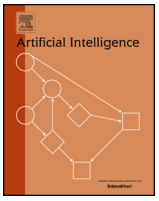
Artificial Intelligence 348 (2025) 104395

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Interpreting capsule networks for image classification by routing path visualization

Amanjot Bhullar [a], Michael Czomko [a], R. Ayesha Ali [a,*], Douglas L. Welch [b]

[a] Department of Mathematics and Statistics, University of Guelph, Guelph, N1G2W1, Ontario, Canada
[b] Department of Physics and Astronomy, McMaster University, Hamilton, L8S4L8, Ontario, Canada

ARTICLE INFO

MSC:
68T45
68T01

Keywords:
Capsule networks
Interpretation
Computer vision

ABSTRACT

Artificial neural networks are popular for computer vision as they often give state-of-the-art performance, but are difficult to interpret because of their complexity. This black box modeling is especially troubling when the application concerns human well-being such as in medical image analysis or autonomous driving. In this work, we propose a technique called routing path visualization for capsule networks, which reveals how much of each region in an image is routed to each capsule. In turn, this technique can be used to interpret the entity that a given capsule detects, and speculate how the network makes a prediction. We demonstrate our new visualization technique on several real world datasets. Experimental results suggest that routing path visualization can precisely localize the predicted class from an image, even though the capsule networks are trained using just images and their respective class labels, without additional information defining the location of the class in the image.

1. Introduction

Convolutional neural networks (CNNs) have achieved state-of-the-art results on many computer vision problems, and are currently the most popular algorithms for image classification. Though it is undeniable that CNNs are capable of producing incredible results, they suffer from some limitations that are suspected to prevent them from being adopted in the long term. These limitations include the difficulty of CNNs to generalize to novel viewpoints and preserve the precise positional information of entities in an image [32]. The max-pooling operation, which is used in CNNs to achieve greater performance, is a source for the loss of precise positional information as it down-samples the image [14]. Further, CNNs may require refinements to better emulate the human visual system, which naturally overcomes some of the challenges CNNs still face. Deeper CNN architectures are often used to mitigate the impact of these limitations. Unfortunately, deeper networks contain more weights, and thus, require larger datasets to train. Automation can then become difficult in specialized fields.

Sabour et al. [32] proposed capsule networks as an alternative neural network for computer vision, and achieved state-of-the-art image classification results on the MNIST dataset [20]. Capsule networks were created to overcome the limitations of CNNs, and are considered to more closely mimic the human visual system. They use a dynamic routing-by-agreement algorithm between layers which allows them to better model hierarchical relationships. In turn, this allows capsule networks to generalize better to novel viewpoints. Also, precise positional information is preserved as max-pooling is not implemented. Capsule network architectures typically require

* Corresponding author.
E-mail addresses: bhullara@uoguelph.ca (A. Bhullar), aali@uoguelph.ca (R. Ayesha Ali).

https://doi.org/10.1016/j.artint.2025.104395
Received 5 October 2022; Received in revised form 7 July 2025; Accepted 10 July 2025

Available online 17 July 2025

fewer trainable weights than CNNs, and as a result, are expected to require a smaller set of images to train. Therefore, capsule networks have become popular in the classification of various types of images recently, particularly for the classification of medical images ([12], [13], [27]).

What the individual dimensions of a given capsule represent are interpreted in Sabour et al. [32] and Shahroudnejad et al. [34]. In short, the image is reconstructed using the decoder network after some of the dimensions of the appropriate capsule(s) have been slightly perturbed [32]. What each dimension represents can be inferred from the way the perturbations affect the reconstructed image. However, asking what entity a given capsule detects is another interesting question. This was not a natural question for Sabour et al. [32] and Shahroudnejad et al. [34] because each MNIST image contains a solitary entity, a single digit. In real world applications this may not be the case. For example, astronomical images may contain several entities such as light echoes, saturated star images and optical path artifacts, within a single image. Interpreting the entities that capsules detect allows a user to determine if the correct information from the image is being used to predict the class of the image and to identify situations in which the network may fail. Once identified, constraints can be applied to the network, or improvements can be made to the training set to decrease the probability of failure. Preventing failure is especially important for applications in which prediction errors can lead to extremely undesirable events, such as in medical image analysis or autonomous driving.

The routing paths that are inherently created in capsule networks can be used to understand which areas of the input image most influence a capsule. In this work, we introduce *routing path visualization*, which represents the routing paths as an image. This visualization shows the regions of the input image that influence a capsule of choice. Routing path visualization (RPV) can be performed on the test set to interpret the entities that a capsule detects, thereby providing a visualization of a particular capsule for each image in the test set. Entities that consistently appear in the routing path visualization images of a particular capsule should be the entities that the capsule detects.

RPV does not require an additional network, such as a decoder network, to be attached to the capsule network, or increase the number of trainable weights of the capsule network. Instead it utilizes the routing paths that are inherently created in capsule networks. RPV generates a map which displays how strongly each region of an image is routed to a given capsule. Consequently, it can be used to interpret the entity that a capsule detects. In contrast, RPV does not interpret the information that each dimension of a capsule encodes, but instead reveals how much of each region in the image is routed to a capsule.

Experimental results suggest that RPV can be used to precisely localize the predicted class from an image when performed on a final layer capsule. Consequently, RPV can also be used for object localization in addition to being an interpretation technique. Localization of the class is useful in medical image analysis as the image expert can cross-check with the RPV image to find abnormal areas. For instance, RPV may find areas of the lung to correspond to lung disease that the image expert may miss. Not only has the routing path that capsule networks inherently create not been used to interpret the entity that a capsule detects, but there is no method for interpreting the entity that a capsule detects.

Our method can be applied to any capsule in the network. RPV offers a method of not only identifying the entities of an input image that impact classification (see sections 5.1 and 5.3), but also to identify the entities detected by intermediate level capsules (see sections 5.2). Our method fills the domain where using a capsule network model is necessary as the training set is too small to train a CNN, and model interpretation is important. Two such examples are provided in section 5, namely a dataset of astronomical images and a dataset of brain tumor MRI scans. Small data sets can leverage generative adversarial networks (GANs) and diffusion models to generate synthetic images that augment the training data ([3], [23]) to mitigate over-fitting during training. However, synthetic training images may not always represent the target classes well, especially when the class is extremely rare, potentially leading to inaccurate models [10]. As a result, relying on less data-hungry models, like capsule networks, may be necessary if synthetic data generation is not a viable option.

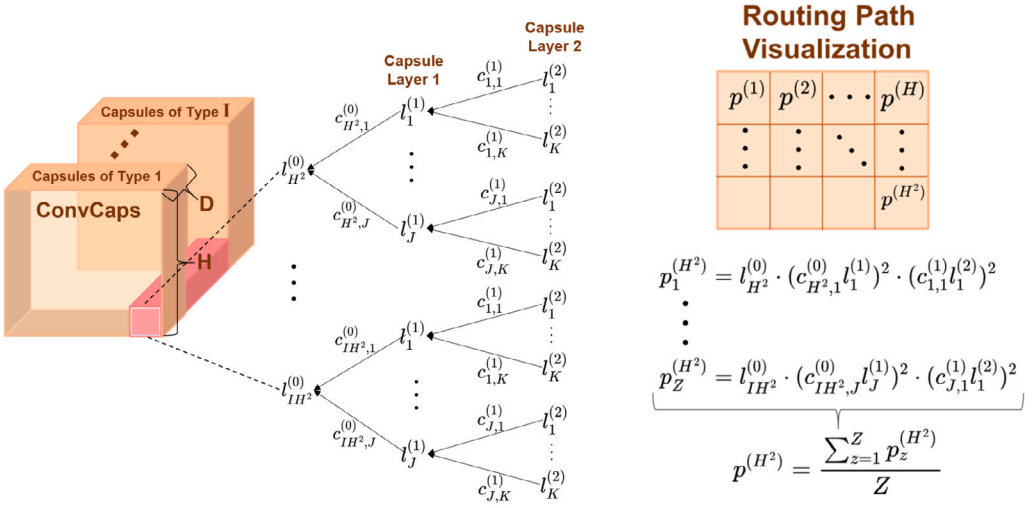
2. Background

In both CNNs and capsule networks, the image is passed through one or more convolutional layers to obtain a tensor representation of the image that is better suited for classification ([32], [21]). Each element of the tensor is called a neuron. The tensor is referred to as a feature map in CNNs and as a convolutional capsule (ConvCaps) layer in capsule networks.

In CNNs, the tensor representation of the image is flattened to a vector of neurons and fed through at least one fully connected layer for classification [21]. Every neuron in the tensor is connected to every neuron in the layer above. The edge weights connecting neurons in adjacent layers are learned during training, and do not change from image to image.

