Towards a Deep Capsule Network

Tianhao Hu*
Institute for Aerospace Studies
University of Toronto
tianhao.hu@mail.utoronto.ca

Abstract

The convolutional neural network is a breakthrough in computer vision, yet it suffers problems of poor translational invariance and lack of information about the detected object's pose. The newly proposed capsule network by Geoffrey Hinton et al. appears to be a remedy to these problems. This one-layer capsule network has the state-of-art accuracy on classifying MNIST dataset, but its capability on more complex images still have room for improvement. In this paper, we use German traffic sign dataset to explore the characteristics of the capsule network. Based the identified problems, we propose a multi-layered capsule network and use MNIST and CIFAR10 dataset in the original paper to benchmark its ability. The experiment shows that the deep capsule network has better ability capturing hierarchical features and has better performance on complex images.

1 Introduction

Object classification has been one of the major challenges in computer vision. The main reason is that the same category of objects can have vastly different appearances. The variation of lighting, background environment and perspective projection make the issue more complicated. Thus, the traditional manually crafted object detectors only have limited success for practical use.

In 2012, the deep convolutional neural network (CNN) was proven to be a very effective algorithm for object classification[Krizhevsky et al., 2012]. One major difference is that CNN can automatically learn the features of the target objects instead of relying on manually engineered features such as the histogram of oriented gradients (HOG)["Dalal and Triggs, 2005]. Although the CNN bases algorithms now can rival human performance on common visual object recognition benchmark task[Russakovsky et al., 2014], it is not without flaws. To ensure CNN to focus on the most important features, max-pooling (sub-sampling) layers are inserted between convolution layers so that only the most dominant features are passed downwards to the deeper layers of the network[Krizhevsky et al., 2012]. The problem is, after several stages of sub-sampling, high-level features have a lot of uncertainty in their poses[Hinton et al., 2011]. This shortcoming makes CNN ineffective in classifying objects in cluttered scenes that is very common in real-world images.

One solution to this problem is to use capsule networks featured with dynamic routing [Sabour et al., 2017]. This new approach can learn the properties of the objects on the image and output them as explicit instantiation parameters. The properties can be color, shape, size, position, orientation, etc. Theoretically speaking, capsule network can resolve the pose ambiguity issue of CNN.

2 Motivation

One of the most important practical applications of computer vision is the perception system of the self-driving car. In order to determine the control commands, the onboard system of the vehicle must

*Student No. 995903063

Code available for download at https://github.com/tianhao-david-hu/deep_capsule_network/

be fully aware of its surroundings, such as the traffic light, traffic signs, vehicles, pedestrians, and drivable road. The leading autonomous vehicle manufacturer, Tesla, extensively use deep CNN for perception[Pirzada", 2015]. Nevertheless, autonomous vehicles still need the supervision of a human driver because CNN is not fully robust and can fail under unexpected circumstances[Lohr", 2016].

As the capsule network has achieved the state-of-art performance on MNIST, the intention of this research is to test how capsule network compares with the existing CNN for self-driving car application and explore any possible improvement. A more robust vision system can definitely save lives.

3 Experiment Design

The capsule network is a rather new architecture. It is only made to the public a month ago at the time this paper is written, and there is no capsule layer implementation in TensorFlow yet. To conduct the experiment, The first step is to program the capsule network and verify it is functional. The capsule network implementation of this paper is based on the work of Geron" [2017].

The implemented the code is then verified with MNIST dataset using the same setting as described in the original paper. Once the implementation is tested, we can then train the capsule network with other datasets.

The dataset of choice is the German Traffic Sign Benchmark[Houben et al., 2013]. This dataset is similar to MNIST, yet it is more difficult to classify. Like MNIST, the traffic signs are the 2D symbol, so that we can do a side-by-side comparison. However, this dataset is more challenging because a) The traffic sign plate can have perspective distortion. b) The background varies and is not solid white anymore. c) The images are colored, and the lighting condition can cause the image color to change drastically. d) The traffic sign suffers motion blur to the vehicle movement. e) The training set is highly unbalanced. f) The traffic sign data set has 43 categorise—much larger than MNIST. However, the traffic sign is not as hard as 3D objects whose appearance can be completely different if looking from a changed angle. All these characteristics of the traffic sign images make it a good intermediate-level experimental dataset for the new capsule network.

