Visual Explanations for Capsule Networks

Anonymous authors
Paper under double-blind review

Abstract

The limited availability of explainability methods for Capsule Networks (*CapsNets*) restricts their adoption in critical domains such as clinical practice or legal document analysis. Although *CapsNets* offer structured and interpretable representations, existing explanation methods have primarily focused on more traditional Convolutional Neural Networks (*CNNs*) and are not directly applicable to capsule-based architectures. To address this issue, we propose a general method (*Caps-CAM*), which generates attribution maps to justify the predictions made by feed-forward *CapsNet* architectures. Unlike prior explanation methods for *CapsNets* that adapt techniques originally designed for *CNNs*, *Caps-CAM* explicitly employs gradient information that reflects the relevance of each capsule to a class of interest. As the gradient can help highlight the most relevant capsules, each selected capsule activation map is weighted by its corresponding gradient. The final attribution heatmap is then generated as a linear combination of weighted activation maps based on their contribution to the target class. Experiments show that *Caps-CAM* serves as an explanation method for *CapsNets* and compare the results of this method with other state-of-the-art explanation techniques previously introduced for *CNNs*. Empirical comparisons w.r.t. state-of-the-art explanation techniques previously introduced for *CNNs*, show that *Caps-CAM* can effectively serve as an explanation method for *CapsNets*. Experiments on standard and real-application data sets show the effectiveness of the introduced *Caps-CAM*.

1 Introduction

Explainability for Deep Neural Networks (*DNN*) aims to enhance transparency by exposing aspects of the inference process that could be informative for its users. Most of these methods were originally designed to explain Convolutional Neural Networks (*CNNs*). Popular methods tailored to this type of architecture include gradient-based methods (Simonyan et al., 2013) and class activation mapping (*CAM*) (Zhou et al., 2016). Gradient-based methods propagate the gradient of a given class in order to emphasize the image regions that have the most significant impact on the prediction of that class. In contrast, *CAM* methods leverage the activation maps from convolutional layers for this purpose, with some variants that also exploit gradient information.

Capsule Networks (*CapsNets*) (Sabour et al., 2017; Hinton et al., 2018) have been introduced as an extension to standard neural networks. Unlike traditional *DNNs*, *CapsNets* are designed to explicitly model hierarchical relationships in the data, which inherently enhances their interpretability capabilities. The growing interest in *CapsNets* has led to their application in various domains, including medical imaging (Ramana et al., 2022; Ayidzoe et al., 2022; Geng et al., 2025; Srinivasan et al., 2025; Sushith et al., 2025), NLP (Wang et al., 2020b; Chen et al., 2024; Liu et al., 2024), and computer vision (Renzulli et al., 2023; Everett et al., 2024; Zeng et al., 2025). This has resulted in an increasing need to explain the decisions made by such networks.

While much work has focused on designing interpretability methods to understand what a capsule-based model has learned (Li et al., 2025; Geng et al., 2025; Tawalbeh & Oramas, 2024; Mitterreiter et al., 2023; Bondarenko et al., 2023; Gu et al., 2021), these methods primarily focus on analyzing internal representations such as capsule activations or *part-whole* relationships. However, understanding what the model has learned does not necessarily provide insight into why a specific decision was made for a given input. This gap highlights the need for explanation methods that can attribute model predictions to specific inputs. Despite

this, relatively fewer efforts have specifically focused on explaining the predictions made by *CapsNets* (Nguyen et al., 2019; Shen & Gao, 2018). Efforts to explain the predictions made by these networks (Nguyen et al., 2019; Ren et al., 2019) have primarily exploited the routing algorithms that characterize capsule architectures. While these methods offer certain advantages, their reliance on the specifics of the routing mechanisms (i.e., dynamic routing) limits their applicability to other architectures based on different routing methods. We argue that eliminating this dependency is crucial for developing explanation methods that are agnostic towards the underlying feed-forward architecture. Another trend in the visual explanation of *CapsNets* has been the adaptation of techniques originally designed for CNN-based models to capsule architectures (Shen & Gao, 2018; Ramana et al., 2022; Tawalbeh & Oramas, 2024). While effective to some extent, these CNN-tailored methods are designed to treat the output of such networks as a scalar, which is normal for standard *CNNs*. Consequently, they ignore key characteristics of *CapsNets* and fail to exploit the (*part-whole*) mechanisms by which *CapsNets* operate.

Building on the above observations and inspired by Wang et al. (2020a), **our main contributions** are firstly, proposing *Caps-CAM*, a general visual explanation method for feed-forward *CapsNet* architectures. The proposed method takes a different approach from *CNNs* based methods by first leveraging gradients to measure the relevance of the capsules on the decision-making process and retaining only the most relevant ones after ranking them. Once the top relevant capsules have been identified, we compute their contribution by multiplying the activation maps and the corresponding top relevant capsules. By selecting only the top relevant capsules prior to computing their contributions, this two-step process eliminates the dependence on specific routing algorithms. Consequently, the proposed method is compatible with different routing and structural *CapsNet* designs, which allows it to be applied across architectures with varying routing mechanisms and structural designs. Moreover, by restricting computations to these most relevant capsules, it reduces the computational overhead associated with processing all capsules. For evaluation, we consider several existing methods such as the work of Selvaraju et al. (2017); Chattopadhay et al. (2018); Wang et al. (2020a), showing that the proposed *Caps-CAM* achieves competitive quantitative performance (in terms of Average % Drop and *AOPC*). On the qualitative side, our explanations show that the considered *CapsNet* architectures exploit both object-specific and context-related features when making predictions. Secondly, testing *Caps-CAM* on a more realistic setting (i.e. the MSTAR/application dataset), demonstrates the practical value of the proposed method. Overall, the experiments conducted on both standard benchmarks and a real-world application dataset demonstrate the capabilities of the proposed *Caps-CAM* method in generating meaningful visual explanations.

2 Related Work

CapsNets (Sabour et al., 2017) were specifically designed for modeling *part-whole* relationships between features in the data used to train them. Several efforts exploited this characteristic, primarily through the routing mechanism, to explain the predictions made by these networks. In line with this, Shahroudnejad et al. (2018) identified the relevant activation paths within the network and used the most relevant units along these paths as means for explainability. Jia & Huang (2020) extended the methodology in the work of Phaye et al. (2018), focusing on explaining global features utilizing a set of relevant capsules. These capsules are selected using routing algorithms that establish connections between the intermediate layers and the later/class capsule. Different from the methods above, which provide explanations by employing the routing to verify the relevant activation paths, the following methods employ the routing paths/mechanism to generate attribution heatmaps. For instance, Tawalbeh & Oramas (2024) produces gradient-based attribution heatmaps to justify the prediction by highlighting the contribution of specific units to a given class. In their method, the selection of these units is determined by the routing algorithm, which makes the resulting attribution maps inherently dependent on this algorithm. Similarly, in Ren et al. (2019), the visual heatmaps are produced by the combination of class-relevant units modulated by their associated coefficients in the routing mechanism of the architecture. In contrast to the work of Ren et al. (2019); Tawalbeh & Oramas (2024), our proposed method eliminates the dependence on such routing algorithms. Instead, we rely on internal activation maps in conjunction with capsule-related gradients to address the explanation task.

