DECAPS: Detail-Oriented Capsule Networks

Aryan Mobiny(✉), Pengyu Yuan, Pietro Antonio Cicalese, and Hien Van Nguyen

Department of Electrical and Computer Engineering, University of Houston, Houston, USA
amobiny@uh.edu

Abstract. Capsule Networks (CapsNets) have demonstrated to be a promising alternative to Convolutional Neural Networks (CNNs). However, they often fall short of state-of-the-art accuracies on large-scale high-dimensional datasets. We propose a Detail-Oriented Capsule Network (DECAPS) that combines the strength of CapsNets with several novel techniques to boost its classification accuracies. First, DECAPS uses an Inverted Dynamic Routing (IDR) mechanism to group lower-level capsules into heads before sending them to higher-level capsules. This strategy enables capsules to selectively attend to small but informative details within the data which may be lost during pooling operations in CNNs. Second, DECAPS employs a Peekaboo training procedure, which encourages the network to focus on fine-grained information through a second-level attention scheme. Finally, the distillation process improves the robustness of DECAPS by averaging over the original and attended image region predictions. We provide extensive experiments on the CheXpert and RSNA Pneumonia datasets to validate the effectiveness of DECAPS. Our networks achieve state-of-the-art accuracies not only in classification (increasing the average area under ROC curves from 87.24% to 92.82% on the CheXpert dataset) but also in the weakly-supervised localization of diseased areas (increasing average precision from 41.7% to 80% for the RSNA Pneumonia detection dataset).

Keywords: Capsule network · Chest radiography · Pneumonia

1 Introduction

Convolutional neural networks (CNNs) have achieved state-of-the-art performance in many computer vision tasks due to their ability to capture complex representations of the desired target concept [4,11,14,16,17,19]. These architectures are composed of a sequence of convolutional and pooling layers, with

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-59710-8_15) contains supplementary material, which is available to authorized users.

A. L. Martel et al. (Eds.): MICCAI 2020, LNCS 12261, pp. 148–158, 2020.
https://doi.org/10.1007/978-3-030-59710-8_15

max-pooling being popularized in the literature due to its positive effect on performance. The max-pooling operation allows CNNs to achieve some translational invariance (meaning they can identify the existence of entities regardless of their spatial location) by attending only to the most active neuron. However, this operation has been criticized for destroying spatial information which can be valuable for classification purposes. Capsule networks (CapsNets) aim to address this fundamental problem of max pooling and has become a promising alternative to CNNs [18]. Previous works have demonstrated that CapsNets possess multiple desirable properties; they are able to generalize with fewer training examples, and are significantly more robust to adversarial attacks and noisy artifacts [5,15]. CapsNets utilize view-point invariant transformations that learn to encode the relative relationships (including location, scale, and orientation) between sub-components and the whole object, using a dynamic routing mechanism to determine how information should be categorized. This effectively gives CapsNets the ability to produce interpretable hierarchical parsing of each desired scene. By looking at the paths of the activations, we can navigate the hierarchy of the parts and know exactly the parts of an object. This property has prompted several research groups to develop new capsule designs and routing algorithms [3,5,10,13].

In recent years, CapsNets have been widely adopted and used in various medical image analysis tasks [2,6,9]. Jiao *et al.* used CapsNet for the diagnosis of mild cognitive impairment [8]. Mobiny *et al.* proposed Fast CapsNet which exploits novel techniques to improve the inference time and prediction performance of CapsNets for the lung nodule classification in 3D CT scans [15]. Lalonde *et al.* proposed SegCaps to expand capsule networks to segment pathological lungs from low dose CT scans [12].

