CaReNet: CapsNet based Regression Network*

(accepted and presented at NCVPRIPG,2019)

Abhinay B, Basanta Sharma, Darshan Gera, S Balasubramanian, and Srikanth Khanna

Sri Sathya Sai Institute of Higher Learning, Anantapur, AP, India
abhinayb.sssihl@gmail.com, basantanickal@gmail.com,
darshangera@sssihl.edu.in, srikanthkhanna@sssihl.edu.in,
sbalasubramanian@sssihl.edu.in
http://sssihl.edu.in

Abstract. We propose a capsule based regression network (CaReNet), a framework that is based on capsule networks (CapsNet), rather than on the conventional convolutional neural networks (CNNs) to determine estimates of continuous variables. The core principles of CaReNet remain that of routing-by-agreement and translation equivariance proposed in the CapsNet architecture for classification problems. However, unlike the CapsNet architecture, the final layer capsules in CaReNet architecture capture the number of values to regress in their dimensionality. An output vector returned by each of these capsules contains all the regressed values while the corresponding activity vector, determined by squashing an output vector, captures the likelihood of the regressed values being present in the corresponding capsule. We show that our novel CaReNet architecture achieves state-of-the-art performance on regressing facial keypoints from images, an important problem that plays a significant role in face recognition systems. The performance of CaReNet has been evaluated on the dataset from Kaggle Facial Keypoints Detection competition. Further, the flexibility in CaReNet architecture allows it to be easily extendable to any other regression problem such as 3D reconstruction of facial models from 2D images.

Keywords: Capsule Networks · Regression · Facial Keypoint Detection.

1 Introduction

Given a continuous dependent variable Y and an independent variable X, regression computes the most probable value of Y for each value of X using a finite set of instances of X and the corresponding instances of Y. Unlike classification which involves the prediction of a class label, regression involves the prediction of a continuous variable. Detecting facial keypoints is one such regression problem that finds its use in tracking and recognizing faces, analyzing facial expressions and in detecting dysmorphic facial signs for medical diagnosis [4]. CNN-based

* Supported by SSSIHL.

regression have achieved more accurate and efficient results compared to the traditional methods in the recent past. For example, traditional methods that detect local features in an image for facial keypoint detection fall into local minima due to ambiguous patches in an image [13], [6]. Cascading CNNs [10] avoid this problem by using multiple CNNs at various levels to predict and fine tune the values. Data augmentation is also used to obtain better results [5]. However, CNNs are limited by the availability of large number of ground truths. For instance, in facial keypoint detection, some of the ground truth keypoints are not disclosed due to difference of angles in view points. As CNNs are not robust to viewpoint variance, these images can lead to misclassification. Removing these images will also not do good as the training set will become smaller.

Capsule network (CapsNet) [9], is a state-of-the-art method in deep learning, based on the principles of viewpoint equivariance and inverse rendering. It has been proven to work well in the domain of classification such as vehicle type recognition [12], traffic sign recognition [2], fire hazard detection [11], etc. Development work of regression models based on CapsNet has just begun [1], [8]. we propose a general capsule based regression network (CaReNet) for facial keypoint detection. We also highlight the improved accuracy obtained by the proposed method on the Kaggle Facial Keypoints Detection dataset [4]. Section 2 discusses the existing work while section 3 elaborates on the proposed CaReNet architecture. Performance of the proposed method is demonstrated in section 4. Section 5 concludes our work and discusses possible future work.

2 Background

2.1 Limitations of CNN

CNNs have three main limitations: Firstly, they need a large dataset to correctly classify or regress from the given data. Secondly, CNNs achieve positional invariance through max pooling and accurately classify even unknown objects with minor changes in positions compared to the objects in the training set. However, positional invariance can lose a lot of important spatial information leading to misclassification. Thirdly, CNNs are not robust to variances in pose. To overcome these limitations, Hinton et al [3], [9] have come up with the concept of positional and view equivariance through the idea of CapsNet.

2.2 Capsule Networks

A CapsNet is essentially a network of many layers of capsules. Each capsule is a group of neurons. Unlike a neuron in a traditional CNN whose I/O is a scalar, a capsule operates on a vector and squashes it using a non-linear activation.

The magnitude of the squashed output vector quantifies the probability of the feature detected by the respective capsule while the vector's direction represents the state, orientation or any other instantiation parameters of the detected feature.