A second group of efforts has focused on the adaptation of methods, initially designed for *CNNs*, for the explanation of *CapsNets*. Along this line, gradient-based approaches have been explored in the work of

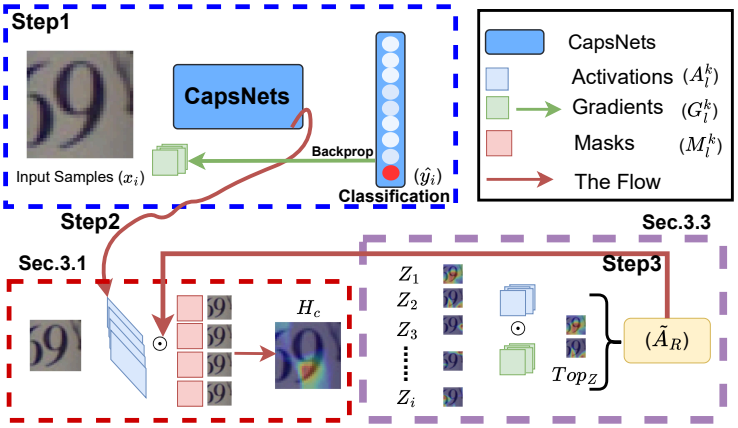

Figure 1: Overview of *Caps-CAM* methodology: Step 1 involves a forward pass through the model. In Step 2, A_l^k is extracted for use in *Score-CAM*. Finally, Step 3 employs $A_l^{k,\intercal}$ and $G_l^{k,\intercal}$ to identify and select the top capsules of *Caps-CAM*. The section refers to where these steps are introduced in the paper.

Tawalbeh & Oramas (2024), while guided backpropagation was applied in Phaye et al. (2018). Additionally, *Grad-CAM* has been adapted to produce visual explanations for capsule-based architectures (Shen & Gao, 2018; Mobiny et al., 2019). Different from the methods mentioned above and similar to *Score-CAM* (Wang et al., 2020a), our proposed method utilizes the relevance of each internal activation channel alongside their corresponding gradients w.r.t the predicted class by weighting them based on the class scores. This produces an explanation heatmap that allows the identification of regions of the input that determine the predicted class.

3 Proposed Method

This section introduces the *Caps-CAM* method for explaining predictions made by *CapsNets*. We first provide background details on *Score-CAM* (Wang et al., 2020a), as our methodology builds upon it. Then, we cover different aspects of the proposed method. To summarize, the proposed method utilizes activation maps while incorporating gradients into the proposed framework in order to compute relevance matrices. Then, the matrices from the most relevant capsules are used to produce explanations in the form of visual heatmaps. Fig. 1 illustrates the pipeline of the proposed *Caps-CAM* explanation method.

3.1 Score-Weighted Class Activation Mapping

Methods based on Class Activation Mapping (CAM) (Zhou et al., 2016; Selvaraju et al., 2017) produce explanation heatmaps through the weighted sum of the activations of the filters in the last convolutional layer of a *CNN* architecture. A common practice is to compute these weights from gradients related to the predicted class. In contrast, *Score-CAM* (Wang et al., 2020a) removes the requirement on gradient information by relying solely on forward passes and internal activations.

Formally, let F denote a pretrained *CapsNet* that maps an input x_i to class scores $S_i = F(x_i)$, from which the predicted class label $\hat{y}_i$ is obtained via $\hat{y}_i = \arg\max_c S_i(c)$. Moreover, let A_l^k denote the activation map of the k-th filter in the convolutional layer l. Hence, the activations $A_l^k(x_i)$ of a given layer l, obtained through the forward pass for an example x_i, are computed as described in Eq. 1.

$$A_l^k(x_i) = F_l^k(x_i), \quad \forall k \tag{1}$$

For *Score-CAM*, the activation maps extracted from the layer prior to the classification part are upsampled to correspond with the size of the input x_i. We denote the resulting upsampled activation map as $\hat{A}_l^k(x_i)$.

These upsampled activation values $\hat{A}_l^k(x_i)$ are then normalized via min-max to the range [0, 1]. This is followed by the occlusion phase, where the input x_i is perturbed using each of the upsampled activation maps $\hat{A}_l^k(x_i)$, producing a new masked input M_l^k (Eq. 2).

$$M_l^k = \hat{A}_l^k(x_i) \odot x_i \tag{2}$$

where $\odot$ represents the element-wise multiplication between the upscaled activation map $\hat{A}_l^k(x_i)$ and the input x_i. Then, the importance of the kth activation map is determined by performing a forward pass through the pretrained network F with the masked input M_l^k and extracting only the output score corresponding to the target class predicted for the original (unmasked) input, defined as $\tilde{S}^k = F(M_l^k)$. The final step involves applying a *ReLU* activation function to the combination of the feature maps $\hat{A}_l^k$ using the weights $\tilde{S}^k$ (Eq. 3). This is done to focus on the features that have a positive contribution to the predicted class c.

$$H_c = ReLU(\sum_{k=1}^{k} \tilde{S}^k \hat{A}_l^k) \tag{3}$$

3.2 Class Activation Mapping for *CapsNets*

In contrast to *CNN* models, *CapsNets* are composed of so-called capsule elements that fundamentally characterize these architectures. While adopting CNN-based methods is possible, doing so poses challenges for directly applying the *Score-CAM* method to these architectures. This difficulty arises from the potential loss of encoded information within each capsule dimension. To address this, additional steps should be taken immediately after performing the forward pass mentioned in Eq. 1. Specifically, the activations $A_l^k \in \mathbb{R}^{1\times\mathbb{Z}}$ (with $\mathbb{Z}$ being the dimensionality of the activations output) are transposed to rearrange their dimensions, yielding $A_l^{k,\intercal}(x_i) \in \mathbb{R}^{1\times Z\times K\times H\times W}$, to incorporate the capsule dimension Z, which represents the capsule elements.

Afterwards, a summation is performed over the capsule dimension Z of the reshaped activations $A_l^{k,\intercal}(x_i)$, resulting in the aggregated tensor across the capsule dimension Z into a single unified matrix $\tilde{A}_l^k(x_i) \in \mathbb{R}^{1\times K\times H\times W}$. By doing so, this aggregation simplifies the representation, reduces the dimensionality of the capsules, and fuses the information they encode. It is important to note that, unlike standard *CAM* methods, where no input masking is applied, similar to *Score-CAM*, the proposed method incorporates a masking step after normalization to adapt the process to capsule-based architectures. Specifically, instead of normalizing the activation maps independently, we normalize them by considering the global min-max values across all matrices $[A_l^{k\,\intercal}(x_i)_1, A_l^{k\,\intercal}(x_i)_2; \ldots; A_l^{k\,\intercal}(x_i)_z]$ associated with the capsule dimension Z. The normalization is performed prior to summing across the capsule dimension and subsequently masking the input using $\tilde{A}_l^k(x_i)$. By doing so, it ensures that all units are calibrated and that no particular unit dominates the aggregated activation map, providing a comparable contribution from the capsule dimensions.

