Limitations of routing-by-agreement based capsule networks

David Peer  
University of Innsbruck  
d.peer@uibk.ac.at

Sebastian Stabinger  
University of Innsbruck  
sebastian@stabinger.name

Antonio Rodriguez-Sanchez  
University of Innsbruck  
antonio.rodriguez-sanchez@uibk.ac.at

Abstract

Classical neural networks add a bias term to the sum of all weighted inputs. For capsule networks, the routing-by-agreement algorithm, which is commonly used to route vectors from lower level capsules to upper level capsules, calculates activations without a bias term. In this paper we show that such a term is also necessary for routing-by-agreement. We will proof that for every input there exists a symmetric input that cannot be distinguished correctly by capsules without a bias term. We show that this limitation impacts the training of deeper capsule networks negatively and that adding a bias term allows for the training of deeper capsule networks. An alternative to a bias is also presented in this paper. This novel method does not introduce additional parameters and is directly encoded in the activation vector of capsules.

1 Introduction

Capsules in capsule networks are groups of neurons that represent an object or a part of an object in a parse tree. The output of a capsule is a so called activation or instantiation vector representing properties of this object. The length of a vector encodes the probability whether an object is present or not. Hinton et al. [2011] describes the idea of such instantiation vectors for which, even if the viewing condition of an object changes, the length of the vector stays the same and only the orientation of the vector changes.

Sabour et al. [2017] introduced the routing-by-agreement algorithm to route instantiation vectors from lower level capsules to upper level capsules. The CapsNet architecture demonstrates the effectiveness of this algorithm. In this architecture the output of a convolutional layer with multiple filters is vectorized to create the input to the first capsule layer. The routing-by-agreement algorithm then routes the vectors from this first capsule layer to the output capsules.

We found that for every input to a capsule layer there exists a symmetric input that can not be distinguished by capsules, even when that symmetric input represents a different class. This symmetric input can be created by simply inverting all input vectors. For shallow network architectures such as the CapsNet mentioned above this is not a problem because a ReLu function is applied to all components of the input vectors ensuring no component is negative. Therefore, the inverse of a vector will never be an input to a capsule. For deeper capsule networks the components of vectors can be positive as well as negative. Thus, the inverse of an input can also occur. An example of a component that is represented using positive and negative values is shown in Figure 1 for different values from $-0.17$ to $+0.17$ in the digit capsule layer. As we can see, $-0.17$ and $+0.17$ represent different instances of the digit $8$. This seems reasonable to us because it increases the degree of freedom of
higher level capsules and therefore decreases the reconstruction error. Sabour et al. (2017) states that "[...] this shift from place-coding to rate-coding combined with the fact that higher-level capsules represent more complex entities with more degrees of freedom suggests that the dimensionality of capsules should increase as we ascend the hierarchy". Therefore we claim that it is a problem to train deeper capsule networks because they cannot distinguish inputs and the inverse of inputs correctly although such inputs can occur. The validity of this claim will be evaluated in this paper.

Our hypothesis is that a capsule network cannot distinguish inputs and the inverse of those inputs, which we will prove in this paper. Using this insight, we are able to alleviate this limitation by changing the original algorithm. We compare two approaches: First, we add a bias parameter to the preactivation. As an alternative approach, we represent instantiation vectors using homogeneous coordinates. We show for both approaches that the limitation disappears in theory and show for multiple datasets and for different architectures that this carries over to practical applications.

Neural networks usually add a bias term to the sum of all weighted inputs. Bias terms are not mentioned by Sabour et al. (2017) for the implementation of CapsNet, presumably because adding a bias is common practice and the authors did not deem it necessary to mention the use explicitly. We evaluated different implementations of the original paper to check if a bias is used. As can be seen in appendix A, 76% of the implementations follow the pseudo code of Sabour et al. (2017) directly and do not add a bias. This leads us to the conclusion that the use of a bias term is currently not generally considered an important part of a capsule network. In addition, some authors, such as Kronenberger and Haselhoff (2018), explicitly state that for capsule networks no bias is used and that "[...] this design of the capsule allows more capabilities in representing its features". We agree that a bias term is not explicitly mentioned in the paper by Sabour et al. (2017), but proof that this still limits the representation of the capsules. Others such as Pechyonkin (2017) or Eldor (2018) state that "[...] there is no bias because it is already included in the W matrix that can accommodate it and other, more complex transforms and relationships". We do not agree with this statement and we will show that a bias or a homogenous representation of vectors is needed by capsule networks.