Upon the completion of the experiment with the traffic sign data set, the outcome is analyzed, and the attempt is made to fix the identified problem. Herein, we propose a deep capsule network that has more than one layer of the capsule. This new network is benchmarked with MNIST and CAFAR10 datasets as the original paper does.

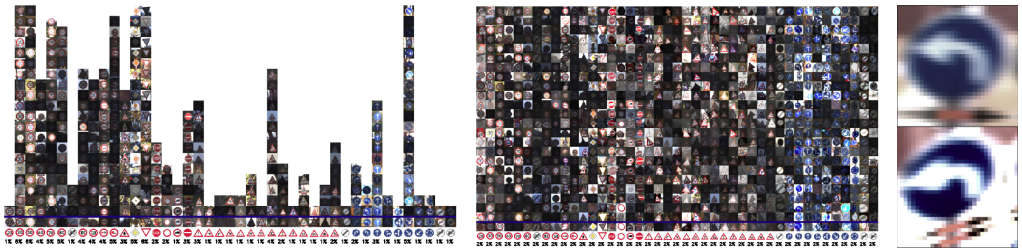
The experimental platform is a PC equipped with an i7-920 processor and a nVidia GTX 1050 Ti video card.

4 Classifying German Traffic Sign Using Standard Capsule Network

In this section, the performance of the capsule network on traffic sign classification will be presented and discussed. The baseline capsule network implementation byGeron" [2017] using TensorFlow is tested to have an accuracy of 99.43% with MNIST if trained with early stopping. Although it is not as good as the accuracy stated on the original paper, it is reasonably close. In the original paper, not all details of dynamic routing are fully disclosed, so it is not easy to reproduce the same result. The most important metric to tell if the implementation is correct is whether the reconstructed images are similar to the figures presented in the original paper. The implementation of this experimentation does meet this criterion.

4.1 German Traffic Sign Dataset

The German traffic sign dataset has images cropped from video streams normalized to a size of 32pixel $\times$ 32pixel $\times$ 3channel. The raw dataset has 34799 training samples, 4410 validation samples, and 12630 testing samples. There are 43 categories of traffic signs, but the number of samples for each category is highly unbalanced. As shown in Figure 1, the data augmentation technique is applied to bring the training sample distribution to the balance. Extra training samples are generated by applying the affine transformation and color manipulation. The augmented training data has 2010 images per category.



(a) Before Augmentation (b) After Augmentation (c) Example

Figure 1: Histogram of Training Data Distribution before and after the Augmentation

4.2 Capsule Network Design

The capsule network architecture used for the traffic sign classification is identical to the original paper[Sabour et al., 2017]. The only difference is hyper-parameters are optimized for this dataset. The kernel size is reduced because the size of 9 is too large and the network can not perceive the detailed features in the traffic signs. The first convolutional layer is made 25% deeper because the traffic sign has more color channels. The number of digitcaps is increased to 43 to match the number of traffic sign categories (Note that the digitcaps should be called traffic-sign-caps here, but we use the term "digitcaps" as the original paper does to keep the terminology consistent). The dimension of the digitcaps is also increased to 40 to parameterize more variations due to the change of viewpoint, lighting and the background. The new hyper-parameters are shown in Figure 2 (modified from the original paper). The decoder structure network remains the same for this dataset. The empirical experiment shows that additional dense layers do not offer any performance gain. An attempt is also made to replace the dense decoder with a convolutional decoder. The result is the overall degradation of classification accuracy.