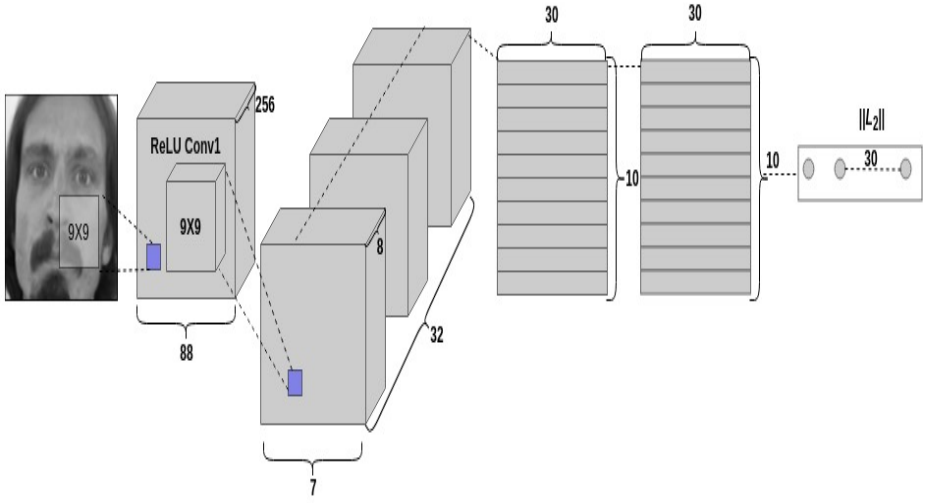


Fig. 1: Our proposed 2D CaReNet encoder architecture.

Sabour et al. [9] proposed a routing-by-agreement algorithm between the layers of capsules to dynamically determine the path taken by the passage of information from one capsule layer to another. For every capsule g_i^l in layer l and capsule g_j^{l+1} in layer $l+1$, a coupling coefficient c_{ij} is iteratively learnt from the agreement between the g_i^l's prediction of g_j^{l+1}'s output and the actual output of g_j^{l+1} (which is captured using cosine similarity).

3 Design of 2DCaReNet for regressing facial keypoints

The architecture of our 2D capsule-based regression network (2D CaReNet) for facial key point detection has an encoder structure (as shown in figure 1) with the following layers similar to that in [9]:

1. Convolutional layer: This layer has 256 convolution kernels of size 9×9 with a stride of 1 and ReLU activation unit.
2. PrimaryCapsule layer: This layer is made up of 32 capsules and each capsule has 8 convolutional units with each having a kernel of size 16×16 and stride of 12.
3. RoutingCapsule layer: This layer is made up of ten 30D capsules. Each of the capsules obtains the input from all the capsules in PrimaryCapsules. Further, each capsule outputs a vector that gives the corresponding activity vector of the capsule when squashed using the squash function given in

equation. As the length of an activity vector determines the probability that an entity exists at the corresponding capsule, we use the squashed vectors or the activity vectors to determine the most probable capsule to be routed dynamically. The non-squashed values are not used for any other purpose in this layer.

4. DetectionCapsule layer: This layer is also made up of ten 30D capsules and the squashed vectors are used to dynamically determine the capsule to be routed to. Further, in this layer, the **non-squashed vector with the highest length** in the corresponding activity vector is considered as the regressed output vector that captures the expected keypoints in its dimensions. Note that both the non-squashed values and the squashed values are used here. The squashed values give the probability of routing to a particular capsule (as proposed in CapsNet) while non-squashed values give the expected key points regressed by that capsule(proposed in this work).

Algorithm 1 Routing algorithm

procedure ROUTING($\hat{\mathbf{u}}_{j|i}, r, l$)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j, \mathbf{s}_j$ ▷ return **both** squashed and non-squashed values

We use routing-by-agreement between PrimaryCapsule layer and the RoutingCapsule layer, and between RoutingCapsule layer and the DetectionCapsule layer. However, unlike the routing algorithm in [9] which returns only the squashed values, our routing algorithm returns **both the squashed values and the non squashed values**. The non-squashed values are used in the final layer, namely, DetectionCapsule layer as the output vectors as explained before.

The loss used to compare the output and the ground truth is MSE instead of the margin loss defined for classification in [9].

