Analyzing the Explanation and Interpretation Potential of Matrix Capsule Networks

Andrei Bondarenko[1], Saja AL-Tawalbeh[2 [0000−0001−6278−0768]],
José Oramas[2 [0000−0002−8607−5067]]

[1] University of Antwerp
[2] University of Antwerp, imec-IDLab

Abstract. The interest in capsule networks, recently proposed as an alternative to convolutional neural networks (CNNs), has seen a steady increase in recent years. This is mainly due to their ability to recognize variations in pose and deformations while requiring less training data compared to classic convolutional neural networks (CNNs). In addition, from an explainability perspective, this novel architecture also shows the potential of being more explainable and interpretable due to its hierarchical, internal representation of learned concepts and its ability to encode class characteristics as pose parameters in the class capsules. However, existing work has mainly focused on studying the first capsule network architecture, while newer architectures, such as Matrix Capsules with EM-Routing, have not received the same attention. Here we conduct a preliminary study of the inner-workings of Matrix Capsule architectures with EM-Routing and perform an analysis of the aspects that differentiate it from regular CNNs. At the same time, we focus our analysis on their interpretability and explainability properties.

Keywords: Capsule Interpretation · Capsule Networks · Path Identification

1 Introduction

Since the deep learning revolution rocked the computer vision community [29,9,20], convolutional (deep) neural networks (CNNs) have been widely and successfully applied to solve many complex computer vision tasks. However, more recently Hinton et al [15] have argued that the max-pooling operation used by these CNNs throws away important information about the precise position and orientation of detected features, which they argue could be used to learn spatial part-whole relationships to make learning equivariant features possible using simpler networks. To improve on these shortcomings a novel neural network building block, called a capsule, was proposed. This block is claimed to be invariant to the pose (position, size, orientation) of the detected features. In a later work, Sabour et al. [27] proposed an iterative algorithm that aims to solve the problem of learning compositional informations. More specifically, linking *parts*, i.e., features learned by lower-level capsules, to *whole* components, i.e., features learned by higher-level capsules.

In recent years, the domain of explainable artificial intelligence (XAI) [1] has been gaining popularity as machine learning models have been shown to be biased towards

dominant classes/features in a given particular dataset [8]. This bias can exist in many shapes and forms [23], and as a result, is often overlooked and undetected until it becomes a problem. As a result, a great deal of research has been devoted to trying to interpret the representations learned by classic deep architectures [30,6,35] and explaining their predictions [36,34,10,12]. Since capsule networks are a novel architecture they have not received the same in-depth dissection compared to their more standard CNN counterparts. This is quite worrisome when noting their application in medical contexts and other critical domains is already being explored [3,17,25,2].

The objective of this work is to study the internal feature representations learned by capsule networks and how higher and lower-level features are related with each other by the routing algorithm. This work aims at providing a preliminary analysis on the more recent matrix capsule architecture introduced by [16] which is rarely addressed in the literature.

To this end, this work studies the two components that differentiate the matrix capsule architecture from regular CNNs, i.e., the matrix capsules and the expectation-maximization algorithm. By applying perturbations to the pose matrices of the matrix capsules in the class capsule layer, we attempt to verify whether the different elements of the pose matrix indeed represent some high-level feature. Next, we verify whether there is any correlation between the activation of certain capsule types and the classes of interest. Finally, the *part-whole* assignments made by the routing algorithm are scrutinized by analyzing the internal routing coefficients and attempting to verify whether some kind of path through the network can be observed when focusing on a more informative subset of capsules.

2 Related work

Existing work related to interpreting capsule networks and explaining their predictions can be categorized into three categories. (i) Works that study the core characteristics of capsule networks and their implications w.r.t. explainability and/or interpretation. (ii) Works that evaluate the explanation and interpretation of capsule network predictions within specific research domains, e,g., biology and medicine. (iii) Finally, works that propose novel architectures that extend or incorporate capsule networks and their routing algorithms with the goal of improving explainability and/or interpretation.

In category (i), [28] analyze the learned features and explanation capabilities of class capsules on the MNIST dataset [21]. There the effect on reconstructed inputs was observed as parameters in the class capsule were modified. In addition, it analyzed the likelihoods predicted by the class capsules for misclassified samples and show that in all cases the correct class has the second highest likelihood. When performing a reconstruction according to the class predicted with the second highest likelihood, they observe that the reconstructions are similar to the ones for the class with the highest likelihood. They argue that this similarity in reconstruction can be used as an explanation for the misclassification of said samples. Our work further buid on this by observing the difference in reconstruction when altering the parameters of the pose matrix in matrix class capsules. One of the experiments performed by Lin et al. [22] shows that capsule networks can produce data representation with interesting attributes. To this

