Text Categorisation using Convolutional Neural Network

•••

Submitted by:
Apurva Bhargava

Submitted to:
Dr. Namita Mittal
Associate Professor
MNIT, Jaipur

Objective

Statistical analysis of a single-channel static convolutional neural network model for text categorisation

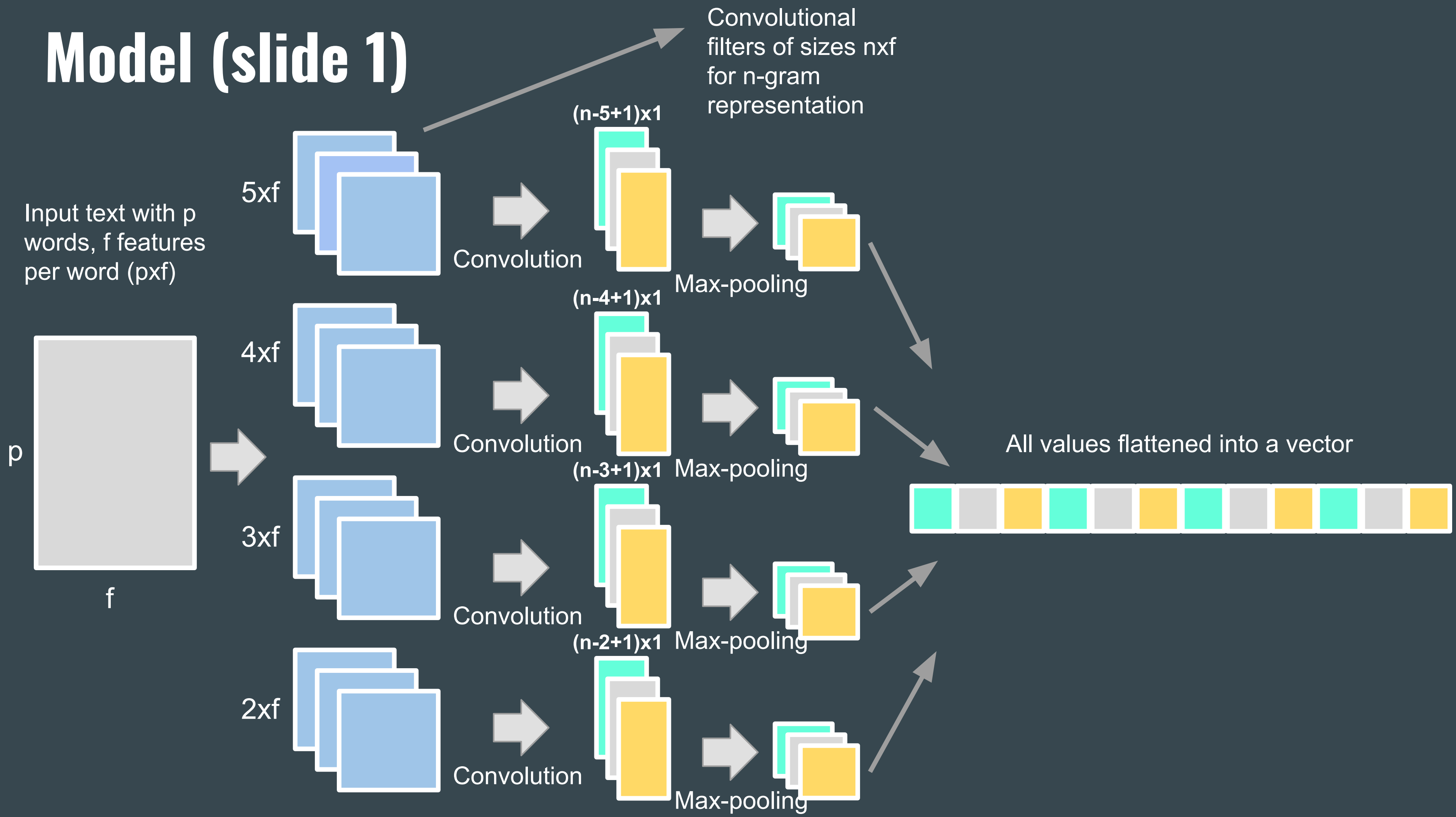
Introduction

CNNs are several layers of convolutions with non-linear activation functions like ReLU or tanh applied to the results.

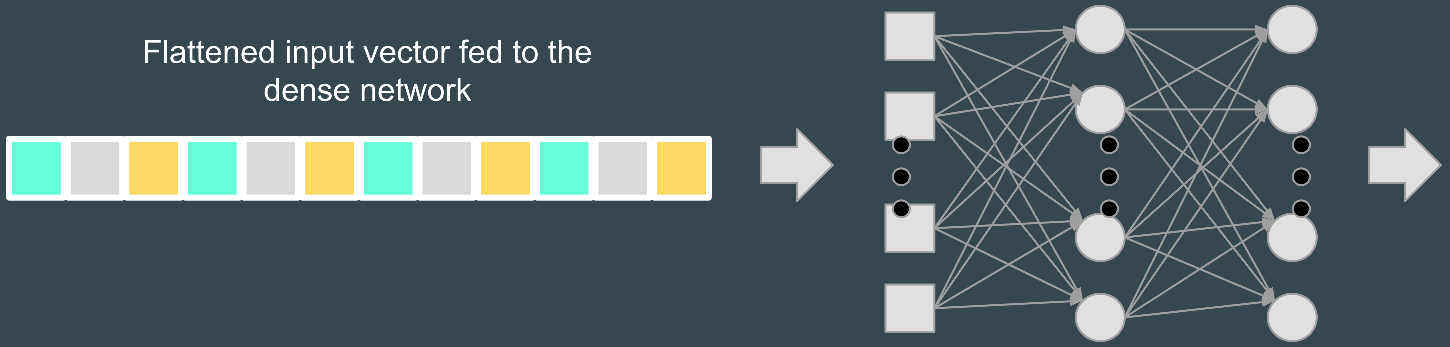
In CNN-static, input features are pre-trained word vector embeddings, which are fixed and do not change over the course of training.

Single channel in CNN implies that there is only word2vec embedding matrix for one input.

Model (slide 1)


Convolutional filters of sizes nxf for n-gram representation
Input text with p words, f features per word (pxf)
p
f
5xf
4xf
3xf
2xf
Convolution
Convolution
Convolution
Convolution
(n-5+1)x1
(n-4+1)x1
(n-3+1)x1
(n-2+1)x1
Max-pooling
Max-pooling
Max-pooling
Max-pooling
All values flattened into a vector

Model (slide 2)



Flattened input vector fed to the dense network

Dense, fully-connected network

Output layer

36 nodes

Evaluation using metrics and checking on test set

Methodology (Environment: Jupyter. Libraries: tensorflow, gensim, matplotlib, nltk)

Data preprocessing

- Non-ASCII characters are removed and a subset of text with central word count range is chosen.
- Pure words are extracted using nltk.RegExpTokenizer. Sentences are padded with ‘unk’ to make the length same.

Feature maps generation

- The sentences for both training and testing sets are fed to gensim Word2Vec for generating word embeddings.
- Algorithm: Skip-gram. Dimensionality through most of the testing: 200

Training

- The word embedding matrix (pxf for a sentence with p words and f features per word) is fed to a convolution layer with variable-sized filters with ReLU activation, followed by a max-pooling layer.
- Output is flattened into vector and fed to the dense layer.