In capsule networks, the tensor representation of the image is composed of vectors, called capsules [32]. A capsule is a group of neurons that detects a specific entity in the image, and encodes the instantiation parameters of the entity if it is present. An entity is defined as an object or object part, and the instantiation parameters can include information on the position, size, or orientation of an entity. A capsule can have a Euclidean length of at most one, and is active if its length is close to one. The length of capsule i in layer q is denoted as $l_i^{(q)}$. A capsule's length can be viewed as the probability that the entity associated with that capsule is present. The tensor representation of the image is flattened to a vector of capsules which are routed to at least one layer of capsules for classification. For our capsule networks, the position of the capsules in the ConvCaps layer is not taken into account when routing. Every capsule in the tensor is routed to every capsule in the layer above. The coupling coefficients determine the extent to which each capsule is routed to each parent capsule. The coupling coefficient between capsule i in layer q and parent capsule j is denoted by $c_{ij}^{(q)}$. These coefficients are not learned during training and change from image to image.

The coupling coefficients are calculated by a routing algorithm which takes as input the set of transformation matrices that are learned during training. There are several variants of the routing algorithm, such as EM routing [16], 3D convolution based dynamic



(a) Tree diagram of routing paths (b) Path back-tracking for visualization

Fig. 1. Routing path visualization for a ConvCaps layer with grid size $H \times H$ and I capsule types, and two succeeding capsule layers containing a J and K number of capsules, respectively. In this example, only the pixel in position H^2, $p^{(H^2)}$, of the visualization image is calculated. The image is used to precisely localize the object that capsule 1 in capsule layer 2 detects.

routing algorithm [29], and approximate routing with master and aide interaction [22]. Here we use the routing algorithm proposed in Sabour et al. [32]. Each capsule in the tensor predicts the instantiation parameters of each parent capsule using the transformation matrices. The prediction of capsule i in layer q of parent capsule j is denoted by $\hat{\mathbf{u}}_{j|i}^{(q)}$. A capsule tends to send most of its output to parent capsules for which the prediction is similar to the predictions of the other capsules in the same layer. When the predictions of multiple capsules agree, a parent capsule becomes active, depending on the architecture of the capsule network. The input of a parent capsule is given by

$$\mathbf{s}_j^{(q+1)} = \sum_i c_{ij}^{(q)} \hat{\mathbf{u}}_{j|i}^{(q)}, \quad i = 1, ..., IH^2 \quad \& \quad j = 1, ..., J. \tag{1}$$

Fig. 1 (a) depicts the routing paths taken by capsules in a network that contains one ConvCaps layer and two capsule layers.

3. Related works

Popular visualization techniques to improve the interpretability and explainability of CNN model include Grad-CAM, Grad-CAM++, Score-CAM, and XGrad-CAM [5,11,33,37]. Grad-CAM++ and its predecessor Grad-CAM both use gradient-based localization techniques to highlight key areas of interest by leveraging the gradients that flow through CNN layers [5,33]. These approaches have become widely adopted for their ability to provide insight into what a CNN is "looking at" when making predictions.

More recently, methods such as Score-CAM have shifted away from gradient reliance due to the inherent noisiness of gradients. Other challenges include gradient vanishing, which can occur due to either saturation in the sigmoid activation function or the flat zero-gradient region in the ReLU activation functions [37]. Score-CAM uses a perturbation-based approach instead, improving stability and interpretability by directly modifying feature maps. Similarly, XGrad-CAM adopts an axiomatic framework, making it more adaptable to a variety of CNN architectures through its general layer-selection methodology, further enhancing visualization flexibility [11].

Black-box saliency models such as SHAP, LIME, RISE, and C-RISE have also gained prominence for their ability to provide model-agnostic interpretability [24,28,31]. These methods typically operate by perturbing inputs and analyzing the resulting changes in output. They offer a way to explain predictions from models for which internal representations are difficult to access. However, while effective for CNNs, They can introduce computational overhead and may lack the spatial granularity provided by CNN-specific visualization methods like Grad-CAM.

Despite the advancements in explainability for CNNs, these methods are primarily designed to work on convolutional layers and do not translate well to capsule networks. Capsule layers differ fundamentally from convolutional layers due to their dynamic routing mechanism, which clusters activations into capsules representing entities and their properties. Adapting existing techniques for capsule networks would require significant modifications, as they are not inherently designed to account for the routing dynamics and hierarchical entity representation that capsule layers provide. This lack of adaptability underscores the need for visualization methods specifically tailored to capsule networks. RPV leverages the routing mechanism unique to capsule networks and provides insights into the learned representations and decision-making process of capsule layers.

4. Methods

4.1. Weight sharing

The feature map is convolved with a set of $N = I \cdot D$ filters to produce a ConvCaps layer of size $H \times H \times N$, which contains $H \cdot H \cdot I$ capsules each of dimension D. Each filter that was used to create the ConvCaps layer detects a particular feature in the feature map. Each capsule in an $H \times H$ grid was created using the same subset of D filters, and as a result, the capsules in a grid detect the same entity. The ConvCaps layer can be thought of as containing I capsule types, each of which detects a certain entity.

The convolution operation contains a form of weight sharing such that local features of the input are detected by the same filter(s). This weight sharing is natural because, as one could argue, the same criteria for detecting features in one location should be applied to other locations of the image [21]. The form of weight sharing in the convolution operation is analogous to the weight sharing we propose for routing in capsule networks.

Analogously, it seems natural to subject capsules of the same type to the same criteria when being routed to capsules in the layer above. Hence, capsules of the same type should use the same transformation matrices for predicting entities in the layer above. This weight sharing not only leads to a simpler model, with fewer trainable weights, but also results in a more flexible model in which the number of weights no longer depends on the size of the input image. Consequently, the capsule network can handle images of any size. For instance, consider a ConvCaps layer A. Suppose the layer succeeding A contains J capsules. Without weight sharing, there is a total of $H \times H \times I \times J$ transformation matrices associated with ConvCaps layer A, and a set of J transformation matrices per capsule in A. With weight sharing, there is a total of only $I \times J$ transformation matrices associated with ConvCaps layer A, along with a set of J transformation matrices per capsule type in A.

In Sabour et al. [32], each capsule in the ConvCaps layer, also known as the primary capsule layer, was given a unique set of transformation matrices. In this work, the capsules in the primary ConvCaps layer that are of the same type share a set of transformation matrices. Our weight sharing technique does not take into account the positions of the capsules and, though developed independently, is similar to that in LaLonde and Bagci [19] for image segmentation.

4.2. Routing path visualization

In a well-trained capsule network, when predicting the class of an image, capsules in the ConvCaps layer containing information relevant to the prediction become active. However, theoretically, some capsules with irrelevant information may also become active due to the network's inability to identify large entities in the early layers. Active capsules containing irrelevant information should be routed to at least one inactive capsule in succeeding layers as inactive capsules weakly affect the final prediction. In more detail, the predictions, $\hat{\mathbf{u}}_{j|i}^{(q)}$, of inactive capsules have small lengths, and as a result, the individual elements of $\hat{\mathbf{u}}_{j|i}^{(q)}$ are small. Equation (1) suggests that the predictions of inactive capsules contribute very little to $\mathbf{s}_j^{(q+1)}$ as $c_{ij}^{(q)}\hat{\mathbf{u}}_{j|i}^{(q)}$ will be small if $||\hat{\mathbf{u}}_{j|i}^{(q)}||$ is small.

Moreover, the ConvCaps layer preserves the relative positions of objects in the image. For example, the information of an object in the upper right corner of an image will remain in the upper right corner in the ConvCaps layer. Hence, if the predicted class is the existence of an object then the ConvCaps that are positioned relative to the object's position in the image should become active as they contain relevant information. Intuitively, in a well-trained network these relevant capsules are routed to the capsule in the final layer that predicts the existence of the object, and all other active capsules are routed to inactive capsules in succeeding layers. Ergo, the routing path can distinguish between active capsules containing relevant and irrelevant information. Inactive capsules are identified using the routing path diagram (see Fig. 1 a), allowing us to filter them out and precisely localize the object by focusing only on the relevant active capsules in the ConvCaps layer. This filtering out procedure is referred to as *routing path visualization*. In turn, the entities that capsules detect can be interpreted, and situations where the network may fail can be identified.

Fig. 1 shows an example of RPV for a ConvCaps layer with grid size $H \times H$ and I capsule types, and two succeeding capsule layers containing J and K capsules, respectively. In this example, the entity that is detected by capsule 1 in the last capsule layer is visualized. This is achieved by calculating each pixel, $p^{(h)}$, of the RPV image and given by

$$p^{(h)} = \frac{\sum_{z=1}^{Z} p_z^{(h)}}{Z}, \tag{2}$$

where $h = 1, ..., H^2$, and Z is the total number of ways the capsules in grid cell h of the ConvCaps layer can be routed to capsule 1 in the last capsule layer. Further,

$$p_z^{(h)} = l_i^{(0)} \cdot (c_{ij}^{(0)} l_j^{(1)})^2 \cdot (c_{j1}^{(1)} l_1^{(2)})^2. \tag{3}$$

Since the lengths of the capsules and the coupling coefficients are always less than 1, $p_z^{(h)}$ is always less than 1.