end, they synthesize a test dataset with known underlying 2D manifolds spanned by geometric translations. They then collect the data representations of this test set in the intermediate layers of a capsule network and a regular CNN. When visualizing the tSNE $\mathbb{R}^2$ embeddings of these representations, they observe that the embeddings of the capsule networks align much better with the internal 2D manifold than those of the regular CNNs. Bhullar A. [7] explores the interpretation of the concepts learned by intermediate and final capsule layers by visualizing the routing path through a convolutional capsule network based on the original architecture [27]. Recent efforts have focused on a deeper study on the learned representation and the hierarchical relationships encoded in CapsNets. On the one hand, [5] conducted a systematic study towards assessing the interpretability of these type of networks. It extended the typical perturbation analysis and extracted the relevant features that define internal paths linking inputs and outputs. Moreover, a methodology is proposed for assessing the existence of part-whole relationships in the internally-encoded representation. Their results suggest that the representation internally encoded in the network is not disentangled. On the other hand, [24] argued that instead of relying on individual neurons to detect simple features like edges or curves, the capsule network aims to recognize higher-level structures and objects by learning capsules that encode the presence and properties of those entities. Therefore, the existence of a part-whole hierarchy in capsule networks is explored through experiments involving parse tree analysis and measuring the response between the parts and the whole. Similar to these efforts, we aim at the analysis of the representation internally encoded in a capsule-based architecture. Different from them, we focus our analysis on the capsule architecture variant based on EM-Routing [16].

In category (ii), [18] show that when using capsule networks for protein structure classification and prediction the prediction vectors encode valuable information about the protein's structure. They show that by modifying the input and observing the changes in the prediction vector it is possible to see which parts of the input the network considers important for classification. In the context of brain tumor classification via radiomics analyses, [4] used input reconstruction after modifying features in the final capsule layer and activation maximization, i,e., finding an input that can maximize the activation of a specific output capsule, to interpret the features learned by capsule networks. They show that the features learned by the network are similar to hand-crafted features used in radiomics.

In category (iii), [13] proposes a novel architecture called graph capsule networks, where the routing part of regular capsule networks is replaced with a multi-head attention-based graph pooling approach. Eliminating the routing part of capsule networks allows them to reuse existing gradient-based explanation methods that were originally created for regular CNNs. [32] combine the capsule network architecture with a trainable multi-head attention layer. They then propose explanation and interpretation methods (they refer to these as local and global interpretation, respectively), based on the routing weights of the capsule network, that yield human-understandable interpretation results. One of our experiments is similar in the sense that we also attempt to explain the predictions made by a capsule network based on the routing weights of the network. [19] focus on the interpretability aspects provided by class capsules, i,e., capsules found in the last layer of classification capsule networks. They propose a new architecture that aims

to address two limitations they identified in said class capsules. The first limitation being that some instantiation parameters of class capsules represent concepts that are irrelevant to classification. The second limitation is that some instantiation parameters within a single class capsule encode overlapping concepts. They address these limitations by combining capsule networks with class-supervised disentanglement learning, which aims to disentangle the latent feature space of a capsule into two complementary subspaces, i,e., class-relevant and irrelevant subspaces. Following this method it is possible to explain the model predictions using distinct, class-relevant concepts. [31] studies the effect of guiding lower-level capsules to represent specific properties by incorporating attribute embeddings into the primary capsule layer. These attributes are then learned by using a binary value, i,e., whether the attribute is present, as the target for the attribute dimensions. This learning is guided by extending the capsule network loss function with an attribute loss function, which is the L1 Loss between the attribute dimensions' values and the corresponding binary target values. A novel architecture was proposed to detect facial action units considering multiple views and model the variation at once [26]. They reconstruct unmasked capsules to explain/model the variation in the dataset. Therefore, they explain the learned representation by visualization these capsules. In our experiments, we proposed masking to verify the relevant path through the network.

An important note to make is that most of the related efforts have focus their analysis in the capsule-based architecture defined by [27]. Different from that, our work focuses on matrix capsules [16].

3 Capsule Networks

This section presents an overview both capsule network architectures (CapsNets) [27,16]

3.1 Capsules and routing-by-agreement

Traditional CNNs are built up of neurons. In a convolutional layer a set of weights, known as a kernel, is replicated across the input space its corresponding layer. As a result, the output of a convolutional layer is a two-dimensional matrix where each element is the result of a linear combination of a part of the input space with the weights from the kernel. This two-dimensional matrix is often called a feature map and the kernel can thus be seen as a replicated feature detector. A nice property that convolutional layers give us is translational equivariance, e.g., the position of an object should not be fixed in order to be detected. A convolutional layer is then often followed by a pooling layer that downsamples the output in order to summarize the presence of features in a region of the feature map. A commonly used pooling operation is max-pooling, which simply summarizes a region as its highest activation. This operation results in (local) translational invariance, i,e., ensuring that the output of a CNN remains the same regardless of how the input is shifted.