3.3 *Caps-CAM*

The proposed method modifies the original *Score-CAM* immediately after extracting the activations A_l^k, prior to the masking step (defined in the previous section). In this step, A_l^k are weighted using the corresponding

Capsule Network	MNIST	SVHN	CIFAR-10	MSTAR
DR	99.1	91.0	74.2	99.4
DR-Conv	99.0	90.5	74.0	99.7
DR-Tra	99.4	91.0	76.0	99.3
EM	98.0	80.0	72.2	86.6

Table 1: *CapsNets* test accuracy (%) on MNIST, SVHN, CIFAR-10, and MSTAR datasets. Dynamic Routing (DR), Dynamic Routing with deep convolutional layers (*DR-Conv*), Dynamic Routing with transformer blocks before the *PC* layer (*DR-Tra*), and Expectation Maximization routing based network (*EM*).

gradients to identify the relevant capsules, ensuring that only the most important ones are retained for subsequent processing. Towards this goal, we first reintroduce the use of gradients. This is motivated by the ability of gradients to capture crucial information regarding units that determine the predicted class. To achieve this, we reshape the activations $A_l^{k,\intercal}(x_i)$, which were previously introduced and preprocessed in Sec. 3.2. Simultaneously, we obtain the gradients $G_l^k(x_i, \hat{y}_i)$ for a predicted class $\hat{y}_i$, which is obtained from the extracted score $S_i(\mathrm{c})$ w.r.t a given input x_i, as the activations for the same input image have already been extracted. Then, we apply the same transpose to gradients, producing the transposed matrix $G_l^{k,\intercal}(x_i, \hat{y}_i)$, and ensure that both matrices, $G_l^{k,\intercal}(x_i, \hat{y}_i)$ and $A_l^{k,\intercal}(x_i) \in \mathbb{R}^{1\times Z\times K\times H\times W}$, have the correct shape, which guarantees their compatibility for the subsequent steps.

With $G_l^{k,\intercal}(x_i, \hat{y}_i)$ and $A_l^{k,\intercal}(x_i)$ in place, we proceed to compute the most influential (relevance) capsules at the dimension Z level in the following step. This is achieved by computing the product between the dimension Z of a given capsule within $G_l^{k,\intercal}(x_i, \hat{y}_i)$ and the corresponding dimensions Z within $A_l^{k,\intercal}$ of the same capsule. This operation indicates how strongly each dimension contributes to the loss, while the activations reflect the magnitude of the response of each dimension. The key insight here is that the relevance of each capsule is determined based on its contribution to the final prediction by employing both matrices. This enables us to identify the relevance score of each capsule $\mathcal{R}_i$ as calculated in Eq. 4. The resulting tensor $\mathcal{R}_i$ contains only the potential capsule relevance.

$$\mathcal{R}_i = A_l^{k,\intercal}(x_i) \cdot G_l^{k,\intercal}(x_i, \hat{y}_i), \quad \forall\, k, \quad \mathcal{R}_i \in \mathbb{R}^{1\times Z\times 1\times 1\times 1} \tag{4}$$

To complement the previous step, we compute the product between the activations $A_l^{k,\intercal}$ and their corresponding relevant capsule scores $\mathcal{R}_i$. This ensures that the contributions of the most relevant capsules in the dimension Z are emphasized over the rest. Therefore, the relevance activation matrix $\mathbf{A}_R^k$ is calculated as:

$$\mathbf{A}_R^k = A_l^{k,\intercal}(x_i) \cdot \mathcal{R}_i, \quad \forall k, \quad \mathbf{A}_R^k \in \mathbb{R}^{1\times Z\times K\times H\times W} \tag{5}$$

By computing the relevance activation matrix $\mathbf{A}_R^k \in \mathbb{R}^{1\times Z\times K\times H\times W}$ this way, we eliminate the dependence on routing algorithms to select the relevant capsules. This design choice improves the compatibility of the proposed method with different capsule-based architectures, which enhances the overall applicability of our method. This is also noted in the study from Byerly et al. (2021), which suggests that the complexity introduced by routing algorithms in *CapsNets* can be avoided, as these networks can be effectively trained without relying on such mechanisms.

Then, we select the top capsules based on their maximum relevance scores, which are obtained from the relevance matrix $\mathbf{A}_R^k$, and which encode the relevance of capsules along the Z dimension. This results in a tensor of indices $\mathbf{I}_{top}$ corresponding to the most relevant capsules (the top capsules), which will be used in subsequent steps. Then, we modify $\mathbf{A}_R^k$ to focus the attention on the relevance of the selected $\mathbf{I}_{top=2}$ capsules, using the previously obtained top ranking ones. Hence, the final relevance activation matrix $\mathbf{A}_{R,Z}^*$ is obtained by retaining the indices corresponding to the $\mathbf{I}_{top=2}$ capsules (Eq. 6).

$$\mathbf{A}_{R,top}^* = \mathbf{A}_R^k[\mathbf{I}_{top}], \quad \mathbf{A}_{R,top}^* \in \mathbb{R}^{1\times 2\times K\times H\times W} \tag{6}$$

Having $\mathbf{A}_{R,top}^*$ appropriately dimensioned, a summation is performed across the capsule dimensions Z, resulting in $\tilde{\mathbf{A}}_R \in \mathbb{R}^{1\times K\times H\times W}$. Once this summation is complete, we first apply a capsule-based normalization step (as explained earlier in Sec. 3.2) to $\tilde{\mathbf{A}}_R$, which ensures that their values are appropriately scaled. Second, we return to the original *Score-CAM* formulation and apply the remaining steps to the modified matrix $\tilde{\mathbf{A}}_R$ (similar to $A_l^k(x_i)$ in the original *Score-CAM* formulation). We then proceed with masking the input using $\tilde{\mathbf{A}}_R$ and combine it with the calculated scores. Retaining only the top capsules is crucial, as some capsules are strongly activated while others exhibit weak responses. The weak capsules should be removed from the computation to prevent their noisy contributions, which can lead to diffuse explanation heatmaps.

4 Evaluation

This section presents the evaluation procedure employed to validate the proposed method, along with the corresponding results. First, *Caps-CAM* is qualitatively evaluated through visualizations on several datasets.

Second, the validity of the visual explanations is assessed, specifically focusing on the importance of the highlighted regions in influencing the outputs of the model.

Implementation details. Our experiments are conducted with NVIDIA GeForce GTX 1060 and DGX-2. The considered architectures are implemented in PyTorch and trained for 60 epochs and a batch size of 32. We use the adam optimizer with the default learning rate, dropout, and weight decay. For the routing algorithm, 3 routing iterations were used in all *CapsNet* architectures. During training, for the above-mentioned datasets, no data augmentation or hyperparameter tuning was applied. For the *EM* network architectures, the learning rate was set to 3×10^{-3}, allowing for effective convergence during training. We employed a weight decay of 2×10^{-7} to prevent overfitting. The code is publicly available at [1].

Datasets. Following Sabour et al. (2017); Hinton et al. (2018); Gu et al. (2021); Byerly et al. (2021); Tawalbeh & Oramas (2024); Mitterreiter et al. (2023), we adopt well-established datasets for the evaluation phase, by considering MNIST (Deng, 2012), SVHN (Goodfellow et al., 2013), and CIFAR-10 (Krizhevsky et al., 2009), each comprising 10 distinct classes.

CapsNet Architectures. We evaluate the proposed method on top of various *CapsNet* architectures, including Dynamic Routing (*DR*) (Sabour et al., 2017), which comprises convolutional, primary capsule, and class capsule layers. In addition, we also employ the Matrix Capsules (*EM*) architecture (Hinton et al., 2018), which incorporates additional layers, such as multiple convolutional capsule layers. We also modified the *DR* architecture by incorporating three additional convolutional layers before the primary capsule layer, resulting in the *DR-Conv* variant. Finally, we integrated a transformer block before the primary capsule layer (*DR-Tra*). These modifications are expected to enhance feature representation. These architectures were trained until they achieved competitive accuracy on the validation sets, demonstrating performance comparable to that reported in the literature (Sabour et al., 2017; Hinton et al., 2018; Xi et al., 2017; Xiong et al., 2019; Yang et al., 2020)(Table. 1). We evaluated the performance on the training and validation sets, with the results reported in the supplementary material (see Table A1). For reproducibility purposes, we provide two tables (Tables A3 and A4 in the supplementary material) detailing the used *CapsNet* architectures. This offers a comprehensive overview of the input and output shapes for both the *encoder* and *decoder* components. Additionally, we provide a summary of the number of parameters used for each architecture (see Table A3 in the supplementary material). This provides insight into the complexity and capacity of each *CapsNet* variant, which is an important consideration in discussions of explainability, as more complex networks often present greater challenges for classification tasks (Zou et al., 2024; Noor et al., 2024).

