Self-Attention Agreement Among Capsules

Rita Pucci
Università degli Studi di Udine
Udine UD, Italia
rita.pucci@uniud.it

Christian Micheloni
Università degli Studi di Udine
Udine UD, Italia
christian.micheloni@uniud.it

Niki Martinel
Università degli Studi di Udine
Udine UD, Italia
niki.martinel@uniud.it

Abstract

At the state of the art, Capsule Networks (CapsNets) have shown to be a promising alternative to Convolutional Neural Networks (CNNs) in many computer vision tasks, due to their ability to encode object viewpoint variations. Network capsules provide maps of votes that focus on entities presence in the image and their pose. Each map is the point of view of a given capsule. To compute such votes, CapsNets rely on the routing-by-agreement mechanism. This computationally costly iterative algorithm selects the most appropriate parent capsule to have nodes in a parse tree for all the active capsules but this behaviour is not ensured by the routing, hence it possibly causes vanishing weights during training. We hypothesise that an attention-like mechanism will help capsules to select the predominant regions among the maps to focus on, hence introducing a more reliable way of learning the agreement between the capsules in a single pass. We propose the Attention Agreement Capsule Networks (AA-Caps) architecture that builds upon CapsNet by introducing a self-attention layer to suppress irrelevant capsule votes thus keeping only the ones that are useful for capsules agreements on a specific entity. The generated capsule attention map is then assigned to classification layer responsible of emitting the predicted image class. The proposed AA-Caps model has been evaluated on five benchmark datasets to validate its ability in dealing with the diverse and complex data that CapsNet often fails with. The achieved results demonstrate that AA-Caps outperforms existing methods without the need of more complex architectures or model ensembles.

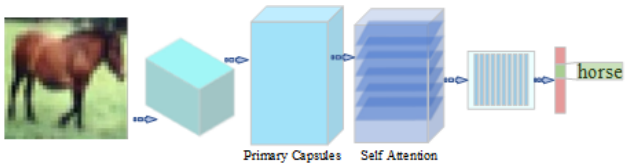

Figure 1. Summarised overview of the proposed AA-Caps. The proposed structure investigate the application of self-attention [49] layer to replace the routing-by-agreement mechanism [45].

1. Introduction

Smart cameras are heavily relying on machine learning tools providing the possibility of transferring the visual recognition abilities to an edge device. Most of such devices leverage Deep Neural Networks (DNNs) [33, 55, 32, 26, 46] architectures due to their excellent results on different challenges. The most famous DNN architecture for visual-related tasks is the Convolutional Neural Network (CNN) [27]. CNNs are the dominating approach on visual recognition tasks with continuous improvements of the state-of-the-art results in many research fields.

Under the ”smart cameras umbrella”, we can find many applications of such methods in: (i) the medical field [5, 14, 15, 17] where they helped to diagnose organs diseases [4, 2, 23, 20]; (ii) bio-informatics for protein discovery [3, 21, 38]; and (iii) conservation ecology research projects [53, 16, 48, 35, 10, 6]. Despite the success of CNNs in different fields, such an architecture still presents many limitations. Among those, there are two well-known issues, one regarding the robustness of CNNs to affine transformations (such as shift in object locations within an image), the other on its limitations with respect to the spatial relationships between object features. The former issue is generally alleviated through data augmentation (but the test

set data can present some unpredictable shifts). The latter is due to the pooling operations maintaining the presence information but ignoring the positional one [47].

Capsule network (CapsNet) [45], a new model based on the capsules concept [18, 50, 24], has been recently proposed to address these two main CNNs limitations. In a nutshell, CapsNets store information at a vector level (through a group of neurons) not via scalars, like traditional DNNs do. Such groups of neurons are known as *capsules*. CapsNets also introduce the concept of routing-by-agreement, where each capsule considers the information extracted from all the previous (parent) capsules to establish if there is an an agreement among them for a particular feature in the image.

Through such a concept, CapsNets [45] achieve state-of-the-art performance on "simple" datasets (e.g., MNIST) but fail with "more complex" data (e.g., CIFAR10). To improve performance on such complex datasets, the seminal work [45] exploited an ensemble procedure, thus requiring additional training time and learnable parameters. Additional works were then proposed with the aim to improve CapsNet and reach CNN performance with a single model. Most of such efforts focused on the expansion of the feature extraction layers [37, 40, 43] (i.e., inclusion of additional convolutional layers before capsules) and on designing complex structures of capsule layers [44, 11, 19, 42, 41]. All such works introduce significant computational burden that limits the exploitation of capsule networks within smart cameras. This motivates a study on improving the CapsNet model performance on complex inputs with special attention to the computational operations.