Testing/ Evaluation

- A smaller labeled dataset of texts is used as testing dataset.
- Loss and accuracy are used as metrics for evaluating model for different settings of hyperparameters..

Hyperparameters considered for analysis

Learning rate

Learning rate specifies the size of movement for each step of gradient descent, i.e., how much the coefficients change on each update

Filter types

A convolution filter of size nxf is used for extracting the representation of a n-gram from the matrix representation. **Filter types=k means 2xf, 3xf, ... (k+1)xf filters.**

Filter count

It is the number of filters used for each filter type. Each filter is initialized by glorot_uniform _initializer, which draws samples from a uniform distribution.

Dropout rate

Dropout refers to ignoring certain number of neurons at random during both forward and backward passes, to avoid over-fitting (used here in dense layer).

Features per word

The gensim word2vec embedding model allows choosing the number of features per word. More features mean better representation.

Datasets

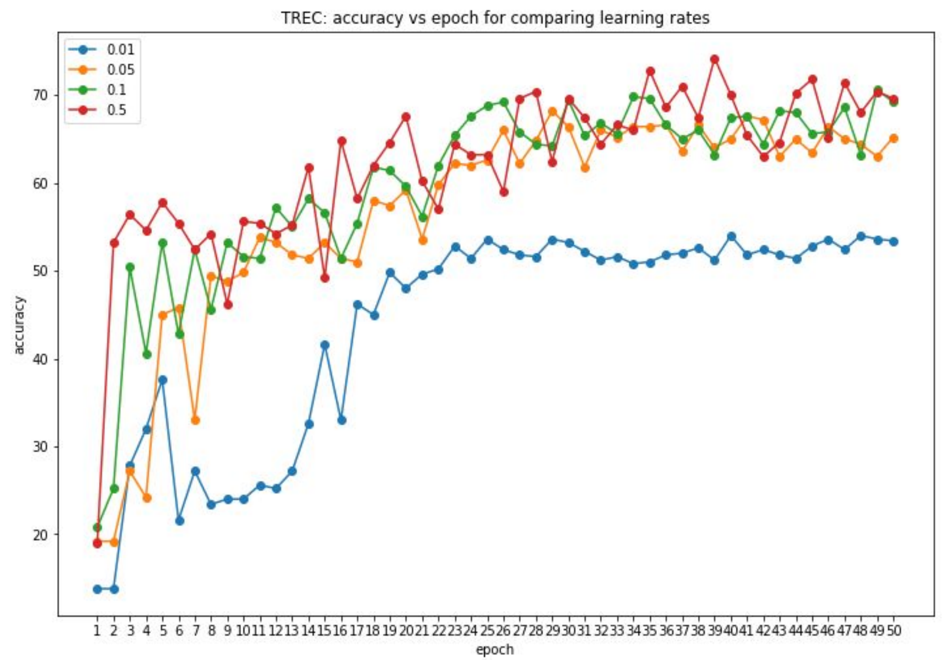
DATASET 1: TREC:

- Text REtrieval Conference dataset QA (Question Answer) dataset's goal is to achieve more information retrieval than just document retrieval by answering factoid, list and definition-style questions.
- Here, 4000 questions are used for training and 500 for testing. There are 6 major categories for classification. Vocabulary: 7546 words.
- Number of words per question = 34. Thus, each input has a matrix representation with dimensions 34xf, where f is number of features per word in word2vec embedding.

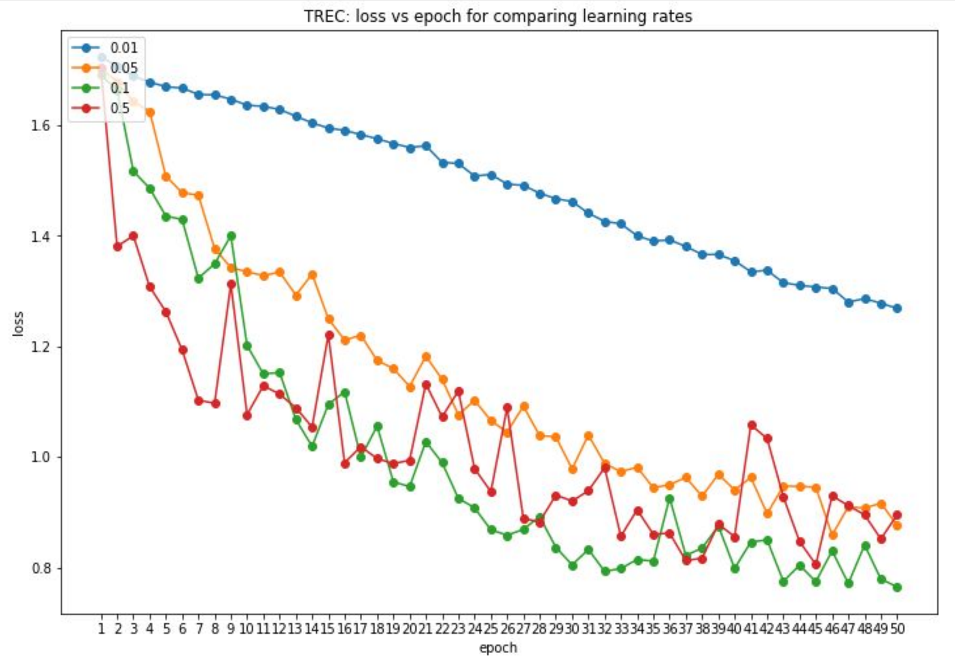
DATASET 2: AGNews:

- AG is a collection of more than 1 million news articles. News articles have been gathered from more than 2000 news sources by ComeToMyHead in more than 1 year of activity. ComeToMyHead is an academic news search engine which has been running since July, 2004.
- Here, a subset with 20000 texts in training set and 4000 in testing set is used. There are 4 categories for classification. Vocabulary: 34884 words.
- Number of words per article = 87. Thus, each input has a matrix representation with dimensions 87xf, where f is number of features per word in word2vec embedding.