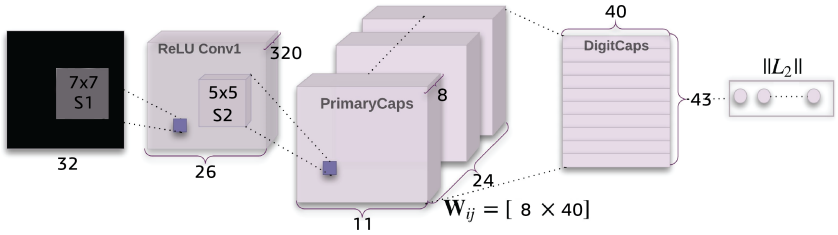


Figure 2: Capsule Network for German Traffic Sign Classification

4.3 Training the Network

Training the capsule network is surprisingly computationally expensive. It does not scale up well for a dataset with much more categories because each primary capsule unit requires an independent affine transformation matrix to predict its value in a digitcap vector. Thus, the network in Figure 2 requires $11 \times 24 \times 40 \times 43 = 454,080$ of 8×40 transformation matrices. If there are 32 samples in a training batch, the number of matrices is then 32 times more. In addition, to take advantage of TensorFlow's parallelized matrix multiplication, the capsule vectors in the primary capsule need to be broadcasted(tiled) for 43 times. The memory allocation associated with tiling also slows down the computation and causes the GPU kernel to idle. Due to the limited space on the video card, the maximum number of samples per batch is only 10. To compensate the noise in the gradient estimation due to small batch size, the training rate is reduced to 0.0001. The runtime for each epoch is about an hour, which is rather long comparing with the standard CNN. Nevertheless, the capsule network is a rather new architecture and is not natively supported by TensorFlow. The slow speed is acceptable at this research stage.

4.4 Result and Discussion

The test accuracy of this model turns out to be 94.8% after ten epochs of training. It might be possible to achieve higher accuracy with more training epochs, but the accuracy starts to level-off already, and the improvement will be marginal. A few sets of different hyper-parameters are also tested, but there is no improvement. In addition, the slow training speed makes exploring a large range of hyper-parameters practically impossible. The state-of-art classification accuracy based on CNN is 99.43%[Stallkamp et al., 2011]. The capsule network might be able to get close if the ensemble technique is adapted. Nevertheless, the capsule network does show an excellent performance on reconstructing the observed image. In Figure 3, the reconstructed image bears a close resemblance to the original images. It is conceivable that the capsule network also tries to reconstruct the background, which is practically the noise. This phenomenon becomes more clear when we manipulate the digitcaps vector's value. In Figure 4, it is possible to change the content in the reconstructed images' background. This indicates that the model is over-fitting and the digitcaps vector has too many degree-of-freedom. However, if the digitcaps vector's dimension is reduced to a smaller number (e.g., 16), the classification accuracy gets much worse, and the network still learns the background.

Figure 3: Input Images(top) and Reconstructed Images(bottom)

Figure 4: Reconstructed Images with Different Instantiation Parameters

5 Deep Capsule Network

In the previous section, we have discovered that it is difficult for a capsule network to match the performance of the standard convolutional neural network with max-pooling if a non-trivial amount of noise is present in the dataset. For the rest of this paper, we will explore if there is any remedy to this problem.

From the research in the past, it is demonstrated and theoretically proved that the deeper neural networks perform much better than the shallow ones[Mhaskar and Poggio", 2016]. Inspired by this phenomenon, we want to experiment whether we can improve the performance of the capsule network if we make it deeper. Note that the capsule network is very shallow compared to the standard CNN.

There are only two hidden layers (although the primary capsule is so much larger than a typical convolution layer of CNN).