Differently from such works, our aim is to maintain unaltered the original CapsNet architecture by introducing only a limited set of operations among capsules. We propose to achieve such an objective by focusing our attention on the interaction between capsules rather than introducing additional computational layers or different sets of capsules. Capsules interact through the routing-by-agreement iterative procedure that at state of the art demonstrate to be effective in identification of entities as parts of objects, and their viewpoints and finally in the classification of images [45, 19]. This routing, due to its iterative nature, has an increasing computational costs, moreover prior work [39] demonstrates that the routing-by-agreement mechanisms may fail to construct a parse tree between each set of entities and object capsules, this is mainly due to the inability of the network to learn routing weights through back-propagation.

Self-attention mechanisms [49] have the ability to access to the entire input to selectively pick out specific elements of it to produce the output. We hypothesise that such an ability can be leveraged to build a novel agreement process among capsules. The intuition is that, using an attention mechanism, a child capsule can build its internal feature representation by selecting the most important features from the set of its parents inputs in a single forward pass without the need of iterative procedures.

Thus, this paper proposes an attention-driven mechanism as an alternative to the CapsNet routing-by-agreement procedure with the aim of improving its performance without the need of resorting to model ensembles or complex alteration of the structure. The motivations that promote this work: i) improve the performance of original CapsNet, in terms of better accuracy and faster convergence; ii) achieve better performance on complicated datasets such as CIFAR-10, SmallNorb, AwA2; iii) we reduce the model complexity for baseline application (MNIST). We introduce the following contributions: (i) we remove the routing-by-agreement and we achieve a non-iterative model based on capsules; (ii) we perform digit capsules votes analysis through attention layers; (iii) we show performance improvements over basic CapsNet architectures without the need of ensembles.

To validate the proposed approach, we have conducted multiple experiments on five benchmark datasets having increasing complexity. Results demonstrate that the proposed approach outperform the original CapsNet in accuracy on CIFA10, SmallNORB, Animals with Attributes benchmark datasets, and `AA-Caps` performs in line with the state of the art with MNIST and SVHN datasets.

2. Related works

2.1. Capsule Network

The idea of capsules was presented in [45] where capsules were proposed as an approach to capture the presence of an entity. A capsule is a set of neurons that collectively produce an activity vector with one element for each neuron to hold that neuron's instantiation value. The CapsNet model presented in [45] consists of layers of capsules applied to the task of classification of images. Authors formalise a training procedure based on routing-by-agreement where each capsule makes prediction over the parent capsule and computes a coupling coefficient between the actual capsule and the parent capsule outputs. Capsule outputs are vectors indicating the presence of an entity within the processed input, while their norms represent the confidence of the indication. Many works at state of the art, present the application of CapsNet structure to cope with image classification using hyperspectral images [12] and medical images [1, 34, 22]. Works focus on the empower of the CapsNet structure to improve the performance of the model with complex datasets (coloured and real images), in DenseNet [40] authors propose to deep the structure adding dense block to the model, the new layers provide an high level of feature extraction while reducing the model interpretability. The idea of stressing out the CapsNet structure to better understand the performances is also presented

in [36, 52, 37]. In [40], authors propose two frameworks where the standard convolutional layers is replaced with densely connected convolutions. This helps in extracting feature maps by different layers and it gives a representative base for the primary capsules entities recognition. In our work, we focus on the routing-by-agreement procedure and we propose `AA-Caps` , a modified CapsNet, based on self-attention mechanism as a valid alternative to routing-by-agreement algorithm. The attention mechanism learns from the votes computed by the primary capsule layer, a compatibility function between low-level features and high-level features. Where the compatibility function is a feed forward neural network with a softmax activation function, the transformers presented in [49] were the first implementation of attention function. We think that the CapsNet needs a smarter and lighter mechanism for training, and, instead of modifying the architecture by a deeper structure, we propose to remove the iterative mechanism of learning with a non-iterative one. This modification makes the training procedure faster compared to the original routing-by-agreement, the learning proves to be more interpretable, and we think that the self-attention mechanism is a validate candidate for capsule training due to the similarity with the routing-by-agreement idea. A detailed discussion on this point will be presented in Section 3.3.

