, D.A.R., Julian, C., Wilson, R., Silvia, C., 2018. Capsule Networks for Protein Structure Classification and Prediction. arXiv:1808.07475v1 [cs.LG].

Jimenez-Sanchez, A., Albarqouni, S., Mateus, D., 2018. Capsule Networks against Medical Imaging Data Challenges. arXiv:1807.07559v1 [cs.CV].

Katebi, R., Zhou, Y., Chornock, R., Bunescu, R., 2018. Galaxy morphology prediction using capsule networks. arXiv:1809.08377v1 [astro-ph.IM] 7, 1–7.

Kauppi, T., Kalesnykiene, V., Kamarainen, J.-K., Lensu, L., Sorri, I., Voutilainen, R., Uusitalo, H., Kälviäinen, H., Pietil̈, J., 2007. DIARETDB1-standard diabetic retinopathy database [WWW Document]. http://www.it.lut.fi/project/imageret/diaretdb1/doc/diaretdb1_techreport_v_1_1.pdf. URL http://www.bmva.org/bmvc/2007/papers/paper-60.html.

Kim, J., Jang, S., Choi, S., 2018a. Text Classification using Capsules. arXiv:1808.03976v2 [cs.CL], 1–9.

Kim, M., Chi, S., 2018. Detection of centerline crossing in abnormal driving using CapsNet. J. Supercomput. https://doi.org/10.1007/s11227-018-2459-6.

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

Koelstra, S., Lee, J.-S., Pun, T., 2011. DEAP: A Database for Emotion Analysis using Physiological Signals. IEEE TRANS. Affect. Comput., 1–15

Krizhevsky, A., Hinton, G., 2009. Learning Multiple Layers of Features from Tiny Images [WWW Document]. URL http://www.cs.toronto.edu/~kriz/cifar.html. (Accessed 6.15.19).

Kumar, A.D., Karthika, R., Parameswaran, L., 2018. Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks. https://arxiv.org/abs/1805.04424.

Kung, H.T., 1982. Why Systolic Architectures [WWW Document]. URL www.eecs.harvard.edu/~htk/.../1982-kung-why-systolic-architecture.pdf. (Accessed 6.6.19).

Kuo, C.C.J., 2016. Understanding convolutional neural networks with a mathematical model. J. Vis. Commun. Image Represent. 41, 406–413. https://doi.org/10.1016/j.jvcir.2016.11.003.

Lalonde, R., Bagci, U., 2018. Capsules for Object Segmentation. arXiv1804.04241v1 [stat.ML], 1–9.

Larsson, G., Maire, M., Shakhnarovich, G., 2017. FractalNet: Ultra-Deep Neural Networks without Residuals. arXiv1605.07648v4 [cs.CV], 1–11.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 52. https://doi.org/10.1038/nature14539.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L. D., 2014. Backpropagation applied to handwritten zip code recognition. Sci. Signal. 7, 541–551. https://doi.org/10.1126/scisignal.2005580.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998a. Gradient-based learning applied to document recognition. In: Proc. OF THE IEEE., pp. 1–46.

LeCun, Y., Bottou, L., Orr, G.B., Muller, K.-R., 1998b. Efficient BackProp [WWW Document]. Neural Networks: tricks of the trade, Springer. URL http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf. (Accessed 6.17.19).

LeCun, Y., Cortes, C., Burges, C.J.C., 1998c. MNIST [WWW Document]. URL http://yann.lecun.com/exdb/mnist/. (Accessed 6.15.19).

LeCun, Y., Huang, F.J., 2004. Learning methods for generic object recognition with invariance to pose and lighting. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Lenssen, J.E., Fey, M., Libuschewski, P., 2018. Group Equivariant Capsule Networks. arXiv:1806.05086v2 [cs.CV].

Li, H., 2018. Cognitive Consistency Routing Algorithm of. arXiv:1808.09062v3 [cs.AI].

Li, S., Ren, X., Yang, L., 2018. Fully CapsNet for Semantic Segmentation. In: PRCV 2018, pp. 392–403. LNCS 11257, https://doi.org/10.1007/978-3-030-03335-4.

Lin, A., Li, J., Ma, Z., 2018. On Learning and Learned Representation with Dynamic Routing in Capsule Networks. arXiv:1810.04041v1 [cs.CV].