In medical image analysis, identifying the affected area and attending to small details is critical to diagnostic accuracy. In this paper, we propose a novel version of CapsNets called Detail-Oriented Capsule Networks (DECAPS) which simulate this process by attending to details within areas that are relevant to a given task while suppressing noisy information outside of the region of interest (or ROI). We can effectively describe our architecture as having both a coarse and fine-grained stage. First, the architecture groups capsules into submodules named capsule heads, each of which is trained to extract particular visual features from the input image. It then employs an inverted routing mechanism in which groups of lower-level capsules compete with each other to characterize the task-specific regions in the image, generating coarse level predictions. The ROIs are then cropped and used in the fine-grained prediction scheme, where the model learns to interpret high-resolution representations of areas of interest. The two predictions are then combined and generate a detail-oriented output which improves performance. We will make DECAPS implementation publicly available.

2 Background on Capsule Networks

A CapsNet is composed of a cascade of capsule layers, each of which contains multiple capsules. A capsule is the basic unit of CapsNets and is defined as a

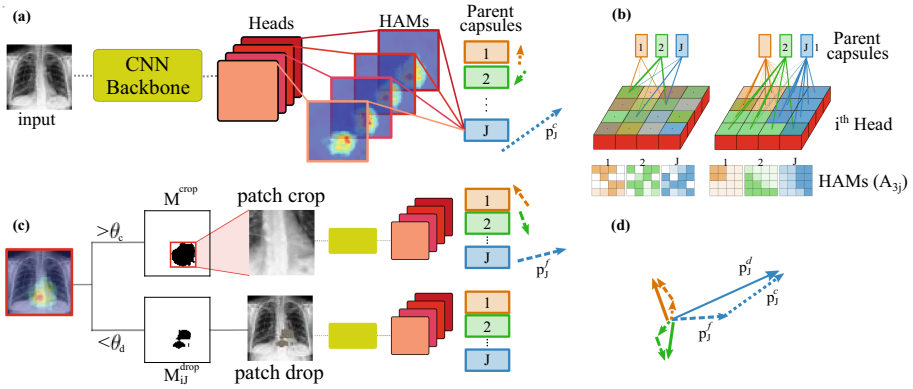

Fig. 1. (a): DECAPS architecture. Head activation maps (HAMs) are presented for the J^{th} class. **(b)**: Dynamic (left) vs. Inverted dynamic routing (right). Inverted dynamic routing places the competition between children capsules of a head yielding discriminative and localized HAMs. **(c)**: Peekaboo training. **(d)**: The distillation process to fine-tune the coarse-grained prediction (p^c) using the fine-grained prediction (p^f).

group of neurons whose output forms a *pose* vector. This is in contrast to traditional deep networks which use neurons as their basic unit. Let Ω_L denote the sets of capsules in layer L. Each capsule $i \in \Omega_L$ has a pose vector p_i^L. The length (the norm or magnitude) of the pose vector encodes the probability that an object of interest is present, while its direction represents the object's pose information, such as location, size, and orientation. The i-th capsule in Ω_L propagates its information to j-th capsule in Ω_{L+1} through a linear transformation $\mathrm{v}_{ij}^L = \mathrm{W}_{ij}^L \mathrm{p}_i^L$, where v_{ij}^L is called a *vote* vector. The pose vector of capsule $j \in \Omega_{L+1}$ is a convex combination of all the votes from child capsules: $\mathrm{p}_j^{(L+1)} = \sum_i r_{ij} \mathrm{v}_{ij}^L$, where r_{ij} are routing coefficients and $\sum_i r_{ij} = 1$. These coefficients are determined by the dynamic routing algorithm [18] which iteratively increases the routing coefficients r_{ij} if the corresponding voting vector v_{ij}^L is similar to p_j^{L+1} and vice versa. Dynamic routing ensures that the output of each child capsule gets sent to proper parent capsules. Through this process, the network gradually constructs a transformation matrix for each capsule pair to encode the corresponding part-whole relationship and retains geometric information of the input data.

Notations: Throughout the paper, r, r, R, **R** represent a scalar, a vector, a 2D matrix, and a tensor (i.e. a higher dimensional matrix; usually a 3D matrix of capsule activations), respectively. Note that multiplying a transformation matrix and a tensor of poses is equivalent to applying the transformation to each pose.