2.2. Transformers

Transformers, or so-called self-attention networks, are a family of deep neural network architectures, where self-attention layers are stacked on top of each other to learn contextualised representations for input tokens via multiple transformations. These models have been able to achieve state of the art on many vision and Natural Language Process tasks. There are many implementation details about the transformer. Still, at a high level, transformer is an encoder-decoder architecture, where each of encoder and decoder blocks consists of a stack of transformer layers. In each layer, we learn to (re-)calculate a representation per input token. This representation is computed by attending to the representations of all tokens from the previous layer. The main component of transformer is the self-attention, and one essential property of it is using a multi-headed attention mechanism. We mainly focus on this component and dig into some of its details as we get back to it when comparing capsule nets with transformers.The primary motivation of using multi-head attention is to get the chance of exploring multiple representation subspaces since each attention head gets a different projection of the representations. In an ideal case, each head would learn to attend to different parts of the input by taking a different aspect into account, and it is shown that in practice, different attention heads compute different attention distributions. Having multiple attention heads in transformers can be considered similar to having multiple filters in CNNs.

The application of self-attention networks for images recognition is an open challenge at the state of the art in computer vision. In [54], authors show the use of self-attention as a basic building block for image recognition models. In [9], authors trained a sequence transformer to auto-regressively predict pixels, achieving results comparable to CNNs on image classification tasks. ViT is a vision transformer model, which applies a pure transformer directly to sequences of image patches proposed in [13], it has achieved state of the art performance on multiple image recognition benchmarks. In addition to basic image classification, transformer has been utilised to address a variety of other computer vision problems, including object detection [7, 56], semantic segmentation [8], image processing [13], and video understanding [30]. In this paper we apply self-attention layer to collaborate with capsules in extracting entities features. The multi-heads are addressed to deal with a dedicated capsule and the mechanism is trained to create the connections between lower layer and upper layer in capsules. There is not a direct manipulation of images trough transformers, but we propose transformers as mechanism to aggregate and interpret entities parts features in order to recognise classes.

3. Proposed model

In this section, we begin by presenting the state-of-the-art mechanism that are the main subjects of our research. In particular, we provide a brief description of the routing-by-agreement mechanism proposed in [45], and an introduction to the self-attention layer [49] functionalities. Then, in Sections 3.3 and 3.4, we describe the similarities between the two mechanisms to provide the theoretical foundations behind the proposed model.

3.1. Capsules and Routing-by-Agreement

A capsule consists of a group of neurons that depicts the properties of various entities present in an image. Such properties are captured while training, and they are automatically selected as more representative for the recognition of the entities. There could be various properties like position, size, texture.

Capsule output is a vector that is obtained by the activation value of each neuron that composes the capsule. This is the activation vector and it is used in the routing-by-agreement procedure where the activation vector at lower layer is sent to all capsules at a higher layer. The activation vector of each capsule of the higher layer makes predictions for the parent capsule. The routing mechanism compares the higher layer prediction with the activation vector of the parent capsule and if the activation vectors matches, there is an agreement and the coupling coefficient between the two capsules is increased.

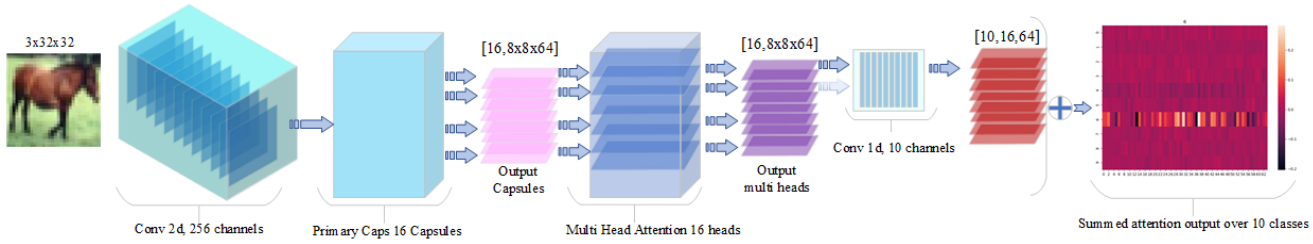

Figure 2. Overview of the proposed `AA-Caps` architecture. The neural processing pipeline begins with a `Conv2d` designated to extract features from the input image that are then fed to the `Primary-Caps` (composed of capsules) trained to identify entities. These are then processed by a `Self-Attention` layer to build the connection among the views of the heads of the layer. A final `Conv1d` maps the computed attention over capsules to the number of classes.

