Assessing Capsule Networks With Biased Data

Bruno Ferrarini[1], Shoaib Ehsan[1], Adrien Bartoli[2], Aleš Leonardis[3], and Klaus D. McDonald-Maier[1]

[1] University of Essex, CSEE, Wivenhoe Park, Colchester CO4 3SQ, UK
{bferra,sehsan,kdm}@essex.ac.uk
[2] Faculté de Médecine, 28 Place Henri Dunant, 63000 Clermont-Ferrand, France
Adrien.Bartoli@gmail.com
[3] University of Birmingham, School of Computer Science, Birmingham B15 2TT, UK
a.leonardis@cs.bham.ac.uk

Abstract. Machine learning based methods achieves impressive results in object classification and detection. Utilizing representative data of the visual world during the training phase is crucial to achieve good performance with such data driven approaches. However, it not always possible to access bias-free datasets thus, robustness to biased data is a desirable property for a learning system. Capsule Networks have been introduced recently and their tolerance to biased data has received little attention. This paper aims to fill this gap and proposes two experimental scenarios to assess the tolerance to imbalanced training data and to determine the generalization performance of a model with unfamiliar affine transformations of the images. This paper assesses dynamic routing and EM routing based Capsule Networks and proposes a comparison with Convolutional Neural Networks in the two tested scenarios. The presented results provide new insights into the behaviour of capsule networks.

Keywords: Capsule Networks · Bias · Comparison · Evaluation

1 Introduction

A robust classification system is expected to give the same prediction for every image of the same class or for images representing the same element in different poses. Machine learning methods, such as Convolutional Neural Networks (CNN), have been used in many classification, detection and recognition tasks [16,10,3]. However, in order to achieve good performance with data driven approaches, well representative data of the visual word are required [19,14,11]. While it is possible to mitigate some bias effects with de-biasing techniques [12] or with data augmentation [23], it is important to use machine

learning approaches with good generalization performance as it contributes to design more robust applications to unseen or underrepresented imaging conditions. This paper focuses on the latter topic and presents a comparison between Convolutional Neural Networks (CNNs) and Capsule Networks (CapsNets) [22,7]. The neurons in a CapsNet are organized in groups denoted as Capsules [8]. In contrast to a single neuron, a capsule can learn a specific image entity over a range of viewing conditions such as viewpoint and rotation. With the use of a routing algorithm to interconnect the capsules, a CapsNet model would be affine invariant and spatially aware. While the behaviour of CNNs with biased data has been extensively investigated [11,14,15], how bias influences CapsNets' performance has received little attention so far.

This paper aims to fill this gap by proposing two experimental scenarios. The first experiment set evaluates a model's classification accuracy with unfamiliar affine transformations. It introduces a capture bias [26] obtained with training and test data having transformation intensities sampled from different distributions. The second test scenario is to assess the variation of a network's performance when trained with a dataset presenting several overrepresented classes with respect to evenly distributed classes. The results are presented for five network models: three dynamic routing-based CapsNet [22] with one, two and three capsule layers respectively, an EM-Matrix routing CapsNet [7] and for a CNN, which represents a comparison baseline.

The rest of this paper is organized as follows. Section 2 provides an overview of related work; Section 3 gives an introduction on capsule networks; Section 4 describes the method and criteria used for the performance evaluation. The results obtained are presented and discussed in Section 5. Finally, Section 6 draws conclusions and proposes possible extensions.

2 Related Work

The impact of bias on data driven methods have been extensively explored in the literature. A review of various types of bias in machine learning datasets is provided in [5]. The problem of bias in popular datasets dissected by cause is presented in [26] and further discussed

in [25] where several de-biasing methods are compared. The generalization performance of CNNs is assessed with unfamiliar scale factor in [11] and with unfamiliar yaw pose and lighting conditions in [14], utilizing face recognition tasks. The analysis of imbalanced data is addressed in [19] and [2]. In [19] several imbalanced datasets are built from CIFAR-10 [15] by means of class down and over-sampling and used to assess CNNs. In [2], the importance of choosing the suitable performance evaluation metric in the presence of imbalanced classes is discussed. To the best of our knowledge, the only work addressing the generalization problem for CapsNets is [6], which demonstrates that dynamic routing based CapsNets generalize faster than CNNs when training data is injected with a few examples of an unfamiliar class. Only a few other works analyze this type of CapsNet but without considering bias or generalization performance: [27] and [20] only test CapsNets with more complex data than those utilized in the original paper [22]. Our paper aims to fill these gaps by proposing an analysis of the generalization performance with unfamiliar affine transformations and imbalanced training data for both the available architectures of CapsNets: dynamic routing [22] (denoted as Vector-CapsNet from now on) and EM-Matrix routing based [7] (MatrixEM-CapsNet).