Given the side-effect of pooling layers of throwing away potentially valuable information, [15] argue that this method my not be most optimal way of learning representations for visual recognition tasks, such as facial identity recognition, that require knowledge of the precise spatial relationships. Instead, they argue that neural networks should use

capsules, i,e., a group of neurons that learns to recognize a certain visual feature, that perform some internal computations on their inputs and encapsulate the results into a small, highly informative vector of outputs. The probability of a feature being detected is then encoded as the length of the output vector, while its *pose*, also referred to as the feature's instantiation parameters, is encoded by the direction in which the vector points.

[27] provide a straightforward implementation of this concept where capsules are represented as vectors. Additionally, they propose an iterative algorithm for linking lower-level capsules to higher-level ones. The total input s_j of a capsule is a weighted sum of prediction vectors $\hat{u}_{j|i}$.

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{1}$$

These prediction vectors are produced by multiplying the output of the capsules in the layer below u_i by a learned weight matrix W_{ij} specific to each pair of capsules in adjacent layers:

$$\hat{u}_{j|i} = W_{ij} u_i \tag{2}$$

The coupling coefficients c_{ij} in Equation 1 are determined as part of the iterative routing algorithm where Ω_l is the set of capsules that forms layer l. The total input s_j is then *squashed*, using the non-linearity depicted in Equation 3, to produce the output vector v_j of the capsule. This ensures that the vectors' length is normalized to the range $[0,1]$.

$$v_j = \frac{||s_j||^2}{1+||s_j||^2} \frac{s_j}{||s_j||} \tag{3}$$

3.2 Matrix capsules and EM-routing

Building on these ideas, [16] later proposed improvements to the capsules concept in the form of matrix capsules together with a new non-linear routing procedure based on the expectation–maximization algorithm. This new form of capsules differs in that it is no longer represented by a single vector. Matrix capsules have two components, a 4x4 pose matrix M, and an activation probability a. This improves on the previous version of capsules in the following ways:

1. Using the length of the pose vector to model the probability that an entity is present required an unprincipled non-linearity (Equation 3) to be used. The new non-linear routing procedure minimizes a more sensible objective function.
2. The new agreement measure, negative log variance of a Gaussian cluster, doesn't saturate at 1 like the cosine of the angle between two pose vectors. This makes it better at discerning between good and very good agreement.
3. By representing the pose as a vector of length n the learned transformation matrices needed to have n^2 parameters. By switching to a matrix representation of n elements for the pose the number of parameters needed for the transformation matrices is also reduced to n.

For this formulation, we again denote the set of capsules that form layer L as Ω_L. The pose matrix M_i of capsule $i \in \Omega_L$ is used to cast a vote V_{ij} for the pose matrix M_j of capsule $j \in \Omega_{L+1}$. This is done by multiplying M_i with a learned transformation matrix W_{ij}.

$$V_{ij} = M_i W_{ij} \tag{4}$$

The set of all votes V is used together with the activation probability vector a as input to the non-linear routing procedure. The output of this procedure is the set of pose matrices M and activation probability vector a for layer Ω_{L+1}. Parameters β_a and β_u are learned discriminatively and parameter λ denotes the inverse temperature which increases every iteration with a fixed schedule. The inverse temperature is a term borrowed from the reinforcement learning field where it is often used for a factor that controls the exploration-exploitation trade-off in optimization algorithms. In the EM-Routing algorithm, it serves a similar purpose in that it forces the routing algorithm to focus more on the more promising capsules (exploitation) in later iterations. The EM routing algorithm is explicit in the fact that it is based on the EM clustering algorithm to assign lower-level capsules to higher-level ones.

4 Experiments

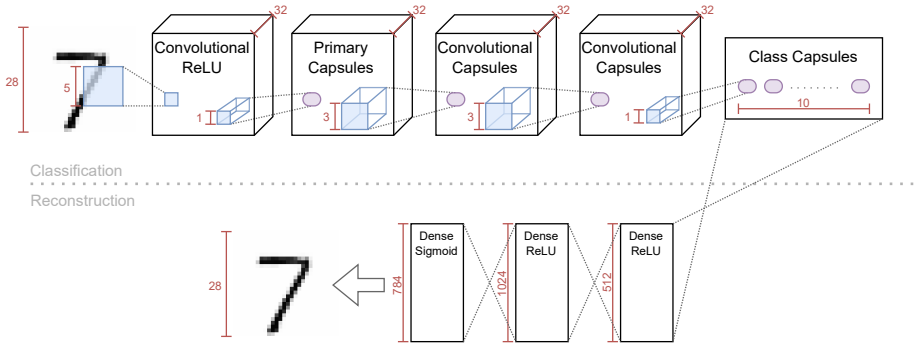

Fig. 1. Architecture of the capsule network under study. The top half of the figure depicts the classification part of the network, while the bottom half represents the reconstruction part. The classification part consists of a regular convolutional layer, a primary capsule layer, two convolutional capsule layers, and a fully connected class capsule layer. The reconstruction part consists of 3 dense layers.

The experiments in this report focus on the matrix capsule architecture presented in [16] and its performance on the MNIST dataset [21]. More concretely, the experiments focus on the two components that differentiate the matrix capsule architecture from regular

CNNs, i,e., the matrix capsules and the *part-whole* associations made by the routing algorithm. We based our experiments on an open-source PyTorch implementation of matrix capsules available at [33].