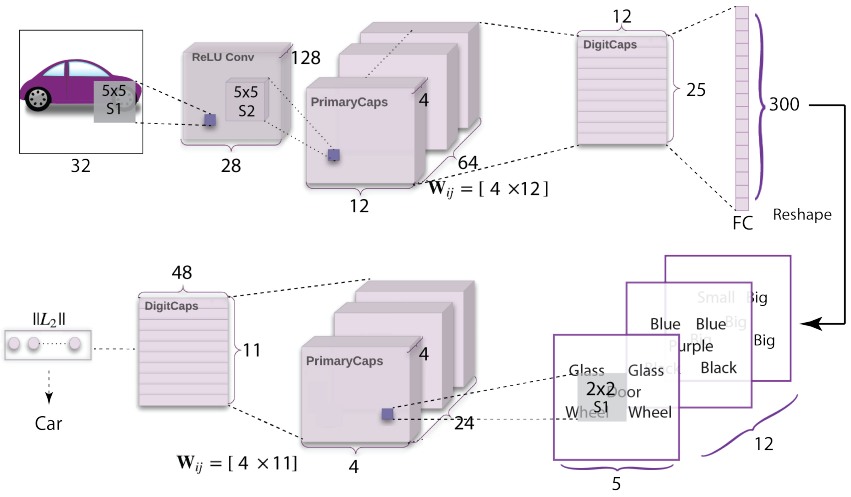


Figure 5: Deep Capsule Network Architecture for CIFAR10 Data Set

In the original paper, there is only one capsule layer in the network. To experiment with the deeper capsule network, a novel two-layer capsule network is designed. Figure 5 is the deep capsule network for CIFAR10 dataset tested in the original paper. The idea behind this architecture is to organize features from a flat image into a hierarchy. For example, when an image of a car is presented to the network, the first capsule network will recognize the components of a car, such as wheels, doors, windows etc. The second capsule network then read these abstract features(e.g., components) and use their properties(e.g., location, color, size, etc.) to determine the category of the entire object(e.g.car) consisting of the low-level components recognized in the first capsule layer. There is a dense layer that has the same number of neurons as its input digitcaps. The reason is that the meanings and the order of instantiation parameters in the digitcaps' vector are not known before the training. The dense layer helps the network to route these parameters to its proper spatial locations for further convolutional operation. As there are more pixels and channels in the image, we simply add an additional dense layer to the decoder (not shown in Figure 5).

To better benchmark the deep capsule network architecture, we also the tested it with MNIST dataset.

Note that due to the limited memory on the video card, the tested deep capsule network is downsized. For example, the capsule dimension of the primary capsule of the second capsule layer should be much larger for a high-level feature or object.

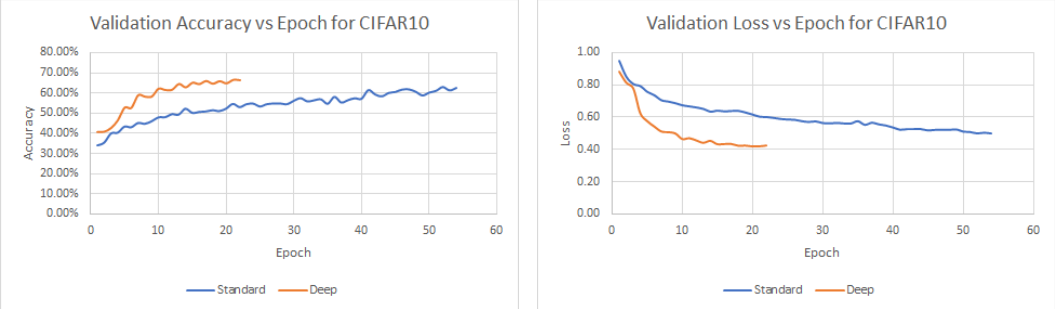
6 Benchmark Deep Capsule Network Using CIFAR10 and MNIST

In the original paper of the capsule network, the authors use both MNIST and CIFAR10 to benchmark the standard(single-layer) capsule network. The detailed configurations are given, and we can repeat their experiment. In this section, we train both deep and standard capsule network with these datasets and compare the result.

6.1 Comparing Deep and Standard Capsule Network on CIFAR10

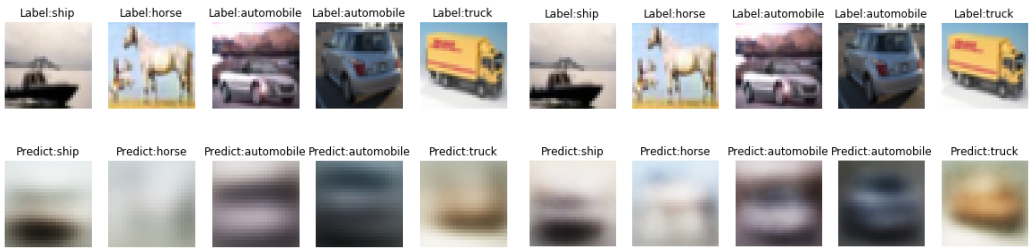
To benchmark the proposed network, both deep and standard capsule networks are trained. The stand network is trained with the setup described in the original paper—using randomly cropped 24×24 image, and the primary capsule number is increased to 64. To further augment the training data, we also add random color variation to the image. For the deep capsule network, the raw 32×32 is sent to it because it is expected to handle more information. The argumentation technique is color variation only.