Liu, Y., Zhou, Y., Wen, S., Tang, C., 2014. A strategy on selecting performance metrics for classifier evaluation. Int. J. Mob. Comput. Multimed. Commun. 6, 20–35. https://doi.org/10.4018/IJMCMC.2014100102.

Ma, D., Wu, X., 2019. TCDCaps: Visual Tracking via Cascaded Dense Capsules. arXiv:1902.10054v1 [cs.CV].

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Yong, Wang, Yunpeng, 2017. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. Sensors (Switzerland) 17, 1–16. https://doi.org/10.3390/s17040818.

Ma, X., Li, Y., Cui, Z., Wang, Y., 2018. Forecasting Transportation Network Speed Using Deep Capsule Networks with Nested LSTM Models. https://arxiv.org/pdf/1811.04745.

Mallea, M.D.G., Meltzer, P., Bentley, P.J., 2019. Capsule Neural Networks for Graph Classification using Explicit Tensorial Graph Representations. arXiv:1902.08399v1 [cs.LG].

Mandal, B., Dubey, S., Ghosh, S., RiteshSarkhel, Das, N., 2019. Handwritten Indic Character Recognition using Capsule Networks. arXiv:1901.00166v1 [cs.CV].

Marchisio, A., Bussolino, B., Colucci, A., Hanif, M.A., Martina, M., Masera, G., Shafique, M., 2018a. X-TrainCaps: Accelerated Training of Capsule Nets through Lightweight Software Optimizations. ArXiv Prepr. arXiv1905.10142, 1–9.

Marchisio, A., Hanif, M.A., Shafique, M., 2018b. CapsAcc: An Efficient Hardware Accelerator for CapsuleNets with Data Reuse. arXiv:1811.08932v1 [cs.DC].

Marchisio, A., Nanfa, G., Khalid, F., Hanif, M.A., Martina, M., Shafique, M., 2019. CapsAttacks: Robust and Imperceptible Adversarial Attacks on Capsule Networks. arXiv:1901.09878v1 [cs.LG].

McCormick, F., 2015. KRAS as a therapeutic target. Clin. Cancer Res. 21, 1797–1801. https://doi.org/10.1158/1078-0432.CCR-14-2662.

Mhaskar, H.N., Micchelli, C.A., 1994. How to Choose an Activation Function [WWW Document]. URL https://papers.nips.cc/paper/874-how-to-choose-an-activation-function.pdf. (Accessed 7.6.19).

Mikolov, T., Corrado, G., Chen, K., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781v3 [cs.CL], 1–12.

Mobiny, A., Nguyen, H. Van, 2018. Fast CapsNet for Lung Cancer Screening. arXiv:1806.07416v1 [cs.CV].

Mukhometzianov, R., Carrillo, J., 2018. CapsNet comparative performance evaluation for image classification. https://arxiv.org/abs/1805.11195 1–14.

Nair, P., Doshi, R., Keselj, S., 2018. Pushing the Limits of Capsule Networks, https://arxiv.org/abs/1804.04241, 1–16.

Neill, J.O., 2018. Siamese Capsule Networks. arXiv:1805.07242v1 [stat.ML], 1–10.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., 2011. Reading digits in natural images with unsupervised feature learning. In: In NIPS Workshop on Deep Learning and Unsupervised Feature Learning, pp. 1–9. https://doi.org/10.2118/18761-MS.

Nguyen, Dai Quoc, Nguyen, T.D., Nguyen, Dat Quoc, Phung, D., 2017. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. https://arxiv.org/abs/1712.02121, https://doi.org/10.18653/v1/N18-2053.

Nguyen, Dai Quoc, Vu, T., Nguyen, T.D., Nguyen, Dat Quoc, Phung, D., 2019. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. arXiv:1808.04122v3 [cs.CL].

Nguyen, H.H., Yamagishi, J., Echizen, I., 2018. Capsule-Forensics: Using Capsule Networks to Detect Forged Images and Videos. arXiv:1810.11215v1 [cs.CV].

Nguyen, T.H., Grishman, R., 2015. Relation Extraction: Perspective from Convolutional Neural Networks Thien. In: NAACL-HLT 2015, pp. 27–69. https://doi.org/10.1002/9780470742778.ch3.

Nielson, M., 2019. Neural Networks and Deep Learning [WWW Document]. URL http://neuralnetworksanddeeplearning.com/chap2.html. (Accessed 6.17.19).