3 Detail-Oriented Capsule Network

In the original CapsNet [18], the vote of each child capsule contributes directly to the pose of all parent capsules. This ultimately has a negative effect on the

quality of the final prediction, due to the noisy votes derived from non-descriptive areas. DECAPS utilize a modified architecture, loss, and routing mechanism that favors votes from ROIs, thus improving the quality of the inputs being routed to the parent capsules. Inspired by the Transformers architecture [22], we group capsules within a grid and call them *Capsule Heads* (see Fig. 1 (a)). One can think of a capsule head as being a grid of capsules that routes information independently of the other heads. Ideally, each capsule head is responsible for detecting a particular visual feature in the input image. To accomplish this, each head shares a transformation matrix W_{ij}^{L} between all capsules for each output class. This contrasts against the original architecture which uses one transformation matrix per capsule, per class. This reduces the required number of trainable parameters by an order of head size (i.e. the number of capsules within a head, 26×26 in our proposed DECAPS); this allows DECAPS to properly scale to large, high-dimensional input images. Additionally, we use the head activation regularization loss (explained in Sect. 3.2) to force the capsules within a head to seek the same semantic concept for each diagnostic task.

Let $\mathbf{P}_{i}^{L} \in \mathbb{R}^{h_L \times w_L \times d_L}$ denote the pose matrix of the capsules of the i^{th} head where h_L and w_L represent the height and width of the head respectively, and d_L is the capsule dimension (i.e. the number of hidden units grouped together to make capsules in layer L). Note that i is the child capsule index in the original CapsNet, but is changed to represent the head index in our architecture. We want the length of the pose vector of each capsule to represent the probability of existence for a given entity of interest in the current input. The capsule outputs are passed through a nonlinear squash function [18] to ensure that the length of the pose vectors is normalized between zero and one. Then we say that $\mathbf{V}_{ij}^{L} = \mathrm{W}_{ij}^{L}\mathbf{P}_{i}^{L}$ is the votes from the capsules of the i^{th} head to the j^{th} parent capsule. To preserve the capsule's location, we perform Coordinate Addition: at each position within a capsule head, the capsule's relative coordinates (row and column) are added to the final two entries of the vote vector [5]. Once we have generated the votes, the routing mechanism determines how information should flow to generate each parent's pose vector.

3.1 Inverted Dynamic Routing

Dynamic routing [18] is a *bottom-up* approach which forces higher-level capsules to compete with each other to collect lower-level capsule votes. We propose an inverted dynamic routing (IDR) technique which implements a *top-down* approach, effectively forcing lower-level capsules to compete for the attention of higher-level capsules (see Fig. 1 (b)). During each iteration of the routing procedure, we use a softmax function to force the routing coefficients between all capsules of a single head and a single parent capsule to sum to one (see Algorithm 1). The pose of the j^{th} parent capsule, p_{j}^{L+1}, is then set to the squashed weighted-sum over all votes from the earlier layer (line 6 in Algorithm 1). Given the vote map computed as $\mathbf{V}_{ij}^{L} = \mathrm{W}_{ij}^{L}\mathbf{P}_{i}^{L} \in \mathbb{R}^{h_L \times w_L \times d_{L+1}}$, the proposed algorithm generates a routing map $\mathrm{R}_{ij} \in \mathbb{R}^{h_L \times w_L}$ from each capsule head to each output class.