3 Capsule Networks

A capsule is a group of neurons whose activity is a tensor which can learn to detect a specific entity over a domain of limited range of viewing conditions such as viewpoint, rotation and lighting [8]. Two Capsule Networks (CapsNets) are proposed in [22] and [7] which are characterized by the architecture outlined as follows. 1) An input stage including one or more regular convolution layers; 2) a single Primary Capsule Layer consisting of a convolutional stage whose neurons are grouped into capsules; 3) one or more Capsule Layers, with the last one as network output, and consists of a capsule per class. Every pair of capsule layers (this includes the Primary layer) are fully connected by means of a routing stage. Routing allows a CapsNet to learn relationships between entities by directing the output of a capsule to the proper parent capsule located in the next level.

For example, a capsule that learned to recognize eyes, will be routed towards the parent capsule for faces but not to a torso capsule.

CapsNets from [22] and [7] have significant differences in their capsule architecture and routing algorithm. The architecture from [22] (Vector-CapsNet) utilizes 1D vector capsules whose length is an hyperparameter. A capsule encodes an entity and its pose like a CNN, deeper capsules encoding higher level entities. The routing stage fully connects two consecutive capsule layers (L and $L+1$), thus the total input of a capsule (j) in $L+1$ depends on the output of every capsule in L. Dynamic routing between capsules works as follows. The output (u_i) of a capsule is multiplied by a transformation matrix W_{ij} to obtain the prediction vector ($\hat{u}_{i|j}$). If the prediction vector is similar to the output of the parent capsule j, then the routing algorithm concludes that i and j are highly related and assigns a high value to the related coupling coefficient (c_{ij}). As the contribution to the total input of j provided by the capsule i is computed as $\hat{u}_{i|j}c_{ij}$, the coupling coefficient expresses how likely capsule i will activate capsule j. Furthermore, the capability of learning relationship between entities that characterize CapsNets is due to a transformation matrix W for each capsule pair $i \in L$ and $j \in \text{Ł} + 1$.

The capsules of the network proposed in [7] (MatrixEM-CapsNet) consist of a scalar activation (a) and a 4×4 pose matrix (M). As in Vector-CapsNet, capsule layers are fully connected. Thus, each capsule i in a layer L is connected to each capsule j in the next layer $L+1$ by means of a 4×4 transformation matrix (W_{ij}) which is learned with an iterative routing algorithm based on EM (Expectation Maximization) clustering and denoted as EM Routing. The prediction of the parent capsule's pose matrix V_{ij} (vote) is computed as the product between M_i and W_{ij} and utilized along with a_i by a routing algorithm to assign routes between capsule i in layer L and capsule j in layer $L+1$ ($\forall i, j$).

The main difference between CapsNet and CNN is how features are routed between layers. CNN utilizes single neurons for representing image features and pooling operations as routing mechanisms. Pooling ensures invariance to small image changes (translation in particular) at the cost of information loss [17] and makes nearly impossible for a CNN to learn relationship between image entities.

4 Experimental Setup

The proposed approach consists of two types of experiment to assess a network's performance with unseen affine transformations and with prominent class imbalance.

4.1 Capture Bias Experiment

Training data and test data are built from the same dataset by applying affine transformations whose intensity is sampled from different distributions. Hence, a model becomes familiar with several image transformations which appear at different intensities in the training and test datasets. For example, if the considered transformation is rotation, the training set would be augmented by a rotation angle sampled in a range, such as $[-20°, +20°]$, while the transformation magnitude for testing would be sampled from a wider range such as $[-90°, +90°]$.
The performance metric utilized for these experiments is classification accuracy, which is the number of correct predictions from all predictions made. Hence, more general models are those achieving higher accuracy on unseen magnitude of a given affine transformation.

In order to provide more comprehensive insights about the influence of unseen imaging conditions, two different criteria for sampling training data are used: uniform and sparse sampling.