Pal, A., Chaturvedi, A., Garain, U., Chandra, A., Chatterjee, R., Senapati, S., 2018. CapsDeMM: Capsule network for Detection of Munro's Microabscess in skin biopsy images. arXiv:1808.06428v2 [cs.CV].

Pattanayak, S., 2017. Pro Deep Learning with TensorFlow-A Mathematical Approach to Advanced Artificial Intelligence in Python. Apress, Bangalore, Karnataka, India.

Paul, J., 2016. Deep learning for brain tumor classification (Masters Thesis). https://doi.org/10.1117/12.2254195.

Peer, D., Stabinger, S., Rodriguez-sanchez, A., 2018. Training Deep Capsule Networks. arXiv:1812.09707v1 [cs.LG].

Phaye, S.S.R., Sikka, A., Dhall, A., Bathula, D., 2018. Dense and Diverse Capsule Networks: Making the Capsules Learn Better. arXiv 1805.04001v1 [cs.CV], 1–11.

Phong, N.H., Ribeiro, B., 2019. Advanced Capsule Networks via Context Awareness. arXiv1903.07497v2 [cs.LG], 1–12.

Popperl, M., Gulagundi, R., Yogamani, S., Milz, S., 2019. Capsule Neural Network based Height Classification using Low-Cost Automotive Ultrasonic Sensors. arXiv1902.09839v1 [cs.CV].

Prakash, S., Gu, G., 2018. Simultaneous Localization And Mapping with depth Prediction using Capsule Networks for UAVs. arXiv:1808.05336v1 [cs.RO].

Radford, A., Metz, L., Chintala, S., 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: ICLR 2016, pp. 1–16.

Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J., Koes, D.R., 2017. Protein-Ligand Scoring with Convolutional Neural Networks. J. Chem. Inf. Model. 57, 942–957. https://doi.org/10.1021/acs.jcim.6b00740.

Raj, D., Sahu, S., Anand, A., 2017. Learning local and global contexts using a convolutional recurrent network model for relation classification in biomedical text. In: 21st Conference on Computational Natural Language Learning (CoNLL 2017), pp. 311–321. https://doi.org/10.18653/v1/k17-1032.

Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S., Rodrigo, R., 2019. DeepCaps: Going Deeper with Capsule Networks. ArXiv Prepr. arXiv1904.09546, 1–9.

Ramasinghe, S., Athuraliya, C.D., Khan, S.H., 2018. A Context-aware Capsule Network for Multi-label Classi cation. arXiv 1810.06231v2 [cs.CV], 1–9.

Rathnayaka, P., Abeysinghe, S., Samarajeewa, C., Manchanayake, I., Walpola, M., 2018. Sentylic at IEST 2018: Gated Recurrent Neural Network and Capsule Network Based Approach for Implicit Emotion Detection. represenarXiv: 1809.01452v1 [cs.CL].

Rawlinson, D., Ahmed, A., Kowadlo, G., 2018. Sparse Unsupervised Capsules Generalize Better. arXiv:1804.06094v1 [cs.CV].

Ren, H., Lu, H., 2018. Compositional Coding Capsule Network with K-Means Routing for Text Classification. arXiv:1810.09177v3 [cs.LG].

Ren, H., Su, J., Lu, H., 2019. Evaluating Generalization Ability of Convolutional Neural Networks and Capsule Networks for Image Classification via Top-2 Classification. arXiv:1901.10112v2 [cs.CV].

Renkens, V., Van, H., 2018. Capsule Networks for Low Resource Spoken Language Understanding. arXiv:1805.02922v1 [eess.AS].

Riedel, S., Yao, L., Mccallum, A., 2014. Modeling Relations and Their Mentions without Labeled Text. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pp. 148–163. https://doi.org/10.1007/978-3-662-44845-8.

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

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., 2015. ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. https://doi.org/10.1007/s11263-015-0816-y.

Sabour, S., Frosst, N., Hinton, G.E., 2017. Dynamic Routing Between Capsules. 31st Conference on Neural Information Processing Systems.

Saha, S., 2018. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [WWW Document]. https://doi.org/10.1080/09640560500294277.

Salamon, J., Bello, J.P., 2016. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. IEEE Signal Process. Lett., 1–5

dos Santos, C.N., Xiang, B., Zhou, B., 2015. Classifying Relations by Ranking with Convolutional Neural Networks. arXiv:1504.06580v2 [cs.CL].

Saqur, R., Vivona, S., 2018. CapsGAN: Using Dynamic Routing for Generative Adversarial Networks. arXiv:1806.03968v1 [cs.CV].

