Dynamically routed capsule nets for sentiment analysis

Valerie Ding
Stanford University
Department of Computer Science
dingv@cs.stanford.edu

Abstract

Capsule nets have seen success in overlapping images and resilience to rotation in the MNIST dataset. Because they have primarily been presented as an alternative to CNNs in image processing tasks, capsule nets are very young in natural language tasks. However, because of their tree structure and alternative approach to highly-dimensional data, they are an attractive approach to explore in semantic analysis, and early research shows promise (Zhao et al., 2018; Wang et al., 2018). We propose a new capsule net implementation for sentiment analysis tasks, extend PrimaryCaps to the Stanford Sentiment Treebank and propose a SentimentCaps parallel to DigitCaps (Sabour et al., 2017), and achieve 76% classification accuracy on the SST dev dataset with our Sentiment CapsNet, on par with RNN and CNN models.

1 Introduction

1.1 Sentiment analysis task

Sentiment analysis, also referred to as sentiment or opinion mining, is a central problem in natural language understanding. Its applications are wide-reaching. Sentiment is interesting from a research perspective: in building machine models to parse highly-dimensional sentiment in language, we advance the human-machine interface and in turn have the opportunity to visualize, through the lenses of powerful, highly-dimensional machine models, just what makes human sentiment human. Sentiment analysis has also been used to gauge and predict economic and societal behavior such as financial market movements and political outcomes (Wang et al., 2012).

In the last two decades, the sentiment analysis task has elicited state-of-the-art approaches ranging from vector space models (Maas et al., 2011) to unsupervised and supervised learning models. Vector representations, single- and phrase-level word embeddings, and unsupervised clustering sheds light on lexical and semantic relations but does not capture sentiment in higher dimensions such as at the sentence level.

The supervised learning approach to sentiment analysis is justified from a predictive classification perspective. Specifically, recurrent neural networks (RNN) and tree-structured recurrent neural networks (TreeRNN) have achieved more than 80% prediction accuracy on a binary sentiment prediction problem, positive or negative (Socher et al., 2013b).

1.2 Recurrent neural network

RNNs leverage sequential hidden units to capture temporal features of data. Hence, RNN architectures have been widely explored in language processing in sequentially-formulated problems (Kumar et al., 2016; Wen et al., 2015).

As applied to classification tasks, supervised neural networks learn internal representations over time that allow them to make classification predictions on previously unseen examples. Neural networks employ forward propagation to make predictions and can use backpropagation to update weights with error gradient descent. Centrally, RNN forward propagation maps input sequences of length n to sequences of hidden units h_t (Pearlmutter, 1995):

$$h_t = f(x_t W_{xh} + h_{t-1} W_{hh}) \tag{1}$$

The hidden representation layer(s) then feed into the prediction layer

$$y = g(h_n W_{hy} + b) \tag{2}$$

1.3 Tree-structured neural network

Recursive architectures such as Recursive Neural Tensor Networks and dynamic memory networks have been proposed in semantic analysis and compositional lexical tasks (Socher et al., 2013a,b; Tai et al., 2015; Kumar et al., 2016). Structuring the neural network as a recursive tree of parent p and child L, R nodes, and maintaining representation dimensionality via parent nodes instead of n-length hidden units, the forward propagation algorithm becomes:

$$p = f(\texttt{concat}(x_L, x_R)W_{wh} + b) \quad (3)$$

$$y = g(h_r W_{hy} + b) \quad (4)$$

where h_r is the the root node representation.

1.4 Capsule nets

Capsule nets leverage properties of both tree-structured networks and convolutional neural networks (CNNs). Whereas traditional CNNs employ max-pooling layers to downsample input dimensionality, capsule nets employ tree-like routing-by-agreement to learn coupling coefficients between children and parents (Sabour et al., 2017).