Explanation Methods. For the implementation of *Caps-CAM* we consider the top-2 most relevant capsules, denoted as $\mathbf{I}_{top}(\mathbf{A}^{*}_{R,top} = \mathbf{A}^{k}_{R}[\mathbf{I}_{top}])$ (Sec. 3.3). The reason for selecting only 2 capsule dimensions out of the 8 dimensions is that, in *CapsNets*, each capsule dimension encodes a distinct characteristic, whose rel-

[1] `Annonymous_To_be_released_upon_acceptance`

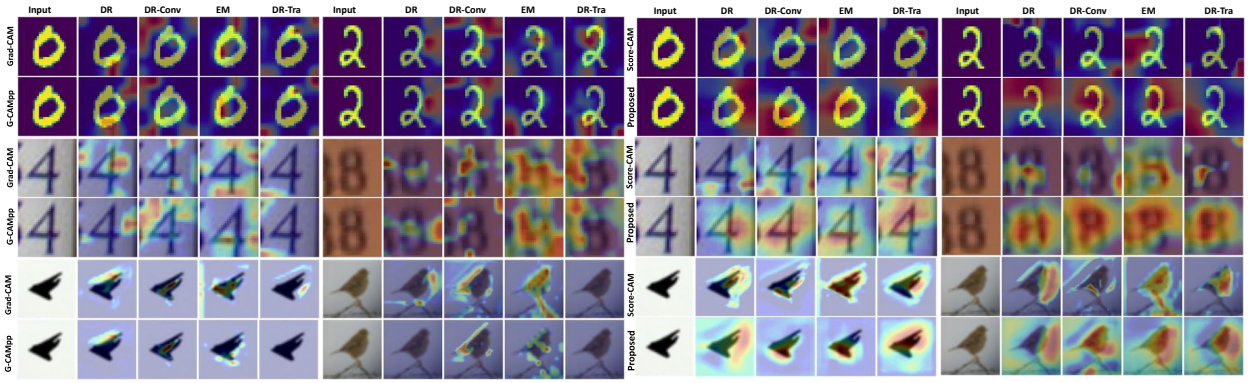

Figure 2: Heatmap visualizations generated by *Grad-CAM*, *Grad-CAM++*, *Score-CAM*, and *Caps-CAM*, highlighting the relevant regions within the MNIST (top), SVHN (middle), and CIFAR (bottom) datasets in the context of classification tasks(predicted correctly).

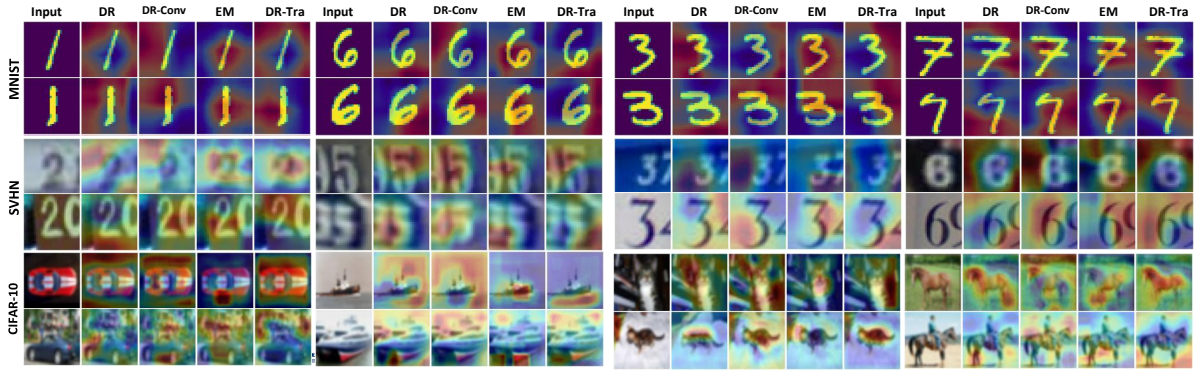

Figure 3: Additional *Caps-CAM* visual heatmaps for correct predictions across datasets (see Fig.A3 in the supplementary material for larger version).

evance can be quantified via its activation strength. Focusing on the entire capsule dimensions can produce diffuse or overlapping activations, which may weaken the resulting explanation map. Considering only these specific capsule dimensions will help focus on the regions that are genuinely relevant when evaluating the generated explanation heatmaps. In line with this, we apply the top-2 capsules to dynamic routing-based *CapsNets*. We adopt the same methodology for the *EM* routing-based architecture with minor modifications due to differences in the respective routing mechanisms. Specifically, *EM* operates with 4×4 (16) dimensions rather than 8 dimensions; from these, we selected 2 dimensions for analysis. We conducted an ablation experiment to justify the choice of employing only a subset of 2 capsule dimensions. Please see the supplementary material for more details (Figures [A7-A14]).

We position our method w.r.t. several CAM-based explanation methods, i.e. *Grad-CAM* (Selvaraju et al., 2017), *Grad-CAM++* (Chattopadhay et al., 2018), and *Score-CAM* (Wang et al., 2020a), which we adapted to make them applicable to *CapsNets*. To achieve this, we reorganize the capsule dimension Z as described in detail in Section 3.2. The matrix $\tilde{A}_l^k(x_i)$ is then used as input for the computation of both methods.

4.1 Qualitative Analysis.

This experiment aims to qualitatively compare the visual heatmaps produced by the proposed *Caps-CAM* method (Sec. 3.3) w.r.t. those from the competing explanation methods. We present some examples in Fig. 2 across the used datasets.

Discussion: As can be noted, the heatmaps produced by *Caps-CAM* exhibit fewer random artifacts and a smoother appearance compared to those generated by other methods, e.g., *Grad-CAM* and *Grad-CAM++*. Moreover, the visualizations produced by the proposed method seem to focus more on object-centric features. For example, in the MNIST dataset (Fig. 2/top), the heatmaps [0, 2] focus on the digit regions while suppressing noise in the background. Similarly, for SVHN (Fig. 2/middle), the heatmaps [4, 8] effectively highlight the digit itself (center parts) while ignoring irrelevant regions (background pixels). In CIFAR-10 examples [airplane, bird] (Fig. 2/bottom), *Caps-CAM* emphasizes key features such as the body of an airplane and the body of a bird, showing the focus of the network on object-centric features. In contrast, as noticed, the heatmaps generated by *Grad-CAM* and *Grad-CAM++* appear sparser compared to those produced by *Score-CAM* and *Caps-CAM*. This difference can be attributed to the incorporation of input masking steps, which may have two effects. On the one hand, since *Score-CAM* computes the weight of each feature map from the class score obtained after masking the input, denser activation maps preserve more information during the masking phase. This typically results in higher class scores in Eq. 3, which in turn favors these denser maps in the final heatmap generation (see Sec. 3.1). On the other hand, this process may capture spatially complementary information and by aggregating contributions from several relevant regions in the context of classification tasks, a denser heatmap is produced.