Algorithm 1. Inverted Dynamic Routing (IDR). Note that i and j are the indices of capsule heads in layer L and $L+1$ respectively.

procedure IDR($\mathbf{V}_{ij}^{L}, n_{\text{iter}}$) ▷ given the votes and number of routing iterations
  $\mathrm{R}_{ij}^{\text{pre}} \leftarrow 0, \quad \forall j$ ▷ initialize the routing coefficients
  for n_{iter} iterations **do**
    $\mathrm{R}_{ij} \leftarrow \texttt{softmax}(\mathrm{R}_{ij}^{\text{pre}})$ ▷ softmax among capsules in head i
    $\tilde{\mathbf{A}}_{ij} \leftarrow \mathrm{R}_{ij} \odot \mathbf{V}_{ij}^{L}$ ▷ $\odot$ is the Hadamard product
    $\mathrm{p}_{j}^{L+1} \leftarrow \texttt{squash}(\sum_i \sum_{xy} \tilde{\mathbf{A}}_{ij})$ ▷ $\sum_{xy}$ is the sum over spatial locations
    $\mathrm{R}_{ij}^{\text{pre}} \leftarrow \mathrm{R}_{ij}^{\text{pre}} + \mathrm{p}_{j}^{(L+1)}.\mathbf{V}_{ij}^{L}$
  end for
  $\mathrm{A}_{ij} \leftarrow \texttt{length}(\tilde{\mathbf{A}}_{ij})$ ▷ `length` computes Eq. 1
  return $\mathrm{p}_{j}^{L+1}, \mathrm{A}_{ij}$
end procedure

The voting map describes the children capsules' votes for the parent capsule's pose. The routing map depicts the weights of the children capsules according to their agreements with parent capsules, with winners having the highest r_{ij}. We combine these maps to generate head activation maps (or HAMs) following

$$\mathrm{A}_{ij} = (\sum\nolimits_{d} \tilde{\mathbf{A}}_{ij}^{2})^{1/2}, \quad \text{where} \quad \tilde{\mathbf{A}}_{ij} = \mathrm{R}_{ij} \odot \mathbf{V}_{ij}^{L} \tag{1}$$

where A_{ij} is the HAM from the i^{th} head to the j^{th} parent, and $\sum_d$ is the sum over d_{L+1} channels along the third dimension of $\mathbf{V}_{ij}^{L}$. A_{ij} highlights the informative regions within an input image corresponding to the j^{th} class, captured by the i^{th} head. IDR returns as many activation maps as the number of capsule heads per output class (see Fig. 1 (a)). Class-specific activation maps are the natural output of the proposed framework, unlike CNNs which require the use of additional modules, such as channel grouping, to cluster spatially-correlated patterns [24]. We utilize the activation maps to generate ROIs when an object is detected. This effectively yields a model capable of weakly-supervised localization which is trained end-to-end; we train on the images with categorical annotations and predict both the category and the *location* (i.e. mask or bounding box) for each test image. This framework is thus able to simultaneously generate multiple ROIs within the same image that are associated with different medical conditions.

3.2 Loss Function

The loss function we define is the sum of two terms and is described as follows:

Margin Loss: We use margin loss to enforce the activation vectors of the top-level capsule j to have a large magnitude if and only if the object of the corresponding class exists in the image [18]. The total margin loss is the sum of the losses for each output capsule as given by

$$L_{\text{margin}} = \sum\nolimits_{j} [T_j \max(0, m^{+} - \|\mathrm{p}_j\|)^2 + \lambda(1 - T_j) \max(0, \|\mathrm{p}_j\| - m^{-})^2] \tag{2}$$

where $T_j = 1$ when class j is present (else $T_j = 0$). Minimizing this loss forces $\|\mathrm{p}_j\|$ of the correct class to be higher than m^+, and those of the wrong classes to be lower than m^-. In our experiments, we set $m^+ = 0.9$, $m^- = 0.1$, and $\lambda = 0.5$.