LEARNING RATE



TREC: accuracy vs epoch for comparing learning rates

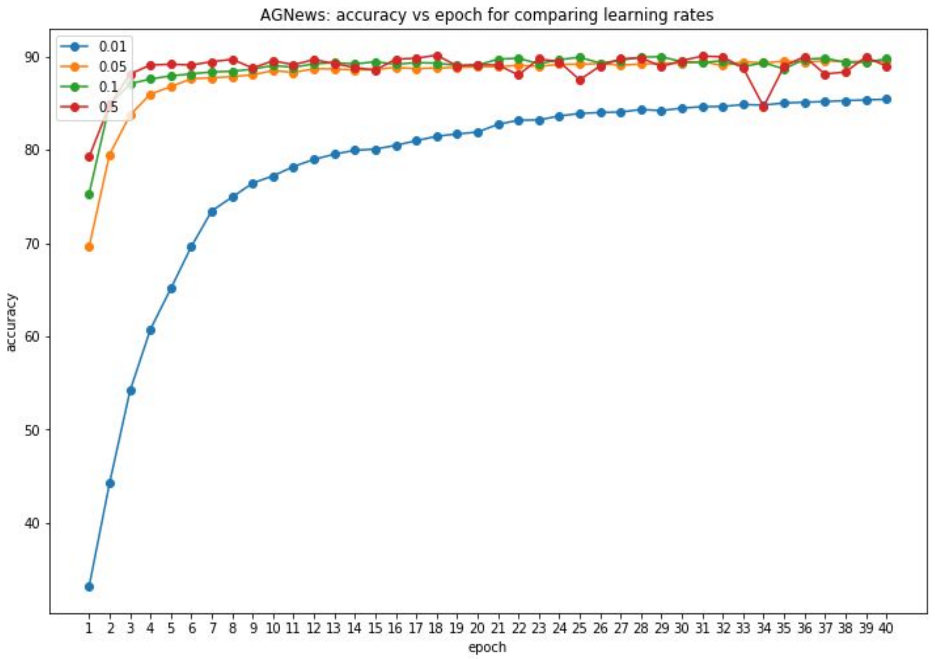


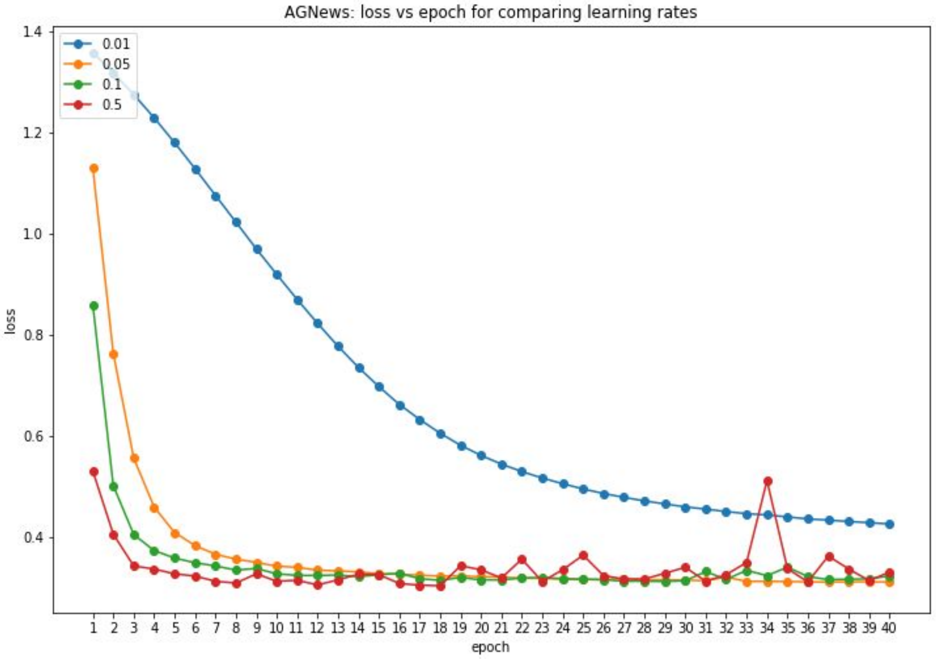
TREC: loss vs epoch for comparing learning rates

Dataset: TREC
Dropout = 0.3
Number of features per word = 200
Filter count = 3
Filter types = 4
Batch size = 100
Epochs = 50
Learning rate: variable

learning rate	accuracy
0.01	54
0.05	68.2
0.1	70.6
0.5	74.2

Decreasing learning rate reduces the step size in training. This leads to slower convergence, i.e., increased training time to reach a certain level of accuracy. As plotted, for 50 epochs, setting a learning rate of 0.01 gave an accuracy of 54% on testing set, whereas for the same number of epochs, learning rate of 0.5 resulted in 74.2% accuracy on testing set.



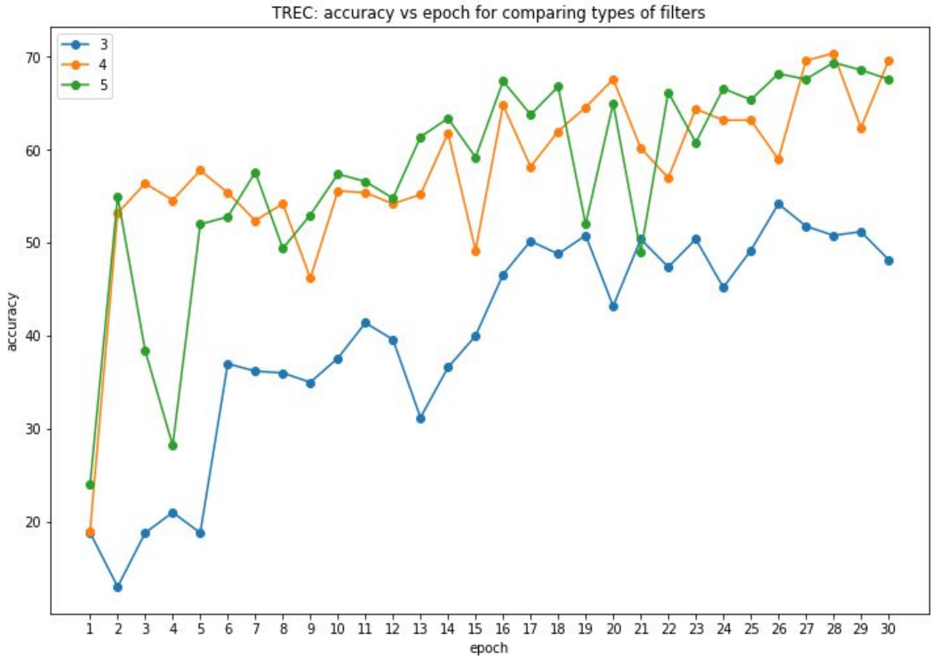


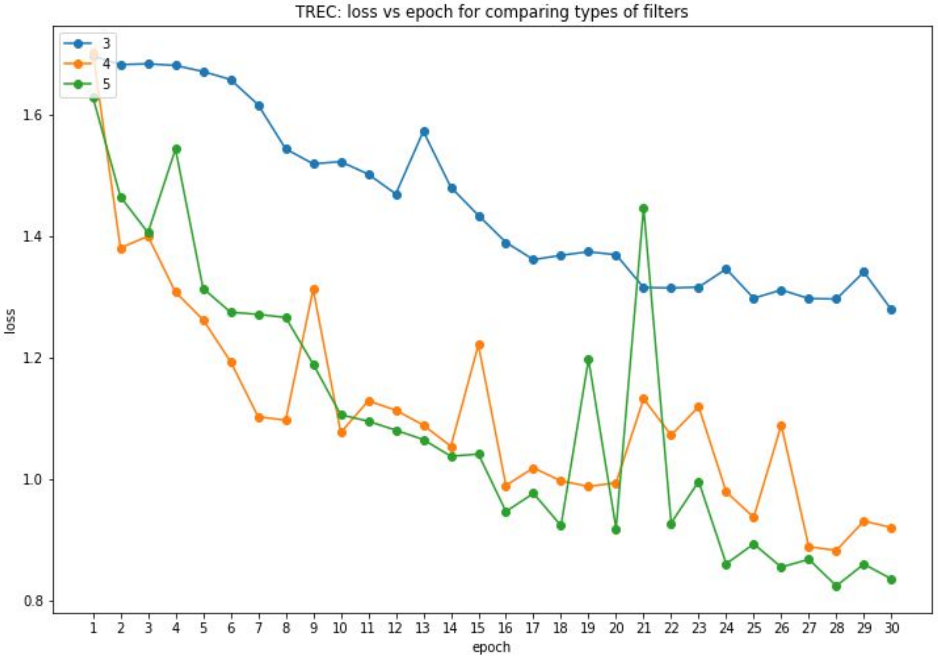
Dataset: AGNews
Dropout = 0.3
Number of features per word = 200
Filter count = 3
Filter types = 4
Batch size = 500
Epochs = 40
Learning rate: variable

learning rate	accuracy
0.01	85.425
0.05	89.55
0.1	89.975
0.5	90.15

As shown in the plots, for the same number of epochs (40), a higher learning rate results in greater accuracy. It may not always be the case, however. A larger learning rate may hinder convergence by taking too large steps that move away from the optimum. A solution is to decrease the learning rate every time the accuracy and loss stop improving.