Squashing function

Capsule output vectors are meant to quantify the probability that the capsule's represented feature (entity) is seen in the input. Thus, capsule j's output vector $\mathbf{v}_j$ given input $\mathbf{s}_j$ is:

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||} \quad (5)$$

Routing-by-agreement

The routing algorithm is defined for each prediction $\hat{\mathbf{u}}_{j|i}$, r iterations, and layer l:

1. `for all` capsule i in layer l and capsule j in layer $l+1$: $b_{ij} = 0$
2. `for r` iterations:
   $\rightarrow$ `for all` capsule i in layer l:
   $\mathbf{c}_i = \texttt{softmax}(\mathbf{b}_i)$
   $\rightarrow$ `for all` capsule j in layer $l+1$:
   $\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
   $\rightarrow$ `for all` capsule j in layer $l+1$:
   $\mathbf{v}_j = \texttt{squash}(\mathbf{s}_j)$
   $\rightarrow$ `for all` capsule i in layer l and capsule j in layer $l+1$:
   $b_{ij} = b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ (agreement term)
3. `return` $\mathbf{v}_j$

Coupling coefficients

Coupling coefficients c_{ij} are connections for wiring between between capsule i in layer l and the capsules in layer $l+1$. The logits b_{ij} are learned during the routing algorithm.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \quad (6)$$

2 Related work and considerations

In implementing and applying capsule nets to the sentiment analysis task, we modeled architecture after the original capsule net paper (Sabour et al., 2017) and decided to implement in Keras after perusing open-source capsule net implementations (for example, CapsuleNet-Keras and CapsNet-Tensorflow on GitHub are both written for MNIST) and noting design decisions. While (Zhao et al., 2018) and recent research in 2018 have begun to explore capsule nets as applied to sentiment analysis, they employ the capsule concept in designs of ConvNets or RNNs. Thus, by implementing capsule nets for the Stanford Sentiment Treebank, we demonstrate an instance beyond MNIST where capsule nets could be useful.

Conv1D layers and max-pooling, along with GloVe word embeddings, has been explored in semantic classification tasks (Zhang et al., 2015). The Stanford Sentiment Treebank is a specialized dataset that requires some data processing to be valid input into our neural networks, which we addressed with Keras Embedding layers and Tokenizer. Our consideration in designing LSTM-ConvNet and LSTM-CapsNet architectures separately was that capsule nets were proposed as an alternative to convolutional neural nets, so we wanted to test them separately. Notably, the original capsule net paper actually proposes a capsule network with a Conv1 layer as the first layer; in SentimentCapsNet, we feed an Embedding layer into an LSTM instead. We experimented wtih Conv1 in the SentimentCapsNet itself but RNN structure is generally better suited to sequential input data of one dimension.

LSTM (long short-term memory) is an adaptive (reset-able) RNN cell (Gers, 1999). We chose to use LSTM as the first layer in SentimentCapsNet due to its success in RNN applications to the sentiment analysis task. Note also that the baseline TfRNNClassifier employs LSTMCells so this is a consistent point of comparison across models.

3 Models

3.1 Distributed feature representations

We use 100-dimensional GloVe word embeddings to featurize SST sequences which are the first layer of inputs to the models. To incorporate this into our neural nets, we use Keras Tokenizer and an initial Embedding layer with embedding dimension of 100.

3.2 Baseline models

We train TreeNN and Tensorflow RNN models to obtain baseline performance metrics.

3.3 Convolutional neural net

We train a convolutional neural net to obtain a benchmark for our capsule net implementation, since capsule nets are posed as an alternative to convolutional neural nets. This CNN architecture consists of three Conv1D layers with 128 filters, kernel size 5, and ReLU activation, using intermediate max-pooling layers.

3.4 LSTM-to-ConvNet

To capture the sequentiality of the SST input data, we propose a RNN-CNN combination neural net by following the Embedding layer with an LSTM with output sequences matching the embedding dimensionality. Then we pass the outputs of that layer as inputs into the ConvNet described in the previous section.

3.5 Sentiment CapsNet

We implement primary capsules and final capsule layers for the sentiment analysis task. Specifically, we implement PrimaryCaps and SentimentCaps analogous to DigitCaps for the Stanford Sentiment Treebank. To do this, we use 1-dimensional convolutional layers to process input sequences and write a custom decoder. We use Keras for the implementations.

The SentimentCapsNet architecture (Figure 1) starts with an Embedding layer, passes to a layer (can be ReLU Conv1 for example, or any type of cell such as LSTM as discussed later). PrimaryCaps is 32 channels of 8-D capsules with 1x1 kernel and stride 2. SentimentCaps, which is akin to DigitCaps in (Sabour et al., 2017), has 2 16D capsules (one for each y label). The dynamic routing occurs between PrimaryCaps and SentimentCaps.