In other words, the capsule in the routing-by-agreement process increases the connection with all the capsules at the lower layer that contribute to the prediction of an entity.

More formally, let $u_i \in \mathbb{R}^{d_u}$ be an output of a capsule i, and j be the parent capsule, the prediction is calculated as:

$$\hat{u}_{i|j} = \mathbf{W}_{ij} u_i \tag{1}$$

where the $\mathbf{W}_{ij} \in \mathbb{R}^{d_u \times d_{\hat{u}}}$ is a weighted matrix applied to compute the affine transformation that given a activation vector u_i provides a prediction vector $\hat{u}_{i|j}$.

To compute the importance of capsule i at lower layer for capsule j at higher layer. This relation is represented by the coupling coefficients c_{ij}. The coupling coefficients c_{ij} are computed applying the softmax function over b_{ij}

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

where b_{ij} is log probability of capsule i being coupled with capsule j.This value is 0 when routing is started, then it is updated while the procedure iterates.

The input vector to parent capsule j is the weighted sum of the probability vectors at capsule i multiplied by the coupling coefficient:

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{3}$$

The output of the capsule vectors represent the probability of an object of been present in the given input or not. These output vectors can exceed value one, depending on the output, so to make the output vector represent a probability, a non linear squashing function is used to restrict the vector length to 1, where s_j is input to capsule j and v_j is the output.

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||} \tag{4}$$

Finally b_{ij} are updated by computing the inner product of v_j and $\hat{u}_{j|i}$. If two vectors agree, the product would be larger leading to longer vector length.

3.2. Self-Attention Layer

In the self-attention layer, the input vector is first transformed into three different vectors: (i) the query vector q, (ii) the key vector k and (iii) the value vector v with dimension $d_q = d_k = d_v = d_{model} = d_m$. The vectors q,k, and v are then concatenated by type into three different matrices, namely, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ each belonging to $\mathbb{R}^{C \times d_m}$ where C is the number of capsules. To compute the attention function between different input vectors the network compute the scores between matrices $\mathbf{Q}$ and $\mathbf{K}$:

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^T \tag{5}$$

These scores determine the degree of attention that we give other vectors when encoding the vector at the current position. The score matrix is then normalised based on the dimension d_m to enhance gradient stability for improved training:

$$\mathbf{S}_N = \frac{\mathbf{S}}{\sqrt{d_m}} \tag{6}$$

The score matrix is transformed into probabilities by the application of the function softmax across the vectors scores and we obtained the weighted value by multiplying the probabilities with the values vectors $\mathbf{V}$.

$$\mathbf{P} = softmax(\mathbf{S}_N) \tag{7}$$

$$\mathbf{Z} = \mathbf{P}\mathbf{V} \tag{8}$$

Vectors with larger probabilities receive additional focus from the following layers. The self-attention layer described above, implement the only encoder of the transformers. The encoder-decoder attention layer in the decoder module is similar to the self-attention layer in the encoder module with some exceptions. In this paper we take into consideration the only encoder module for the `AA-Caps` proposed network.

Figure 3. Datasets samples. The images shown are samples from the five dataset that we used to validate the proposed AA-Caps. The selected dataset are shown clockwise from the top left: MNIST, SmallNORB, Animals with Attributes 2 (AwA2), CIFAR10, and Street View House Numbers (SVHN). All of these datasets are benchmarks at the state of the art.

3.3. Routing-by-agreement and Self-Attention Layer

In this section, we want to analyse the similarities between the two mechanisms. The routing-by-agreement and the self-attention layer are built on the same idea of extracting a probability distribution that describes the relation among the lower layer and the upper layer. In capsule networks, the dynamic routing determines the connection from the lower layer to the higher layer of capsules, in par with this the self-attention layer decides how to attend to different parts of the input and how information from different parts contribute to the updates of representations. It is possible to map the attention weights the self-attention layer to assignment probabilities in capsule net. The probabilities distribution differs between the two mechanisms: in capsule nets, the assignment probabilities are computed bottom-up, in self-attention the attention is computed top-down. i.e. the attention weights in self-attention are distributed over the representations in the lower layer, but in capsule nets, the assignment probabilities are distributed over the higher layer capsules. The attention probabilities are computed based on the similarity of the representations in the same layer, but this is equivalent to the assumption that the higher layer is first initialised with the representations from the lower layer and then it is updated based on the attention probabilities computed by comparing these initial representations with the representations from the lower layer.