Uniform Sampling Let T be an affine transformation, D_r a training dataset, D_e the relative test dataset, R_r and R_e two magnitude ranges such that $R_r \subset R_e$. A network is trained with D_r whose every sample s, is augmented with $T(s, t_r)$ where t_r is the magnitude uniformly sampled from R_r: $t_r \in R_r$. Our tests consist of running the model along the complete axis of transformation range R_e. Thus, a set of magnitudes are sampled at fixed size steps starting from the lower bound of R_e until the end of the range. For each $t_{e_i} \in R_e$, the complete dataset D_e is transformed with $T(t_{e_i})$ and used to compute a network's accuracy. This process results in a curve showing the relationship between transformation magnitude and a model's accuracy.

Sparse Sampling Let T be an affine transformation, D_r a training dataset, D_e the relative test dataset, R_r and R_e two magnitude ranges such that $R_r \subset R_e$. A subset of n of values are chosen from R_r to form a set $K = \{t_{r_1}, t_{r_2}, \dots t_{r_n}\}$. A network is trained with D_r whose sample s is augmented with $T(s, t_r)$ where t_r is the magnitude uniformly sampled from K: $t_r \in R_r$. Our test procedure is the same as in the Uniform Sampling experiment.

4.2 Imbalanced Data

A model trained with imbalanced classes presents a bias towards the overrepresented ones, which results in more frequent prediction of such majority classes [5]. The performance measure is the Matthews Correlation Coefficient (MCC) for multiple classes [9] as it is proven to be more insensitive to imbalanced data than accuracy [2]. MCC value can fall in $[-1, +1]$, where $+1$ corresponds to a perfect classification. A network is trained with both balanced and imbalanced data and the resulting MCC values are compared. Better models are expected to have a narrower gap between MCC scores of balanced and imbalanced data.

5 Results

Table 1: Models assessed: cnn-wp is a CNN similar to the comparison baseline from [22], vcaps-s, vcaps-d and vcaps-t are Vector-CapsNet with single, double and triple capsule layers respectively, caps-em is a MatrixEM-CapsNet.

Model	Layers
cnn-wp	$\mathrm{C}(5,1,256); \mathrm{P}(3,1); 2 \times [\mathrm{C}(5,1,256); \mathrm{P}(3,2)]; \mathrm{F}(328,192,10)$
vcaps-s	$\mathrm{C}(9,1,256); \mathrm{C}(9,2,256); \mathrm{Pr}(1152,8,3); \mathrm{Cps}(10,16)$
vcaps-d	$\mathrm{C}(9,1,64); \mathrm{C}(9,2,64); \mathrm{Pr}(288,8,3); \mathrm{Cps}(20,10); \mathrm{Cps}(10,16)$
vcaps-t	$\mathrm{C}(9,1,128); \mathrm{C}(9,2,128); \mathrm{Pr}(1152,4,3); 2 \times [\mathrm{Cps}(32,8)]; \mathrm{Cps}(10,16)$
caps-em	$\mathrm{C}(6,2,32)$; A:32; B:24; C:24; D:24; E:10; K:3;

Results are presented for several models as listed in Table 1: cnn-wp is a CNN with three layers and max pooling, vcaps-s, vcaps-d and vcaps-t are Vector-CapsNet with one, two and three layers of capsules respectively and caps-em is a MatrixEM-CapsNet. All the networks are implemented with Tensorflow [1]. In particular, vcaps-s, vcaps-d and vcaps-t are built on top of the source code provided by the authors of Vector-CapsNet [21], while caps-em is derived from the code shared at [28]. The cnn-wp model is implemented from scratch and has similar architecture and hyperparameters as the comparison baseline from [22] used to evaluate Vector-CapsNet on the MNIST dataset [18]. For the notation in Table 1, the following convention is utilized. $\mathrm{C}(k, s, o)$ represents a convolutional layer with kernel k, stride s and o filters; $\mathrm{P}(k, s)$ indicates a max pool layer with kernel k and stride s; $\mathrm{F}(i, h, o)$ is a fully connected network with a single hidden layer of h neurons; $\mathrm{Pr}(c, l, r_i)$ indicates a Primary Capsule Layer having c capsules with length l and utilizing r iterations for the routing algorithm; $\mathrm{Cps}(c, l, r)$ represents a capsule layer and c, l and r have the same meaning as for $\mathrm{Pr}(c, l, r)$. Except for an additional convolutional layer at the start, caps-em has the same architecture as proposed in [7] but uses less capsules per layer. While in [7] the hyperparameters A, B, C, D are all equal to 32, our implementation reduces the complexity of the network by setting B, C and D to 24. This compromise was necessary to run caps-em with at least 2 routing iterations on our 8GB RAM graphics card. The models have been trained with the Adam [13] optimizer with default parameters ($\beta_1 = 0.9$ and $\beta_2 = 0.999$) and with an initial learning rate of 0.001 for Vector-CapsNet and cnn-wp, and 0.0005 for MatrixEM-CapsNet. The loss function to train vcaps-s, vcaps-d and vcaps-t is Margin Loss [22] with parameters $m^+ = 0.9, m^-, \lambda = 0.5$. The Spread Loss [7] has been used for caps-em with margin m increasing from 0.2 up to 0.95 in around 10 epochs. Regularization has been obtained with a reconstruction stage consisting of a neural network with two hidden layers of 512 and 1024 units respectively.