A second trend can be observed in Fig. 3, which shows additional examples produced by *Caps-CAM*. In these examples, *Caps-CAM* highlights different relevant features of the input images. There, we notice that for the MNIST/SVHN datasets, it focuses on the background [MNIST/6-SVHN/2,3,6], while for CIFAR-10, the method highlights the sky and the sea [boat] as well as the sky and the ground [horse/bottom class], where these features contribute to the classification output. These observations suggest that, in the context of classification tasks, the network considers background features as relevant cues for its decision-making phase.

Fig. 4 presents visual heatmaps generated by *Caps-CAM* for cases in which *CapsNets* produced incorrect predictions. For MNIST, the network often confuses visually similar classes, and the corresponding explanations show that attention is concentrated on regions shared across these classes. For example, as shown in Fig. 4, the digits [4, 7, 9]/right exhibit overlapping shape patterns and the produced visual heatmaps highlight these shared components, while failing to emphasize discriminative regions that could resolve the ambiguity. For digit [3]/right, the heatmaps exhibit covered regions in the upper curved regions, which are also characteristic of digits [5] and [8]. This indicates that the model might rely on partial shape cues rather than globally discriminative features, resulting in misclassification. We also note a similar behavior when the model confuses classes with similar characteristics in digits [2, 3, 4, 7]/SVHN, where the heatmaps highlight non-object-centric features (i.e., background) of these digits. Furthermore, in both *DR-Conv* and *DR-Tra*, the network confuses digit [9]/SVHN, likely due to the presence of additional numbers within the input images, as it is covered by the heatmap. In some cases, the visual explanations highlight regions of the ground-truth digit despite incorrect predictions. For instance, in [3,4]/MNIST, the heatmaps emphasize the upper curved region of digit [3], which closely resembles the top loop structure of digit [8], the predicted class. At the same time, the highlighted central stroke of digit [4] corresponds to a stroke configuration commonly associated with digit [9], providing a visual explanation for the model's confusion. Similarly, in digits [3,7]/SVHN, the explanations focus on the central stroke of digit [3] and on the central and lower strokes of digit [7]. Notably, the elongated lower stroke of digit [7] resembles the vertical structure of digit [1], which helps explain why the model confuses these classes. For CIFAR-10, a higher degree of confusion between classes, which the explanation maps clearly reveal. For example, the visual heatmaps often highlight background regions [sky, grass, or street] that are shared across different categories, such as [airplane, car], [airplane, ship], [deer, horse], and [ship, frog].

The observations above suggest that *Caps-CAM* is capable of identifying both object-specific and context-specific features, which may be important for the overall prediction of the network. In addition, we observed differences in the visual explanations generated by *DR* (Sabour et al., 2017) and *EM* (Hinton et al., 2018)

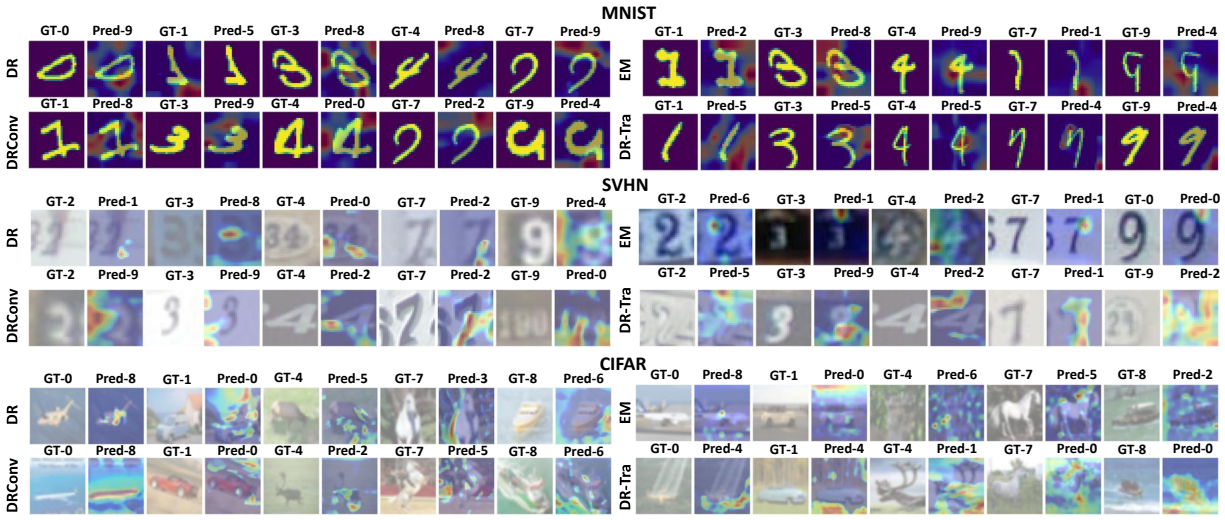

Figure 4: Additional heatmaps generated by the *Caps-CAM* method when *CapsNets* produced incorrect predictions across the evaluated datasets. For CIFAR, (0-airplane), (1-car),(4-deer), (7-horse),(8-ship).

routing-based algorithms. We conjecture that these differences might be related to how the routing mechanisms define *part-whole* relationships, the characteristics of the routing algorithm, and the complexity of the dataset. These differences result in different characteristics of the explanation heatmaps. For instance, DR-based heatmaps tend to emphasize the most salient parts of an object that maximize agreement across capsules. In contrast, EM-based visual heatmaps appear smoother and more centered around the object or digit, as EM routing employs the clustering algorithm.

4.2 Quantitative Analysis

We quantify the validity of the generated explanation heatmaps to the decision-making process followed by the considered *CapsNets* models using the Average % Drop and the Area Over the Perturbation Curve (*AOPC*) metrics. A brief overview of these metrics is provided here; detailed definitions and implementation procedures are available in the supplementary material (Eq. A1 and Eq. A2). Following prior efforts Chattopadhay et al. (2018); Samek et al. (2016), we considered only the correctly predicted inputs when computing both metrics. This is motivated by the fact that explanations for misclassified classes would reflect spurious evidence supporting the incorrect class, and are therefore less informative for evaluation purposes.

Average % Drop (avg % Drop) (Chattopadhay et al., 2018). This metric evaluates the reliability of an explanation by quantifying the change in the prediction confidence of the model after masking the input according to a class-specific visual heatmap. When the heatmap accurately identifies regions that are relevant to the decision of the model, suppressing less informative areas results in little or no reduction in confidence, yielding a low Average % Drop. A lower avg % drop indicates a better explanation heatmap.

Area Over the Perturbation Curve (*AOPC*) (Samek et al., 2016). This metric quantifies the average change in model output in response to progressive input perturbations. Following the work of Gu & Tresp (2021), we applied a similar perturbation setting to the generated explanation heatmaps using a progressive perturbation strategy guided by the explanation heatmaps produced by the method being evaluated. At each perturbation step, a 5×5 patch centered at the most relevant unperturbed spatial location is replaced with random noise. Once a location is perturbed, it is excluded from subsequent steps. This procedure is repeated for a fixed number of (15) perturbation steps. A higher *AOPC*, implies a better explanation heatmap, as it implies that the most relevant locations identified by the heatmap correspond to the information that is most relevant to the features exploited by the network.