We also implement a custom decoder (Figure 2) to interpret the output of the capsule network. The decoder first feeds SentimentCaps to a fully-connected (Dense) layer with ReLU activation, then through a fully-connected dense softmax classifier for final output.

3.6 LSTM-to-SentimentCapsNet

Because we are attempting to explore the viability of capsule networks on this semantic analysis task, we replace the ConvNet in our previous RNN-CNN combination model with a capsule network. This required an original implementation of capsule networks for the SST dataset, since existing capsule network frameworks are tailored to MNIST and image classification tasks.

4 Initial staging experiments

4.1 Data

We use the Stanford Sentiment Treebank (SST) train/dev data and the binary problem expressed via grouping the labels (scores) into positive and negative categories (Socher et al., 2013b).

4.2 TreeNN

TreeNN trained on the SST train set over 100 epochs and with learning rate $\lambda = 0.05$, achieves F-score = 0.51, 0.49 on negative and 0.52 on positive, with precision and recall around the 0.50 mark in both cases (Table 1).

	prec	recall	f1	support
negative	0.50	0.49	0.49	428
positive	0.51	0.52	0.52	444
overall	0.51	0.51	0.51	872

Table 1: TreeNN classification report.

4.3 RNN (LSTM)

An LSTM trained on the SST train set with the same hyperparameters as the TreeNN yields higher performance metrics (Table 2). Thus we focus on RNN over TreeNN-based architectures when feeding into NN architectures.

5 Results

5.1 Evaluation

We trained and evaluated the models in Keras using the SST train/dev split. We can confidently focus on dev set classification accuracy for each

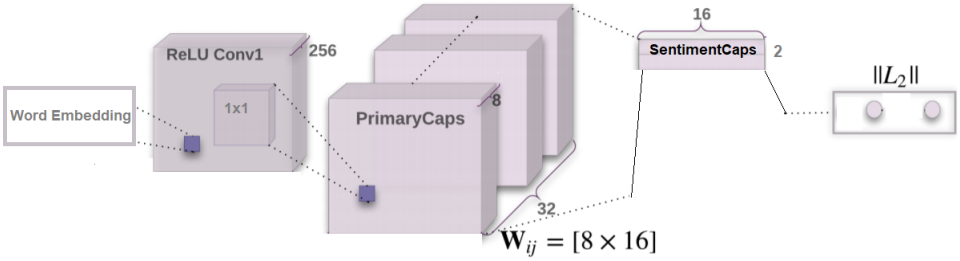

Figure 1: SentimentCapsNet architecture. Diagram adapted from (Sabour et al., 2017).

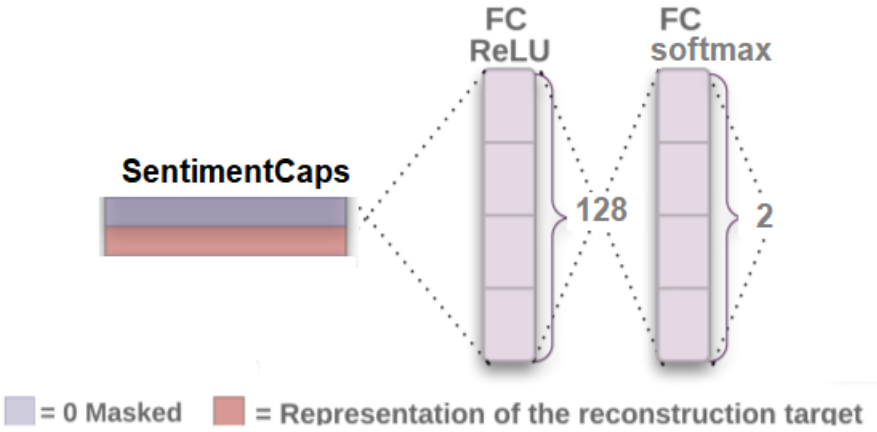

Figure 2: Custom decoder. Diagram adapted from (Sabour et al., 2017).

	prec	recall	f1	support
negative	0.76	0.70	0.73	428
positive	0.73	0.79	0.76	444
overall	0.75	0.75	0.75	872

Table 2: LSTM classification report.

model because the positive and negative classes are balanced in the SST train and dev datasets, and moreover Keras removed recall, precision, and f1 from their model metrics citing unreliability due to batch approximation.