The decoder structure is similar to the CapsNet classification architecture as in [9]. The dimensions of the three fully connected layers are 30×1024, 1024×4096 and 4096×9216 respectively. The reconstruction loss is calculated using sum of squared errors (SSE). The over all loss is calculated as in [9]:

$$loss = MSE_{encoder} + 0.00005 * SSE_{decoder} \tag{1}$$

Here, the decoder loss is scaled down by 0.00005 so that the it does not dominate the encoder loss during the training phase. In the absence of scaling, the few regressed points try to decode the the whole of the original image leading to an under-fitted model and inaccurate results. On scaling down the decoder

Hyper-parameters	Values
Data split	80% - 20%
Optimizer	Adam
Learning rate	0.0001
Batch size	16
Weight initialization	Uniform distribution with mean 0 and variance 1
Total training epochs	300

Table 1: Hyperparameters set (using random search where applicable) for CaReNet

loss, the decoder only captures and reconstructs the most relevant features in the original image (just as in [9]).

4 Performance Evaluation

4.1 Dataset

In our implementation of CaReNet for facial keypoints detection, we used the Kaggle dataset [4], a dataset that consists of 7049 96 × 96 images out of which, only 2140 images have all the 15 key points labelled.

4.2 Training

With the given dataset, we performed three experiments:

1. In the first experiment, we created a test set named, $test_{split}$ containing 100 (i.e 1.5 %) of the given 7049 training images with complete labels. This test set was essential for performing quantitative analysis in the evaluation phase as the test images provided by Kaggle do not have any labels. The rest of the training set named, $training_{split}$ was used in the training phase for both quantitative and qualitative analysis.
2. In the second experiment, we created a training set named $train_{allLabels}$ containing the 2140 images of the given training set with all the labels present. The test evaluation was done on the given 1783 test images provided in the Kaggle platform.
3. In the third experiment, we used all of the images provided for training and performed the quantitative model evaluation on the test set provided in the Kaggle platform.

The given test images were also used for qualitative analysis in all the experiments. The hyper-parameters of the architecture model were obtained using *random search* and were set to the various values as shown in the table 1.

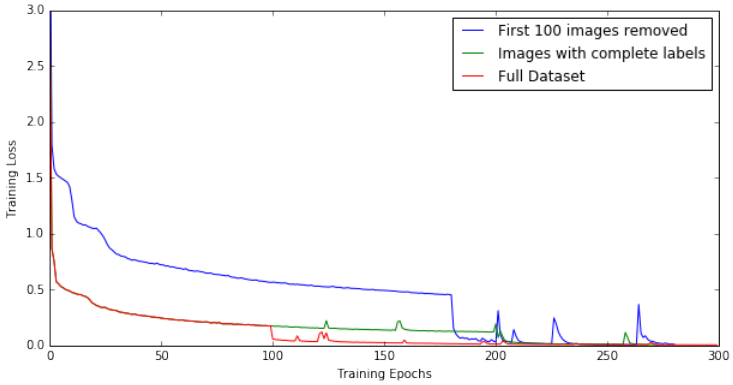


Fig. 2: Training evaluation of CaReNet with various experiments & their training sets. CaReNet converges after 200 epochs in all the experiments

4.3 Results

Training error In the training phase, the quantitative analysis between the output vector o and the ground truth vector g was performed using MSE (defined in equation 2) averaged over all the training images. Figure 2 shows the training evaluation of CaReNet with all the datasets used in the various experiments. Clearly, the model converges after 200 training epochs in all of the experiments. The average MSE converged to 0.0107, 0.008 and 0.1756 when our model was built on the $training_{split}$, $training_{allLabels}$ and the given complete training set respectively. However, this only shows that the model works well on training set. The next subsection evaluates the model on a test set.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(g_i - o_i)^2 \tag{2}$$

Test error / Prediction accuracy In the test phase, we evaluate the prediction accuracy of the output o compared to the ground truth g using the root-mean-square error (RMSE) (defined in equation 3).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(g_i - o_i)^2} \tag{3}$$

As there are two convolutional layers in our network, we used a **baseline model** that also contained two convolutional layers of 256, 128 channels with respective kernels of size 6×6, 4×4 and strides of size 4 and 2. Two fully connected layers of sizes 1152, 512 interspersed with a ReLU unit follow the last convolutional layer. Table 2 shows the results obtained with these models when run on the various training sets and the test sets.