We also want to emphasize that, as other works have noted [11], we experienced difficulties in reproducing and achieving the same performance as the original paper on matrix capsules [16].

4.1 Architecture

The architecture under study is depicted in the top half of Figure 1. It starts with a regular convolutional layer with a 5x5 kernel, 32 channels, a stride equal to 2, and a ReLU activation function. This layer is followed by the primary capsule (PrimCap) layer, which has 32 capsule types. This layer performs a learned linear transformation to determine the pose matrix and activations for each of the primary capsules. From the PrimCap layers onward routing takes place between layers. The number of iterations of the routing algorithm that were performed was fixed at 2. The PrimCap layer is followed by two convolutional capsule (ConvCap) layers. Both have a kernel size of 3x3 and 32 capsule types. They only differ in their stride where the first ConvCap layer has a stride of 2 and the second ConvCap layer has a stride of 1. The last ConvCap layer is connected to a capsule layer which contains one capsule per output class. We refer to this layer as the class capsule (ClassCap) layer.

For some of the experiments this network was extended with a decoder similar to the one used with the first capsule architecture proposed by Sabour et al. [27]. The decoder is depicted on the bottom half of Figure 1. It features three fully connected layers, with the first two using the ReLU activation function and having sizes 512 and 1024, respectively. The final layer of the decoder uses the sigmoid activation function and is of size 784, which is equal to the size of a flattened input image. The output of the final layer can then be reshaped to form a reconstructed input image. By adding this decoder the network is also trained to reconstruct the input based on the elements of the class capsules' pose matrices. These reconstructions can then be used to verify which features are encoded by the elements of the pose matrices.

4.2 Training

The MNIST dataset provides predefined train and test sets, with each having 60K and 10K samples, respectively. Additionally, 10% (6K samples) of the training set was used as the validation set. Training duration was set to 30 epochs in batches of 64 and 256 samples for the training and test, respectively.

To gauge the impact of adding a decoder to the network on the classification performance, the network was trained twice. Once without the decoder part and a second time with the decoder extension. For training the network without the decoder the spread loss was used. The spread loss maximizes the gap between the activation of the target class a_t and the activation of other classes. This loss is computed using Equation 5 where m is the minimal margin between class activations.

$$L_i = (max(0, m - (a_t - a_i)))^2, L = \sum_{i \neq t} L_i \tag{5}$$

For training the network with the decoder the total loss was set to equal the sum of the spread loss and a scaled reconstruction loss. The reconstruction loss is simply the mean squared error between the actual and the reconstructed input. It is scaled down (by a factor of 5e$-$4) in order to prevent it from dominating the training process. The performance of both final models was then measured using the test set. The results have been summarized in Table 1. The final decoder model was then used for the rest of the experiments. We can see that adding the decoder impacts the classification performance slightly, however, not to a degree that is detrimental to the rest of the experiments.

Table 1. Best model performance in terms of test set accuracy of the model trained with and without the decoder attached.

Model	Decoder	No Decoder
Accuracy	97%	98%

4.3 Pose matrix

The first aspect of matrix capsules is the pose matrix. Hinton et al. [16] call it that because they argue that the network *could* learn to encode the relationship between an entity and the viewer, i.e., the pose, using said matrix. By perturbing the elements of this pose matrix and observing how the reconstruction of a digit changes we can get a sense of what is encoded by each element. To this end a sample was taken for each class and fed to the classification part of the network to get a predicted pose matrix. Each element of this pose matrix was then perturbed independently by intervals of 0.05 in the range $[-0.3, 0.3]$. We chose this interval and step size in order to get results that are comparable to the dimension perturbation reconstructions done in [27], where a the same step size was used over a smaller range, i.e., $[-0.25, 0.25]$

Similar to the vector representation introduced in [27] we observed that some elements encode combinations of global variations while others encode localized parts of digits. We show 3 examples of these localized parts in Figure 2, where the top row shows a change in the curvature of the top part of the digit 7, the middle row shows variation in the length of the bottom curve of digit 5, and the bottom row shows how a parameter controls how open or closed the top part of a digit 4 is. So the pose matrices do indeed encode the configuration of features, but not necessarily a physical pose, as could be inferred from the naming used for this element.