FILTER TYPES

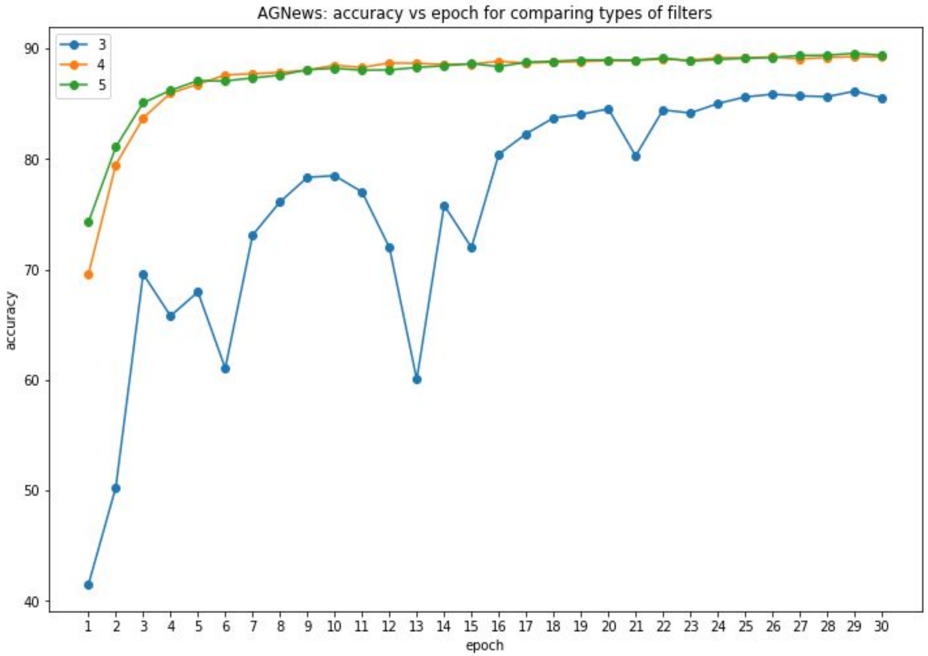


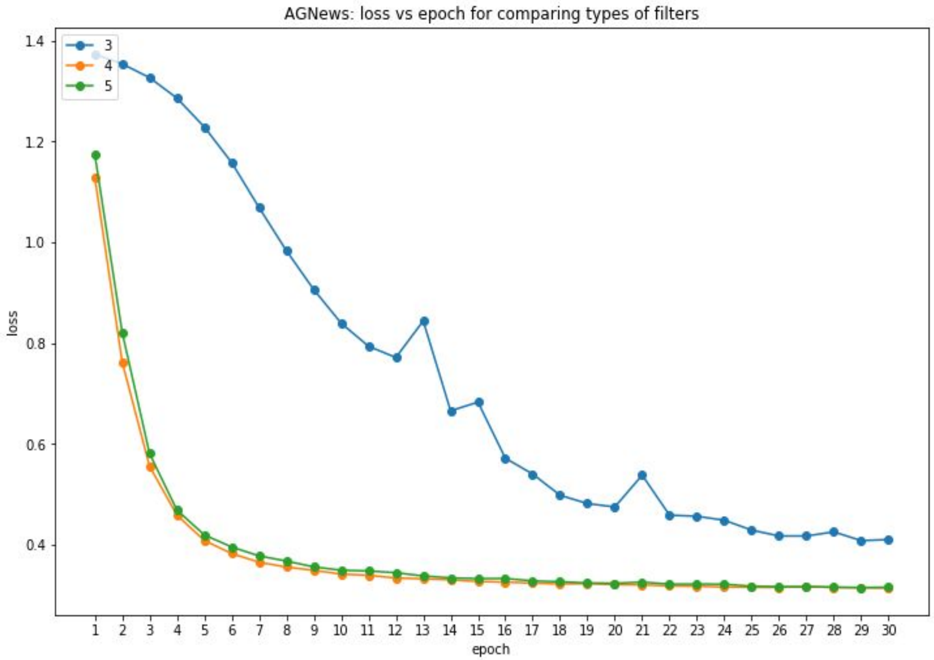


Dataset: TREC
Dropout = 0.3
Number of features per word = 200
Filter count = 3
Filter types: variable
Batch size = 100
Epochs = 30
Learning rate = 0.5

Filter types	accuracy
3	54.2
4	70.4
5	69.4

Increasing the types of filters improves representation by incorporating n-grams with larger n and increases the number of features used as input to the dense layer. Thus, increasing the filter types improves accuracy, as seen in the plots when accuracy improves from 54.2% to 70.4% when more filter types are used. Usually, representations upto 5-grams are sufficient.