Head Activation Regularization: We propose a regularization loss function to supervise the head activation learning process. We want each head activation map A_{ij} to capture a unique semantic concept of the j^{th} output category. Inspired by center loss [23], we define a feature template $\mathrm{t}_{ij} \in \mathbb{R}^{d_{L+1}}$ for the i^{th} semantic concept of the j^{th} output category. We compute the semantic features $\mathrm{f}_{ij} \in \mathbb{R}^{d_{L+1}}$ using the information routed from the i^{th} head to the j^{th} output category. While the magnitude of f_{ij} represents the presence of the desired semantic concept, the orientation captures the instantiation parameters (i.e. pose, deformation, texture, etc.). We, therefore, want to regularize the orientation of f_{ij} for a given capsule head to guarantee that it is capturing the same semantic concept among all training images. Each value of t_{ij} is initialized to zero and is updated using a moving average as

$$\mathrm{t}_{ij} \leftarrow \mathrm{t}_{ij} + \gamma(\hat{\mathrm{f}}_{ij} - \hat{\mathrm{t}}_{ij}), \quad \text{where } \mathrm{f}_{ij} = 1/n_i \sum\nolimits_{xy} \tilde{\mathbf{A}}_{ij}, \tag{3}$$

where $\hat{\mathrm{f}}_{ij}$ and $\hat{\mathrm{t}}_{ij}$ are the normalized vectors, γ is the update step, which we set to 10^{-4}, while n_i represents the number of capsules in head i. To accomplish this, we penalize the network when the orientation of a head's features f_{ij} deviate from the template t_{ij} following

$$L_{\mathrm{HAR}} = \frac{1}{IJ} \sum\nolimits_i \sum\nolimits_j (1 - \texttt{cosine}(\mathrm{f}_{ij}, \mathrm{t}_{ij})) \tag{4}$$

where I and J represent the total number of child and parent capsules.

3.3 Peekaboo: The Activation-Guided Training

To further promote DECAPS to focus on fine-grained details, we propose the Peekaboo strategy for capsule networks. Our strategy boosts the performance of DECAPS by forcing the network to look at all relevant parts for a given category, not just the most discriminative parts [20]. Instead of hiding random image patches, we use the HAMs to guide the network's attention process. For each training image, we randomly select an activation map A_{ij} for each recognized category. Each map is then normalized in the range $[0, 1]$ to get the normalized HAM $\mathrm{A}^*_{ij} \in \mathbb{R}^{h_L \times w_L}$. We then enter a two step process: patch cropping, which extracts a fine-grained representation of the ROI to learn how to encode details, and patch dropping, which encourages the network to attend to multiple ROIs. In patch cropping, a mask $\mathrm{M}^{crop}_{ij} \in \mathbb{R}^{h_L \times w_L}$ is obtained by setting all elements of A^*_{ij} which are less than a cropping threshold $\theta_c \in [0, 1]$ to 0, and 1 otherwise. We then find the smallest bounding box which covers the entire ROI, and crop it from the raw image (Fig. 1 (c)). It is then upsampled and fed into the network to generate a detailed description of the ROI. During the patch dropping procedure, M^{drop}_{ij} is used to remove the ROI from the raw image by using a dropping threshold $\theta_d \in [0, 1]$. The new patch-dropped image is then fed to the network

for prediction. This encourages the network to train capsule heads to attend to multiple discriminative semantic patterns. At test time, we first input the whole image to obtain the coarse prediction vectors (p_j^c for the j^{th} class) and the HAMs A_{ij} from all capsule heads. We then average all maps across the heads, crop and upsample the ROIs, and feed the regions to the network to obtain the fine-grained prediction vectors (p_j^f). The final prediction p_j^d, referred to as distillation (Fig. 1 (d)), is the average of the p_j^c and p_j^f.

4 Experiments and Results

Implementation Details: In our experiments, we use Inception-v3 as the backbone and take the *Mix6e* layer output as the CNN feature maps. We then compress the feature maps using 1×1 convolutional kernels to generate 256 maps. We split the maps into four capsule heads, each of which includes a grid of 64-dimensional capsules. These heads employ the described inverted routing procedure to route into the final 16-dimensional class capsules. The best performance was achieved using 3 routing iterations, $\theta_c = 0.5$, and $\theta_d = 0.3$. The network is trained using the Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and a learning rate of 10^{-4} which is fixed for the duration of training.