5.2 ConvNet

The ConvNet with three 1-dimensional convolutional layers peaks at 77% dev set classification accuracy at training epoch 8 (Figure 3). The lowest accuracy performance is 68% in the first epoch.

The ConvNet reaches approximately peak performance at the fourth epoch, consistent with previous ConvNet findings on semantic analysis which reach peak performance at the second training epoch, for example. In fact, one reason why we did not need to train for more than 10 epochs for any model was that the GloVe embeddings already do decently in establishing a classification model. However, we still want to train for a number of epochs to establish consistency, a learning trajectory, and it is important when comparing to something like caps nets, which is dynamically routing probability weights and thus necessitates more careful consistency checks.

5.3 LSTM-to-ConvNet

The LSTM-to-ConvNet peaks at 80% dev set classification accuracy at training epoch 8 and 10 (Figure 5). The lowest accuracy performance is 61% at epoch 3.

5.4 LSTM-to-CapsNet

The LSTM-to-SentimentCapsNet peaks at 76% dev set classification accuracy at training epoch 7 (Figure 7). The lowest accuracy performance is 66% at epoch 3.

5.5 Models summary

Looking at the compiled performance of dev set classification across models (Figure 9), we see

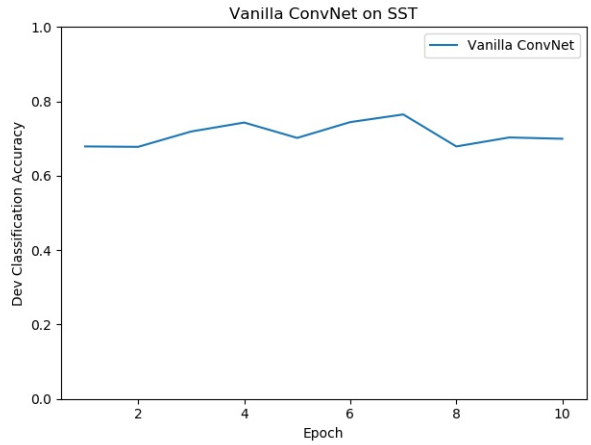

Figure 3: ConvNet SST dev set accuracy over training epochs.

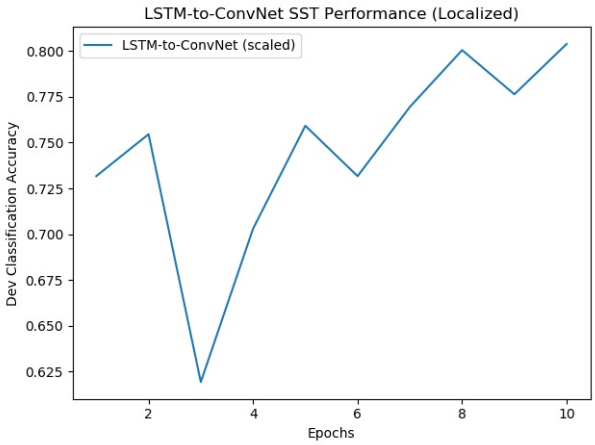

Figure 6: LSTM-to-ConvNet SST dev set accuracy over training epochs, scaled for clarity.

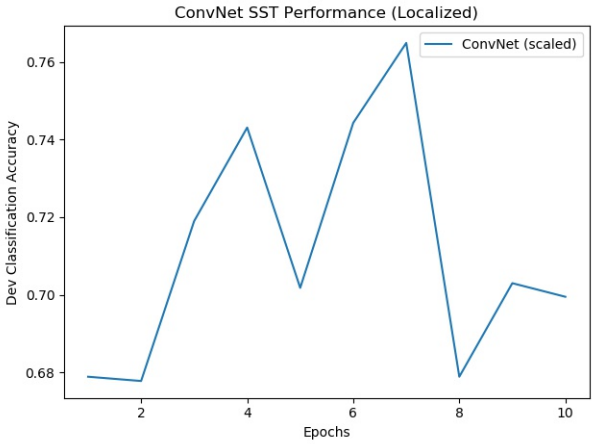

Figure 4: ConvNet SST dev set accuracy over training epochs, scaled for clarity.