3.4. Capsules and Attention Heads

In networks base on capsules, each pair of capsules types from two adjacent layers has a different transformation matrix. So each instantiation of different capsules types has a different point of view over the capsules at the previous layer. In par with this, in a self-attention layer, there are multiple attention heads and each attention head uses a set of transformation matrices to obtain the projection of key, value, and query. So, each attention heads works on a different projection of the representations in the lower layer. The purpose of the two mechanisms is to have different kernels in convolutional neural networks in order to have different representations/points of view. Capsules with different types have a different point of view, but at the lower layer, the assignment probabilities for a capsule are normalised over all capsules in the higher layer regardless of the type. Hence we have one assignment distribution per capsule in the lower layer. In the self-attention layer, each head processes independently its input, with the result of a separate attention distribution for each position in the higher layer. The outputs of the heads are only combined at the last step where they are simply concatenated and linearly transformed to compute the final output of the multi-headed attention block.

3.5. AA-Caps Architecture

Figure 2 shows the architecture proposed for AA-Caps. The architecture consists of a *2D Convolutional layer*

(Conv2d), a *Primary Capsule Layer* (Primary-Caps), a *Self-Attention layer* (Self-Attention), a *1D Convolutional layer* (Conv1d) and it ends with a *classification layer* based on the attention outputs. The input image of the model is defined over m channels, the number of channels depends on the dataset to be considered. The image is fed into the Conv2d that applies a 2D convolution on m channels and extracts 256 feature maps. It applies a $kernel = 9 \times 9$ and $stride = 1 \times 1$. This layer is the original feature map extraction described for CapsNet([45]). The Primary-Caps layer takes the 256 feature maps and identify the entities present in the image. Each capsule in the Primary-Caps provides probability vectors with dimension $d_{\hat{u}}$. We obtain a tensor of $[C, W \times H \times d_{\hat{u}}]$, where for each capsule $c \in C$, there is a map $W \times H$ of activation vectors with dimension $d_{\hat{u}}$. We reshape the tensor obtained by the capsules in order to feed each head into Self-Attention with one map from a capsule. We think that feeding each head with a map from a single capsule, focus the head into the identification of the entities in which the capsule is specialised and let the Self-Attention layer collect from each head the main features identified by the capsules. In particular, the input to attention head i is the map from the i capsule and we extract the linear transformation of the $\mathbf{K}$ (Key), $\mathbf{Q}$ (Query) and $\mathbf{V}$ (Value) as described in Section3.2, and the output of attention head i is the attention function:

$$attention = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_m}})\mathbf{V} \tag{9}$$

that is described in equations presented in Section 3.2. The representation of each position in the upper layer of capsules is a weighted combination of all the representations in the lower layer. In order to compute these weights, the attention distributions, each attention head, computes the similarity between the query in each position in the upper layer to the keys of all positions in the lower layer. Then, the distribution of attention over all positions is computed by applying Equation 9 on these similarity scores. Each position in the Self-Attention layer, have a distribution of attention. The values at all positions are combined using the attention probabilities of the head. These probabilities are concatenated and transformed linearly to compute the output of the multiple head attention component. The Self-Attention layer is implemented with the Transformer Encoder Layer with C heads. We finally apply the Conv1d with $kernel = 1 \times 1$ and $stride = 1 \times 1$ and map the distributed attention over Y channels, Y is the number of classes in the datasets. The summation provides the final classification matrix $\mathbf{X} \in \mathcal{R}^{Y \times d_{\hat{u}}}$, in Figure 2 on the right.

3.6. Loss Function

The obtained matrix $\mathbf{X}$ represents for each row a class in the pool of classes of the dataset and for each column the attention raised by the heads in Self-Attention . We think that the class represented with the vector of major intensity represents the prediction of the model. We apply the marginal loss introduced in [45] to compute the error in prediction.

$$\begin{aligned}\mathcal{L}_{\text{margin}} = \sum_{c=1}^{C} \Big(& T_c \ \max(0, m^+ - \hat{x}_c)^2 + \\ & \lambda(1 - T_c) \ \max(0, \hat{x}_c - m^-)^2 \Big)\end{aligned} \tag{10}$$

where T_c equals 1 if the input datum belongs to class c, 0 otherwise, $\hat{x}_c$ is the magnitude of the c-th row of matrix $\mathbf{X}$, *i.e.* the c-th row of $\mathbf{X}$, which represents the prediction for class c. The $m^+ = 0.9$ and $m^- = 0.1$ are hyperparameters controlling the margins, and the initial weights for absent classes is controlled with $\lambda = 0.5$.