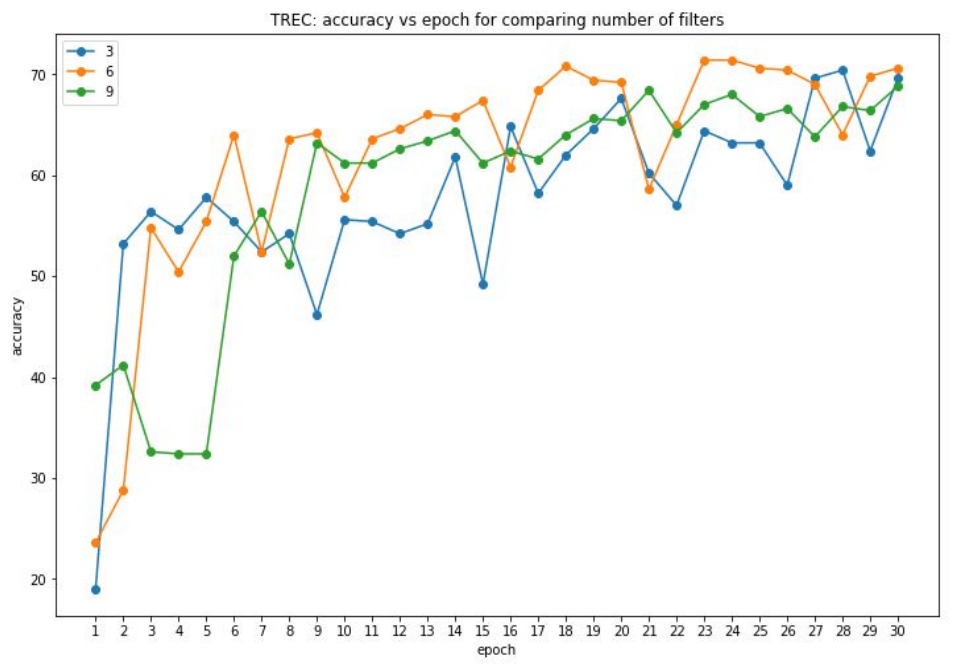


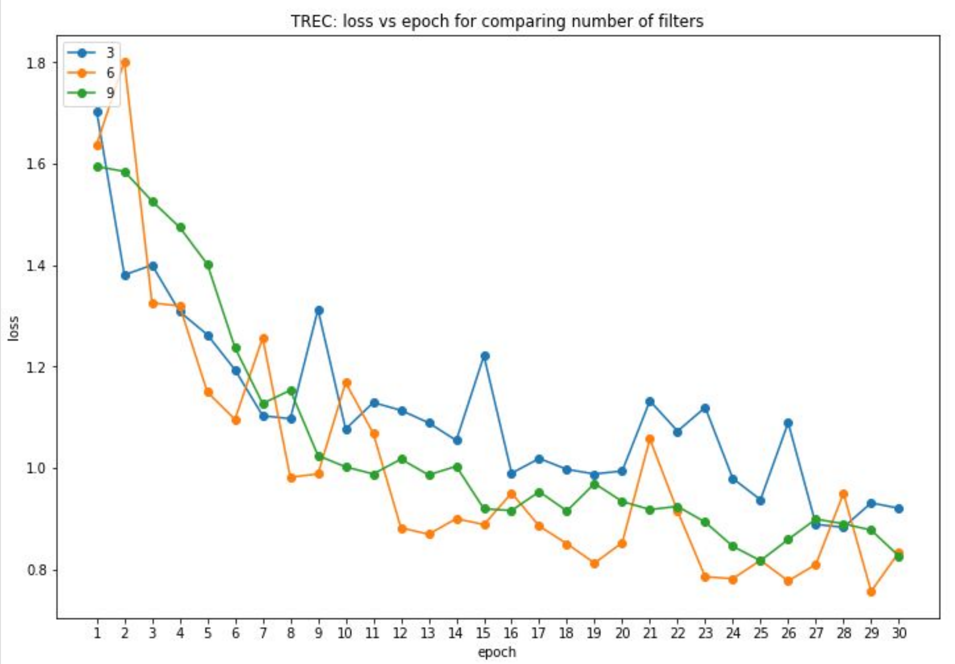
Dataset: AGNews
Dropout = 0.3
Number of features per word = 200
Filter count = 3
Filter types: variable
Batch size = 500
Epochs = 30
Learning rate = 0.05

Filter types	accuracy
3	86.15
4	89.275
5	89.575

Dimensions of filters for Filter types=3 are 2x200, 3x200, 4x200. For Filter types=4, an additional 5x200 filter is also used and for Filter types=5, additional 5x200 and 6x200 filters are also used. As shown in the plots, the accuracy improves by using more types of filters. There isn’t any significant improvement in accuracy on using filters with dimensions larger than 5x200.

FILTER COUNT

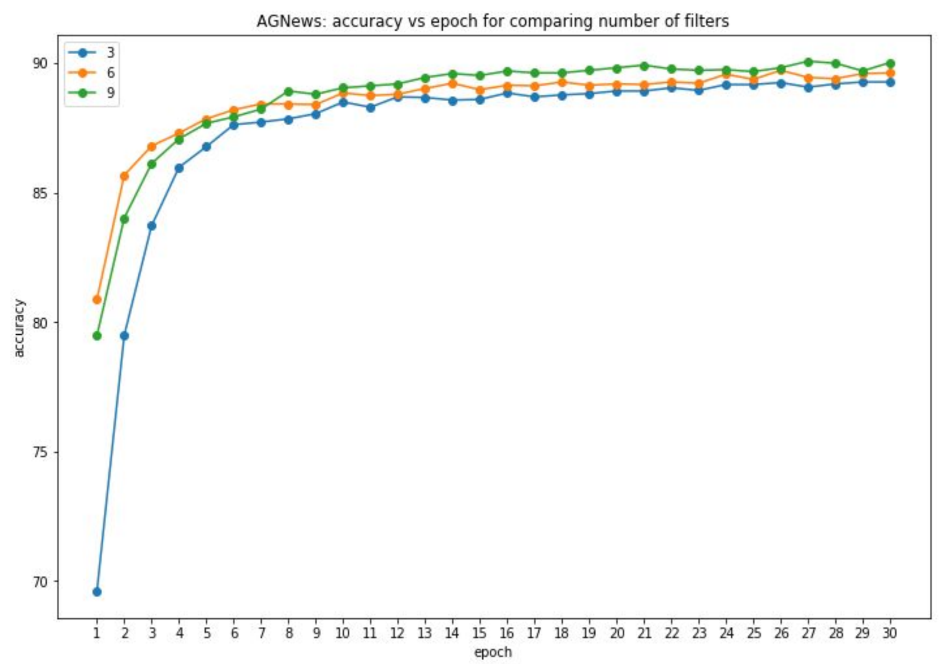

TREC: accuracy vs epoch for comparing number of filters

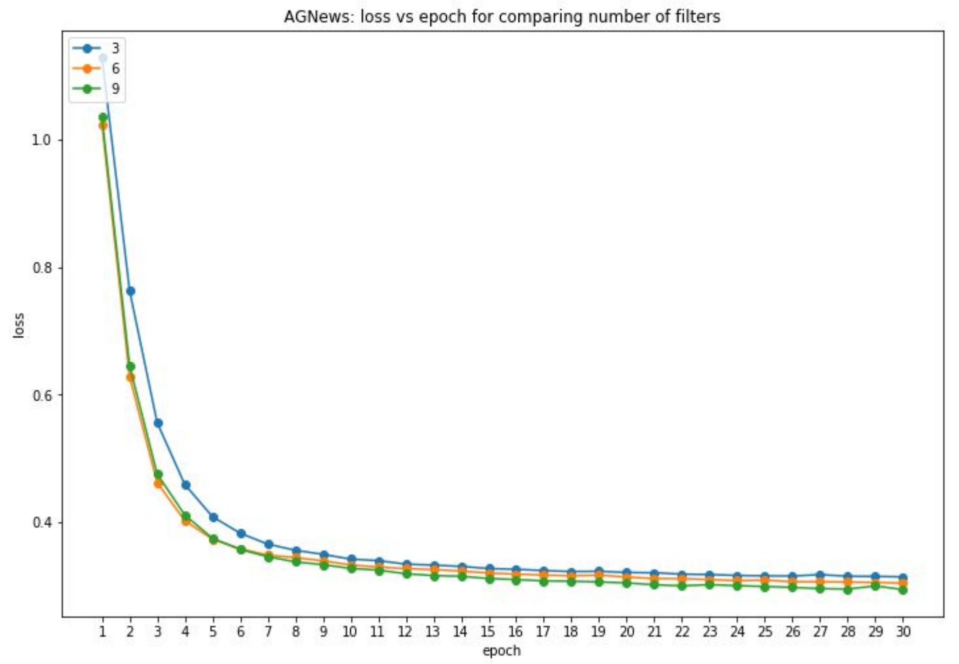

TREC: loss vs epoch for comparing number of filters

Dataset: TREC
Dropout = 0.3
Number of features per word = 200
Filter count: variable
Filter types = 4
Batch size = 100
Epochs = 30
Learning rate = 0.5

Filter count	accuracy
3	70.4
6	71.4
9	68.8

Increasing the number of filters of each type increases the number of features used as input to the dense layer. Thus, increasing the filter count improves accuracy, as seen in the plots when accuracy improves to 71.4% when filter count is increased from 3 to 6. However, a large number of features may lead to overfitting, especially since the dataset is relatively smaller.