Datasets: We use two datasets to evaluate the performance of the proposed DECAPS architecture. The CheXpert [7] radiography dataset is used for the detection of 5 selected pathologies, namely Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural effusion (see Table 1 of [7] for more information on the data distribution). We also report our results on the RSNA Pneumonia detection data which includes bounding box annotations of the affected regions in the images [1]. It is important to note that our approach only uses the category labels (not the bounding boxes) for the localization of pneumonia localization.

Evaluation Metrics: We use the area under ROC curve (AUC) to report the prediction accuracy on the CheXpert dataset. We use the mean intersection over union (mIoU) to evaluate the localization accuracy of the model on the RSNA dataset. We also compute the average precision (AP) at different IoU thresholds. At each threshold, a true positive (TP) is counted when a predicted object matches a ground truth object with an IoU above the threshold. A false

Table 1. Prediction performance of models trained on the CheXpert dataset. For each model, average result is reported over the best 10 trained model checkpoints.

	Cardiomeg.	Edema	Consolid.	Atelectasis	Pleural Eff.	mAUC (%)
Inception-v3 [21]	0.841(±0.052)	0.876(±0.055)	0.891(±0.044)	0.833(±0.032)	0.921(±0.038)	87.24
DenseNet121 [7]	0.832(±0.047)	0.941(±0.031)	0.899(±0.037)	0.858(±0.042)	0.934(±0.027)	89.28
DenseNet121+HaS [20]	0.849(±0.041)	0.940(±0.055)	0.904(±0.039)	0.867(±0.050)	0.938(±0.024)	89.96
CapsNet [18]	0.835(±0.033)	0.915(±0.038)	0.890(±0.035)	0.845(±0.031)	0.949(±0.033)	88.68
DECAPS	0.852(±0.048)	0.935(±0.039)	0.897(±0.028)	0.865(±0.045)	0.946(±0.022)	89.90
DECAPS+Peekaboo	**0.895**(±0.044)	**0.972**(±0.027)	**0.913**(±0.033)	**0.883**(±0.029)	**0.978**(±0.019)	**92.82**

positive (FP) indicates a predicted object had no associated ground truth object. A false negative (FN) indicates a ground truth object had no associated predicted object. We then calculated AP as TP/(TP+FP+FN) over all test samples [1].

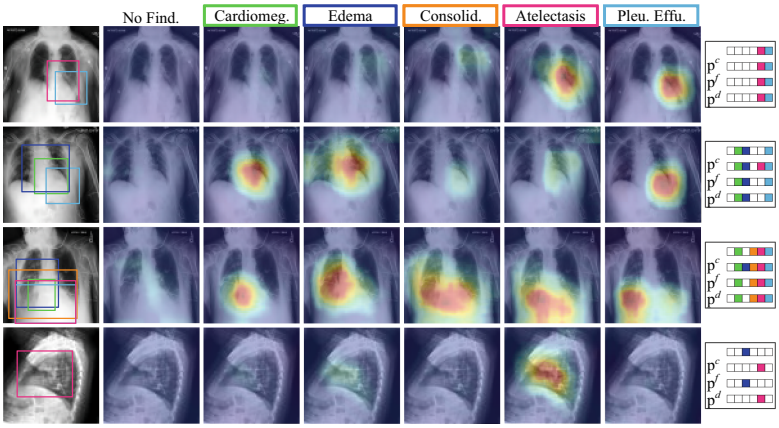

Fig. 2. Qualitative results on the CheXpert dataset. True pathologies (T), coarse (p^c), fine-grained (p^f), and distilled (p^d) predictions are presented for each case.