Equation (3) measures how much of capsule i in the ConvCaps layer is routed to capsule 1 in the second capsule layer through a particular path. The term $l_i^{(0)}$ will be large if capsule i in the ConvCaps layer is active, the term $(c_{ij}^{(0)} l_j^{(1)})^2$ will be large if capsule i in the ConvCaps layer is mostly routed to an active parent capsule j, and the term $(c_{j1}^{(1)} l_1^{(2)})^2$ will be large if capsule j in the first capsule layer is mostly routed to capsule 1 in the second capsule layer. The squaring amplifies the difference between active and inactive capsules. Equation (2) normalizes this value over all paths of all capsules in grid cell h.

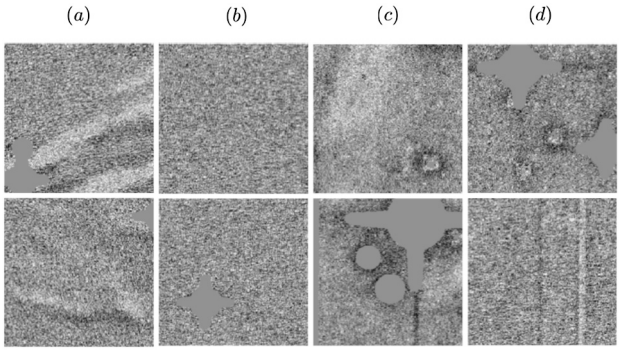


Fig. 2. Examples of types of entities present in the CFHT dataset. Column (a) is of images that clearly contain at least one light echo, column (b) is of images that clearly do not contain at least one light echo, column (c) is of images that contain light echoes and artifacts, and column (d) is of images that contain entities which have similar characteristics to light echoes.

RPV recycles the coupling coefficients and capsule lengths calculated during prediction to estimate how much information is passed by a set of I ConvCaps capsules, which correspond to a region of the input image, to any capsule in a subsequent layer. On the other hand, the importance weights produced in Grad-Cam are averaged over the spatial dimensions of a feature map resulting in the inability of Grad-Cam to properly identify which regions of the input image were used to make the prediction, the units of each feature map may have different "importance".

In Chen et al. [6] it was concluded that the routing algorithm, which calculates the coupling coefficients, is unnecessary as the coupling coefficients can be learned implicitly by the network. Paik et al. [26] came to a similar conclusion as they found that removing the routing algorithm altogether and uniformly assigning the coupling coefficients gives similar performance to including a routing algorithm. This suggests that a capsule network without a routing algorithm can achieve top tier performance. It will also train more quickly as the routing algorithm is a computationally expensive step. However, using a routing algorithm where the link strengths between capsules are known permits improved model interpretability. Furthermore, it is possible that the routing algorithm allows a capsule network to be constructed with fewer weights, as the network does not need to learn how to calculate the link strengths between capsules. The aforementioned is valued when interpretability is important and the training set is very small.

5. Experiments

We applied RPV to interpret capsule network classifications from three real world datasets, labeled MNIST [20], CFHT [25], and BRAIN [7]. These latter datasets provide more realistic applications for which the training size is small and RPV may be needed.

5.1. Description of datasets

The MNIST data consists of images of single digits 0 through 9. This example illustrates how RPV can help interpret entities in images and how our notion of interpretability differs from that of previous work ([32], [34]). The MNIST capsule network had a similar architecture to the network in Sabour et al. [32], except that the network used filters of size 3 instead of 9 because the ConvCaps layer does not preserve the relative positions of objects when it has a grid size that is much smaller than the size of the image. Training was stopped when the validation accuracy was good as opposed to state-of-the-art as the objective was to evaluate RPV, not prediction accuracy per se. The MNIST dataset comprises a total of 70,000 images, split into 50,000 for training, 10,000 for validation, and 10,000 for testing. Each image is a 28×28 grayscale image representing handwritten digits. Additionally, the dataset is relatively balanced across the ten classes, ensuring that each digit (0–9) is fairly represented.

The CFHT data requires classifying astronomical images as containing or not containing at least one supernova light echo. Images were obtained from a 2011 astronomical survey performed by the Canada-France-Hawaii Telescope, a charge-coupled device mosaic imager, MegaCam. The primary objective was to search for supernova light echoes, locations on the sky where interstellar dust has been illuminated by the brief outburst of a supernova [25]. Such locations are highlighted in difference images (Fig. 2) based on the difference between two exposures obtained at the same telescope pointing (i.e., same celestial coordinates) at different times, after compensating for differences in depth and image quality.

Difference imaging can also produce a variety of artifacts, some of which are objects that mimic the characteristics of light echoes [25]. Artifacts can be caused by but are not limited to effects of telescopic pointing offsets, light from bright stars, scattering of optics and the imager, diffraction, and cosmic rays (Fig. 2 $(c), (d)$). It is of interest to find difference images that contain entities that have a high probability of being a light echo, and to localize such entities since visual examination of large numbers of difference images is far too time-consuming. Supernovae light echoes reveal themselves in difference images as roughly equally-bright positive and negative flux diffuse features, displaced in by tens of pixels in a given direction (Fig. 2 (a)).

The CFHT data were derived from a survey of approximately 13,000 2048×4612 difference images that were manually classified over the course of one year. Only 22 of these images were found to contain at least one supernova light echo [25] and were used to produce the CFHT data. In particular, the light echoes in these 22 light echo containing difference images were manually masked

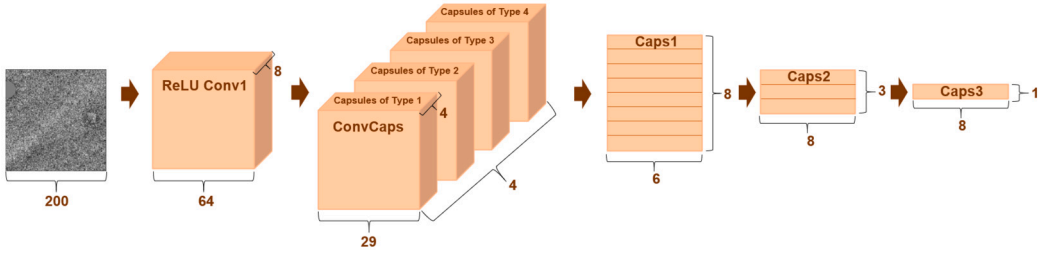


Fig. 3. The capsule network architecture that was trained using the CFHT dataset.

and then cut into 4885 difference images of size 200×200. If a cropped image contained at least 2500 pixels of mask then it was classified as containing a light echo. From among these 4885 cropped images, 175 of the light echo containing images and 175 of the remaining images (which may contain other artifacts) were randomly selected to form the final CFHT dataset of 350 images sized 200×200. The dataset consists of images and their respective binary labels indicating whether the image does or does not contain at least one light echo. In this analysis, the data was randomly split into a training set of 250 images, and a validation and test set of 50 images each. The dataset was not augmented with rotations and transformations of the original cropped images because the part-whole relationship learned by capsule networks allows for them to generalize well to novel viewpoints [15].

The CFHT capsule network architecture is shown in Fig. 3. To elaborate, a 200×200 image was convolved with a set of eight 9×9 filters using a stride of three to produce the feature map. The feature map was convolved with a set of sixteen 5×5 filters using a stride of 2 to produce the ConvCaps layer. The subsequent capsule layers contain 8, 3, and 1 capsules of dimension 6, 8, and 8, respectively. The network was trained such that the length of the single capsule in capsule layer 3 was approximately 1 when at least one supernova light echo was present in the image, and approximately 0 otherwise. This is achieved by minimizing the margin loss function, L, and given by

$$L = T \max(0, m^+ - ||\mathbf{v}||)^2 + \lambda(1 - T) \max(0, ||\mathbf{v}|| - m^-)^2,$$

where $T = 1$ if at least one light echo is present in the image, $m^+ = 0.9$, $||\mathbf{v}||$ is the length of the capsule in capsule layer 3, $\lambda = 0.5$, and $m^- = 0.1$ [32].

The BRAIN data requires classifying MRI images of the brain as containing one of three types of tumors. The U.S. Food and Drug Administration (FDA) requires safety validation for software as a medical device (SaMD). In fact, a rigorous analysis showing that a SaMD correctly processes input data and generates accurate, reliable and precise output data is conducted before approval by the FDA [36]. The low number of trainable weights in capsule networks relative to CNNs makes them ideal for classifying medical images as medical training sets are often small in size, which may result in an over-fitted CNN model. RPV with capsule networks is a suitable choice for SaMDs designed for image classification as it can interpret how the input is processed within intermediate layers of the network (see Section 4.2 CFHT) and at the output layer (see Section 4.1 MNIST). For demonstration, a capsule network model was trained on a small dataset of brain tumor MRI images, and the model was interpreted with RPV.