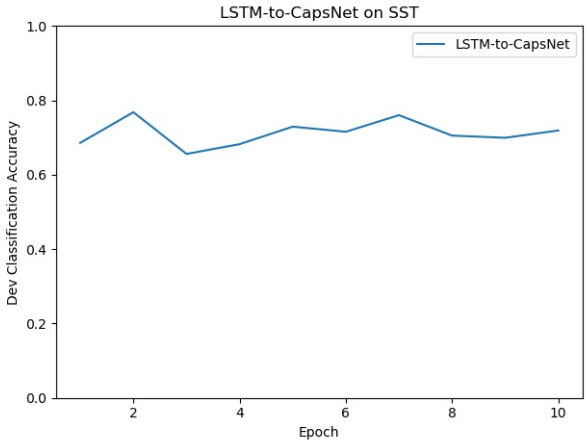

Figure 7: LSTM-to-CapsNet SST dev set accuracy over training epochs.

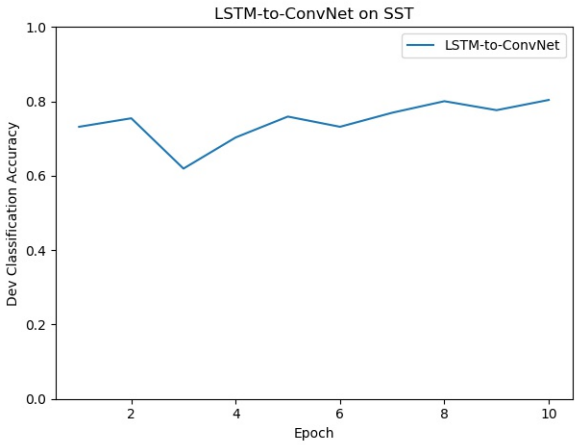

Figure 5: LSTM-to-ConvNet SST dev set accuracy over training epochs.

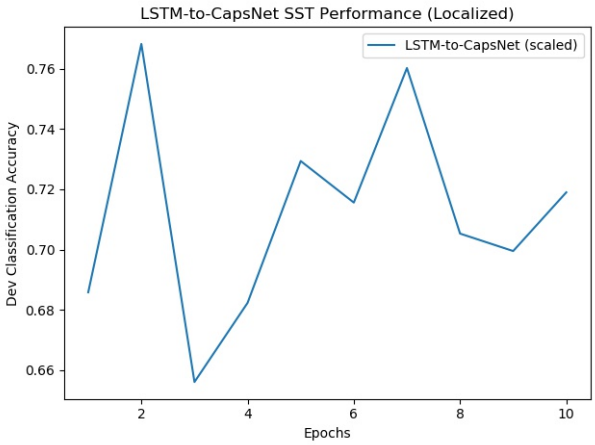

Figure 8: LSTM-to-CapsNet SST dev set accuracy over training epochs, scaled for clarity.

that LSTM-to-CapsNet is able to match performance of ConvNets and the RNN-CNN combination, LSTM-to-ConvNet, and notably is able to outperform all other models at epoch 2 and maintain until epoch 7 where it matches the 3-layer ConvNet in classification accuracy performance. Compared to TreeNN, we see that all models are able to massively outperform. This is where capsule nets are notably advantageous, given their direct analogy to TreeNN as conceptually a tree-structured network.

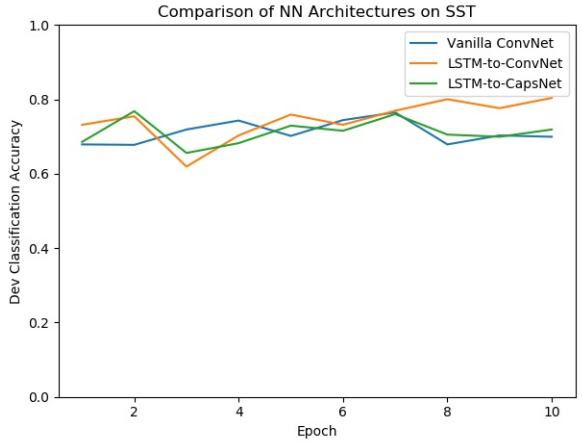

Figure 9: Multiple model SST dev set accuracy over training epochs.