Results on CheXpert Dataset: The quantitative classification results are summarized in Table 1. We compare our results with Inception-v3 (which is the backbone used in our framework), DenseNet121 (the best performing baseline CNN according to [7]), a DenseNet121 model trained with the Hide-and-Seek (HaS) strategy [20] to boost the weakly-supervised localization of the model, and the vanilla CapsNet with the same backbone. The vanilla DECAPS architecture yielded significantly higher classification accuracies than the baseline networks and achieves performance on par with DenseNet121+HaS. Adding the proposed Peekaboo method to our framework significantly improves the prediction and localization performance of the model. Examples are shown in Fig. 2. Each HAM is activated when the model detects a visual representation associated with the pathology of interest. It highlights the ROI which will be cropped and passed through the fine-grained prediction stage to then generate the distilled prediction. The first row in Fig. 2 shows an example with accurate classifications, while the second and third rows show samples that benefit from fine-grained prediction and distillation (atelectasis and edema are correctly removed). The fourth row shows a failure case that is diagnosed as Edema, but predicted as Atelectasis (see more examples and ablation study in the supplementary material section).

Results on RSNA Dataset: We compare our qualitative results with a weakly-supervised localization approach [20], as well as a supervised (Faster RCNN [17]) detection method as shown in Table 2. The prediction and localization metrics are computed at two levels for the DECAPS model: level-1 refers to the coarse

Table 2. Test prediction accuracy (%), mean intersection over union (mIoU), and average precision (AP) over various IoU thresholds for RSNA Pneumonia detection.

	Unsupervised	%Acc	mIoU	$AP_{0.3}$	$AP_{0.4}$	$AP_{0.5}$	$AP_{0.6}$
Inception-v3+HaS	✓	87.14	0.314(±0.321)	0.417	0.370	0.241	0.194
Faster-RCNN		92.77	**0.611(±0.125)**	**0.887**	**0.853**	**0.718**	**0.561**
DECAPS (level-1)	✓	86.25	0.401(±0.176)	0.642	0.537	0.460	0.322
DECAPS (level-2)	✓	**94.02**	0.509(±0.130)	0.800	0.771	0.594	0.471

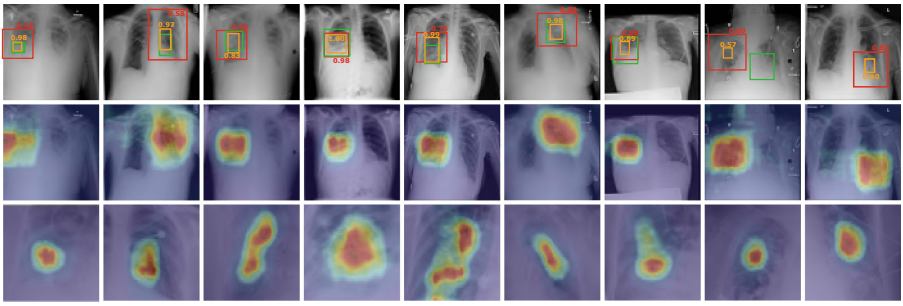

Fig. 3. Qualitative results on the RSNA dataset. **Top:** Input images with ground truth (green), level-1 (red), and level-2 (orange) bounding boxes. **Middle** Course-grained (level-1) activation maps. **Bottom:** Fine-grained (level-2) activation maps. (Color figure online)

prediction on the whole image while level-2 refers to the localization result of the fine-grained prediction (i.e. the ROI within the cropped region, examples shown in Fig. 3). We observe that the fine-grained prediction stage significantly improves the weakly-supervised localization performance over the coarse prediction stage and the baseline weakly supervised method (Inception-v3+HaS). We also note that the prediction accuracy of the fine-grained prediction stage exceeds the supervised Faster-RCNN prediction diagnosis, while localization accuracy is lower. We hypothesize that this is due to the coarse nature of the ground truth bounding boxes which also capture superfluous information from other tissues.

5 Conclusions

In this work, we present a novel network architecture, called DECAPS, that combines the strength of CapsNets with detail-oriented mechanisms. DECAPS is first applied to the whole image to extract global context and generate saliency maps that provide coarse localization of possible findings. This is analogous to a radiologist roughly scanning through the entire image to obtain a holistic view. It then focuses in the informative regions to extract fine-grained visual details from the ROIs. Finally, it employs a distillation process that aggregates information from both global context and local details to generate the final prediction. DECAPS achieves the highest accuracies on CheXpert and RSNA Pneumonia

datasets. Despite being trained with only image-level labels, DECAPS are able to accurately localize the region of interests which enhances the model's interpretability. We expect our method to be widely applicable to image detection and recognition tasks, especially for medical image analysis tasks where small details significantly change the diagnostic outcomes.