The MRI brain dataset consists of a total of 3064 2D slices of size 512×512 from 233 patients. The dataset also contains masks of each image and coarse tumor boundaries that highlight the region of the tumor. More information about the dataset can be found in Cheng et al. [7]. Of the 3064 slices, 708 are of meningiomas, 1426 are of gliomas, and 930 are of pituitary tumors. Gliomas are malignant, whereas pituitary tumors are benign and meningiomas are mostly benign. The data were split on a patient-wise basis. Only a total of 378 images (126 of each tumor type) were used to train the capsule network to show the applicability of capsule networks and RPV for very small datasets. The validation set was comprised of 162 images that were equally class balanced, and the test set contained 2509 images that were not class balanced. As with the MNIST example, a search was not conducted to find a capsule network architecture that optimized accuracy because the purpose of this experiment was to interpret how the network classifies images as opposed to producing state-of-the-art results.

The original images of size 512×512 were down-sampled to size 128×128, similar to Afshar et al. [2] and Sultan et al. [35]. The BRAIN architecture was also trained on a class balanced set of 1603 images to ensure that the subpar accuracy, when compared to Afshar et al. [2], was not due to a small training set. It was found that when the BRAIN architecture was trained using the larger dataset the network converged to a similar loss and test accuracy, suggesting that the network architecture was limiting the accuracy as opposed to the training set.

The architectures for the three models are given in Table 1, where we see that model complexity was highest for the 10-class MNIST data (633,600 weights), followed by the 3-class BRAIN data (511,556 weights) and then the 2-class CFHT data (5,984 weights). For the MNIST and BRAIN datasets, images were classified to contain the digit or tumor type associated with the capsule having largest length. For the CFHT dataset, an image was classified as containing a light echo if the length of the light-echo-detecting capsule was greater than 0.5; this cutoff was found to work well in practice. Accuracy was then calculated as follows,

$$\frac{\texttt{True Positives} + \texttt{True Negatives}}{\texttt{No. of Images in Set}} \times 100\%.$$

The capsules in the final capsule layer were interpreted for the MNIST and BRAIN capsule networks, while the capsules in the second layer were interpreted for the CFHT capsule network. This intermediate capsule layer was chosen because the final capsule layer

Table 1
Capsule network architecture, defined by the feature map (F, N, S), ConvCaps layer (F, I, D, S), and capsule layers (C, D), where F is the length of each filter, N is the number of filters used, S is the stride, I is the number of capsule types, D is the number of dimensions per capsule, and C is the number of capsules in each capsule layer. Weights correspond to the total number of trainable weights in each model.

Data	Convolutional Layers		Capsule Layers			Weights
	Feature Map	ConvCaps	1	2	3	
MNIST	(3, 256, 1)	(3, 32, 8, 1)	(10, 16)			633,600
CFHT	(9, 8, 3)	(5, 4, 4, 2)	(8, 6)	(3, 8)	(1, 8)	5,984
BRAIN	(5, 16, 2)	(5, 42, 10, 2)	(36, 12)	(24, 14)	(3, 16)	511,556

Table 2
Prediction accuracy for the validation and test sets. Duration is the time taken to train each model. Steps is the total number of times the weights were updated during training, and Time shows the time taken to classify a single image, averaged over 50 images.

Data	Weights	Accuracy		Training		Time
		Val	Test	Duration	Steps	
MNIST	633,600	92%	91%	12 h	1600	0.11 s
CFHT	5,984	88%	92%	< 1 h	9200	0.02 s
BRAIN	511,556	70%	67%	24 h	5000	9.22 s

contains a single capsule. As such, the RPV of the final layer capsule would be a simple addition of the routing path visualizations of the capsules in the second capsule layer.

Each model was trained using a single NVIDIA Pascal P100 GPU on TensorFlow [1] with the Adam optimizer [17]. Margin loss was minimized during training without an additional reconstruction loss [32]. Although the resulting capsules in the final layer may not encode all instantiation parameters of the class, RPV still works because it interprets the entity that a capsule itself detects, not what the individual dimensions of the capsule represent. It is important to note that reconstruction loss could act as a regularization term that encourages the model to learn a more optimal set of weights [32]. Although reconstruction loss may be used in our models to improve model accuracy, its absence should not interfere with RPV.

5.2. Experimental results

Summary statistics for each model are reported in Table 2, where Capsule networks for both MNIST and CFHT data showed above 90% test accuracy, though the average classification time for an MNIST image took 5 times longer than that of a CFHT image (0.11 s for MNIST vs 0.02 s for CFHT). The test accuracy of the BRAIN capsule network was lower (67%) and the time to classify a single image was higher (9.22 s) compared to the other datasets.

The RPV images from capsule layer 1 for a sample set of 5 MNIST images is shown in Fig. 4 and is representative of the results on all images in the test set. In general, the contour of the digit in an image was routed to the correct digit capsule, presumably because the contour of a digit provided the most useful information as all digits are the same color. Regions of the image containing the 2 and 0 were also routed to digit capsules 8 and 6, respectively (Fig. 4). The network, at times, seemed to have difficulty distinguishing a 2 from an 8, or a 0 from a 6.

For the CFHT data, Fig. 5 reveals how much of each region in an image was routed to each of the three capsules in capsule layer 2 on a sample of 5 images from the test set. These results are representative of the results on all 50 images in the test set. The brighter the region in the RPV image, the more that region was routed to that capsule. Light echoes appear brighter and darker than the (zero-mean) background, but do not have a specific shape. When groups of supernova light echoes were seen, they generally were linearly distributed across the difference image. It is clear that capsule 3 detected light echoes as it precisely localized them from the image. When a light echo was not present in the image, the RPV image for capsule 3 was mostly empty, even though the network was trained using only images and their respective class labels. Typically, a more supervised approach would be required to train a network for object localization, such as providing the set of bounding boxes or pixel coordinates that contain the object [30].

Capsule 1 appeared to detect a combination of light echoes and artifacts, per Figs. 5(a) and (b). However, capsule 2 appeared to detect the difference in brightness around bright stars, per Figs. 5(c) and (e), because the RPV image was mostly empty in the absence a bright star (Figs. 5(a), (b), and (d)). Consequently, the model may fail when given difference images containing bright stars. In other words, all capsules in capsule layer 2 are routed to the single capsule in capsule layer 3 which should be detecting light echoes. Since the light from bright stars is being routed to the light-echo-detecting capsule, the light-echo-detecting capsule may use that information to incorrectly make a prediction. This is an instance where it may be advantageous to manually disable capsule 2 to improve model robustness on future data.

The confusion matrix associated with the BRAIN test set, shows that glioma tumors were often incorrectly classified as meningioma (Table 3). RPV was used on the test set to interpret the entities detected by the capsules in the last capsule layer. The results from a

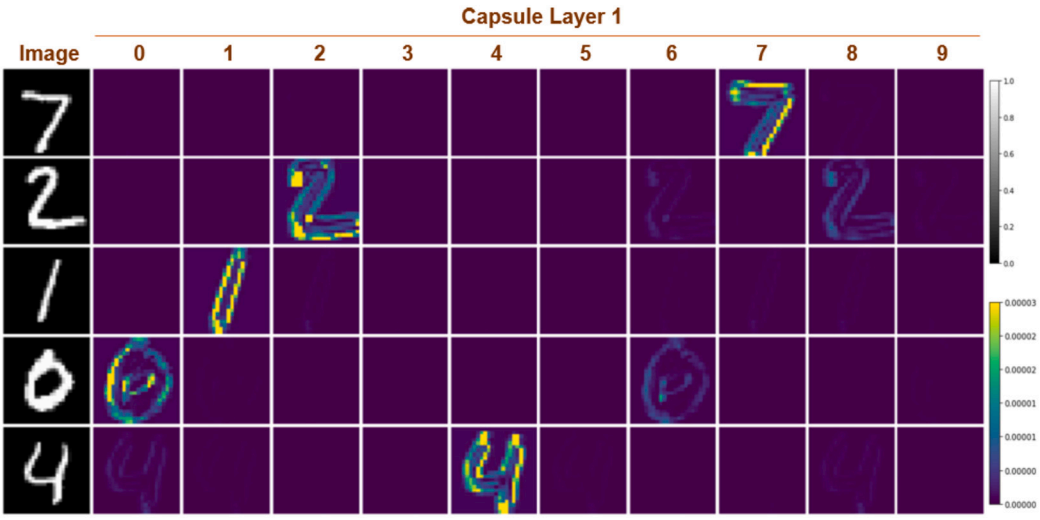


Fig. 4. Routing path visualization to interpret the entities that the capsules in capsule layer 1 detect. Sample test images are of size 28×28, and RPV images of capsule layer 1 are of size 24×24.

