Reproducing a Convolutional Neural Network for Sentence Classification

Alison Caulfield
McGill University
alison.caulfield@mail.mcgill.ca

Chloe Grosdidier
McGill University
chloe.grosdidier@mail.mcgill.ca

Michael Vaquier
McGill University
michael.vaquier@mail.mcgill.ca

***Abstract*— In this paper, we reproduce a simple Convolutional Neural Network, which serves as a baseline for the sentence classification task in a project by researcher Yoon Kim [1]. This model, CNN-rand, is trained using randomized word embeddings. We also implement two other models from Kim's paper, CNN-static and CNN-non-static, which train on top of the pretrained word embeddings word2vec. We test the accuracy of our models with two of the datasets used by Kim. We successfully reproduce the published results from Kim's paper, and obtain accuracies within 0.45% of Kim's baselines. Furthermore, we perform extensive hyperparameter fine-tuning to understand the effects of each hyperparameter, and improve upon the accuracies obtained by the baseline model. Some of the best results from these tests improved the baseline's accuracy by at most 2.02%. We also show that a simpler model that only uses one convolution, rather than 3 parallel convolutions, can achieve similar results. We use an incremental fine-tuning strategy to find a combination of hyperparameters that work well together. Furthermore, we explore the effects of using a uniform distribution to initialize the randomized word embeddings, as opposed to using a uniform distribution. Finally, we implement modifications to the architecture of the model. Although the modified model architecture achieves a strong accuracy, it is not able to surpass the accuracy of the baseline.**

I. INTRODUCTION

A. Background

Sentence classification is a key component of natural language processing. Convolutional Neural Networks (CNN) are commonly used in image classification tasks. However, with the use of word embeddings, they can also be applied to sentence classification, due to their ability to understand and interpret local relations of hierarchical and temporal structures. [2] They can also extract important features in a sentence with their convolutional layers.

Word embeddings are vectors that map words to real values. They can be used to capture the semantical and morphological information of a word. Word2Vec is a pre-trained set of word vectors, which were trained using an unsupervised neural model by Mikovlov et al. [7] They have a dimensionality of 300 and have been used by numerous researchers to accomplish sentence classification tasks.

B. The Task

Researcher Kim examines how various simple CNN models perform on the sentence classification task, when trained with pretrained word vectors. [1] Kim develops a CNN model, which uses one convolutional layer with multiple kernel sizes. His baseline model (CNN-rand) uses randomly initialized word embeddings, which have a dimensionality of 300, that are fine-tuned during training. He then trains a model using the pretrained word2vec embeddings (CNN-static), which keeps the word vectors static throughout training and changes only the other parameters. He also trains a model (CNN-non-static), which uses the word2vec embeddings and fine-tunes them throughout training. His final model (CNN-multichannel) explores the use of multiple channels of word embeddings. Kim concludes that a simple CNN model, trained using pretrained word vectors, can perform well on a variety of datasets. Furthermore, Kim finds it is not always the best strategy to use multiple channels. The accuracy of Kim's models are tested using 7 different datasets. For 4 out of 7 of these models, Kim is able to achieve state-of-the-art performance at the time.

In this paper, we implement three of the four CNN models used in Kim's paper: CNN-rand, CNN-static and CNN-non-static. We reproduce the results obtained by Kim's models. We perform rigorous tests on each of their hyperparameters. We compare the performance of the models when using different distributions to initialize the randomized word vectors. We then explore the effects of modifying the architecture of the model, by performing a cross-feature convolution before max pooling. We test our models using 2 of the 7 datasets used by Kim, one which Kim's model beats the state-of-the-art performance and one for which the it does not.

II. RELATED WORK

The model proposed in Kim's paper has been expanded and modified by various research teams. In particular, Yin and Schutze implemented MVCNN, which is a CNN for sentence classification built using ideas from Kim's paper. [3] Like Kim's model, MVCNN uses pretrained word embeddings, implements multiple kernel sizes in each convolutional layer and uses multi-channels. Their proposed architecture differs from Kim's, by it's implementation of dynamic k-max pooling (rather than global max pooling) and it's use of multiple convolutional layers. They discuss an issue with using global max pooling, which inspired our modifications to Kim's model's architecture. Yin and Schutze test the accuracy of their model using the Subjectivity Dataset, which is also used in both Kim's paper and in this paper. MVCNN achieves an accuracy of 93.9% on this dataset.