Acknowledgments. This research was supported by the National Science Foundation (NSF-IIS 1910973).

References

1. RSNA pneumonia detection challenge. https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview/evaluation. Accessed 15 Mar 2020
2. Afshar, P., et al.: 3D-MCN: a 3D multi-scale capsule network for lung nodule malignancy prediction. Sci. Rep. **10**(1), 1–11 (2020)
3. Ahmed, K., Torresani, L.: Star-caps: capsule networks with straight-through attentive routing. In: Advances in Neural Information Processing Systems, pp. 9098–9107 (2019)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
5. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. In: International Conference on Learning Representations (2018)
6. Iesmantas, T., Alzbutas, R.: Convolutional capsule network for classification of breast cancer histology images. In: Campilho, A., Karray, F., ter Haar Romeny, B. (eds.) ICIAR 2018. LNCS, vol. 10882, pp. 853–860. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93000-8_97
7. Irvin, J., et al.: ChexPert: a large chest radiograph dataset with uncertainty labels and expert comparison. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 590–597 (2019)
8. Jiao, Z., et al.: Dynamic routing capsule networks for mild cognitive impairment diagnosis. In: Shen, D., et al. (eds.) MICCAI 2019. LNCS, vol. 11767, pp. 620–628. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32251-9_68
9. Jiménez-Sánchez, A., Albarqouni, S., Mateus, D.: Capsule networks against medical imaging data challenges. In: Stoyanov, D., et al. (eds.) LABELS/CVII/STENT -2018. LNCS, vol. 11043, pp. 150–160. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01364-6_17
10. Kosiorek, A., Sabour, S., Teh, Y.W., Hinton, G.E.: Stacked capsule autoencoders. In: Advances in Neural Information Processing Systems, pp. 15486–15496 (2019)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
12. LaLonde, R., Bagci, U.: Capsules for object segmentation. arXiv preprint arXiv:1804.04241 (2018)
13. Mobiny, A., Lu, H., Nguyen, H.V., Roysam, B., Varadarajan, N.: Automated classification of apoptosis in phase contrast microscopy using capsule network. IEEE Trans. Med. Imaging **39**(1), 1–10 (2019)
14. Mobiny, A., Singh, A., Van Nguyen, H.: Risk-aware machine learning classifier for skin lesion diagnosis. J. Clin. Med. **8**(8), 1241 (2019)

15. Mobiny, A., Van Nguyen, H.: Fast CapsNet for lung cancer screening. In: Frangi, A.F., Schnabel, J.A., Davatzikos, C., Alberola-López, C., Fichtinger, G. (eds.) MICCAI 2018. LNCS, vol. 11071, pp. 741–749. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00934-2_82
16. Ravindran, A.S., Mobiny, A., Cruz-Garza, J.G., Paek, A., Kopteva, A., Vidal, J.L.C.: Assaying neural activity of children during video game play in public spaces: a deep learning approach. J. Neural Eng. **16**(3), 036028 (2019)
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
18. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems (2017)
19. Shahraki, F.F., Prasad, S.: Graph convolutional neural networks for hyperspectral data classification. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 968–972. IEEE (2018)
20. Singh, K.K., Lee, Y.J.: Hide-and-seek: forcing a network to be meticulous for weakly-supervised object and action localization. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 3544–3553. IEEE (2017)
21. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
22. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
23. Wen, Y., Zhang, K., Li, Z., Qiao, Yu.: A discriminative feature learning approach for deep face recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 499–515. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_31
24. Zheng, H., Fu, J., Mei, T., Luo, J.: Learning multi-attention convolutional neural network for fine-grained image recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5209–5217 (2017)