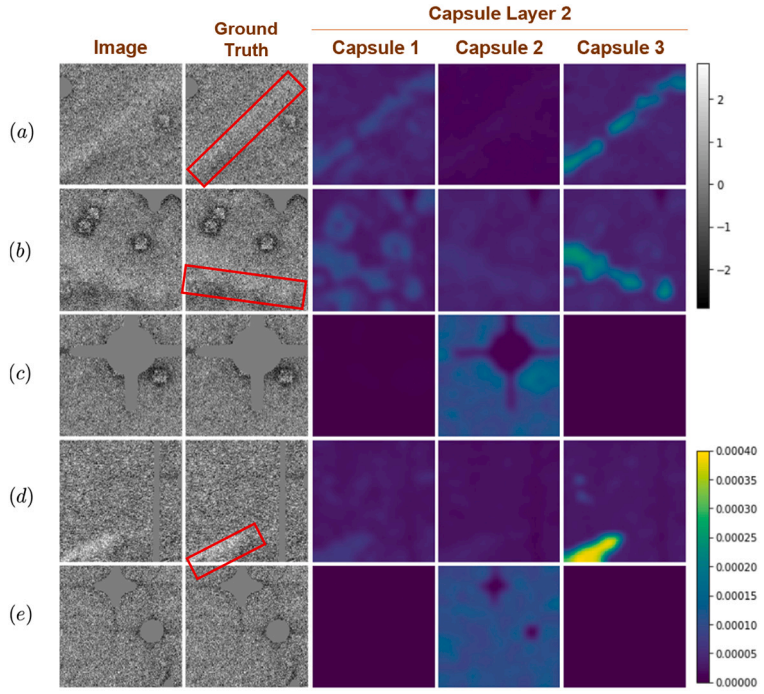


Fig. 5. Routing path visualization of entities detected in capsule layer 2 detect for sample images from test set. Sample test images are of size 200×200, and RPV images of capsule layer 2 are of size 29×29. Images (a), (b), and (d) contain light echoes; (c) and (e) contain stars. All images show varying amounts of interstellar dust or artifacts. The red rectangles indicate locations of visually-identified light echo features in the difference images. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 3
Confusion matrix displaying the classification results of the test set.

True Class	Predicted Class		
	Meningioma	Glioma	Pituitary Tumor
Meningioma	377	71	80
Glioma	427	711	108
Pituitary Tumor	66	74	595

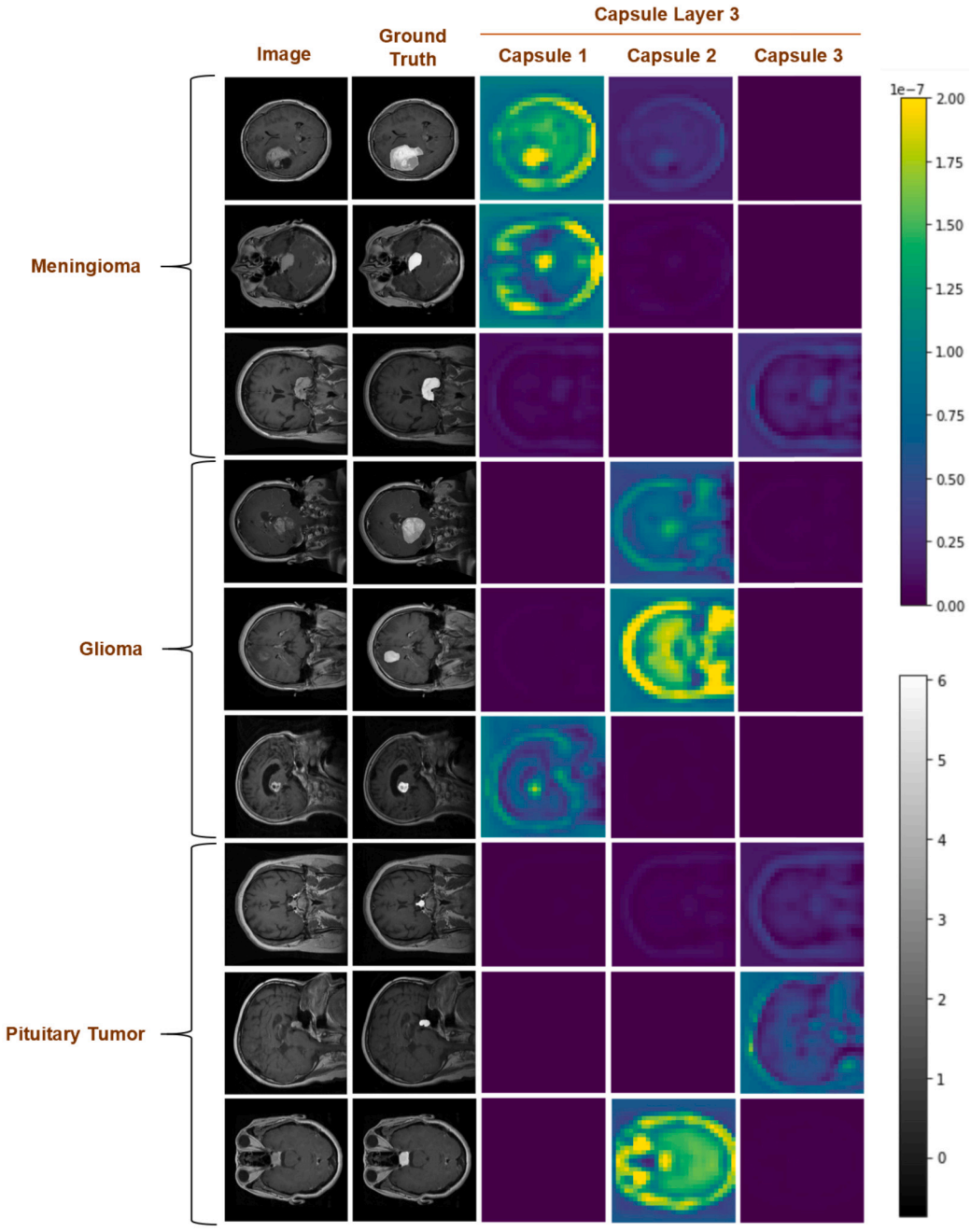


Fig. 6. Routing path visualization to interpret the entities detected by capsules in capsule layer 3. Sample images from test set of size 512×512, and RPV images of size 29×29. The first three rows are ground truth examples of meningioma, the next three are of glioma, and the last three are of pituitary tumors. The ground truth column highlights the location of the tumor.

sample set of 9 images are shown in Fig. 6, and are representative of the results on all images in the test set. In capsule layer 3, capsule 1 detected meningioma, capsule 2 detected glioma, and capsule 3 detected pituitary tumors. The capsule with the brightest RPV was the predicted class. For example, row 3 displays an incorrect classification of meningioma as a pituitary tumor; row 6 displays an incorrect classification of glioma as meningioma; row 9 displays an incorrect classification of a pituitary tumor as glioma; and the remaining rows display correct classifications.

From the ground truth and routing path visualizations in Fig. 6, it is clear that the network used the location of the tumor to predict tumor type, with the exception of row 5, where the tumor was not very visible in the input image. Ideally, the capsule network should use features localized to the region of the tumor in an input image for classification. However, RPV shows that the network weighed the outline of the skull just as much as the tumor location when classifying the images. Fig. 6 suggests that the network may have

improperly learned to use the slice at which the MRI was imaged to predict the type of tumor. It is possible that certain tumor types are often imaged at certain slices instead of random slices, thereby resulting in a flawed dataset.

To mitigate such possible sources of misclassification, images could be generated at random slices, thus influencing the model to not correlate certain slices with certain brain tumors. Alternatively, MRIs of brains that do not contain tumors could be imaged at the same slices that were used to generate the BRAIN dataset and then incorporated into the training data, again, influencing the model to not correlate certain slices with certain brain tumors. Regardless, it is evident that RPV can be used to interpret the predicted classes from a capsule network and identify the nature and potential sources of misclassification.

6. Visualization technique comparison

To the best of our knowledge, there is no existing visualization method specifically designed for capsule networks to interpret capsule layers. To validate the effectiveness of RPV, we compared it to state-of-the-art visualization techniques, including GradCAM, GradCAM++, and Score-CAM [5,33,37]. These latter visualizations techniques are readily available in the `tf_keras_visualization` Python package and are tailored for CNNs. Comparisons were made using the Cats and Dogs dataset on Kaggle [8], which provides a practical real-world example involving the classification of visually similar categories that have familiar, nuanced distinctions.

6.1. Dataset selection

The Cats and Dogs dataset contains 25,000 labeled images of cats and dogs, from which 600 training images, 200 validation images, and 600 test images, were randomly selected such that there would be an even split between cats and dogs in each subset. Cats and dogs share several anatomical features—such as ears, eyes, mouths, and noses—but exhibit distinct variations in the structure and orientation of these features. This characteristic makes the dataset ideal for evaluating how models identify and focus on specific features during classification tasks. The similarity between the classes also allows for a thorough visual assessment of the interpretability of the methods.