In the original paper, 7 fold ensemble is used to bring the accuracy up. As ensemble technique does not provide any insight into the network itself, we choose not to do it.



(a) Accuracy (b) Loss

Figure 6: Accuracy and Loss vs. Epoches during Training for CIFAR10



(a) Standard Capsule Network (b) Deep Capsule Network

Figure 7: Comparison of Reconstruction Quality between Standard and Deep Capsule Network Top: Input Images; Bottom: Reconstructed Images

The test accuracy of the standard capsule network is 63.64% after 55 epochs of training, and the deep capsule network has 66.63% accuracy after 22 epochs of training. It may not seem to be much improved, but from Figure 6, we can see that the deep capsule network still has room for improvement if training keeps going. We have to stop the training due to the time constraint of this research project.

In addition, the deep capsule network also has better reconstruction outputs as we can see in Figure7. The reconstructed image from standard capsule network is very blurry and unrecognizable, but the deep capsule network actually manages to reconstruct the main building blocks of the target object.

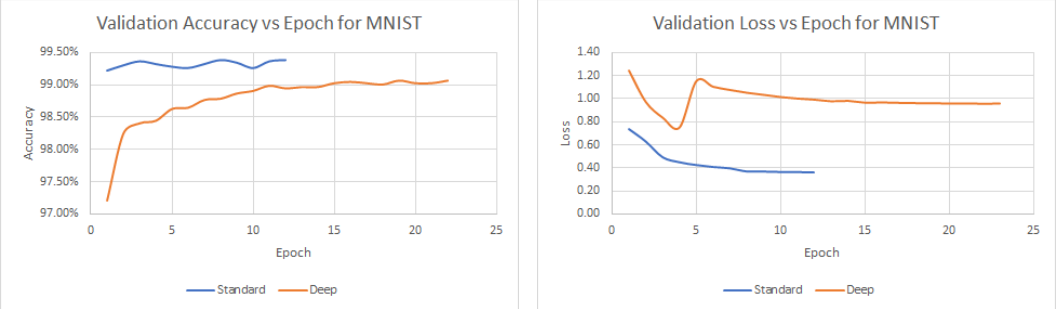
Unfortunately, the deep network still tries to reconstruct the background that is unrelated to the target object.

6.2 Comparing Deep and Standard Capsule Network on MNIST

We also repeat the same test with the MNIST dataset. The topology of the network stays the same, yet the size of the deep capsule networks' layers are reduced proportionally to match the resolutions of the MNIST dataset.

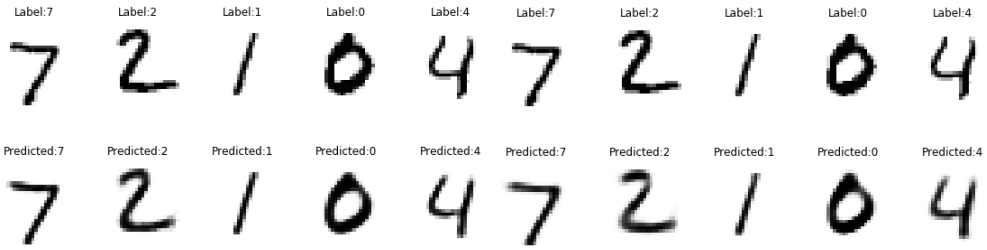
Similarly, we exam the accuracy and loss curves shown in Figure 8. The standard capsule network does learn very fast as mentioned in the original paper and achieves a high accuracy within two epochs of training. The deep capsule network starts off with low accuracy, but as the training goes on, it starts to catch up. The deep capsule network has trainable parameters four times more than the standard ones, and it can not be trained as fast.

There is a kink in the loss curve of the deep capsule network. It is not an error. The deep capsule network is originally trained on CPU, which turns out to be too slow. We are forced to switch over to GPU based training. However, we must use a much smaller batch size due to the limited memory on the video card. It is very obvious that the smaller batch size has a huge impact on the training quality. Looking at Figure 8, we can expect the deep capsule network to have better performance if it is trained on a better video card such as nVidia GTX 1080Ti used in the original paper.