Sastry, S., 2018. Recurrent Capsule Network for Image Generation. vixra.org/pdf/1804.0112v1.pdf, pp. 1–9.

Scherer, D., Müller, A., Behnke, S., 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In: 20th International Conference on Artificial Neural Networks (ICANN), Thessaloniki, Greece, pp. 92–101. https://doi.org/10.1007/978-3-642-15825-4_10.

SETA Project [WWW Document], 2016. A ubiquitous data Serv. Ecosyst. better Metrop. Mobil. URL http://setamobility.weebly.com/. (Accessed 6.17.19).

Shahroudnejad, A., Mohammadi, A., Plataniotis, K.N., 2018. Improved Explainability of Capsule Networks: Relevance Path by Agreement. arXiv:1802.10204v1 [cs.CV].

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: ICLR 2015 Conference Proceedings, pp. 1–14. ArXiv:1409.1556v6 [Cs.CV].

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., Stefanovic, D., 2016. Deep neural networks based recognition of plant diseases by leaf image classification. Comput. Intell. Neurosci. 2016. https://doi.org/10.1155/2016/3289801.

Sorokin, D., Gurevych, I., 2017. Context-Aware Representations for Knowledge Base Relation Extraction. In: 2017 Conf. Empir. Methods Nat. Lang. Process, pp. 1784–1789. https://doi.org/10.2116/bunsekikagaku.24.244.

Su, J., Vargas, D.V., Sakurai, K., 2019a. One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. 1–15. https://doi.org/10.1109/tevc.2019.2890858.

Su, J., Vargas, D.V., Sakurai, K., 2019b. Attacking convolutional neural network using differential evolution. IPSJ Trans. Comput. Vis. Appl. 11, 1–16. https://doi.org/10.1186/s41074-019-0053-3.

Sukittanon, S., Surendran, A.C., Platt, J.C., Burges, C.J.C., 2004. Convolutional networks for speech detection. 8th International Conference on Spoken Language Processing.

Sun, W., Zhao, H., Jin, Z., 2017. A facial expression recognition method based on ensemble of 3D convolutional neural networks. Neural Comput. Appl. 1–18. https://doi.org/10.1007/s00521-017-3230-2.

Tobing, E., Murtaza, A., Han, K., Yi, M.Y., 2018. EP-CapsNet : Extending Capsule Network with Inception Module for Electrophoresis Binary Classification. In: 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering (BIBE). IEEE, pp. 327–333. https://doi.org/10.1109/BIBE.2018.00071.

Toutanova, Kristina, Chen, D., 2015. Observed versus latent features for knowledge base and text inference. In: 3rd Workshop on Continuous Vector Space Models and Their Compositionality (CVSC), pp. 57–66. https://doi.org/10.18653/v1/w15-4007.

Upadhyay, Y., Schrater, P., 2018. Generative Adversarial Network Architectures For Image Synthesis Using Capsule Networks. arXiv:1806.03796v4 [cs.CV].

Verma, S., Zhang, Z., 2018. Graph Capsule Convolutional Neural Networks. Joint ICML and IJCAI Workshop on Computational Biology, Stockholm, Sweden.

Vesperini, F., Gabrielli, L., Principi, E., Squartini, S., 2018. Polyphonic Sound Event Detection by using Capsule Neural Networks. J. Sel. Top. SIGNAL Process. X, 1–13.

Vu, T., Nguyen, D.Q., Johnson, M., Song, D., Willis, A., 2017. Search personalization with embeddings. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) https//arxiv.org/abs/1612.03597, 10193 LNCS, 598–604. https://doi.org/10.1007/978-3-319-56608-5_54.

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014

Wang, D., Liu, Q., 2018. An optimization view on dynamic routing between capsules. In: 6th International Conference on Learning Representations (ICLR 2018) Call for Workshop Papers, pp. 1–4.

Wang, M., Xie, J., Tan, Z., Su, J., Xiong, D., Bian, C., 2018a. Towards Linear Time Neural Machine Translation with Capsule Networks. arXiv:1811.00287v1 [cs.CL].

Wang, Q., Qiu, J., Zhou, Y., Ruan, T., Gao, D., Gao, J., 2018b. Automatic Severity Classification of Coronary Artery Disease via Recurrent Capsule Network. arXiv:1807.06718v2 [cs.CL].