4.4 Activations

The second aspect of the matrix capsules that was studied is their internal activations. The goal is to determine whether the activations alone could be used to explain the predictions made by the network or to identify any capsule types that encoded discriminative features, i.e., features that can be used to uniquely identify certain classes. To this end we computed first order statistics (minimum, maximum, mean, and standard deviation)

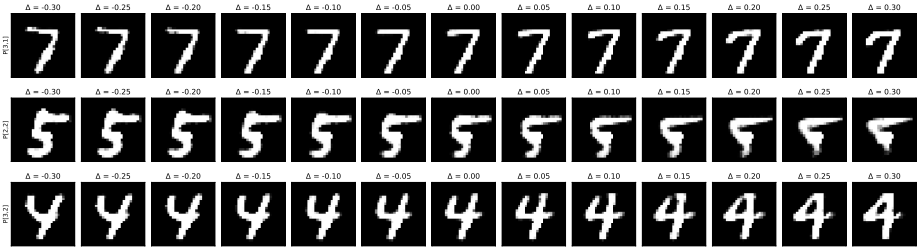

Fig. 2. Examples of pose matrix perturbations that influence localized parts. Top row shows the change in curvature of a 7's top line. Middle row shows variation in length of a 5's bottom curve. Bottom row shows how a pose parameter controls how open or closed the top of a 4 is.

of the activations of each capsule type per class for each layer over the test set. By plotting these class-specific statistics we can verify whether the activation range of a given capsule types deviates strongly for a specific class. If a strong deviation is present for a certain class, that could suggest that that capsule type encodes a feature that is unique to that class. The results of this experiment are presented in Figures 3 through 6. Each of the subplots in each of the figures shows the aggregated activation values when processing samples belonging to a single class.

The activations of the ClassCaps at the end of the architecture, i.e., Figure 6, clearly show that the class capsules each encode one of the classes because the mean activation of a single class capsule is always the highest. However, as we move one step back through the network and look at the activations in the second ConvCaps layer, i.e., Figure 5, we see that the activations are squeezed into a much tighter range. In addition, the mean activation curves look very similar between the different classes. This makes it hard to identify capsules that encode discriminating features. A similar observation can be made for the earlier capsule layers, i.e. Figures 3 and 4. This suggests that the activations of the matrix capsules on their own are not a good proxy for explaining predictions made by the network.

This experiment did allow us to determine valid activation value ranges per capsule type, which can be used in future experiments to ensure that any perturbations that are made to these activation values remain within a valid range. For example, we can see that most activation values do not go far beyond 0.5 so it would not make sense to perform perturbations that go far beyond that value. Moreover, focusing on a significantly narrower range would lead to an incomplete inspection of the representation space.

4.5 Routing coefficients

The final aspect of matrix capsules that differentiates them from regular CNNs is the routing process, which produces routing coefficients in order to associate low-level capsules to higher-level ones. As such, we can try to find a parse tree-like structure [14,27] of capsules for each class capsule, by going backwards through the network and following the connections between capsules with the highest routing coefficients. This structure could then tell us which combination of low-level features form the conceptS

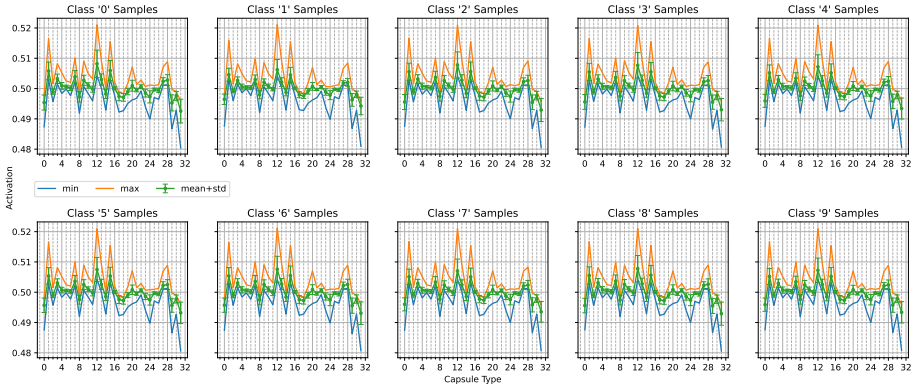

Fig. 3. Minimum, maximum, and mean (+ standard deviation) activations per capsule type for the PrimCaps layer. Each plot considers only the samples of one of the classes.

represented by the class capsule. Since the model we are studying, and as a result the number of connections we need to consider for the parse tree, is quite large, we first try to identify capsules that are more informative than others, i.e. those which contributed the most to the classification result. To this end, push the training set through the network and collect the values of the routing coefficients after the last routing iteration, for each capsule layer in the network. Then, we compute the mean routing coefficient over all samples, for each each pair of connected capsules. We use the variance of the routing coefficients of connections originating from a lower-level capsule as a proxy for importance. The intuition behind this is that capsules that have a similar mean routing weight to each capsule in the next layer are less important since they are not casting a distinct vote for a higher-level concept i.e., are remaining indecisive with their vote. While a capsule with a higher variance in mean routing weight will have a more distinct vote towards a single (or a subset of) capsule(s) in the next layer, thus influencing the final classification result more. Once we had ranked the input capsules for each layer in terms of their importance, we evaluate the classification performance, in terms of accuracy, of the model while masking all but the top-k most important input capsules in a layer. We performed the experiment for all top-i where $i \in [1,n]$, with n being the number of capsules in the layer. The range of experiments was performed for each layer independently. The results of these experiments have been plotted in Figure 7, where we can see the accuracy of the model in terms of the number of top-k most important capsules left unmasked. We can see that the last two layers the top-50% most important capsules provide the largest influx in performance, even nearing the performance of the full model. For the first ConvCaps layer, the increase in performance is a little more gradual. This aligns with the idea that the primary capsules, which serve as input to the first ConvCaps layer, encode very low-level features that are shared more between concepts encoded in the next layer. In later layers, the concepts get more refined and can thus serve as more unique, defining features for concepts/classes in the next layer.