4. Experiments

We validate AA-Caps over five benchmark dataset at state of the art. We split each dataset in training and test dataset following the split proposed by the authors of the datasets. The results shown in this section are obtained by training and testing AA-Caps separately for each dataset. We do not apply any pre-trained layers to maintain the original structure of the CapsNet model [45].

4.1. Datasets

To validate the proposed approach, we have considered five different benchmarks, namely: Handwritten digits dataset (MNIST), CIFAR10, SmallNorb, Animals with Attributes 2 (AwA2), Street View House Numbers (SVHN). Figure 3 shows some image samples from these data sources.

MNIST [28] dataset consists of 70000 images with dimension 28×28. The dataset has been split in 60000 and 10000 images for training and testing respectively. The dataset is a collection of greyscale images of handwritten numbers classified among 10 classes.

SVHN contains 73000 and 26000 real-life digit images for training and testing.

CIFAR10 [25] is a well known standard dataset for image recognition experimentation, it consists of 60000 images from 10 classes of objects from different contexts. We maintain the dataset split in training and test suggested by the dataset authors: 50000 images in training set and 10000 images in test set. The images have dimension 32×32 and they are defined over three colour channels (RGB colour space).

Model	MNIST	SVHN	CIFAR10	SmallNORB	AwA2
Baseline CapsNet	99.67% (100E)	93.23% (100E)	68.70%	89.56% (50E)	12.1% (100E)
AA-Caps (Ours)	99.34% (100E)	92.13% (100E)	71.60%	89.72% (50E)	23.97% (100E)

Table 1. Summary of evaluation results. The model is validated over different bechmark to prove the contribution provided respect to the original CapsNet. We present results obtained with MNIST, SVHN, CIFAR10, SmallNORB, and AwA2 datasets.

Model	Description	Parameters	Test Acc.
CapsNet (Baseline)	Conv - Primary Capsules - Final Capsules	8.2M	99.67%
AA-Caps (Ours)	Conv - Primary Capsules - Self-Attention - Conv	6.6M	99.34%

Table 2. Comparison of CapsNet model with AA-Caps . The table presents a brief description layers that compose the structure of baseline CapsNet compared to the structure of AA-Caps, the number of trainable parameters, and the accuracy achieved by the model after 100 epochs on MNIST dataset.

SmallNORB [29] consists of 24300 image 96×96 stereo grey-scale images defined over 2 colour channels. We resized the images to 48x48 and during training processed random 32x32 crops, and central 32x32 patch during test.
AwA2 [51] consists of 37322 images of 50 animals classes. The images are collected from public sources, that makes the dataset challenging due to the uncontrolled images.

4.2. Results

We demonstrate the potential of AA-Caps on multiple datasets and compare it with the CapsNet architecture. The CaspNet architecture considered for this evaluation is the baseline CapsNet, it does not include the reconstruction module. All the experiments are performed using NVIDIA TITAN RTX with 8GB RAM. We run all our models for 50 - 100 epochs. The initial learning rate is set to 0.0001 with RAdam as optimiser [31].

SVHN dataset The dataset is defined on $C = 10$, due to the complexity of images we set the dimension of activation vector $d_{\hat{u}} = 64$. In Table 1, AA-Caps provide an accuracy of 92.13% in 100 epochs, this result is competitive with CapsNet result trained at the same epoch. We think that it is an interesting result that needs to be deeply investigated.

CIFAR10 The proposed model is compared with the CapsNet without the ensemble models. This choice is made to compare the two models with similar conditions. The model for AA-Caps is the same as what we use for SVHN dataset. The model outperforms the original CapsNet, performing well on accuracy with 71.60% compared to the original 68.70%.

SmallNORB The dataset is defined on $C = 5$, and we use a activation vector of $d_{\hat{u}} = 32$ due to the reduced number of channels (the images are defined over 2 channels). The performance of AA-Caps is compared with CapsNet for the first 50 epochs, and it outperforms the CapsNet at the state of the art with the accuracy of 89.72%.