All images were first converted to grayscale to reduce computational complexity and ensure consistency in color channels. They were then resized to $96 \times 96 \times 1$ using bicubic interpolation. This resizing ensured uniform input dimensions across the dataset while balancing the need to retain sufficient visual detail. Each pixel value was normalized by dividing by 255, ensuring pixel intensities ranged from 0 to 1, per standard preprocessing for most neural network architectures. A capsule network and a CNN were each fit to the preprocessed training data.

Given the susceptibility of CNNs to overfitting on small datasets [9], data augmentation was used exclusively for the CNN model to simulate additional variability in the training data. This augmentation included random rotations, shifts, shear transformations, zooming, and horizontal flipping. These techniques effectively expanded the dataset size and variability, providing a slight advantage to the CNN model. In contrast, capsule networks have demonstrated robustness when trained on limited labeled datasets [18], negating the immediate need for similar augmentation. This distinction ensures that both models are trained under conditions that align with their respective strengths, fostering a fairer comparison of their performance and visualization capabilities.

6.2. Model selection

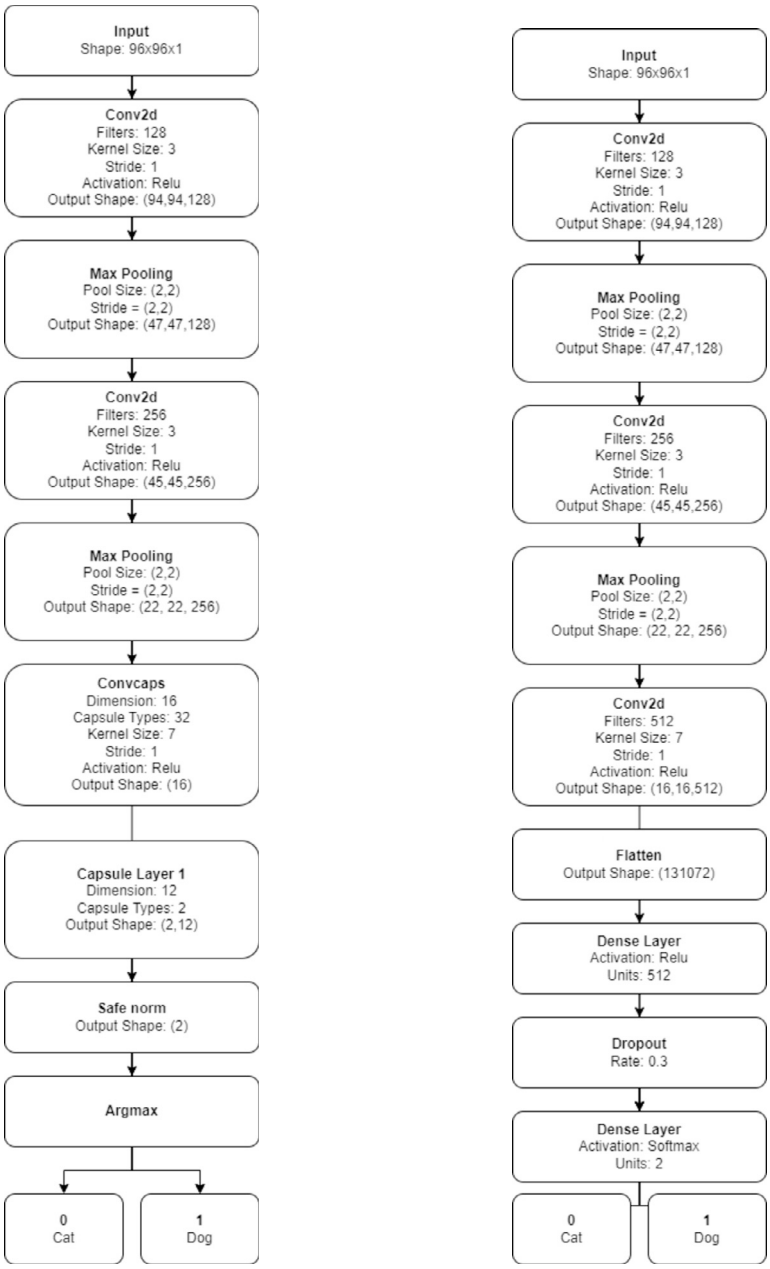
To ensure a fair comparison, we designed both models to share a similar CNN backbone with 3 convolutional layers and 2 max pooling layers, as depicted in Fig. 7 — an unconventional choice for capsule architectures. The inclusion of more than one convolutional layer is known to reduce feature map sizes and the number of parameters, thereby enabling the use of larger input images without encountering resource limitations [4]. Indeed, using 3 convolutional layers facilitated striking a balance between feature extraction and computational efficiency. For the capsule network, the third convolutional layer was the ConvCaps layer. There was also a final capsule layer with six routing iterations. In short, the primary difference between the two network architectures lay in the CNN's dense layers for classification versus the capsule network's capsule layer, which predicts the existence of entities through its routing mechanism.

For the loss function, the capsule network employed margin loss, while the CNN used categorical cross-entropy. Both models used the Adam optimizer with a learning rate of 0.0001, and the final classification layer for both architectures employed a softmax activation function. The capsule network was trained with a batch size of 8, while the CNN used a batch size of 16, as these configurations provided optimal results for the dataset. To achieve fair and robust results that minimized extraneous variables, both models were trained until the validation loss plateaued for 20 consecutive epochs.

Despite having significantly fewer parameters, the capsule network achieved superior accuracy on the Cats and Dogs dataset. This demonstrates the effectiveness of capsule layers in capturing hierarchical spatial relationships, particularly in scenarios where training data is limited. These results provide a strong foundation for evaluating the utility of RPV in visualizing the decision-making process within capsule networks, bridging the gap left by traditional CNN-centric visualization methods.

6.3. Visualization speed comparison

First, the visualization methods were compared with respect to their computational efficiency using two benchmarks: the total execution time, both including and excluding inference time. All tests were performed sequentially on a local machine using Tensor-



(a) Capsule Network Model (b) CNN Model

Fig. 7. The capsule network model with 6,731,776 trainable parameters, and CNN model with 73,829,890 trainable parameters. The capsule network achieved a test accuracy of 74.50% and a test loss of 0.16, compared to the CNN's test accuracy of 70.50% and test loss of 0.58.

Flow with the DirectML package and an AMD RX 5700 GPU (8 GB). The execution times for the CNN visualization methods were measured using the `tf_keras_vis` library, starting from the moment each method was invoked until a result was returned.

Note that Grad-CAM and Grad-CAM++ rely on gradient extraction from input images, while Score-CAM generates outputs by masking inputs and computing predictions for each mask. To ensure fair timing comparisons, model inference time was included in the total duration for all methods. For RPV, the time spent specifically on its visualization process was isolated and measured, subtracting the baseline inference time to evaluate the algorithm's computational overhead.

As shown in Table 4, RPV consistently outperformed the other methods in speed. In the most challenging scenarios, RPV was 4.3 times faster than Grad-CAM on average. In the best-case scenario, RPV achieved a remarkable 85.3 times speed improvement over Score-CAM. These results highlight the efficiency of RPV in generating meaningful visualizations, especially for resource-constrained environments.

Table 4
Performance Comparison of Visualization Methods in Terms of Algorithm and Total Execution Time.

Visualization Method	Non-Inference Time (Avg. ± 1 SD)	Total Time (Avg. ± 1 SD)
RPV	0.0150 ± 0.0063	0.4583 ± 0.1917
GradCAM	0.0695 ± 0.0025	0.0736 ± 0.0030
GradCAM++	0.0723 ± 0.0026	0.0765 ± 0.0032
ScoreCAM	1.2805 ± 0.0246	1.6951 ± 0.0296