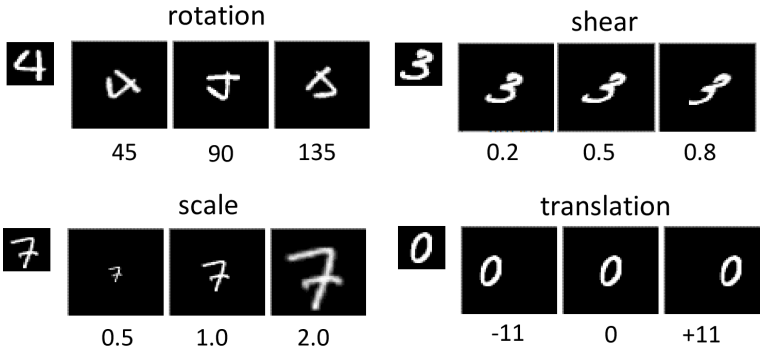

Fig. 1: Several MNIST images as they are transformed and padded for testing a model accuracy.

5.1 Generalization Performance on Unfamiliar Affine Transformations

Generalization performance with uniformly sampled affine transformations (Section 4.1) has been assessed utilizing affMNIST [24] as training data and MNIST [18] for tests. AffMNIST is a dataset obtained from MNIST by applying to each image several uniformly sampled transformations, namely rotation in $[-20°, 20°]$, scale between 0.8 and 1.2, shear along the x axis in $[-0.2, 0.2]$ and translation. As compared to MNIST, which has 28 pixel images, affMNIST has 40 pixel images in order to fit scaled up digits. Accuracy data is obtained for each transformation using the MNIST test set with the following extended ranges: rotation $[-90°, 90°]$, scale factor $[0.5, 2.0]$, horizontal shear $[-0.8, 0.8]$ and horizontal translation (x axis) $[-13, 13]$. As test required wider range of transformations with respect to those available during training, the models have been fed with 56 pixel images obtained by zero-padding affMNIST images. Padding allowed us to test the models with scale factors up to 2.0 and wider translations than those present in affMNIST without any crop to MNIST digits. Figure 1 shows some samples from MNIST as they are transformed and padded for testing a model accuracy.

The results for uniform sampling experiments are shown in Figure 2 where the accuracy as a function of an affine transformation is plotted for each model. The most prominent difference among models occurs with unfamiliar scales where vcaps-t outperforms both