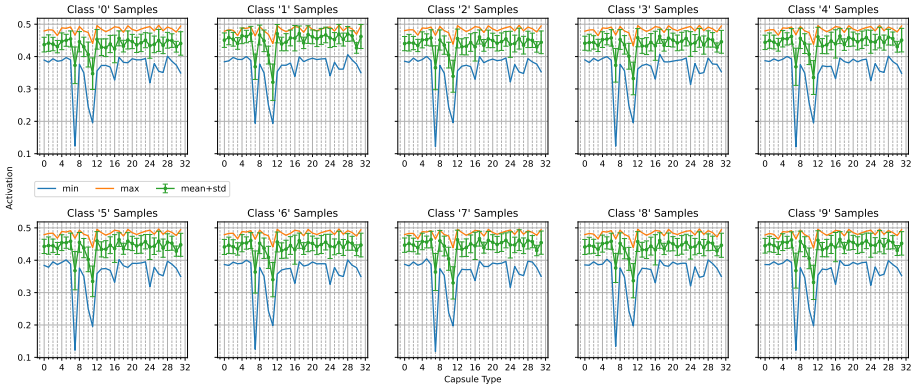

Fig. 4. Minimum, maximum, and mean (+ standard deviation) activations per capsule type for the first ConvCaps layer. Each plot considers only the samples of one of the classes.

Based on these results, we chose a threshold top-k for each layer and evaluated the classification performance of the network when masking capsules in multiple layers. For the ConvCaps1, ConvCaps2 and ClassCaps layers we chose the thresholds 5398, 668, and 217, respectively. Relatively speaking, this amounts to masking about 14%, 42%, and 58% of the capsules in each layer, respectively. These thresholds were chosen in such a way that they were as small as possible while maximizing the classification performance of the network when a mask with said k is applied to the respective layer. We then tried all possible combinations of applying masks in multiple layers and evaluated classification performance. The results of these experiments are summarized in Table 2.

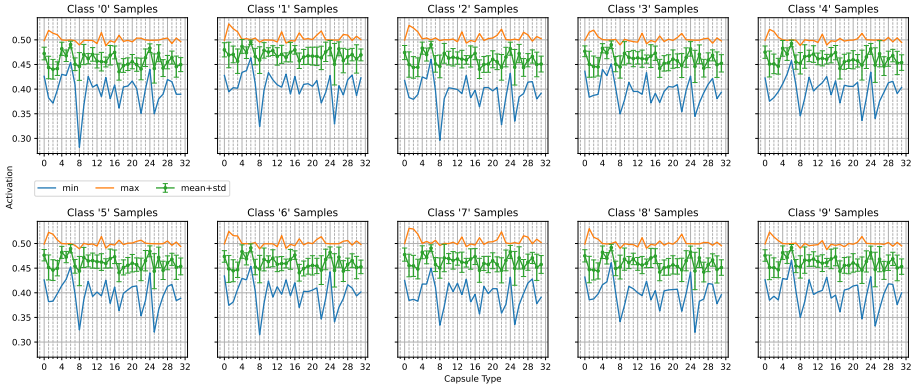

Fig. 5. Minimum, maximum, and mean (+ standard deviation) activations per capsule type for the second ConvCaps layer. Each plot considers only the samples of one of the classes.

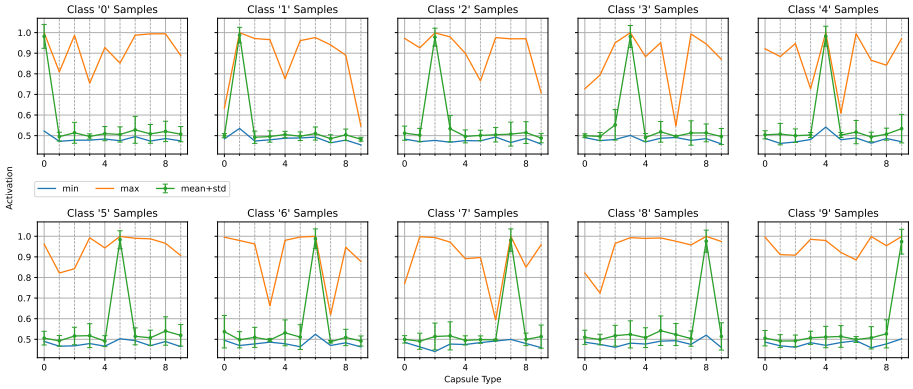

Fig. 6. Minimum, maximum, and mean (+ standard deviation) activations per capsule type for the ClassCaps layer. Each plot considers only the samples of one of the classes.