The paper is structured as follows: In the next section we summarize related work. In section 3 we proof that a capsule network can not distinguish inputs and the inverse of those inputs. Section 4 introduces two methods to avoid the limitation of symmetric inputs. In section 5 we show experimentally that this limitation negatively impacts the training of deeper capsule networks and we show that the proposed methods alleviate this problem. In section 6 we conclude our findings.

2 Related Work

Capsules are introduced by Hinton et al. (2011). The author also showed that such capsules can be trained by backpropagating the difference between the actual and the target outputs. Building of this work, Sabour et al. (2017) introduced an algorithm called routing-by-agreement (algorithm 1) to route output vectors from one capsule layer to another. To demonstrate this iterative routing the authors introduced an architecture called CapsNet which consists of multiple convolutional layers, followed by a PrimaryCaps and a DigitCaps layer. This network achieved state of the art performance on the MNIST and CIFAR10 dataset.

In a follow up work, Hinton et al. (2018) introduced the Expectation-Maximization-Routing algorithm. Despite this new algorithm, the routing-by-agreement algorithm is still popular because of its ease of implementation in common deep learning frameworks. In addition, the algorithm can easily be adapted to new tasks as shown by Mobiny and Nguyen (2018) who use the routing-by-agreement algorithm for lung cancer screening. Duarte et al. (2018) detects action in movies using the same
We refer to [Sabour et al. (2017)] with capsules $i$ in layer $l$ and $j$ in layer $l+1$ with $r$ routing iterations and predictions $\hat{u}_{j|i}$.

Algorithm 1 Routing-by-agreement algorithm as presented by [Sabour et al. (2017)].

$\forall$ capsules $i$ in layer $l$ and $j$ in layer $l+1$ with $r$ routing iterations and predictions $\hat{u}_{j|i}$

1: procedure ROUTINGBYAGREEMENT($\hat{u}_{j|i}, r, l$)
2: \forall b_{ij}, b_{ij} \leftarrow 0
3: \textbf{for} $r$ iterations $\textbf{do}$
4: $c_{ij} \leftarrow \exp(b_{ij})$
5: $s_j \leftarrow \sum_i c_{ij} \hat{u}_{j|i}$
6: $v_j \leftarrow \frac{||s_j||^2}{1+||s_j||^2} \cdot s_j$
7: $b_{ij} \leftarrow b_{ij} + v_j \cdot \hat{u}_{j|i}$
8: \textbf{end for}
9: \textbf{end procedure}

This list of new applications also shows that the routing-by-agreement algorithm is still the most widely used routing algorithm for capsule networks. To the best of our knowledge our work is the first to prove the limitation of capsule networks using the routing-by-agreement algorithm.

### 3 Limitations of capsule networks

In this section we prove that a capsule network, using routing-by-agreement (see algorithm 1), cannot distinguish inputs $u_1, u_2, ..., u_J$ and their symmetric or inverse counterpart $-u_1, -u_2, ..., -u_J$.

#### 3.1 Notation

A lower level layer produces $I$ output vectors $u_1, u_2, ..., u_I$ which are the input of the current layer $l$ with $J$ capsules. We call $-u_1, -u_2, ..., -u_J$ the inverse input. The input to the routing-by-agreement algorithm are predictions $\hat{u}_{j|i}$ from the lower level capsule $i$ to the upper level capsule $j$, which are calculated by $\hat{u}_{j|i} = W_{ij} u_i$, where $W_{ij}$ is the transformation matrix.

The output of algorithm 1 are activation vectors $v_j$ which are calculated from the preactivations $s_j$. The preactivation is a vector with an arbitrary length, which does not guarantee that $0 \leq ||s_j|| \leq 1$. Therefore, the nonlinear squash function $(v_j = \frac{||s_j||^2}{1+||s_j||^2} \cdot s_j)$ ensures that $||v_j||$ represents a probability.

The algorithm uses the coupling coefficients $c_{ij}$ from lower level capsules $i$ to upper level capsules $j$ to calculate the preactivations. The coupling coefficients are calculated from the logits $b_{ij}$, which are updated iteratively in the routing algorithm to couple with the most appropriate parent capsule as the routing proceeds.