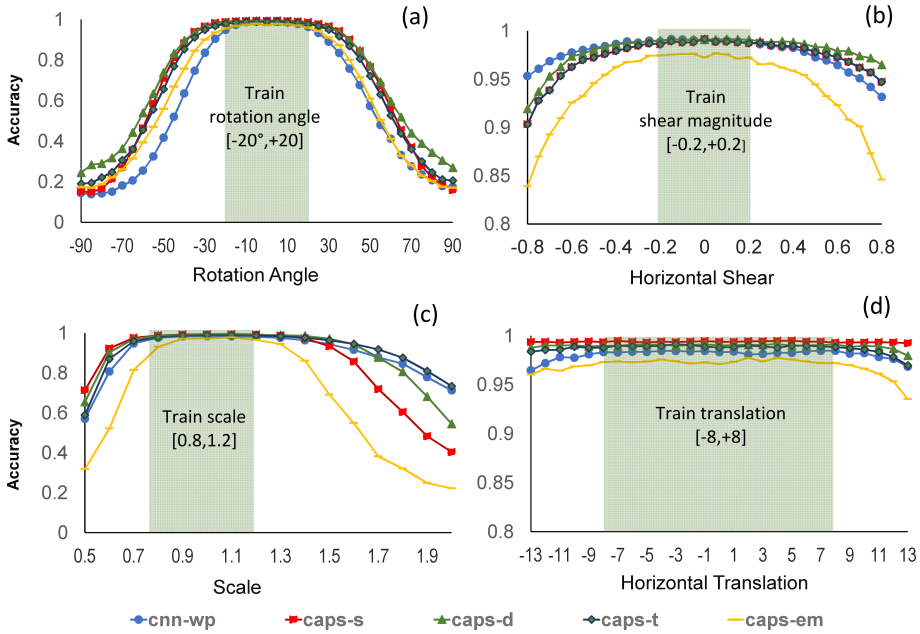

Fig. 2: Accuracy as a function of Rotation Angle (a), shear along the x axis (b), scale factor (c) and horizontal Translation (d). The green area indicates the affine transformation range available in training data (affMNIST).

cnn-wp and the other capsule networks. A closer look at the scale plot (Figure 2.c) allows us to infer a positive relationship between the number of capsule layers in Vector-CapsNets and generalization performance with unfamiliar scale factors. Indeed, vcaps-t achieves better accuracy at each unfamiliar scale than vcaps-s and vcaps-d for scale factors larger than 1.2, which is the largest scale present in affMNIST. On the contrary, for small test scale this trend is inverted and it appears that Vector-CapsNet has the slowest decay in accuracy among the considered models. Also with rotation, CapsNets generalize better than other types of networks, keeping the accuracy above 90% in the interval $[-35°, 35°]$, which is 15° wider than the sample interval for the rotation used to generate affMNIST.

The same four affine transformations have been considered in sparse sampling experiments. Model training is carried out by augmenting MNIST samples with a single transformation a time whose

intensity is sampled in a finite set. Hence, rotation is sampled in $\{-90°, 0, +90°\}$, scale in $\{0.5, 2.0\}$, horizontal shear in $\{-0.5, 0.5\}$ and horizontal translation in $\{-11, 11\}$.

The models do not present significant differences with respect to each other for rotation and horizontal shear (Figure 3). In particular, the networks show a very good generalization performance to unseen shear magnitudes. In fact, just including two values for shear in the training set, yields an almost flat accuracy plot along all shear test range. Generalization performance with sparse shear sampling is coherent with the results obtained with uniform sampling. Indeed, the models' accuracy has a flat trend along the entire test interval $\{-0.8, 0.8\}$. Similarly to the uniform sampling scenario, the scale results show that deeper Vector-CapsNets generalize better than the other models with unfamiliar scale factors.

The results from sparse translation experiments show that cnn-wp and the three considered Vector-CapsNet have a prominent accuracy drop in the middle of the test interval, while caps-em has stable accuracy on the entire test interval. The reason for the performance gap between caps-em and Vector-CapsNet is probably due to the routing algorithm, which is the main difference between these two types of network (Section 3).

Table 2: The models' accuracy with MNIST and affMNIST and the models' MCC with balanced (BAL) and imbalanced (I-BAL) datasets. GAP shows the difference between BAL and I-BAL MCC values.

model	MNIST	affMNIST	BAL	I-BAL	GAP
cnn-wp	0.9923	0.9926	0.9258	0.9021	-0.0237
caps-s	0.9958	0.9999	0.9202	0.8973	-0.0229
caps-d	0.9935	0.9981	0.9336	0.8929	-0.0407
caps-t	0.9933	0.9999	0.9139	0.9004	-0.0135
caps-em	0.9827	0.9961	0.8899	0.7483	-0.1416