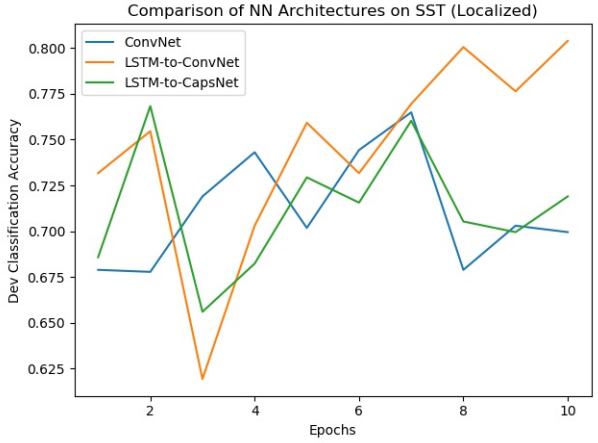

Figure 10: Multiple model SST dev set accuracy over training epochs, scaled for clarity.

6 Discussion

We began this work suspecting that capsule nets, due to their success in image classification tasks, would be a contender to convolutional neural networks. However, we were uncertain on how they would fare on a sentiment analysis task, and against RNNs. Our results show us that our capsule net architectures indeed compete well and outperform ConvNets, and are able to handle sequential data surprisingly well given that the capsule dynamic routing is actually meant to increase the flexibility from standard max-pooling procedures and thus address a shortcoming of ConvNets, but not necessarily address any shortcoming with RNNs. Compared to the TreeNN baseline, any one of our neural net models outperforms by on the order of 20-30% classification accuracy, but this is most notable for capsule nets given their tree structure being most comparable to TreeNN in design and conceptually.

Because our capsule nets are dynamically routed, we draw the parallel to tree-structured networks which can also be extended to train directional weights dynamically. Compared to the TreeNN baseline, however, any combination of RNN, ConvNet, and our SentimentCapsNet is able to significantly outperform.

A common critique of capsule nets is asking why it works. It actually arguably makes more conceptual sense to apply it to one-dimensional vectors (such as word embeddings) first, as we have done here, before attempting it on two- or higher-dimensionality spaces such as images. The first successful implementation of capsule nets on the MNIST dataset is a meaningful data point that is in many ways corroborated by our results.

However, there is room for improvement. State of the art models are able to consistently achieve 80+% classification accuracy while our SentimentCapsNet models hover around 70-80% classification accuracy. It is important to continue exploring, to better understand the architectures we are employing and what datasets they may be better or worse suited to.

7 Uncertainty analysis

Although we reported classification performance metrics, in evaluation it is also important to think about the methods and datasets from a conceptual, theoretical level.

Capsule nets are inherently convolutional. That is why they are compared to ConvNets. One of the biggest proposed advantages is an alternative to max-pooling in dynamic routing. One way to

interpret this is the search for a method that not only learn model weights but also maintains the complexity and dimensionality, also without overfitting.

In fact, the decoder is meant to counteract overfitting to noisy data. This is something that RNNs do not automatically have structured into them. Uncertainty in the form of noisy data or otherwise confusing text could be where capsule nets could perform, coupling the resilience of ConvNets with the dynamic learning of tree-structured networks.

In initial experiments, the ReLU decoder layer before the softmax decoder layer proved critical to the dev set classification performance of SentimentCapsNet, changing accuracy by on the order of 20%. This is one way to quantify the effect of the decoder, but different designs of the decoder could be interesting to explore.

Another point is considering what it would take to reach state of the art model classification performance. There is about a +10% classification accuracy remaining to consistently outperform state of the art models. This might be explained by deep neural net architectures (Socher et al., 2013b). We use three-layer ConvNets in our ConvNet and LSTM-ConvNet models. Deep capsule nets might be a natural progression but would require work on multiple decoders to counteract overfitting and ensure learning of relevant features, not noise in data.

Additionally, in conducting research we must be aware of the utility of the work we are doing. This paper focuses on a new implementation of capsule nets to the sentiment analysis task with a twofold goal, one to understand if capsule nets could be viable (the results appear positive), and secondly to push at the years-old sentiment analysis task. While the second goal is currently dominated by deep neural nets, we achieved the first goal and demonstrated that there is room to explore deep capsule nets as a result of this work.