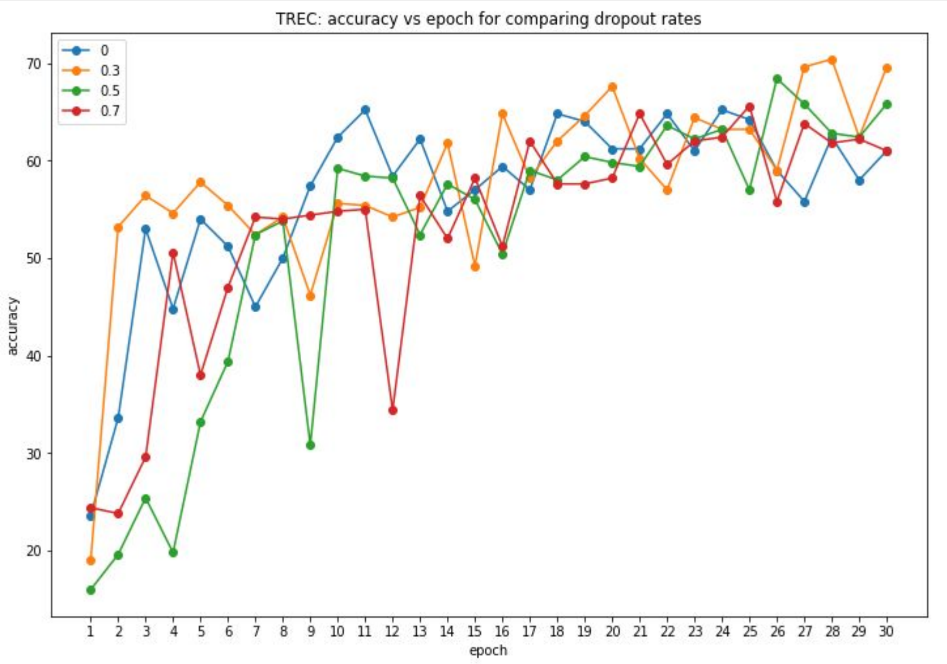


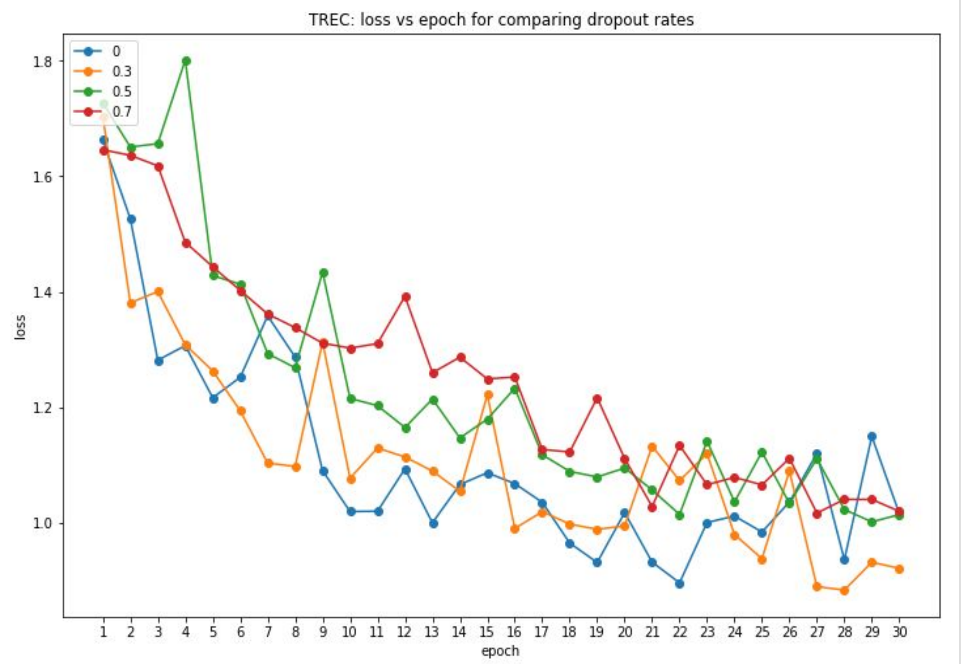
Dataset: AGNews
Dropout = 0.3
Number of features per word = 200
Filter count: variable
Filter types = 4
Batch size = 500
Epochs = 30
Learning rate = 0.05

Filter count	accuracy
3	89.275
6	89.725
9	90.075

The testing accuracy consistently improved from 89.275 to 89.725 to 90.075 when the number of filters was increased from 3 to 6 to 9. Since the AGNews training set is considerably larger with 20000 news articles, with 87 words per article, a filter count of 9 does not cause overfitting, unlike the case of TREC. A larger number of features can adequately represent it without overfitting.