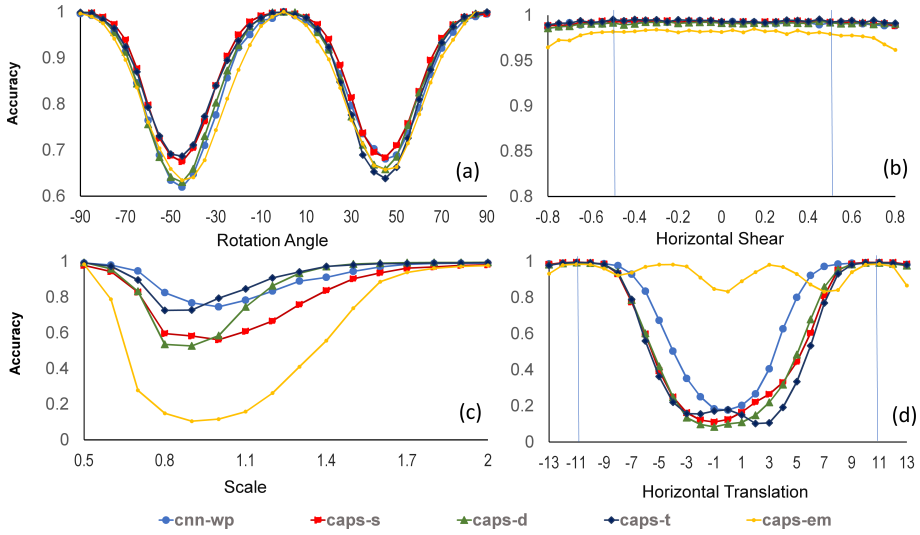

Fig. 3: Effect of sparse sampling of affine transformations in the training data. Accuracy is represented as a function of Rotation Angle (a), shear along the x axis (b), scale factor (c) and horizontal Translation (d).

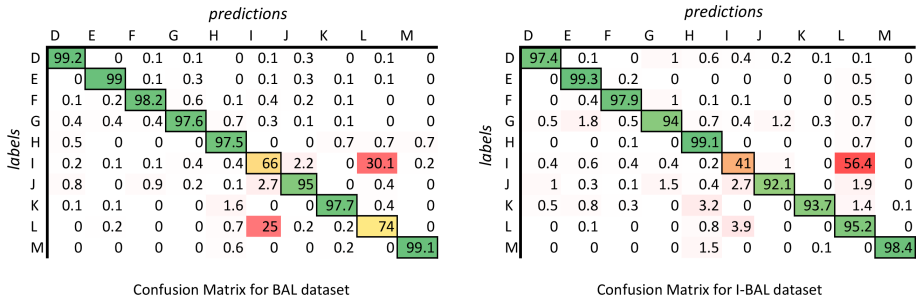

Fig. 4: Confusion matrices of the vcaps-t model for BAL and I-BAL. The over-represented classes E, H and L are more often predicted by the model trained with I-BAL thus, this results in misclassification increase.

5.2 Performance Analysis with Imbalanced Data

The datasets utilized for these experiments have been generated from EMNIST-Letters [4], which consists of 26 balanced classes of handwritten letters with 4800 samples each. The balanced dataset (BAL)

is a subset of EMNIST including 10 of its classes (D to M) with 2400 samples each, while for the imbalanced dataset (I-BAL) classes have been down-sampled to 600 images, except for E, H and L which have the same 4800 samples from EMNIST-Letters. Figure 4 shows the confusion matrices of vcaps-t for BAL and I-BAL. As expected, the three overrepresented classes, E, H and L, are predicted more often. This is particularly evident for classes that are similar to each other such as L and I. Indeed, the similarities between lowercase L letters and uppercase I letters result in several misclassifications even with BAL datasets where I is predicted as L in 30.3% cases and I is called L in 25% cases. In I-BAL, L is overrepresented as compared to I, which is wrongly classified as L more than half of the time (56.4%). MCC for all the models are summarized in Table 2. The least robust model to imbalanced data is caps-em, with a gap between BAL and I-BAL of 0.1416. cnn-wp and vcaps-s have similar results while vcaps-t capture the best performance with a gap of 0.0135, which is about one half of vcaps-s' gap.

The number of capsule layers alone does not explain the better performance of vcaps-t over vcaps-s. Indeed, vcaps-d outperforms the other networks with BAL (MCC of 0.9306) but it also has the widest gap with unbalanced data among Vector-CapsNet: 0.0407. Several double layer architectures were examined other than vcaps-d, but it was neither possible to find a better model nor to precisely determine the factor that influences the performance the most. For example, replacing the two capsule layers of vcaps-d (Table 1) with $2 \times \text{Cps}(128, 4, 3)$ increased the learnable parameters from $5M$ to $8.9M$ however, the performance decreased sightly from 0.938 for BAL to 0.8938 for I-BAL (with a gap of 0.0442) in our experiments.