We refer to $v_j^+$ as the activation vector, calculated by the routing algorithm, using the inputs $u_i$, and to $v_j^-$ as the inverse activation vector, calculated with the inverse inputs $-u_i$. To indicate the activations calculated in iteration $r$ of the iterative routing algorithm we write $v_j^{+r}$ or $v_j^{-r}$ respectively. For example, the activations calculated in the first routing iteration for inverse inputs are referenced with $v_j^{-1}$. The routing algorithm starts from iteration 1 and not from iteration 0 such that e.g. $u_{j|i}^{0+}$ shows the initialization of the activation vectors. We use the same notational convention to refer to the preactivations $s_j$, the logits $b_{ij}$, the routing coefficients $c_{ij}$ and the predictions $\hat{u}_{j|i}$ during specific iterations.

#### 3.2 Proof

**Lemma 1.** The prediction $\hat{u}_{j|i}^+$ is inverted ($\hat{u}_{j|i}^- = -\hat{u}_{j|i}^+$) when the input $u_i$ is inverted.
Proof.

\[ \hat{u}_{ji}^- = W_{ij}(-u_i) = -W_{ij}u_i = -\hat{u}_{ji}^+ \]

\[ \square \]

**Lemma 2.** Assume that \( c_{ij}^+ = c_{ij}^- \), then the activation \( v_j^+ \) of a capsule \( j \) can be inverted \( (v_j^- = -v_j^+) \) by inverting all inputs \( u_1, u_2, ... u_I \).

**Proof.** For the preactivation \( s_j^- = \sum_i c_{ij}^- \hat{u}_{ji}^- \), and the inverse inputs \( u_i \):

\[
\begin{align*}
\hat{s}_j^- &= \sum_i c_{ij}^- \hat{u}_{ji}^- \\
&= \sum_i c_{ij} (-\hat{u}_{ji}^+) \quad \text{[Lemma 1]} \\
&= \sum_i c_{ij}^+ (-\hat{u}_{ji}^+) \quad \text{Assumption} \\
&= -\sum_i c_{ij}^+ \hat{u}_{ji}^+ = -s_j^+ \quad \text{Definition of } s_j^+
\end{align*}
\]

and therefore

\[
\begin{align*}
v_j^- &= \frac{||s_j^-||^2}{1 + ||s_j^-||^2} s_j^- \\
&= \frac{|| - s_j^+ ||^2}{1 + || - s_j^+ ||^2} -s_j^+ \\
&= -\frac{||s_j^+||^2}{1 + ||s_j^+||^2} \frac{s_j^+}{||s_j^+||} = -v_j^+ \\
&\quad \text{Definition of } v_j^+
\end{align*}
\]

\[ \square \]

We have seen that the activation of a capsule can be inverted by inverting all inputs under the assumption that \( c_{ij}^+ = c_{ij}^- \). We will now show that the property \( c_{ij}^+ = c_{ij}^- \) holds in every iteration of **algorithm 1**.

**Lemma 3.** The routing-by-agreement algorithm produces coupling coefficients \( c_{ij}^{r+} = c_{ij}^{r-} \) in every routing iteration \( r \in \mathbb{N} \) for \( r \geq 1 \).

**Proof.** First we will show that \( b_{ij}^{k+} = b_{ij}^{k-} \) for any routing iteration \( k \), using a proof by induction on the routing iterations.

**Base step:** For routing iteration \( 1 \) and the initialization of \( b_{ij}^{0+} = b_{ij}^{0-} = 0 \) as shown in **line 2** we can show that

\[ c_{ij}^{1+} = \frac{0}{\sum_i \exp(0)} = c_{ij}^{1-} \]

Therefore, **Lemma 2** is applicable which implies that

\[
\begin{align*}
b_{ij}^{1-} &= 0 + (v_j^{1-} \cdot \hat{u}_{ji}^{1-}) \quad \text{Definition of } b_{ij}^{1-} \\
&= 0 + (v_j^{1-} \cdot -\hat{u}_{ji}^{1+}) \quad \text{Lemma 1} \\
&= 0 + (-v_j^{1+} \cdot -\hat{u}_{ji}^{1+}) \quad \text{Lemma 2} \\
&= 0 + (v_j^{1+} \cdot \hat{u}_{ji}^{1+}) = b_{ij}^{1+}
\end{align*}
\]