AwA 2 The Awa2 dataset is the more complex dataset considered in this work, the images in the dataset are in high definition and they are defined over 50 classes of animals. We set $C = 50$ and we use a activation vector dimension of 64. The performance of the original CapsNet is low and after 100 epochs it obtains an accuracy of 12.1%. We observe a strong improvement in performance with AA-Caps that achieves 23.97%. We think that the proposed model provides a high improvement taking into consideration the complexity of the dataset.

4.3. Conclusions

We proposed a new model based on Capsule Network, AA-Caps that replaces the routing-by-agreement mechanism with self-attention layer. The direct connection between activity matrices from capsules and the multi-head attention create dedicated heads that analyse the similarities between capsules output. The effectiveness of this architecture is demonstrated by state-of-the-art performance (99.34%) on MNIST data with an important decrease of memory space, almost 2M parameters, making the model in line with the conventional CapsNets. Similar behaviour is observed on digit in real-life with SVHN dataset, AA-Caps performance is competitive with the original CapsNet, 92.13% and 93.23% respectively. Although AA-Caps performed better (71.60%) than the baseline CapsNet model on CIFAR10 data. The AA-Caps outperform CapsNet also on SmallNORB dataset by 0.18% (89.72%) and Animals with Attributes 2 dataset by 11.96% (23.97%). Performance on real-life, complex data (CIFAR-10, AwA 2, etc.) is known to be substandard compared to simpler datasets like MNIST. We think that the AA-Caps performs well on the benchmarks considered in the study and that proves the improvement in performance compared to existing CapsNet.

References

[1] Parnian Afshar, Arash Mohammadi, and Konstantinos N Plataniotis. Brain tumor type classification via capsule networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3129–3133. IEEE, 2018.

[2] Esad Akar, O Marques, WA Andrews, and B Furht. Cloud-based skin lesion diagnosis system using convolutional neural networks. In *Intelligent Computing*, pages 982–1000, 2019.

[3] José Juan Almagro Armenteros, Konstantinos D Tsirigos, Casper Kaae Sønderby, Thomas Nordahl Petersen, Ole Winther, Søren Brunak, Gunnar von Heijne, and Henrik Nielsen. Signalp 5.0 improves signal peptide predictions using deep neural networks. *Nature biotechnology*, 37(4):420, 2019.

[4] A Asuntha and Andy Srinivasan. Deep learning for lung cancer detection and classification. *Multimedia Tools and Applications*, pages 1–32, 2020.

[5] Ibtissam Bakkouri and Karim Afdel. Computer-aided diagnosis (cad) system based on multi-layer feature fusion network for skin lesion recognition in dermoscopy images. *Multimedia Tools and Applications*, pages 1–36, 2019.

[6] Roberto Barbuti, Stefano Chessa, Alessio Micheli, and Rita Pucci. Identification of nesting phase in tortoise populations by neural networks. extended abstract. In *The 50th Anniversary Convention of the AISB, selected papers*, pages 62–65, 2013.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[8] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[9] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[10] Stefano Chessa, Alessio Micheli, Rita Pucci, Jane Hunter, Gemma Carroll, and Rob Harcourt. A comparative analysis of svm and idnn for identifying penguin activities. *Applied Artificial Intelligence*, 31(5-6):453–471, 2017.

[11] Adrien Deliège, Anthony Cioppa, and Marc Van Droogenbroeck. Hitnet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules. *arXiv preprint arXiv:1806.06519*, 2018.

[12] Fei Deng, Shengliang Pu, Xuehong Chen, Yusheng Shi, Ting Yuan, and Shengyan Pu. Hyperspectral image classification with capsule network using limited training samples. *Sensors*, 18(9):3153, 2018.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[14] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[15] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24, 2019.

[16] Konstantinos P Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[17] Seung Seog Han, Gyeong Hun Park, Woohyung Lim, Myoung Shin Kim, Jung Im Na, Ilwoo Park, and Sung Eun Chang. Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network. *PloS one*, 13(1):e0191493, 2018.

[18] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

[19] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. 2018.

[20] Le Hou, Youlong Cheng, Noam Shazeer, Niki Parmar, Yeqing Li, Panagiotis Korfiatis, Travis M Drucker, Daniel J Blezek, and Xiaodan Song. High resolution medical image analysis with spatial partitioning. *arXiv preprint arXiv:1909.03108*, 2019.