8 Future work

We used 100-dimensional GloVe word embeddings in our models, 10 training epochs, and certain configurations of convolutional layers and capsule layers. It would be important to train these models with other hyperparameters for more extensive testing.

Even though Keras deprecated the built-in precision, recall, and F-score metrics tracking, there could be utility in redesigning or re-implementing, since these metrics help us understand model specific performance on classes.

This paper focused on the binary classification task due to want for solid baselines and rigorous benchmarks, but it would be interesting to see how SentimentCapsNet fares on the fine-grained sentiment classification task. This requires some refactoring and neural network redesign, which leads to the next point of potential future work.

There is huge need to open-source a generalizable implementation of capsule nets. Much of the work involved in this paper was implementing and testing the implementation of capsule nets for the sentiment analysis dataset, which is not currently open-sourced anywhere to our knowledge. Huadong Liao is beginning this initiative on GitHub, attempting to create a Tensorflow-like library for capsule nets, but it currently only supports MNIST.

Our SentimentCapsNet implementation is a step forward, but in order to truly advance research in this field we should look to working on open-source generalizable capsule net implementations. We can also look into other text-parsing related applications of SentimentCapsNet, such as seq2seq translation, and extend to voice applications. There are a high number of related applications that are confirmed to be interesting from demonstrating the viability and performance of capsule nets on tasks of sentiment analysis dimensionality.

9 Conclusions

We implemented capsule nets as first proposed by (Sabour et al., 2017) for the MNIST image classification task, for the sentiment analysis task. This involves implementing primary caps, capsule layers, and the custom decoder for capsule net output.

We trained each model (3-Layer ConvNet, RNN-CNN LSTM-to-ConvNet, and our LSTM-to-SentimentCapsNet) over 10 epochs on the SST train set and consistently achieved 70-80% dev set classification accuracy, demonstrating the viability of combining NN architectures in the sentiment analysis task, and more importantly demonstrating the competitiveness of capsule nets in sentiment analysis. There is incredible uncertainty surrounding capsule nets and most commonly repeated note by researchers currently is to implement capule networks and test them across different datasets.

We did that by implementing SentimentCapsNet in this paper, applying it to the Stanford Sentiment Treebank, and noting that SentimentCapsNet outperforms RNN and CNN models in the first half of the epochs and then begin to parallel CNN models in behavior with more epochs.

Ultimately, we successfully implemented capsule nets for the Stanford Sentiment Treebank dataset, with our own implementation of PrimaryCaps, proposing SentimentCaps to parallel DigitCaps, and a custom decoder for our SentimentCapsNet. SentimentCapsNet is competitive with RNN and ConvNet models. Applications that SentimentCapsNet appears advantageous for, based on our results, are in cases of limited training time or processing power, and preferable over pure ConvNet models, which agrees with the compared-to-ConvNet results in (Sabour et al., 2017).

We plan to open-source our implementation of capsule networks for the Stanford Sentiment Treebank sentiment analysis. We hope to contribute to continued development of open-source capsule net libraries, as we have demonstrated its viability and competitiveness in sentiment analysis tasks in this paper and capsule nets could have a place alongside RNNs and ConvNets in many other tasks.

Acknowledgments

I would like to thank Kelsey Josund for helpful feedback every step of the way. Thanks to Prof. Chris Potts, Prof. Bill MacCartney, and the rest of the CS224U teaching staff. I especially referred to the CS224U class SST, TreeNN, RNN and Tensorflow modules extensively to understand the dataset and implement my baseline classifiers, as well as understand the limitations and capabilities of Tensorflow. I ultimately used Keras due to the breadth of resources and tutorials available for constructing custom neural nets. Thanks to Xifeng Guo, who open-sourced his CapsNet implementation in Keras for the MNIST dataset and brought attention to the need to implement and test on other datasets, which motivated my implementation of Capsule Nets for the SST dataset. Thanks as well to Huadong Liao, who open-sourced his MNIST CapsNet in Tensorflow.

References

F.A. Gers. 1999. Learning to forget: continual prediction with lstm. *IET Conference Proceedings*, pages 850–855(5).

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

B. A. Pearlmutter. 1995. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228.

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

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *CoRR*, abs/1804.00538.