**Inductive hypothesis:** \( b_{ij}^{(k-1)+} = b_{ij}^{(k-1)-} \)
Inductive step: In iteration \( k \), Lemma 2 is applicable because by using the inductive hypothesis we can show that
\[
c^{k-}_{ij} = \frac{\exp(b^{(k-1)-}_{ij})}{\sum_i \exp(b^{(k-1)-}_{il})} = \frac{\exp(b^{(k-1)+}_{ij})}{\sum_i \exp(b^{(k-1)+}_{il})} = c^{k+}_{ij}
\]
Therefore we can prove that
\[
b^{k-}_{ij} = b^{(k-1)-}_{ij} + \left( v^{k-}_{j} \cdot \tilde{u}^{-}_{ji} \right) \quad \text{Definition of } b^{k-}_{ij}
\]
\[
b^{(k-1)+}_{ij} + \left( v^{k+}_{j} \cdot \tilde{u}^{+}_{ji} \right) = b^{k+}_{ij} \quad \text{Lemma 2 and inductive hypothesis}
\]
This proof by induction shows that in every iteration \( k \) the property \( b^{k+}_{ij} = b^{k-}_{ij} \) holds. Therefore, for the coupling coefficients in the next iteration
\[
c^{(k+1)-}_{ij} = \frac{\exp(b^{(k+1)-}_{ij})}{\sum_i \exp(b^{(k+1)-}_{il})} = \frac{\exp(b^{(k+1)+}_{ij})}{\sum_i \exp(b^{(k+1)+}_{il})} = c^{(k+1)+}_{ij}
\]

Lemma 4. For arbitrary inputs the output of a capsule layer is inverted whenever all inputs are inverted.

Proof. Lemma 2 shows that the activation vector can be inverted by inverting all inputs under the assumption that \( c^{k-}_{ij} = c^{k+}_{ij} \). Lemma 3 shows that \( c^{k-}_{ij} = c^{k+}_{ij} \) at any iteration \( k \). Therefore Lemma 2 is always applicable and outputs are inverted whenever all inputs are inverted.

Theorem 1. A capsule network cannot distinguish inputs and their inverse.

Proof. By recursively applying Lemma 4 at every layer we see that an inverted input produces an inverted output at the last layer such that \( v^{-}_{j} = -v^{+}_{j} \). The classification of an input is based on the length of the output vector. But \( ||v^{-}_{j}|| = ||v^{+}_{j}|| \) holds and therefore a capsule network cannot distinguish the input and the inverse input.

4 Solving the routing-by-agreement limitation

In this section we show two different methods to avoid the previously proven limitation. The first method, using a bias, is generally used in neural networks. The second method, using homogeneous representations, are specially designed for capsule networks.

Bias. The first method targets Lemma 2 so that activation vectors can not be inverted while preserving their lengths. A different length ensures that the classification is different.

To accomplish this, we introduce bias parameters to the calculation of the preactivation with \( s_{j} = \left( \sum_i c_{ij} \tilde{u}^{-}_{ji} \right) + b_{j} \). So for \( b_{j} \neq 0 \).
Figure 2: Network architecture(s) used in this paper. Hidden capsule layers (green) are added to produce networks from 2 up to 6 capsule layers. A network without hidden layers is the CapsNet architecture from Sabour et al. (2017).

\[
\begin{align*}
    s_j^- &= \left( \sum_i c_{ij} u_{ij}^- \right) + b_j & \text{New definition of } s_j \\
    &= \left( \sum_i c_{ij} \left( -\hat{u}_{ij}^+ \right) \right) + b_j & \text{Lemma 1} \\
    &= - \left( \sum_i c_{ij} \hat{u}_{ij}^+ \right) + b_j & \text{Assumption of lemma 2} \\
    &\neq - \left( \sum_i c_{ij} \hat{u}_{ij}^+ \right) + b_j = -s_j^+ & \text{New definition of } s_j
\end{align*}
\]

This enables the network to learn non zero bias parameters such that the calculated preactivation vector is different for inputs and their inverse inputs. This leads to the activation vectors and the length of the activation vectors being different. Therefore, the network is able to learn a bias so that inputs and symmetric inputs can be distinguished.