DROPOUT RATE

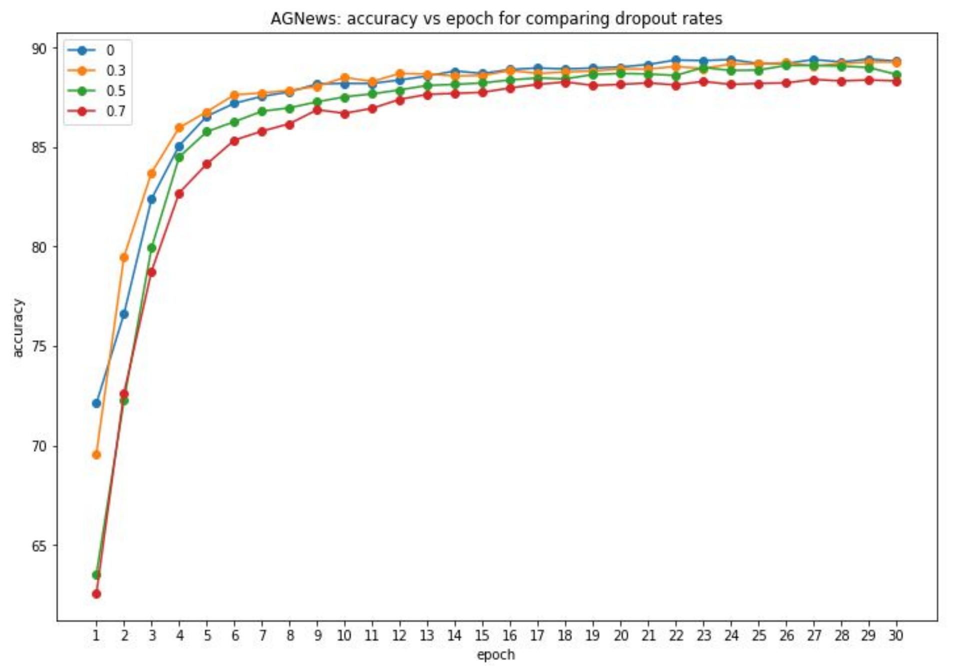


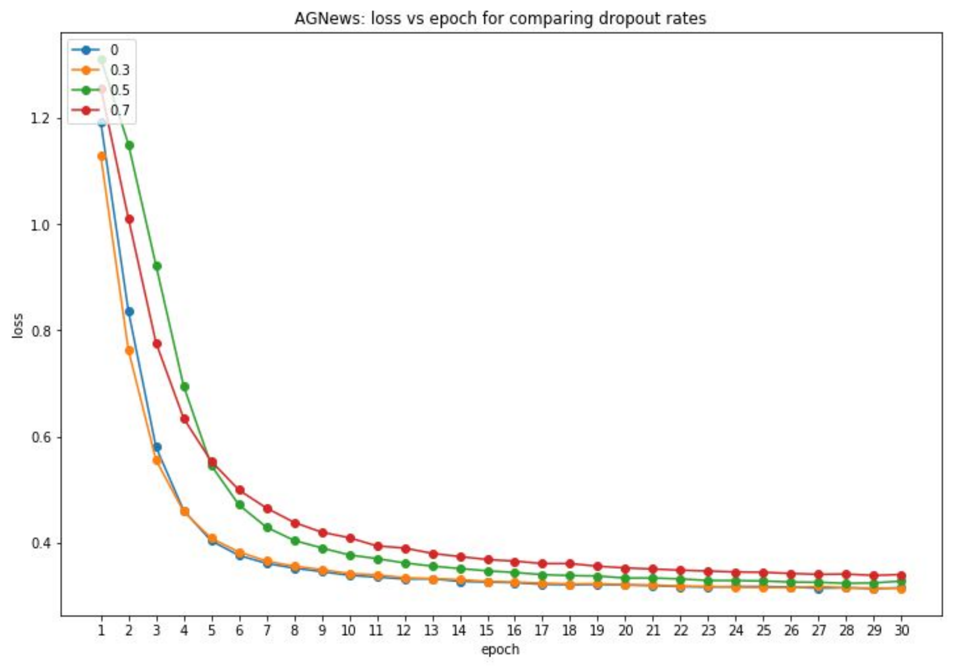


Dataset: TREC
Dropout: variable
Number of features per word = 200
Filter count = 3
Filter types = 4
Batch size = 100
Epochs = 30
Learning rate = 0.5

dropout	accuracy
0	65.2
0.3	70.4
0.5	68.4
0.7	65.6

For TREC, after 30 epochs, the maximum accuracy is seen in the case where dropout rate is set to 0.3. For dropout = 0 (no dropout case), the model tends to overfit the training data and thus leads to low accuracy on testing data. For larger dropouts, underfitting is a likely cause for decreased accuracy and increased loss.



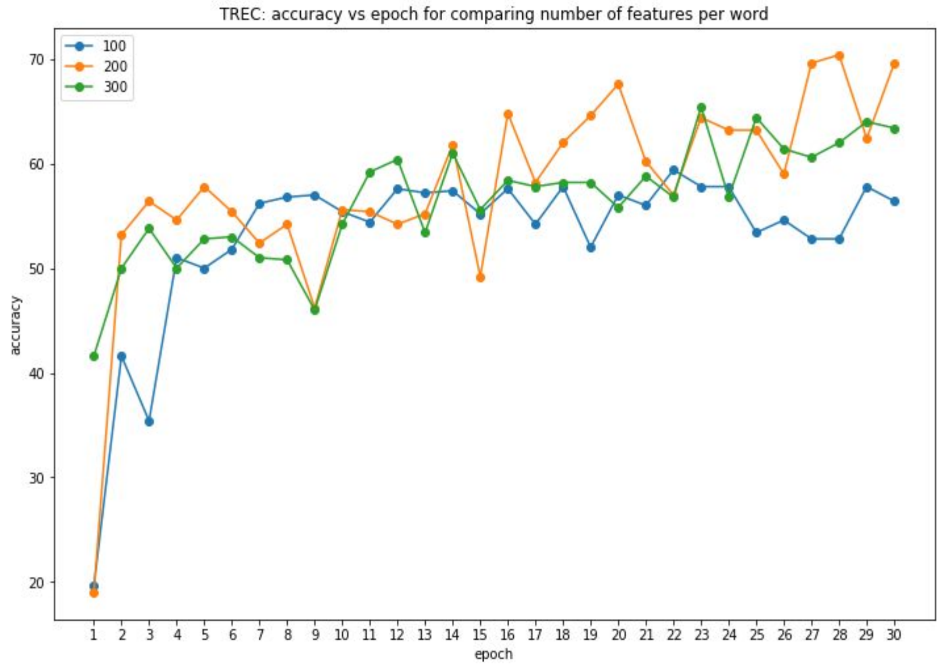


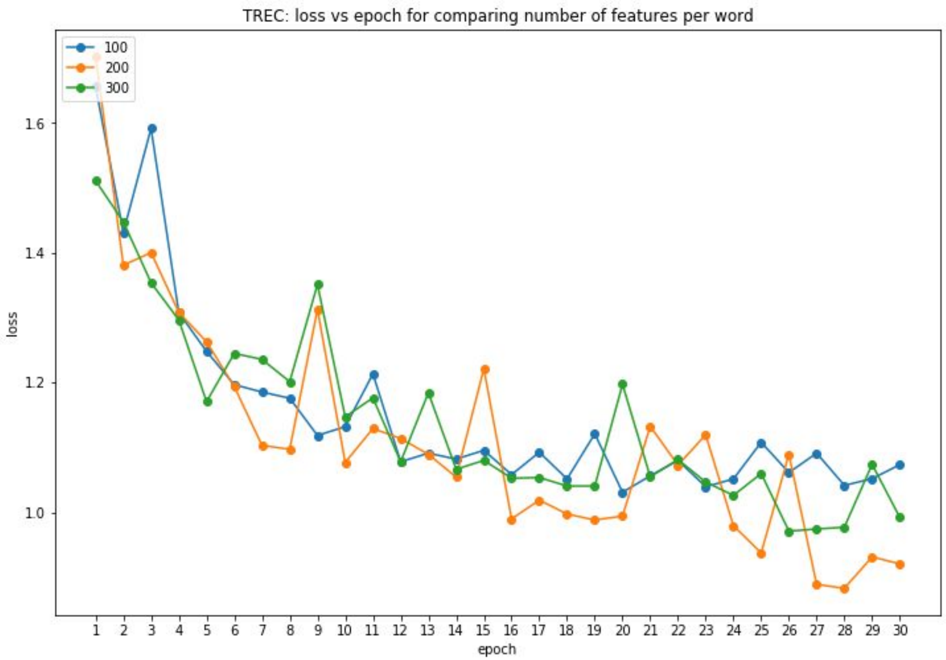
Dataset: AGNews
Dropout: variable
Number of features per word = 200
Filter count = 3
Filter types = 4
Batch size = 500
Epochs = 30
Learning rate = 0.05

dropout	accuracy
0	89.425
0.3	89.275
0.5	89.1
0.7	88.4

The dropout rates for 0 and 0.3 result in satisfactory accuracies. Larger dropout rates lead to underfitting and in turn, reduced accuracy. Considering the large size of the AGNews training set, increasing dropout does not result in significant changes in accuracies.