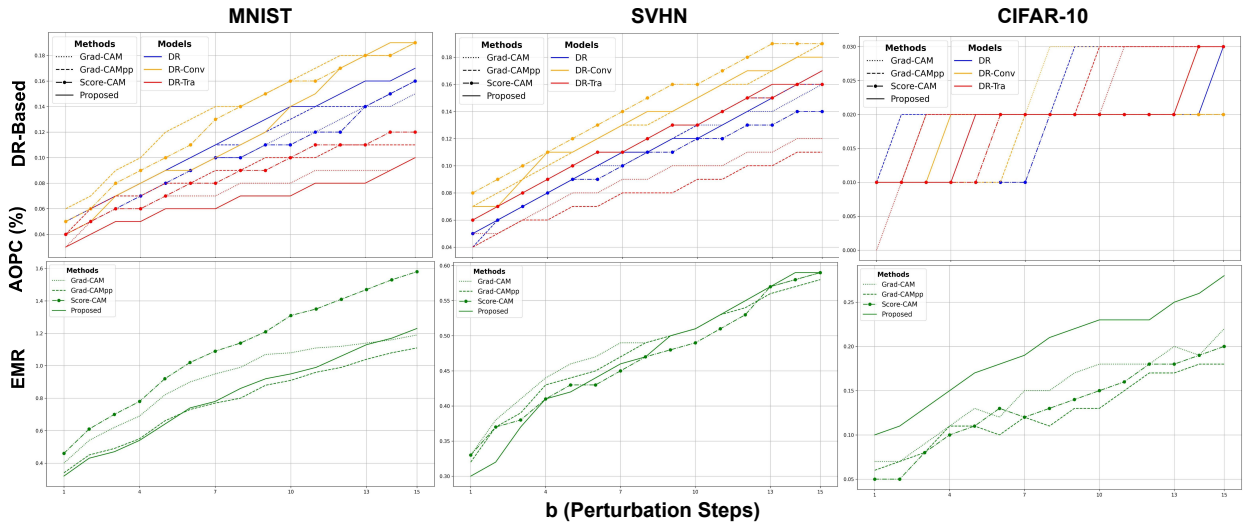

Figure 5: AOPC (%) values computed over 15 perturbation steps for visual explanations generated by various explanation methods applied to *CapsNet* architectures across the datasets. Refer to the appendix for the actual numbers (Tables A5,A6 and A9).

Dataset	**MNIST**				**SVHN**				**CIFAR-10**				**MSTAR**			
Method	*DR*	*DR-Conv*	*DR-Tra*	*EM*	*DR*	*DR-Conv*	*DR-Tra*	*EM*	*DR*	*DR-Conv*	*DR-Tra*	*EM*	*DR*	*DR-Conv*	*DR-Tra*	*EM*
Grad-CAM	30	33	23	31	46	51	44	31	45	47	46	43	52	53	52	42
Grad-CAM++	27	27	21	32	48	49	43	31	43	43	43	47	46	51	51	38
Score-CAM	0.3	1	15	**11**	14	18	13	33	17	19	**14**	47	21	27	33	**19**
Caps-CAM (Proposed)	**0.2**	**0.6**	**14**	46	**3**	**9**	**9**	**31**	**14**	**17**	15	**35**	**3**	**1**	**21**	37

Table 2: Average % drop (lower is better) for different interpretability methods across MNIST, SVHN, CIFAR-10, and MSTAR datasets. The models compared are Dynamic Routing (*DR*), DR with deep *conv* layers (*DR-Conv*), DR with transformer blocks before the *PC* layer (*DR-Tra*), and EM-routing (*EM*).

For easier comparison between the different configurations, explanation heatmaps are produced for correctly predicted labels, and the methods are evaluated accordingly. Table 2 reports the mean percentage drop scores averaged across all classes for each dataset and evaluated architecture.

Discussion: When focusing on *avg % Drop*, notably, the *DR* model achieves the best performance when utilizing *Caps-CAM* across the datasets. Table 2 shows that *Caps-CAM* consistently outperforms other well-known methods across all architectures and datasets, with two exceptions. Specifically, *Caps-CAM* shows a higher average drop on both the *EM* architecture for the MNIST dataset and on the *DR-Tra* architecture for the CIFAR-10 dataset. While the heatmaps visually align with the digits in MNIST (Fig. 2/top and Fig. 3/top), the high average drop indicates that the highlighted regions alone may not fully justify the predictions made by the model. This suggests that the model may rely on additional patterns or background cues not captured by the heatmaps. These observations underscore the importance of explanation methods that are reliable both visually and quantitatively. Visually reliable explanations highlight the regions that the model genuinely relies on when making a prediction, whereas quantitatively reliable explanations are reflected in metrics such as Average % Drop, with lower values indicating higher explanation quality. Considering both criteria together provides a more comprehensive evaluation of the reliability of the generated heatmaps. Addressing this gap can significantly enhance the reliability of the proposed method.

Compared to gradient-based methods, *Caps-CAM* clearly outperforms *Grad-CAM* and *Grad-CAM++*. This improvement stems from two key factors. First, the masking of the input using relevance activation maps ensures that only the most informative regions of the input contribute to the explanation, which reduces noise from irrelevant regions. Second, the use of calculated confidence scores prioritizes capsules that are most predictive of the output, further emphasizing important regions. Together, these mechanisms enhance the reliability of the visual heatmaps, which explains the improved performance compared to *Grad-CAM* and *Grad-CAM++*.

Regarding *AOPC*, similar to avg.% drop metric, the *AOPC* metric exhibits consistent behavior across most cases (Fig. 5), regardless of the explanation method or the *CapsNet* architecture used for evaluation (see Table A5/MNIST/DR, Table A6/SVHN/DR, Table A6/CIFAR-10/DR, and Table A9/bottom/EMR in the supplementary material). Across the datasets, *Caps-CAM* consistently outperforms other explanation methods (Table 5,6, and 9), with the exception of *Score-CAM*, which achieves higher *AOPC* scores when the *DR-Conv* (SVHN) (Fig. 5-top) (Table 6) and *EM* (MNIST) (Fig. 5-bottom) (Table 9) architectures are used to generate the visual explanations. Furthermore, *Caps-CAM* is specifically tailored for *CapsNets*, which facilitates producing visual explanations that leverage the relevant features, resulting in heatmaps that are more contextually aware compared to gradient-based methods originally designed for *CNN* architectures.

4.3 Use Case: Moving and Stationary Target Acquisition and Recognition

To demonstrate the practical value of the proposed *Caps-CAM* method, we test it on a more realistic setting, specifically, Moving and Stationary Target Acquisition and Recognition from satellite images (Touafria & Yang, 2019; Dai et al., 2023). For this purpose, we conduct experiments on the MSTAR dataset.[2] Training on this dataset required data augmentation in order to mitigate overfitting. Please see the supplementary material for more details.