**Homogeneous representation of instantiation vectors** This method is inspired by homogeneous coordinates and therefore we call this representation of vectors a homogeneous representation. In this representation we set the last dimension of all intermediate activation vectors to 1. This method ensures that lemma 1 does not hold because it is not possible to produce an inverse prediction \( \hat{u}_{ij}^- \) from \( \hat{u}_{ij}^+ \) when the last parameter is fixed to 1. Note also that due to this additional constant component, the representation of the probability of intermediate activation vectors shifts from \( 0 \leq ||v_j|| \leq 1 \) to \( 1 \leq ||v_j|| \leq 2 \).

5 Experimental evaluation

In this section we evaluate the impact of the limitation proved in section 3. We compare the original method with the methods that avoid this limitation for different networks and different datasets. We used the MNIST dataset by LeCun and Cortes (2010), fashionMNIST by Xiao et al. (2017) and smallNorb by LeCun et al. (2004).

Our implementation is based on the original implementation of Sabour et al. (2017) which is available on GitHub. We uploaded our implementation and a link to the datasets to GitHub.

1Sabour S. (2018, September 11). *Original GitHub Repository of CapsNet TensorFlow implementation.* From https://github.com/Sarasra/models/tree/984fbc754943c849c55a57923f4223099a1ff88

2Original link removed for review. The source code and a link to the dataset is added to the supplementary material (see file README.md)
Table 1: Test accuracy for different capsule architectures trained on different datasets. RBA is the routing algorithm proposed by Sabour et al. (2017). RBA + bias uses additional bias weights to calculate the preactivation and Homogeneous RBA uses a homogeneous representation of activation vectors.

| CapsLayer | RBA MNIST | RBA MNIST | RBA + bias MNIST | RBA + bias MNIST | Homogeneous RBA MNIST | Homogeneous RBA MNIST |
|-----------|-----------|-----------|------------------|------------------|-----------------------|-----------------------|
| 2         | 99.2      | 90.8      | 93.8             | 99.2             | 90.8                  | 94.1                  |
| 3         | 98.9      | 91.1      | 92.3             | 99.2             | 91.5                  | 93.8                  |
| 4         | 98.6      | 91.2      | 94.8             | 99.0             | 91.5                  | 91.5                  |
| 5         | 09.8      | 10.0      | 16.9             | 98.3             | 90.9                  | 95.8                  |
| 6         | 09.8      | 10.0      | 20.0             | 98.8             | 91.1                  | 89.5                  |

are based on the CapsNet model. We add additional hidden capsule layers as shown in figure 2 to produce deeper capsule networks. During training no data augmentation is performed. All experiments were done using TensorFlow 1.12.0 by Abadi et al. (2015) on a workstation with a single Nvidia-Titan XP GPU. We used the following hyperparameters for training: A batch size of 128, the Adam optimizer with a decay rate of 0.96, a learning rate of 0.001 and 3 routing iterations as proposed by Sabour et al. (2017). Every model is trained for 20k steps before it is evaluated on the test set of the respective dataset.

Results for the original algorithm, the method with bias and the homogenous representation of the vectors can be seen in table 1. We trained architectures from 2 up to 6 capsule layers for the MNIST, fashionMNIST and smallNorb dataset.

We could gain the following insights from the experiments:

1. For up to 4 layers all methods are successfull.
2. The accuracy of RBA for more than 4 capsule layers is close to random. Note that MNIST and fashionMNIST has 10 different classes whereas smallNorb has only 5 classes.
3. In all cases, RBA + bias is able to successfully classify the given task.
4. In all cases, homogeneous RBA is also able to successfully classify the given task although no additional parameters are introduced.

This experiment supports our claim that the limitation that we proved in section 3 influences the training of deeper capsule networks negatively and that fixing this limitation allows a successful training.

6 Conclusion

The routing-by-agreement algorithm does not include a bias parameter. The majority of the community follows this approach and does not use a bias parameter in their implementations. We found that without such a term the representation of activation vectors is limited and hypothesized that this becomes a problem for deeper capsule networks.

We proved this limitation theoretically for the commonly used routing-by-agreement algorithm. Following this theoretical proof we showed that a bias term avoids this limitation and introduced a different representation that also avoids this limitation without the introduction of new parameters.

In the experimental section we have seen that this limitation influences the training of deeper capsule networks negatively. The experiments support our claim that the theoretical limitation impacts the training of deeper capsule networks in practice. We therefore suggest to add a bias term to preactivations or to use a homogeneous representation of activation vectors as this enables future developments of deeper capsule network architectures.
A Evaluation of capsule network implementations

To evaluate whether a bias parameter is generally used in implementations of capsule networks, we evaluated the code of 17 different frameworks using TensorFlow, Keras, CNTK, and PyTorch and different languages such as Python or R. Not a single implementation uses a homogenous representation as presented in this paper and therefore we left this out in Table 2.