[21] José Jiménez, Miha Skalic, Gerard Martinez-Rosell, and Gianni De Fabritiis. K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *Journal of chemical information and modeling*, 58(2):287–296, 2018.

[22] Amelia Jiménez-Sánchez, Shadi Albarqouni, and Diana Mateus. Capsule networks against medical imaging data challenges. In *Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 150–160. Springer, 2018.

[23] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[24] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*, pages 15486–15496, 2019.

[25] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[27] Y. LeCun, B. E Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten

digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1990.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:II–104 Vol.2, 2004.

[30] Zekang Li, Zongjia Li, Jinchao Zhang, Yang Feng, and Jie Zhou. Bridging text and video: A universal multimodal transformer for video-audio scene-aware dialog. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[31] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[32] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[33] Niki Martinel, Christian Micheloni, and Gian Luca Foresti. The evolution of neural learning systems: a novel architecture combining the strengths of nts, cnns, and elms. *IEEE Systems, Man, and Cybernetics Magazine*, 1(3):17–26, 2015.

[34] Aryan Mobiny and Hien Van Nguyen. Fast capsnet for lung cancer screening. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 741–749. Springer, 2018.

[35] Gota Morota, Ricardo V Ventura, Fabyano F Silva, Masanori Koyama, and Samodha C Fernando. Machine learning and data mining advance predictive big data analysis in precision animal agriculture. *Journal of Animal Science*, 2018.

[36] Rinat Mukhometzianov and Juan Carrillo. Capsnet comparative performance evaluation for image classification. *arXiv preprint arXiv:1805.11195*, 2018.

[37] Prem Nair, Rohan Doshi, and Stefan Keselj. Pushing the limits of capsule networks. *Technical note*, 2018.

[38] Xiaoyong Pan and Hong-Bin Shen. Predicting rna–protein binding sites and motifs through combining local and global deep convolutional neural networks. *Bioinformatics*, 34(20):3427–3436, 2018.

[39] David Peer, Sebastian Stabinger, and Antonio Rodriguez-Sanchez. Training deep capsule networks. *arXiv preprint arXiv:1812.09707*, 2018.

[40] Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti Bathula. Dense and diverse capsule networks: Making the capsules learn better. *arXiv preprint arXiv:1805.04001*, 2018.

[41] Rita Pucci, Christian Micheloni, Gian Luca Foresti, and Niki Martinel. Deep interactive encoding with capsule networks for image classification. *Multimedia Tools and Applications*, 79(43):32243–32258, 2020.

[42] Rita Pucci, Christian Micheloni, Gian Luca Foresti, and Niki Martinel. Fixed simplex coordinates for angular margin loss in capsnet. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3042–3049. IEEE, 2021.

[43] Rita Pucci, Christian Micheloni, Vito Roberto, Gian Luca Foresti, and Niki Martinel. An exploration of the interaction between capsules with resnetcaps models. In *Proceedings of the 13th International Conference on Distributed Smart Cameras*, pages 1–6, 2019.

[44] Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo. Deepcaps: Going deeper with capsule networks. In *CVPR*, pages 10725–10733, 2019.

[45] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[46] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[47] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[48] Michael A Tabak, Mohammad S Norouzzadeh, David W Wolfson, Steven J Sweeney, Kurt C VerCauteren, Nathan P Snow, Joseph M Halseth, Paul A Di Salvo, Jesse S Lewis, Michael D White, et al. Machine learning to classify animal species in camera trap images: applications in ecology. *Methods in Ecology and Evolution*, 10(4):585–590, 2019.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[50] Dilin Wang and Qiang Liu. An optimization view on dynamic routing between capsules. 2018.

[51] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[52] Canqun Xiang, Lu Zhang, Yi Tang, Wenbin Zou, and Chen Xu. Ms-capsnet: A novel multi-scale capsule network. *IEEE Signal Processing Letters*, 25(12):1850–1854, 2018.

[53] Pengfei Xu, Songtao Guo, Qiguang Miao, Baoguo Li, Xiaojiang Chen, and Dingyi Fang. Face detection of golden monkeys via regional color quantization and incremental self-paced curriculum learning. *Multimedia Tools and Applications*, 77(3):3143–3170, 2018.

[54] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[55] Tao Zhou, Zhixin Li, Canlong Zhang, and Huifang Ma. Classify multi-label images via improved cnn model with adversarial network. *Multimedia Tools and Applications*, pages 1–20, 2019.

[56] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.