Wang, Y., Sun, A., Han, J., Liu, Y., Zhu, X., 2018c. Sentiment Analysis by Capsules. In: International World Wide Web Conference Committee, pp. 1165–1174.

Wong, S.C., Gatt, A., Stamatescu, V., McDonnell, M.D., 2016. Understanding Data Augmentation for Classification: When to Warp? 2016 International Conference on Digital Image Computing: Techniques and Applications. https://doi.org/10.1109/DICTA.2016.7797091.

Wu, J., 2017. Introduction to convolutional neural networks. Natl. Key Lab Nov. Softw. Technol. 1–31. https://doi.org/10.1007/978-3-642-28661-2-5.

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

Xia, C., Zhang, C., Yan, X., Chang, Y., Yu, P.S., 2018. Zero-shot User Intent Detection via Capsule Neural Networks. metharXiv 1809.00385v1 [cs.CL].

Xiang, C., Zhang, L., Zou, W., Tang, Y., Xu, C., 2018. MS-CapsNet: A Novel Multi-Scale Capsule Network. IEEE Signal Process. Lett. 1. https://doi.org/10.1109/LSP.2018.2873892.

Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747v2 [cs.LG], 1–6.

Yang, Z., Wang, X., 2019. Reducing the dilution: An analysis of the information sensitiveness of capsule network with a practical solution. caparXiv 1903.10588v2 [cs.LG].

Zhang, C., Li, Y., Du, N., Fan, W., Yu, P.S., 2018a. Joint Slot Filling and Intent Detection via Capsule Neural Networks. arXiv:1812.09471v1 [cs.CL].

Zhang, D., Wang, D., 2015. Relation Classification via Recurrent Neural Network. modarXiv 1508.01006v2 [cs.CL].

Zhang, H., Meng, L., Wei, X., Tang, Xiaoliang, Tang, Xuan, Wang, X., Jin, B., Yao, W., 2019a. 1D-Convolutional Capsule Network for Hyperspectral Image Classification. arXiv:1903.09834v1 [cs.CV], 1–13.

Zhang, L., Edraki, M., Qi, G., 2018b. CapProNet : Deep Feature Learning via Orthogonal Projections onto Capsule Subspaces. arXiv:1805.07621v2 [cs.CV].

Zhang, W., Tang, P., Zhao, L., 2019b. Remote Sensing Image Scene Classification Using CNN-CapsNet. Remote Sens. 11, 1–22. https://doi.org/10.3390/rs11050494.

Zhang, X., 2019. Fluorescence microscopy image classification of 2D HeLa cells based on the CapsNet neural network. Med. Biol. Eng. Comput. https://doi.org/10.1007/s11517-018-01946-z.

Zhang, X., Li, P., Jia, W., Zhao, H., 2018c. Multi-labeled Relation Extraction with Attentive Capsule Network. arXiv:1811.04354v1 [cs.CL].

Zhang, X., Zhao, S.-G., 2018. Cervical image classification based on image segmentation preprocessing and a CapsNet network model. Int. J. Imaging Syst. Technol. 1–10. https://doi.org/10.1002/ima.22291.

Zhao, W., Ye, J., Yang, M., Lei, Z., Zhang, S., Zhao, Z., 2018a. Investigating Capsule Networks with Dynamic Routing for Text Classification. arXiv1804.00538v4 [cs.CL].

Zhao, Y., Birdal, T., Deng, H., Tombari, F., 2018b. 3D Point-Capsule Networks. arXiv:1812.10775v1 [cs.CV].

Zhao, Z., Kleinhans, A., Sandhu, G., Patel, I., Unnikrishnan, K.P., 2019. Capsule Networks with Max-Min Normalization. arXiv:1903.09662v1 [cs.CV], 1–15.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A., 2017. Scene parsing through ADE20K dataset [WWW Document]. Scene Parsing through ADE20K Dataset-. Proc. IEEE Conf. Comput. Vis. Pattern Recognition. https://doi.org/10.1109/CVPR.2017.544.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., Xu, B., 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In: 54th Annu. Meet. Assoc. Comput. Linguist, pp. 207–212. https://doi.org/10.18653/v1/p16-2034.

Zhou, Y., Liu, Y., 2018. Correlation analysis of performance metrics for classifier. Decis. Mak. Soft Comput. www.worldscientific.com 487–492.

Please cite this article as: M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty et al., Capsule Networks – A survey, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2019.09.014