(a) Accuracy (b) Loss

Figure 8: Accuracy and Loss vs. Epoches during Training for MNIST



(a) Standard Capsule Network (b) Deep Capsule Network

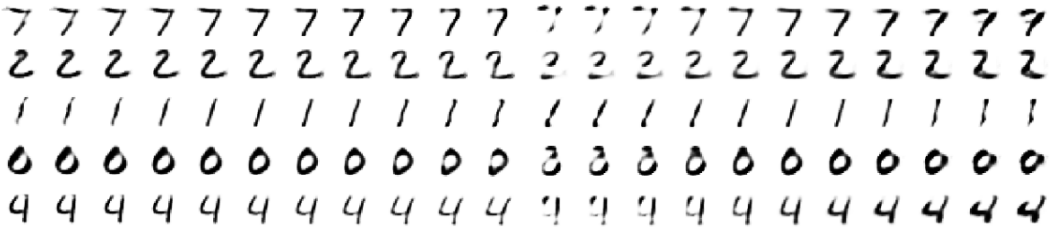
Figure 9: Comparison of Reconstruction Quality between Standard and Deep Capsule Network Top: Input Images; Bottom: Reconstructed Images

As we can see in Figure 9, the reconstruction of the input images from both networks look almost identical, although the deep capsule network's output is more smooth and less noisy.

The comparison of artificially generated images using instantiation parameters are the most interesting part of this experiment and confirms that the deep capsule network is working as intended. Figure 10 is the output generated from the decoder by varying the one of the output digitcap vector's value. On this figure, there are two important characteristics to notice.

The output of the standard capsule network mainly changes one property of the target object. In this case, it is the localized skew. On the contrary, the deep capsule network can manipulate a combination of properties. Not only it changes the localized skew, but it also varies the stroke thickness. This phenomenon shows that deep capsule layer does manage to learn the hierarchy of features. If one higher hierarchy property changes, more than one low-level properties vary with it simultaneously.

Secondly, the deep capsule network allows discontinued stroke. This is not observed in the output from the standard capsule network at all. It appears that the deep capsule network has learned the subcomponents of a character.



(a) Standard Capsule Network (b) Deep Capsule Network

Figure 10: Comparison of Artificially Generated Images between Two Networks

7 Conclusion

In this paper, we benchmark the standard capsule network's performance on a more challenging German traffic sign dataset that has many similarities to MNIST. A degradation of performance is observed. A deep capsule network is then proposed to improve the performance of standard capsule network. The experiments show that the deep capsule network does have the ability to acquire the hierarchy of features and has a higher accuracy on classifying more complex images.

8 Future Work

Both the standard and deep capsule networks have the tendency of learning the noise(e.g., background) of the image. A noise rejection mechanism is still needed to be invented and included in the capsule network. We could use max-pooling, but that will defy the purpose of capsule network which is designed to get rid of the pooling layer. Still, it is interesting to check if adding max-pooling can remove the background from the reconstructed image.

All deep capsule networks in this research are trained with batch sizes smaller than 8 due to the limit capacity of hardware that we have access to. The quality of training is rather poor and we do not have time to train them with more epochs. Nor do we have resource to do cross-validation. All the work should be repeated on a faster GPU that can at least host a batch size of 32. We do expect to see a better result with a higher batch size and epochs.

References

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, USA, 2012. Curran Associates Inc.

Navneet "Dalal and Bill" Triggs. "histograms of oriented gradients for human detection". In *"Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01"*, "CVPR '05", Washington, DC, USA, 2005. IEEE Computer Society.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21734-0.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. *CoRR*, abs/1710.09829, 2017.

"Usman Pirzada". "exclusive: The tesla autopilot-an in-depth look at the technology behind the engineering marvel". Technical report, "Dec" 2015.

"Steve Lohr". "a lesson of tesla crashes computer vision can not do it all yet". Technical report, "Sept" 2016.

"Aurelien Geron". "capsule networks", Dec 2017.

"Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel". "detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark". In *"International Joint Conference on Neural Networks"*, number "1288", 2013.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

"Hrushikesh Mhaskar and Tomaso A. Poggio". "deep vs. shallow networks : An approximation theory perspective". *"CoRR"*, "abs/1608.03287", 2016.