We can see that masking the inputs of the two last layers together, i.e., ConvCaps2 and ClassCaps, impacts the classification performance the most, while combinations, where the inputs of the ConvCaps1 layer are masked, have a very minor difference in performance compared to the same (sub-)model with without said mask. The sub-model with all layer masks applied sees a total relative drop in performance of 5.89%, compared to the original model. The fact that the model performance has not completely deteriorated after masking about 20% of the total number of capsules in the network suggests that this is a good step towards finding a method for determining the critical routing paths through the network. A potential avenue to explore based on these results is to focus on a single class and base the importance ranking of the capsules on the impact on the classification performance of that class. This way we could potentially determine a sub-model that contains all of the critical routing paths for a given class. The features encoded by the capsules on this path could then be used to explain the predictions made by the network.

5 Discussion & Conclusion

In this work, we conducted a preliminary study focusing on the three aspects that differentiate matrix capsules with EM-routing from CNNs. We approached this analysis by assessing the extend to which these aspects could be used as a proxy for explaining the predictions made by the network and interpreting the concepts it encodes internally. First, we studied the pose matrices of the class capsules. Via reconstructions of perturbed pose matrices, we show that the parameters of these matrices encode instantiation parameters of the classes they represent. This aspect can thus be used for interpreting the concepts learned by the network, similar to the observations made by other works [27,28,19]. However, this observation is different from what the original ”pose” term introduced in [16] may suggest to the reader. More specifically the use of this term in combination the dataset of 3D objects used for its validations, suggest that the the ”pose” encoded by

Table 2. Classification performance (accuracy) of sub-models derived by masking input capsules in different layers. When masks were applied, the top-5398 input capsules were left unmasked in the ConvCaps1 layer, the top-668 in the ConvCaps2 layer, and top-217 in the ClassCaps layer.

Masked			Classification Performance (%)
ConvCaps1	ConvCaps2	ClassCaps	
			97
x			97
	x		96
		x	97
x	x		96
x		x	96
	x	x	92
x	x	x	91

these matrices is nothing more than that of the object in the 3D space. We believe this combination lends itself to confusion, and in fact our preliminary results suggest that what these parameters encode is heavily dataset dependent. As such, we consider the term *instantiation parameters*, used by earlier works, to be more general and appropriate.

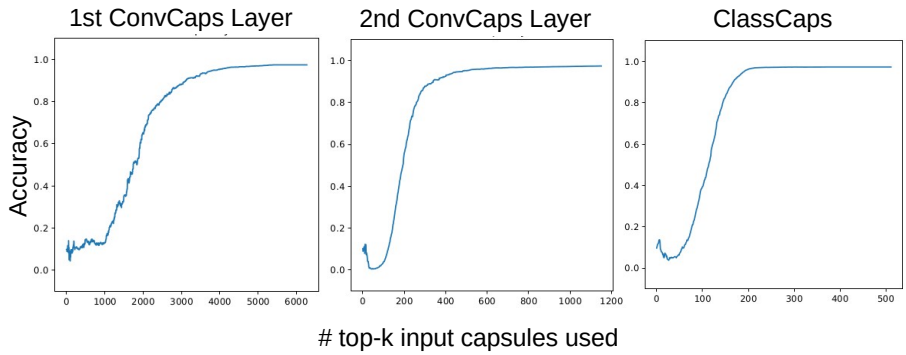

Fig. 7. Classification performance of the network when varying the top-k input capsules left unmasked. Each plot represents the classification performance curve of the network while masking a subset of the input capsules for each layer independently.

Second, we focused our analysis on the internal activations of capsules within the network. We observed that the activations do not differ enough between different classes in earlier layers of the network to be able to identify capsules that are important to a specific class. This leads us to believe that the capsule activations, on their own, are not a good proxy for explaining the predictions made by the network.

Additionally, we took a closer look at the routing coefficients computed by the EM-Routing algorithm. By using them as a proxy for the importance of input capsules