6 Conclusions

The analysis of capsule networks has received little attention. This paper aimed to provide novel insights into this new type of neural network and proposed several experiments to assess the performance of a network with biased data. Overall, CapsNet outperforms CNNs in most of the cases but not by a large gap. Our results have allowed us to infer that the number of capsule layers (depth) influences generalization performance, this is particularly evident in scale plots

(Figure 2.c) where the accuracy at unseen scales improves with a network depth. Apart from this, the influence of a CapsNet's hyperparameters is not totally understood and would deserve a more detailed and specific analysis. On imbalanced data vcaps-t outperforms all the other networks by a consistent gap but the contribution of the triple capsule layer of vcaps-d remains unclear, which is affected by imbalance data more than vcaps-s. Finally, the worst model in any scenario is caps-em with the exception of sparse translation (Figure 3). However, it is worth mentioning that the caps-em implementation it not from its authors and includes less capsules than the model originally proposed in [7]. Indeed, our Tensorflow implementation is very demanding in terms of RAM and caps-em is the most complex model that can fit in an 8GB Graphics card. A natural extension of this work would include MatrixEM-CapsNet once an official implementation is available. Furthermore, new insights would be provided from a more specific analysis of the relationship between hyperparameters and generalization performance such as the depth and the distribution of capsules among a CapsNet's layers.

Acknowledgment

This work has been supported by the UK Engineering and Physical Sciences Research Council EPSRC [EP/K004638/1, EP/R02572X/1 and EP/P017487/1]

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Akosa, J.: Predictive accuracy: A misleading performance measure for highly imbalanced data. In: Proceedings of the SAS Global Forum (2017)
3. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: CNN architecture for weakly supervised place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5297–5307 (2016)

4. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373 (2017)
5. Glauner, P., Valtchev, P., State, R.: Impact of biases in big data. arXiv preprint arXiv:1803.00897 (2018)
6. Gritsevskiy, A., Korablyov, M.: Capsule Networks for low-data transfer learning. arXiv preprint arXiv:1804.10172 (2018)
7. Hinton, G., Frosst, N., Sabour, S.: Matrix capsules with em routing (2018)
8. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: International Conference on Artificial Neural Networks. pp. 44–51. Springer (2011)
9. Jurman, G., Furlanello, C.: A unifying view for performance measures in multi-class prediction. arXiv preprint arXiv:1008.2908 (2010)
10. Kalliatakis, G., Stamatiadis, G., Ehsan, S., Leonardis, A., Gall, J., Sticlaru, A., McDonald-Maier, K.D.: Evaluating deep Convolutional Neural Networks for material classification. arXiv preprint arXiv:1703.04101 (2017)
11. Karianakis, N., Dong, J., Soatto, S.: An empirical evaluation of current convolutional architectures' ability to manage nuisance location and scale variability. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4442–4451 (2016)
12. Khosla, A., Zhou, T., Malisiewicz, T., Efros, A.A., Torralba, A.: Undoing the damage of dataset bias. In: European Conference on Computer Vision. pp. 158–171. Springer (2012)
13. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Kortylewski, A., Egger, B., Schneider, A., Gerig, T., Morel-Forster, A., Vetter, T.: Empirically analyzing the effect of dataset biases on deep face recognition systems. Preprint (2017)
15. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
16. 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)
17. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
18. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
19. Masko, D., Hensman, P.: The impact of imbalanced training data for Convolutional Neural Networks (2015)
20. Nair, P., Doshi, R., Keselj, S.: Pushing the limits of Capsule Networks. Technical note (2018)
21. Sabour, S.: Dynamic routing between capsules, source code (2017), `https://github.com/Sarasra/models/tree/master/research/capsules`, last accessed: 05.02.2019
22. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems. pp. 3859–3869 (2017)
23. Savinov, N., Seki, A., Ladicky, L., Sattler, T., Pollefeys, M.: Quad-networks: unsupervised learning to rank for interest point detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
24. Tieleman, T.: affMNIST (2013), `https://www.cs.toronto.edu/~tijmen/`, last accessed: 05.02.2019
25. Tommasi, T., Patricia, N., Caputo, B., Tuytelaars, T.: A deeper look at dataset bias. In: Domain Adaptation in Computer Vision Applications, pp. 37–55. Springer (2017)

26. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1521–1528. IEEE (2011)
27. Xi, E., Bing, S., Jin, Y.: Capsule Network performance on complex data. arXiv preprint arXiv:1712.03480 (2017)
28. Zhang, S.: Matrix-capsules-em-tensorflow, source code (2018), `https://github.com/www0wwwjs1/Matrix-Capsules-EM-Tensorflow`, last accessed: 05.02.2019