As we can see in Table 2, 76% of the implementations directly follow the pseudo code of Sabour et al. (2017) without adding a bias parameter to the preactivation and only 24% use a bias term.

| Reference            | Implementation                                                      | Bias |
|----------------------|---------------------------------------------------------------------|------|
| Sabour et al. (2017) | https://github.com/Sarasra/models                                  | Yes  |
|                      | https://github.com/sookkek/dynamic_routing_between_capsules        | No   |
|                      | https://github.com/southworkscom/CapsNet-CNTK                      | No   |
|                      | https://github/xFengGuo/CapsNet-Keras                              | No   |
|                      | https://github/yechengli/LightCapsNet                               | No   |
|                      | https://github/gram-ai/capsule-networks                            | No   |
|                      | https://github.com/naturomics/CapsNet-Tensorflow                    | Yes  |
|                      | https://github.com/bourdakos1/capsule-networks                      | No   |
|                      | https://github.com/JunYeopLee/capsule-networks/                    | Yes  |
| Zhao et al. (2018)   | https://github.com/andyweizhao/capsule_text_classification          | No   |
|                      | https://github.com/BoonHuan-Xin/capsnet-mmnet                      | No   |
|                      | https://github/4flabel/capsnet                                     | No   |
|                      | https://github.com/jeasi817/adv_attack_capsnet                      | No   |
|                      | https://www.kaggle.com/fizzbuzz/beginner-a-guide-to-capsule-networks| No   |
| Rajasegaran et al. (2019) | https://github.com/brjahu/deepcaps/                             | Yes  |
| Nguyen et al. (2019) | https://github.com/daiquocnguyen/CapsE                              | No   |
| Xinyi and Chen (2019) | https://github.com/XinyiZ001/CapsGNN                                | No   |

Note: Some implementations call the prior probability \( b_{ij} \) a bias in their implementation.

References

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., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Duarte, K., Rawat, Y. S., and Shah, M. (2018). Videocapsulenet: A simplified network for action detection. abs/1805.08162.

Eldor, T. (2018). Capsule neural networks. http://web.archive.org/web/20190521155956/https://towardsdatascience.com/capsule-neural-networks-part-2-what-is-a-capsule-84665418929f. Accessed: 2019-05-21.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In ICANN.

Hinton, G. E., Sabour, S., and Frosst, N. (2018). Matrix capsules with em routing. In ICLR.

Kronenberger, J. and Haselhoff, A. (2018). Do capsule networks solve the problem of rotation invariance for traffic sign classification? In ICANN (3), volume 11141, pages 33–40. Springer.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In CVPR (2), pages 97–104.

Mobiny, A. and Nguyen, H. V. (2018). Fast capsnet for lung cancer screening. In Frangi, A. F., Schnabel, J. A., Davatzikos, C., Alberola-López, C., and Fichtinger, G., editors, MICCAI (2), volume 11071 of Lecture Notes in Computer Science, pages 741–749. Springer.
Nguyen, D. Q., Vu, T., Nguyen, T. D., Nguyen, D. Q., and Phung, D. (2019). A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), page to appear.

Pechyonkin, M. (2017). Understanding hinton’s capsule networks. http://web.archive.org/web/20190521160127/https://pechyonkin.me/capsules-2/ Accessed: 2019-05-21.

Pöpperl, M., Gulagundi, R., Yogamani, S., and Milz, S. (2019). Capsule neural network based height classification using low-cost automotive ultrasonic sensors. CoRR, abs/1902.09839.

Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S., and Rodrigo, R. (2019). Deepcaps: Going deeper with capsule networks.

Rawlinson, D., Ahmed, A., and Kowadlo, G. (2018). Sparse unsupervised capsules generalize better. abs/1804.06094.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, NIPS, pages 3859–3869.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xinyi, Z. and Chen, L. (2019). Capsule graph neural network. In International Conference on Learning Representations.

Zhao, W., Ye, J., Yang, M., Lei, Z., Zhang, S., and Zhao, Z. (2018). Investigating capsule networks with dynamic routing for text classification. EMNLP-18, abs/1804.00538.