Model	Training set		RMSE on	
	Name	Size	$test_{split}$	Kaggle test set
Baseline	$training_{split}$	6949 images	2.355	3.314
Baseline	$training_{allLabels}$	2143 images	-	3.349
Baseline	Kaggle training set	7049 images	-	3.350
CaReNet	$training_{split}$	6949 images	2.217	3.100
CaReNet	$training_{allLabels}$	2143 images	-	3.108
CaReNet	Kaggle training set	7049 images	-	3.113

Table 2: Evaluation of models (refer section 4.3) using various training sets (refer section 4.2)

In the first experiment, we obtained a total RMSE of 2.217 on the 100 images of $test_{split}$ and a RMSE of 3.100 on the Kaggle test set using the Kaggle platform. In the second and third experiments, we obtained a RMSE of 3.108 and 3.113 respectively on the Kaggle test set using Kaggle platform. In all the cases, the errors obtained are lesser than the corresponding baseline models. Further, such low test errors that have been obtained on our 4-layer network were previously attained by much deeper networks [7] that are convolutional. Another observation is that both the baseline and CaReNet models performed best when trained on $training_{split}$ dataset. However, the growth of error differs significantly between these models when the training set changes from $training_{split}$ to $training_{allLables}$. The ratio of test error growth in the baseline model and CaReNet is $4:1$. Further, the difference between the test errors of CaReNet in the two experiments is infinitesimal while the size of the dataset reduces by 77%. This emphasises that CaReNet not only performs better than CNN, but also achieves optimal performance even on small datasets. However, the time taken to train the current implementation of CaReNet is much higher than that of CNN, a set back we wish to overcome as part of our future work.

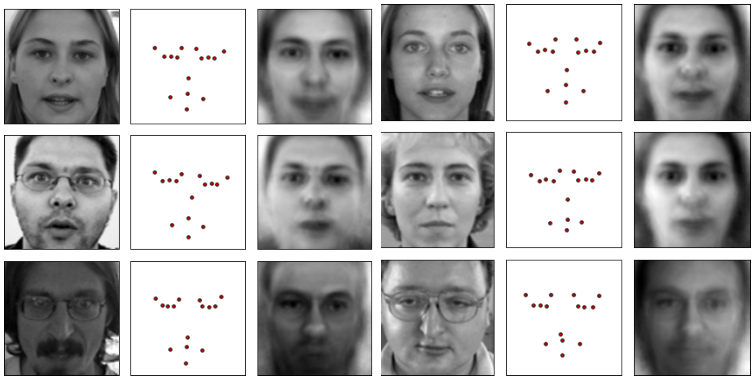
Prediction Visualization Results of first experiment on the labelled $test_{split}$ set are shown in figure 3. In the second and third experiments, we relied solely on the visual perception of the test images with no labels. Figure 4 shows that our model has regressed very appealing facial keypoints when trained on all the training images.

As part of the qualitative analysis, we have also visualized the reconstructed image returned by the decoder structure using the encoded keypoints. The reconstructed images were very similar in all the experiments and are shown in figure 5 when our model is trained on $test_{split}$ and run on the Kaggle test set. The figure shows that the reconstructed images, learnt solely from the keypoints with the given image as the ground truth, capture the areas around the keypoints (like the corner of eyes, tip of the nose) sharply and accurately while blurring the rest of the areas of the input image. Hence, the decoder structure uses the keypoints obtained from the encoder and acts like a filter that blurs out

Fig. 3: Results with $test_{split}$ images. Model output is plotted using red dots(.) and ground truth is using green stars(*).

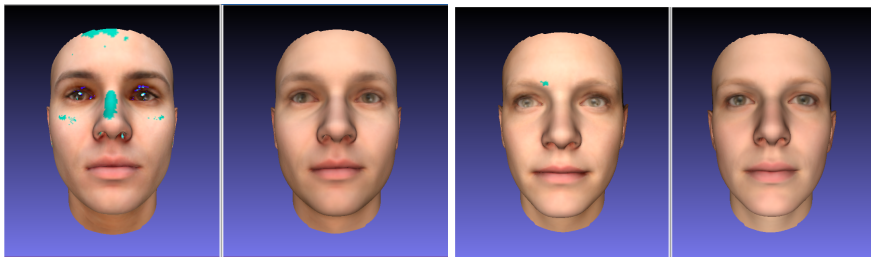
Fig. 4: Results with test images. Model output is plotted using red dots(.)

all the features in the input images which do not lie in the neighbourhood of the keypoints.