FEATURES PER WORD

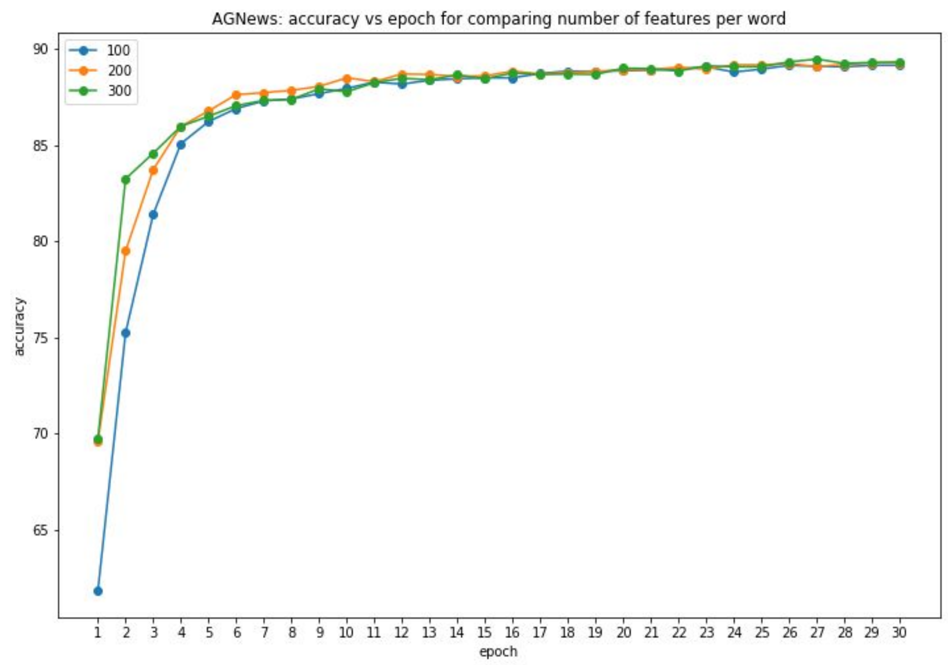


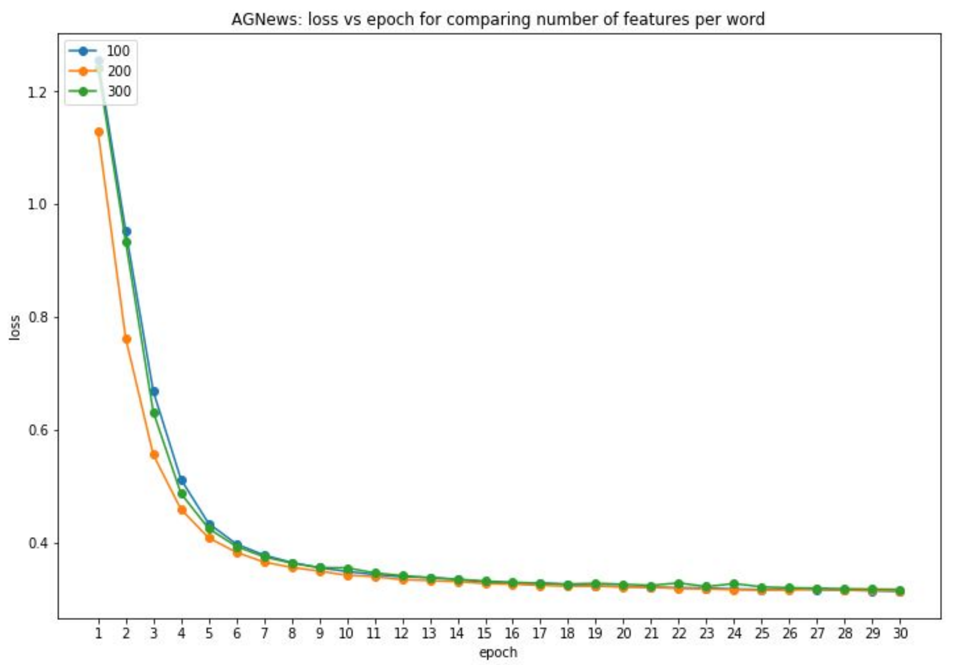


Dataset: TREC
Dropout = 0.3
Number of features per word: variable
Filter count = 3
Filter types = 4
Batch size = 100
Epochs = 30
Learning rate = 0.5

features	accuracy
100	59.4
200	70.4
300	65.4

Increasing the dimensionality of the Word2Vec embedding from 100 to 200 or 300 results is greater accuracy, as seen in the plots. A large variety of texts are required to create the vector embedding. Since TREC has a relatively lesser number of samples with short questions, a larger vector might result in overfitting. A larger vector also increases the training time.


AGNews: accuracy vs epoch for comparing number of features per word


AGNews: loss vs epoch for comparing number of features per word

Dataset: AGNews
Dropout = 0.3
Number of features per word: variable
Filter count = 3
Filter types = 4
Batch size = 500
Epochs = 30
Learning rate = 0.05

features	accuracy
100	89.15
200	89.275
300	89.475

As seen in the plots, increasing the dimension of the word vector results in improved accuracy and less loss. It increases to 89.475 for 300. Since the dataset is big with a large lexicon, using larger Word2Vec vector is effective. Usually, for a vocabulary of ~50000, 200 is a good enough size.

Conclusions

- Selection of learning rate requires consideration of accuracy-time tradeoff. Smaller learning rates will lead to better accuracy but the convergence is slow. Training with very large learning rates may not see any convergence at all. Varying the learning rate as training progresses is a good solution.

- Generally, using more types of filters for more n-gram representations will improve accuracy of categorisation, but at the cost of increased training duration. Usually, a 5-word window is sufficient to represent most logically connected group of words. Inclusion of filters larger than that may lead to overfitting in the case of smaller datasets.

- Increasing count of filters increases the number of feature maps, which leads to greater accuracy, again, at the cost of increased training time. It should be tuned according to the requirements, while also ensuring that the increase in features does not lead to overfitting.

- The dropout rate selection depends on the size of dataset. A smaller dataset or a dataset with large variety of examples require no dropout as the features are not memorized by the model. A larger dataset or one with frequently occurring similar examples should be tested with different dropout rates. A very high dropout might lead to underfitting.

- Accuracy of prediction improves with number of features per word. However for a vocabulary upto ~35000 words, size=100 works just as well and for ~70000 words, size=200 does.

References

[1] Kim, Yoon. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 10.3115/v1/D14-1181.

[2] Glorot, Xavier & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research - Proceedings Track. 9. 249-256.

[3] Hinton, Geoffrey E.; Srivastava, Nitish; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan R. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". arXiv:1207.0580 [cs.NE].

[4] Mahdi Rezaeinia, Seyed & Ghodsi, Ali & Rahmani, Rouhollah. (2017). Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis.

[5] http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[6] Zhang, Lei & Wang, Shuai & Liu, Bing. (2018). Deep Learning for Sentiment Analysis : A Survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 10.1002/widm.1253.

[7] https://radimrehurek.com/gensim/models/word2vec.html

[8] https://www.tensorflow.org/tutorials/estimators/cnn

[9] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html

[10] https://trec.nist.gov/data/qa.html

[11] https://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html