[2] `https://www.sdms.afrl.af.mil/index.php?collection=`

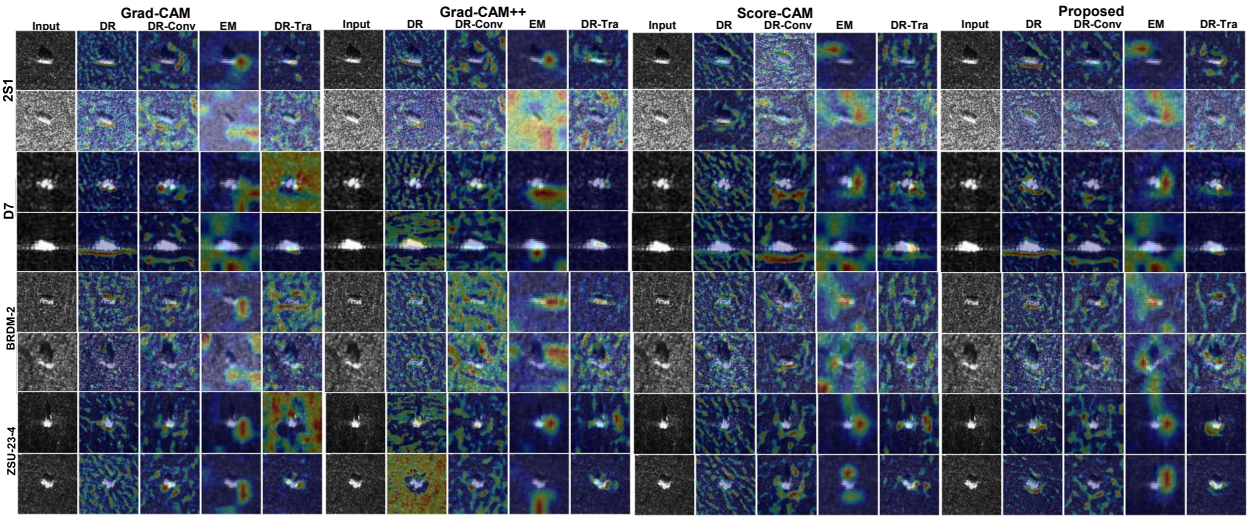

Figure 6: Heatmap visualizations generated by the proposed *Caps-CAM* method, compared to state-of-the-art methods, highlighting the relevant regions w.r.t classification problem within the MSTAR dataset (see Fig. A1 and A2 in the supplementary material for a larger version).

The following discussion presents qualitative examples of the generated explanations and quantitatively evaluates their validity considering the two previously introduced metrics, i.e., avg % drop and *AOPC*. For the latter, we report *AOPC* scores over 15 perturbation steps.

Qualitative Analysis. Fig. 6/right depicts examples of the explanation heatmaps generated by the proposed *Caps-CAM* method, alongside other methods from the literature, evaluated across several *CapsNet* architectures. As shown in Fig 6, the visual explanations produced by *Caps-CAM* generally exhibit fewer random artifacts compared to those generated by *Grad-CAM* and *Grad-CAM++*. This reflects that *Caps-CAM* leverages the structured capsule representations and relevance-weighted activations, which focus only on the capsules that contribute most to the predicted class. As a result, irrelevant regions are effectively suppressed, producing cleaner heatmaps that better reflect on the decision of the network. Furthermore, the proposed method provides an indication that the considered models focus not only on object-centric features, but on contextual features as well. This can be noticed in the case of class 2S1(top-right) (Fig. 6/right), where the visualizations focus on the shadow of the vehicle. A similar behavior is observed for the BRDM-2 class using the *DR-Tra* architecture. Another notable observation is found in the D7 class (Fig. 6/right), where the visual explanations tend to focus on the ground pressure marks left by the vehicle, likely due to its significant weight. A similar behavior is observed in the ZSU-23-4 class. As both classes correspond to heavy military vehicles, this pattern supports our interpretation. Additional examples illustrating this behavior are provided in Fig. 7. More figures supporting our findings can be found in the supplementary material.

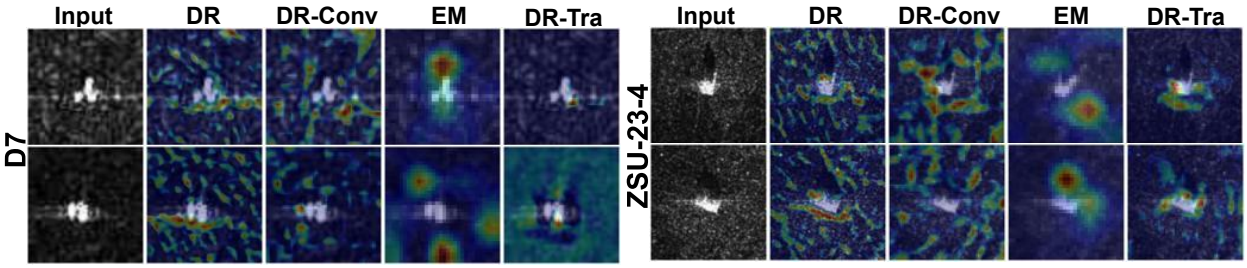

Figure 7: Additional explanation heatmaps generated by the proposed *Caps-CAM* method for heavy tank classes in the MSTAR dataset.

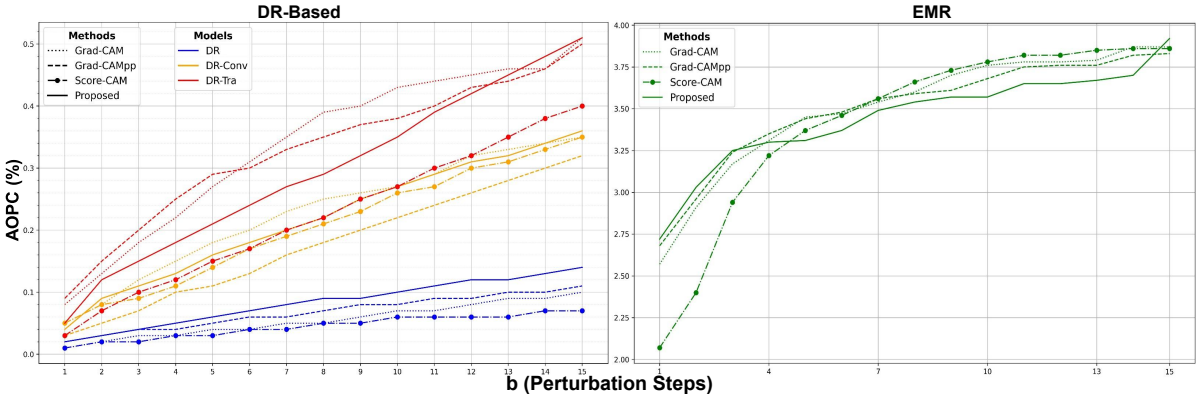

Figure 8: AOPC (%) values over 15 perturbation steps metric for visual explanation generated by explanation methods on *CapsNets* trained on the MSTAR dataset. Refer to the appendix for the actual numbers (Tables A8 and A9).

Quantitative Analysis. When focusing on the *avg % drop* metric, the proposed *Caps-CAM* method continues to demonstrate superior performance across most architectures. As shown in Table 2, *Caps-CAM* achieves the lowest drop scores on the *DR*, *DR-Conv*, and *DR-Tra* architectures, outperforming all other baseline methods, including *Score-CAM*, *Grad-CAM*, and *Grad-CAM++*. However, an exception is observed with the *EM* architecture. where *Score-CAM* outperforms *Caps-CAM*. This aligns with the trend previously observed on MNIST, where *Caps-CAM*'s performance on the *EM* architecture was also less effective. This may result from the unique routing behavior in the *EM* variant, which impacts the reliability of *Caps-CAM*'s relevance scores. These results highlight the stability of *Caps-CAM* in dynamic routing-based capsule architectures, particularly when applied to complex real-world datasets such as MSTAR.

When looking at the *AOPC* (Fig. 8), we observe that the *AOPC* curves for all explanation methods are relatively close for each architecture (see Table A8/DR and Table A9/top/EMR in the supplementary material for the actual scores). Nevertheless, *Caps-CAM* outperforms the other methods, despite not relying on the classification layer. This behavior may indicate that *Caps-CAM* explanations are generated using the most relevant capsules, identified through the product between activation maps and their corresponding gradients. This process highlights more relevant features that contribute to improved explanation quality.