(a) (b)

Fig. 5: Results of CaReNet - In both (a) and (b): Left: Input image, Middle: Encoded keypoints, Right: Image decoded from the keypoints.



(a) (b) a

Fig. 6: Results CaReNet on 3D reconstruction - In both (a) and (b): Left: Ground truth, Right: Output.

Though the facial keypoint detection using Kaggle dataset is sufficient to prove that CaReNet works for regression, we carried out a similar experiment on 3D facial reconstruction from 2D images by regressing the shape and texture. We used 300W-3D [14], a dataset that contains images from the 300-W dataset and the corresponding 3DMM parameters. Hyper-parameters such as the length of the capsules in detectionCapsule layer were changed and the results obtained are shown in figure 6. Clearly, this shows that the architecture is flexible and can be easily modified for other regression problems.

5 Conclusion and future work

We have proposed a framework for regression network called capsule based regression (CaReNet) which uses capsule networks (CapsNet) rather than the traditional convolutional neural networks (CNNs) to determine estimates of con-

tinuous variables. CaReNet is based on the principles of routing-by-agreement and translation equivariance proposed in the CapsNet architecture for classification problems. However, unlike the CapsNet architecture that captures the localized part of the target object in the dimensions of its final layer, CaReNet captures the number of values to regress in their dimensionality. Each of these capsules select an output vector that contains the regressed values based on the corresponding activity vector that captures the likelihood of the regressed values being present in that output vector. We achieved impressive performance on the dataset from Kaggle Facial Keypoints Detection competition that were achievable with much deeper convolution networks in the past. This shows that our novel CaReNet architecture can achieve state-of-the-art performance on regressing facial keypoints from images. Further, our architecture is very flexible and can be easily extended to other regression domains such as 3D reconstruction through inverse rendering.

The implementation for CaReNet can be found here.

References

1. Deng, C., Zhang, L., Cen, Y.: Retrieval of chemical oxygen demand through modified capsule network based on hyperspectral data. Applied Sciences **9**(21), 4620 (2019)
2. Guan, H., Yu, Y., Peng, D., Zang, Y., Lu, J., Li, A., Li, J.: A convolutional capsule network for traffic-sign recognition using mobile lidar data with digital images. IEEE Geoscience and Remote Sensing Letters (2019)
3. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: International Conference on Artificial Neural Networks. pp. 44–51. Springer (2011)
4. Kaggle: Facial keypoints detection dataset. https://www.kaggle.com/c/facial-keypoints-detection/data
5. Kimura, M., Yamashita, T., Yamauchi, Y., Fujiyoshi, H.: Facial point detection based on a convolutional neural network with optimal mini-batch procedure. In: IEEE International Conference on Image Processing (ICIP). pp. 2860–2864 (2015)
6. Liang, L., Xiao, R., Wen, F., Sun, J.: Face alignment via component-based discriminative search. In: European conference on computer vision. pp. 72–85. Springer (2008)
7. Longpre, S., Sohmshetty, A.: Facial keypoint detection. Stanford University (2016)
8. Ramírez, I., Cuesta-Infante, A., Schiavi, E., Pantrigo, J.J.: Bayesian capsule networks for 3d human pose estimation from single 2d images. Neurocomputing (2019)
9. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems. pp. 3859–3869 (2017)
10. Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3476–3483 (2013)
11. Yaloveha, V., Hlavcheva, D., Podorozhniak, A., Kuchuk, H.: Fire hazard research of forest areas based on the use of convolutional and capsule neural networks. In: 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON). pp. 828–832. IEEE (2019)
12. Zhang, Z., Zhang, D., Wei, H.: Vehicle type recognition using capsule network. In: 2019 Chinese Control And Decision Conference (CCDC). pp. 2944–2948. IEEE (2019)

13. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2879–2886 (2012)
14. Zhu, X., Lei, Z., Liu, X., Shi, H., Li, S.Z.: Face alignment across large poses: A 3d solution. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 146–155 (2016)