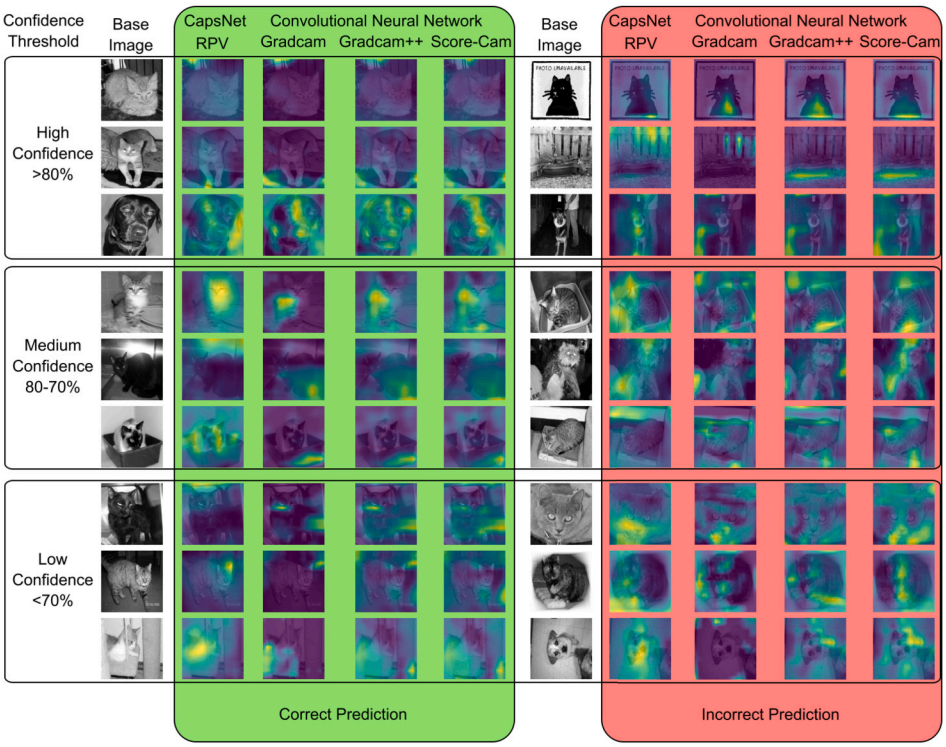


Fig. 8. Comparison of visualization masks generated by RPV and CNN visualization methods for Cat and Dog data. White columns show original image. Correctly classified images shown in green (left). Incorrectly classified images shown in red (right). Top, middle and bottom rows correspond to high, medium and low confidence predictions, respectively.

6.4. Visualization comparison

Fig. 8 presents a comparative analysis of the visualization masks generated by RPV and the CNN visualization methods, with key regions identified by the models being highlighted on several true positive and false positive classifications, by model confidence. Fig. 8 provides a detailed insight into how models interpret features differently. For example, in high-confidence correct predictions, features like the dog's nose, ears, and collar are consistently highlighted across methods, with RPV maintaining precision. In medium-confidence predictions, such as the bottom cat (row 6), RPV demonstrates the capsule network's ability to localize the cat's body, while CNN-based methods show the CNN's emphasis on less relevant background features, indicating differences in learned representations.

6.5. Masking and model confidence comparison

An image masking technique inspired by Grad-CAM++ [5] was used to challenge the performance of the visualization methods. Selective regions of the images were masked based on their corresponding activation maps, allowing for a systematic assessment of how different image areas contributed to the model's confidence. In particular, two masking strategies were employed:

(i) masking the regions highlighted by the activation map, and (ii) masking regions outside the highlighted areas. In the first strategy, where key regions identified by the activation map were removed, a significant drop in confidence was anticipated, as the model is forced to rely on less relevant features. Conversely, in the second strategy, where non-highlighted areas are masked, confidence is expected to remain relatively stable since these regions have minimal influence on the model's predictions.

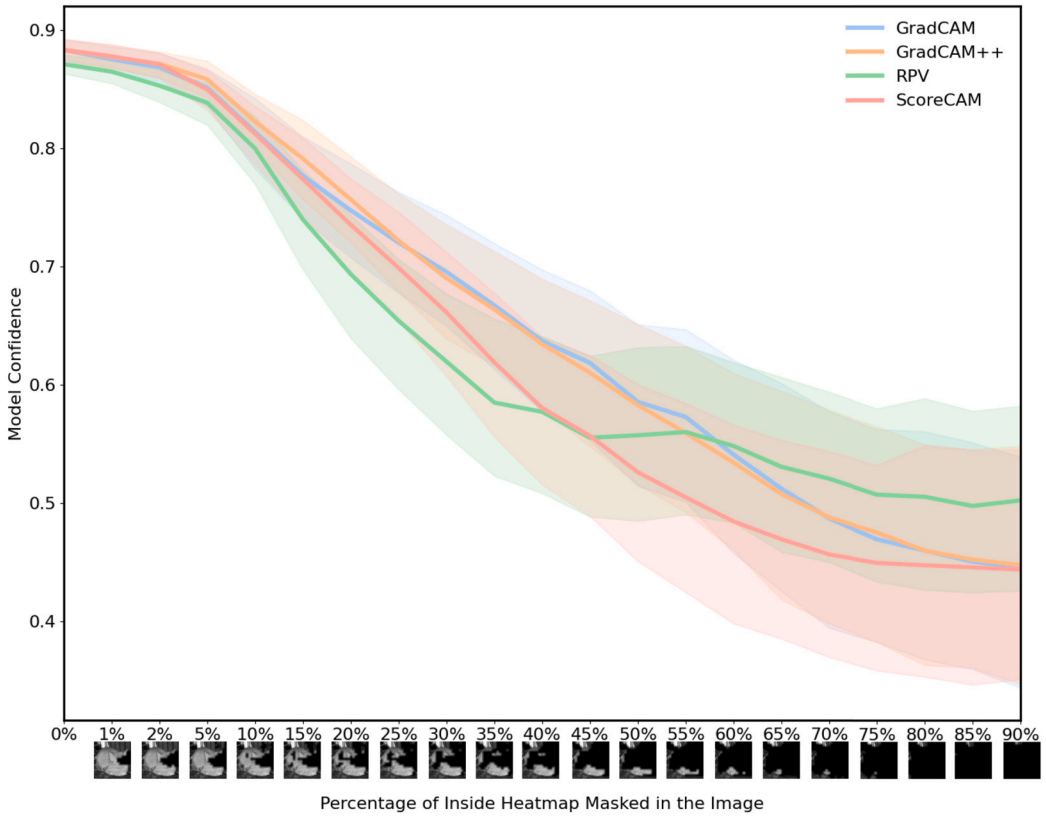


Fig. 9. Confidence drop when masking important regions. RPV demonstrates a sharper confidence decline compared to other methods, suggesting better localization of key features. For reference, image sequence along x-axis provides the binary mask for the cat in row 6 of Fig. 8.

We evaluated confidence changes across a range of percentage thresholds, and gradually increased the masked area. This approach provided insights into the robustness of highlighted regions and the model's sensitivity to partial occlusion. One-hot encoding was used to select pixels in the activation maps with pixel value above a given threshold. When applied to the original image, masked pixels corresponded to areas being occluded, effectively nullifying their contribution by multiplying pixel values by 0, and non-masked regions preserved their original pixel values. This binary masking ensured consistent behavior and avoided potential anomalies had weighted masking been applied instead.

Figs. 9 and 10 plot the model confidence associated with each visualization method against percent occlusion under the two masking strategies. When relevant regions were sequentially masked out (Fig. 9), RPV showed a sharper confidence decline compared to other methods. This finding supports its superior localization accuracy. On the other hand, when non-critical regions were sequentially masked out (Fig. 10), RPV showed a smaller drop in model confidence, even at higher thresholds, highlighting its robustness when masking non-critical areas. At higher percent occlusion values, where large portions of the image were masked, all methods exhibited greater variability due to the loss of contextual information.

In summary, these results show that RPV consistently performed better than or remained competitive with other visualization methods across a variety of thresholds. RPV effectively localized meaningful features by identifying and prioritizing critical regions within an image, thereby providing a more reliable and explainable interpretation of model behavior compared to CNN visualization methods.

7. Conclusions

Understanding how capsule networks make predictions is vital, especially in applications that directly impact human well-being, such as medical image analysis and autonomous systems. While previous efforts have largely focused on interpreting the features encoded within individual capsule dimensions [32], this study broadens the scope by introducing routing path visualization. RPV not only facilitates the interpretation of entities detected by intermediate and final layer capsules but also identifies situations where the network may fail, thereby enabling precise localization of predicted classes. This capability is particularly relevant for capsule networks trained on small datasets, as the weight-sharing mechanism reduces the risk of overfitting while maintaining robust performance.