5 Conclusion

We propose an explanation method, *Caps-CAM*, specifically designed for justifying the outputs of *CapsNet* architectures. Comparative analysis with existing state-of-the-art methods shows the superiority of *Caps-CAM*. More precisely, both qualitative visualizations and quantitative evaluations confirm that *Caps-CAM* outperforms other methods, in most cases by effectively highlighting the most relevant regions of input images at different levels of importance in the context of classification tasks. A limitation of this work is that *Caps-CAM* produces visual explanations specifically from capsule routing backbones. However, extending this method to other backbone architectures (e.g., graph-based backbones) remains unexplored and represents a potential direction for future research. *Caps-CAM* could be extended to multimodal tasks and focus on the methods for combining capsule relevance across modalities. Furthermore, the trade-off between explanation methods and model complexity requires further investigation to gain a more comprehensive understanding of the observed model behavior.

References

Mighty Abra Ayidzoe, YU Yongbin, Patrick Kwabena Mensah, Jingye Cai, and Faiza Umar Bawah. Visual interpretability of capsule network for medical image analysis. *Turk. J. Electr. Eng*, 2022.

Andrei Bondarenko, Saja AL-Tawalbeh, and José Oramas. Analyzing the explanation and interpretation potential of matrix capsule networks. In *AIMLAI at ECML/PKDD*. Springer, 2023.

Adam Byerly, Tatiana Kalganova, and Ian Dear. No routing needed between capsules. *Neurocomputing*, 2021.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *IEEE/WACV*, 2018.

Yutong Chen, Xia Li, Yang Liu, Peng Bi, and Tiangui Hu. Relational extraction from biomedical texts with capsule network and hybrid knowledge graph embeddings. *Symmetry*, 16(12):1629, 2024.

Qijun Dai, Gong Zhang, Biao Xue, and Zheng Fang. Capsule-guided multi-view attention network for sar target recognition with small training set. *IEEE Geoscience and Remote Sensing Letters*, 20:1–5, 2023.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Miles Everett, Aiden Durrant, Mingjun Zhong, and Georgios Leontidis. Capsule network projectors are equivariant and invariant learners. *arXiv preprint arXiv:2405.14386*, 2024.

Xinyu Geng, Jiaming Wang, Xiaolin Huang, Fanglin Chen, and Jun Xu. Parsecaps: An interpretable parsing capsule network for medical image diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 3122–3130, 2025.

Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

Jindong Gu and Volker Tresp. Interpretable graph capsule networks for object recognition. 2021.

Jindong Gu, Volker Tresp, and Han Hu. Capsule network is not more robust than convolutional network. In *CVPR*, 2021.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *ICLR*, 2018.

Bohan Jia and Qiyu Huang. De-capsnet: A diverse enhanced capsule network with disperse dynamic routing. *Applied Sciences*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

Peizhang Li, Jiyuan Ru, Qing Fei, Zhen Chen, and Bo Wang. Interpretable capsule networks via self attention routing on spatially invariant feature surfaces. *Scientific Reports*, 15(1):13026, 2025.

Han Liu, Changya Li, Xiaotong Zhang, Feng Zhang, Wei Wang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. Depression detection via capsule networks with contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 22231–22239, 2024.

Matthias Mitterreiter, Marcel Koch, Joachim Giesen, and Sören Laue. Why capsule neural networks do not scale: Challenging the dynamic parse-tree assumption. In *AAAI*, 2023.

Aryan Mobiny, Hengyang Lu, Hien V Nguyen, Badrinath Roysam, and Navin Varadarajan. Automated classification of apoptosis in phase contrast microscopy using capsule network. *IEEE transactions on medical imaging*, 2019.

Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. Use of a capsule network to detect fake images and videos. *arXiv preprint*, 2019.

Khondaker Tasrif Noor, Antonio Robles-Kelly, Leo Yu Zhang, Mohamed Reda Bouadjenek, and Wei Luo. A consistency-aware deep capsule network for hierarchical multi-label image classification. *Neurocomputing*, 604:128376, 2024.

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

Kadiyala Ramana, Madapuri Rudra Kumar, K Sreenivasulu, Thippa Reddy Gadekallu, Surbhi Bhatia, Parul Agarwal, and Sheikh Mohammad Idrees. Early prediction of lung cancers using deep saliency capsule and pre-trained deep learning frameworks. *Frontiers in Oncology*, 2022.

Hao Ren, Jianlin Su, and Hong Lu. Evaluating generalization ability of convolutional neural networks and capsule networks for image classification via top-2 classification. *arXiv preprint arXiv:1901.10112*, 2019.

Riccardo Renzulli et al. Hierarchical object-centric learning with capsule networks. 2023.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *NeurIPS*, 2017.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE/ICCV*, 2017.

Atefeh Shahroudnejad, Parnian Afshar, Konstantinos N Plataniotis, and Arash Mohammadi. Improved explainability of capsule networks: Relevance path by agreement. In *IEEE/GlobalSIP*, 2018.

Yan Shen and Mingchen Gao. Dynamic routing on deep neural network for thoracic disease classification and sensitive area localization. In *Machine Learning in Medical Imaging: 9th International Workshop, MLMI 2018, Held in Conjunction with MICCAI*, 2018.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2013.

M Nuthal Srinivasan, Mohamed Yacin Sikkandar, Maryam Alhashim, and M Chinnadurai. Capsule network approach for monkeypox (capsmon) detection and subclassification in medical imaging system. *Scientific Reports*, 15(1):3296, 2025.

Mishmala Sushith, N Malligeswari, M Anlin Sahaya Infant Tinu, and M Jaiganesh. Hypergraph-based capsule temporal memory network for efficient and explainable diabetic retinopathy detection in retinal imaging. *Scientific Reports*, 2025.

Saja Tawalbeh and José Oramas. Towards the characterization of representations learned via capsule-based network architectures. *Neurocomputing*, 2024.

Mohamed Touafria and Qiang Yang. Sar image classification via capsule networks. In *Proceedings of the 3rd international conference on computer science and application engineering*, pp. 1–5, 2019.

Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *IEEE/CVF*, 2020a.

Zhengyang Wang, Xia Hu, and Shuiwang Ji. icapsnets: Towards interpretable capsule networks for text classification. *arXiv preprint arXiv:2006.00075*, 2020b.

Edgar Xi, Selina Bing, and Yang Jin. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*, 2017.

Yi Xiong, Guiping Su, Shiwe Ye, Yuan Sun, and Yi Sun. Deeper capsule network for complex data. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Shuai Yang, Feifei Lee, Ran Miao, Jiawei Cai, Lu Chen, Wei Yao, Koji Kotani, and Qiu Chen. Rs-capsnet: an advanced capsule network. *IEEE Access*, 8:85007–85018, 2020.

Ru Zeng, Yan Song, and Yanjiu Zhong. An interpretable unsupervised capsule network via comprehensive contrastive learning and two-stage training. *Pattern Recognition*, 158:111059, 2025.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *IEEE/CVPR*, 2016.

Binghui Zou, Chunjie Cao, Longjuan Wang, Sizheng Fu, Tonghua Qiao, and Jingzhang Sun. Facile: A capsule network with fewer capsules and richer hierarchical information for malware image classification. *Computers & Security*, 137:103606, 2024.