in each layer, we were able to determine a subset of capsules that contributed to most of the classification performance, for each layer. For some layers, we were able to mask more than 50% of the input capsules without experiencing a significant drop in classification performance. Finally, we selected a subset of input capsules for each layer that maximized the number of capsules being masked, while minimizing the loss in performance, and experimented by applying these masks to multiple layers at the same time. We saw a relative drop in performance of at most 5.89%, leading us to believe that using the routing coefficients as a proxy for input capsule importance is a good first step towards a robust explanation method for capsule networks.
Potential future work will focus on further refining the use of routing coefficients as a proxy for input capsule importance based explanation method and alternative methods to estimate activation paths.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE access **6**, 52138–52160 (2018)
2. Afshar, P., Heidarian, S., Naderkhani, F., Oikonomou, A., Plataniotis, K.N., Mohammadi, A.: Covid-caps: A capsule network-based framework for identification of covid-19 cases from x-ray images. Pattern Recognition Letters **138**, 638–643 (2020)
3. Afshar, P., Mohammadi, A., Plataniotis, K.N.: Brain tumor type classification via capsule networks. In: 2018 25th IEEE international conference on image processing (ICIP). pp. 3129–3133. IEEE (2018)
4. Afshar, P., Plataniotis, K.N., Mohammadi, A.: Capsule networks' interpretability for brain tumor classification via radiomics analyses. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 3816–3820. IEEE (2019)
5. AL-Tawalbeh, S., Oramas, J.: Towards the characterization of representations learned via capsule-based network architectures. arXiv preprint arXiv:2305.05349 (2023)
6. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6541–6549 (2017)
7. Bhullar, A.: Interpreting Capsule Networks for Image Classification by Routing Path Visualization. Master's thesis, University of Guelph (2020)
8. Buolamwini, J., Gebru, T.: Gender shades: Intersectional accuracy disparities in commercial gender classification. In: Conference on fairness, accountability and transparency. pp. 77–91. PMLR (2018)
9. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3642–3649 (2012). https://doi.org/10.1109/CVPR.2012.6248110
10. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE international conference on computer vision. pp. 3429–3437 (2017)
11. Gritzman, A.D.: Avoiding implementation pitfalls of "matrix capsules with em routing" by hinton et al. In: International Workshop on Human Brain and Artificial Intelligence. pp. 224–234. Springer (2019)
12. Grün, F., Rupprecht, C., Navab, N., Tombari, F.: A taxonomy and library for visualizing learned features in convolutional neural networks. International Conference on Machine Learning (ICML) Workshops, 2016 (2016)

13. Gu, J.: Interpretable graph capsule networks for object recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1469–1477 (2021)
14. Hinton, G.E., Ghahramani, Z., Teh, Y.W.: Learning to parse images. Advances in neural information processing systems **12** (1999)
15. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: International conference on artificial neural networks. pp. 44–51. Springer (2011)
16. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. In: International conference on learning representations (2018)
17. Iesmantas, T., Alzbutas, R.: Convolutional capsule network for classification of breast cancer histology images. In: International Conference Image Analysis and Recognition. pp. 853–860. Springer (2018)
18. de Jesus, D.R., Cuevas, J., Rivera, W., Crivelli, S.: Capsule networks for protein structure classification and prediction. arXiv preprint arXiv:1808.07475 (2018)
19. Jung, D., Lee, J., Yi, J., Yoon, S.: icaps: An interpretable classifier via disentangled capsule networks. In: European Conference on Computer Vision. pp. 314–330. Springer (2020)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012), https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
22. Lin, A., Li, J., Ma, Z.: On learning and learned data representation by capsule networks. IEEE Access **7**, 50808–50822 (2019). https://doi.org/10.1109/access.2019.2911622, https://doi.org/10.1109/access.2019.2911622
23. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. ACM Comput. Surv. **54**(6) (jul 2021). https://doi.org/10.1145/3457607, https://doi.org/10.1145/3457607
24. Mitterreiter, M., Koch, M., Giesen, J., Laue, S.: Why capsule neural networks do not scale: Challenging the dynamic parse-tree assumption. arXiv preprint arXiv:2301.01583 (2023)
25. Mobiny, A., Nguyen, H.V.: Fast capsnet for lung cancer screening. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 741–749. Springer (2018)
26. Onal Ertugrul, I., Jeni, L.A., Cohn, J.F.: Facscaps: Pose-independent facial action coding with capsules. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 2130–2139 (2018)
27. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 3859–3869. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
28. Shahroudnejad, A., Afshar, P., Plataniotis, K.N., Mohammadi, A.: Improved explainability of capsule networks: Relevance path by agreement. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP). pp. 549–553 (2018). https://doi.org/10.1109/GlobalSIP.2018.8646474
29. Shrestha, A., Mahmood, A.: Review of deep learning algorithms and architectures. IEEE Access **7**, 53040–53065 (2019). https://doi.org/10.1109/ACCESS.2019.2912200
30. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
31. Van Bruggen, S.: Interpretable Capsule Networks: Incorporating Semantic Knowledge with Guided Routing. Master's thesis, University of Amsterdam (2019)
32. Wang, Z., Hu, X., Ji, S.: icapsnets: towards interpretable capsule networks for text classification. arXiv preprint arXiv:2006.00075 (2020)

33. Yang, L.: Matrix Capsules with EM Routing. https://github.com/yl-1993/Matrix-Capsules-EM-PyTorch, accessed: 2022-02-17
34. Zeiler, M.D., Taylor, G.W., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: 2011 international conference on computer vision. pp. 2018–2025. IEEE (2011)
35. Zhang, Q.s., Zhu, S.C.: Visual interpretability for deep learning: a survey. Frontiers of Information Technology & Electronic Engineering **19**(1), 27–39 (2018)
36. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)