The present analysis showed that RPV generates reliable visualizations when the network is trained to high accuracy, as demonstrated with the MNIST and CFHT datasets. However, even in scenarios with less optimal performance, such as the BRAIN dataset,

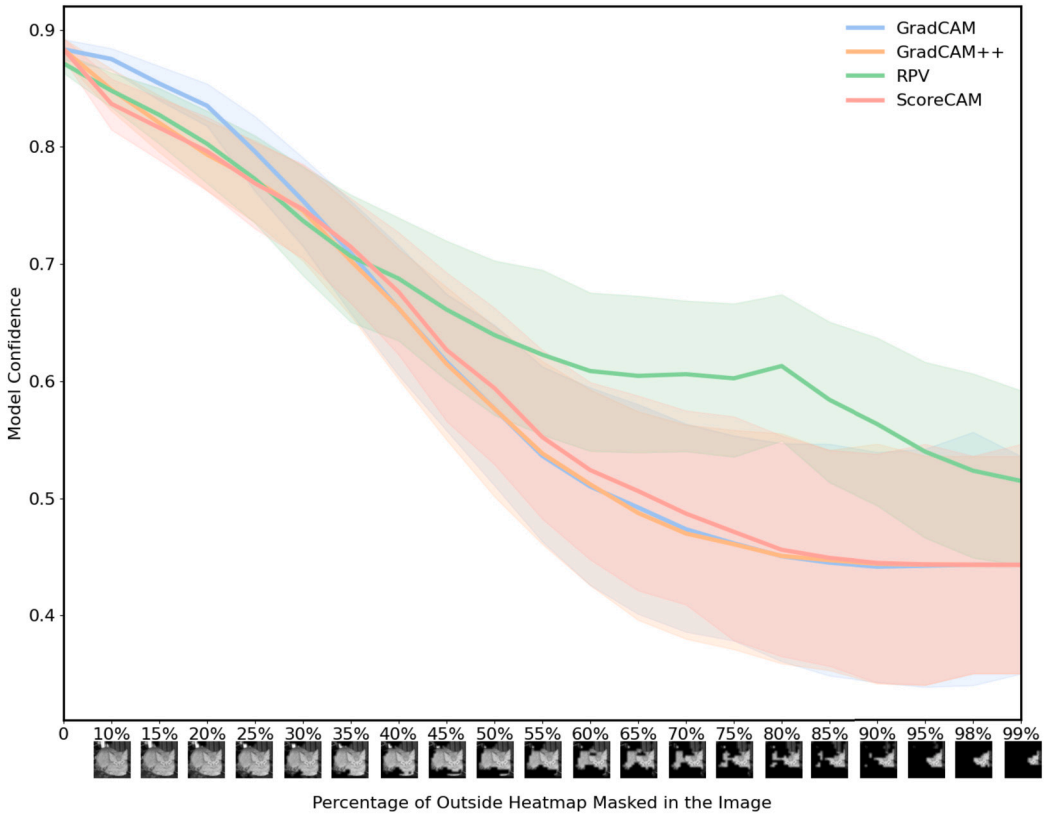


Fig. 10. Confidence drop when masking non-critical regions: RPV maintains a lower confidence drop after masking 40% of the background compared to other methods, reflecting its robustness. For reference, image sequence along x-axis provides the binary mask for the cat in row 6 of Fig. 8.

RPV provided meaningful explanations for model classifications that could inform improvements to the model itself or to the data collection. This versatility in making predictions or informing future predictions, highlights RPV's potential for broad applicability.

Future directions include integrating a decoder network to reconstruct images from intermediate capsule layers, which could enhance model performance by introducing a reconstruction loss as a regularization term. Exploring alternative routing mechanisms, such as the EM routing algorithm or Coordinate Addition [16], could further preserve positional information and improve network performance. Finally, extending the architecture to include multiple ConvCaps layers presents an intriguing avenue for enhancing the granularity and interpretability of routing path visualizations.

This work underscores the importance of developing interpretable AI models like capsule networks, particularly for critical applications, and lays a strong foundation for advancing both theoretical understanding and practical implementation in this domain.

CRediT authorship contribution statement

Amanjot Bhullar: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Michael Czomko:** Writing – review & editing, Writing – original draft, Visualization, Software, Formal analysis, Data curation, Conceptualization. **R. Ayesha Ali:** Writing – review & editing, Writing – original draft, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **Douglas L. Welch:** Writing – review & editing, Investigation, Data curation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Amanjot Bhullar reports was provided by University of Guelph.

Acknowledgement

This research was enabled in part by support provided by Compute Ontario (www.computeontario.ca) and Compute Canada (www.computecanada.ca). The images analyzed were based on observations obtained with MegaPrime/MegaCam, a joint project of CFHT and CEA/DAPNIA, at the Canada-France-Hawaii Telescope (CFHT) which is operated by the National Research Council (NRC)

of Canada, the Institut National des Sciences de l'Univers of the Centre National de la Recherche Scientifique of France, and the University of Hawaii.

Data availability

Data will be made available on request.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: large-scale machine learning on heterogeneous distributed systems, arXiv preprint, arXiv:1603.04467, 2016.
[2] P. Afshar, K.N. Plataniotis, A. Mohammadi, Capsule networks for brain tumor classification based on mri images and coarse tumor boundaries, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 1368–1372.
[3] K. Baek, H. Shim, Commonality in natural images rescues gans: pretraining gans with generic and privacy-free synthetic data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 7854–7864.
[4] B. Bian, D. Laughlin, K. Sato, Y. Hirotsu, Synthesis and structure of isolated l1/sub 0/ fept particles, IEEE Trans. Magn. 36 (5) (2000) 3021–3023, https://doi.org/10.1109/20.908663.
[5] A. Chattopadhay, A. Sarkar, P. Howlader, V.N. Balasubramanian, Grad-cam++: generalized gradient-based visual explanations for deep convolutional networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, pp. 839–847.
[6] Z. Chen, X. Li, C. Wang, D. Crandall, P-capsnets: a general form of convolutional neural networks, arXiv preprint, arXiv:1912.08367, 2019.
[7] J. Cheng, W. Huang, S. Cao, R. Yang, W. Yang, Z. Yun, Z. Wang, Q. Feng, Enhanced performance of brain tumor classification via tumor region augmentation and partition, PLoS ONE 10 (10) (2015) e0140381.
[8] W. Cukierski, Dogs vs. Cats, https://kaggle.com/competitions/dogs-vs-cats, 2013, Kaggle.
[9] H. Dishar, L. Muhammed, A review of the overfitting problem in convolution neural network and remedy approaches, J. Al-Qadisiyah Comput. Sci. Math. 15 (2023) 155, https://doi.org/10.29304/jqcm.2023.15.2.1240.
[10] A. Figueira, B. Vaz, Survey on synthetic data generation, evaluation methods and gans, Mathematics 10 (15) (2022) 2733.
[11] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, B. Li, Axiom-based grad-cam: towards accurate visualization and explanation of cnns, arXiv preprint, arXiv:2008.02312, 2020.
[12] E. Goceri, Analysis of capsule networks for image classification, in: International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, 2021.
[13] E. Goceri, Capsule neural networks in classification of skin lesions, in: International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, 2021, pp. 29–36.
[14] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, vol. 1, MIT Press, Cambridge, 2016.
[15] G.E. Hinton, A. Krizhevsky, S.D. Wang, Transforming auto-encoders, in: International Conference on Artificial Neural Networks, Springer, 2011, pp. 44–51.
[16] G.E. Hinton, S. Sabour, N. Frosst, Matrix capsules with em routing, in: International Conference on Learning Representations, 2018.
[17] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[18] M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty, B.Y. Edward, Capsule networks – a survey, J. King Saud Univ, Comput. Inf. Sci. (ISSN 1319-1578) 34 (1) (2022) 1295–1310, https://doi.org/10.1016/j.jksuci.2019.09.014, https://www.sciencedirect.com/science/article/pii/S1319157819309322.
[19] R. LaLonde, U. Bagci, Capsules for object segmentation, arXiv preprint, arXiv:1804.04241, 2018.
[20] Y. LeCun, The mnist database of handwritten digits, http://yann.lecun.com/exdb/mnist/, 1998.
[21] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[22] H. Li, X. Guo, B.D. Ouyang, X. Wang, Neural network encapsulation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 252–267.
[23] Y. Lu, H. Wang, W. Wei, Machine learning for synthetic data generation: a review, arXiv preprint, arXiv:2302.04062, 2023.
[24] S. Lundberg, A unified approach to interpreting model predictions, arXiv preprint, arXiv:1705.07874, 2017.
[25] B.J. McDonald, The search for supernova light echoes from the core-collapse supernovae of AD 1054 (crab) and AD 1181, master's thesis, McMaster University, 2012.
[26] I. Paik, T. Kwak, I. Kim, Capsule networks need an improved routing algorithm, in: Asian Conference on Machine Learning, PMLR, 2019, pp. 489–502.
[27] S. Pawan, J. Rajan, Capsule networks for image classification: a review, Neurocomputing 509 (2022) 102–120.
[28] V. Petsiuk, Rise: randomized input sampling for explanation of black-box models, arXiv preprint, arXiv:1806.07421, 2018.
[29] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, R. Rodrigo, Deepcaps: going deeper with capsule networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10725–10733.
[30] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[31] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[32] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in: Advances in Neural Information Processing Systems, 2017, pp. 3856–3866.
[33] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.
[34] A. Shahroudnejad, P. Afshar, K.N. Plataniotis, A. Mohammadi, Improved explainability of capsule networks: relevance path by agreement, in: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2018, pp. 549–553.
[35] H.H. Sultan, N.M. Salem, W. Al-Atabany, Multi-classification of brain tumor images using deep neural network, IEEE Access 7 (2019) 69215–69225.
[36] U.S. Food, Drug Administration, Proposed regulatory framework for modifications to artificial intelligence/machine learning (ai/ml)-based software as a medical device (samd). Online, accessed 29 January 2021.
[37] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, X. Hu, Score-cam: score-weighted visual explanations for convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 24–25.