Kim's baseline has clearly played an important role in the development neural networks for sentence classification techniques. It has served as a baseline for numerous other

research projects, including recent work done by Thomson on neural encoders, which encode graph representations of natural language sequence encoding networks [5], and Tiang, Qin and Liu's work on document modeling for sentiment classification with gated recurrent neural networks. [6]

Simple baselines, which do not use neural networks, have also been explored for sentiment classification. Researchers Wang and Manning implement a variety of Naive Bayes and Support Vector Machine models, with a focus on fine-tuning their parameters. [4] They also test the accuracy of their model using the Subjectivity Dataset, and found that some of their simple models have state-of-the-art performance. They show that using bi-grams with Multinomial Naive Bayes achieves an accuracy of 93.6%, on the subjectivity data.

III. DATASETS AND MODEL ARCHITECTURE

A. Datasets

We used two of the benchmark datasets that were used in Kim's paper: the movie review dataset and the subjectivity dataset. [1] Kim's model was able to beat the state-of-the-art methods at the time for the movie reivew dataset, but was not able to do so for the subjectivity dataset. We chose these datasets, because they both have a fairly similar size, which is around 10000 samples each and an average sentence length of around 21 words.

- The movie review dataset (**MR**) is a binary dataset, where movie reviews made on the IMDb website are classified into the two categories: negative reviews and positive reviews. [8] The MR dataset has 10662 samples, 5331 of which are positive reviews and 5331 of which are negative reviews.
- The subjectivity dataset (**SUBJ**) is also a binary dataset, where snippets from movie reviews from rotten tomatoes and plot summaries from IMDb are classified into the two categories: subjective and objective. [9] The SUBJ dataset consists of 10000 samples, which are also evenly split into the two categories. 5000 of the reviews are subjective and 5000 of the reviews are objective.

B. Model Architecture

We implemented the baseline model from Kim's paper, which is a single layered convolutional neural network (CNN). [1] We use the *keras* library to implement the model. The model starts with an embedding layer followed by parallel convolution layers, each with a different kernel size and a global max pooling layer. Then, the outputs of the three branches are aggregated, dropout is applied, and then fed into a fully connected layer. A softmax output is used. Figure 1 shows the model's architecture, and was adapted from [1].

IV. PROPOSED APPROACH

A. Hyperparameter Fine-Tuning

In his paper, Kim conducted all experiments using the same set of hyperparameters, which were selected via grid-search against one dataset. [1] The hyperparameters chosen for Kim's model are as follows:

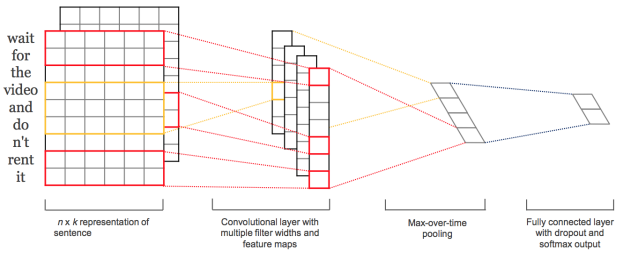

Fig. 1. Baseline Model Architecture [1]

- Activiation Function: rectified linear units (ReLU)
- Kernel Sizes: three one-dimensional kernels with window sizes of 3, 4, and 5 and 100 feature maps each
- Dropout: dropout rate of 0.5
- L2 Regularization: l_2 constraint of 3 (Note that for the implementation of our model, we used the *keras* library, which only permits modifying the λ variable of the L2-regularisation. We found that $\lambda = 0.001$ achieved results that were closed to the reported results of Kim's models').
- Minibatch Size: 50
- Optimizer: Adadelta

In order to rigorously test the model's hyperparameters, we conducted numerous grid searches on both datasets, for each of CNN-rand, CNN-static and CNN-non-static. For each hyperparameter, we iteratively modified it, ran the model for 50 epochs, then reported the results. Only one hyperparameter was modified at a time and all other hyperparameters were held constant at the value used in Kim's baseline (listed above) throughout the tests. By running grid search, we attained accuracies that push the limits of what is possible with the model's architecture. We also were able to understand the effects of each individual hyperparameter on it's performance.

Specifically, the hyperparameters that we ran grid search on include the size of the feature maps, the number parallel convolutions used, the size of each kernel in a convolution, the strength of L2-regularisation, the optimizer used and the dropout rate.

B. Incremental Hyperparameter Fine-Tuning

Due to dependencies between certain hyperparameters, we cannot find the best set of hyperparameters by simply combining the results of the individual hyperparameter fine-tuning. [10] Optimal parameters need to be chosen based on the values assigned to the other parameters. Therefore, we use an incremental hyperparameter fine-tuning strategy, in order to find the set of hyperparameters with the highest accuracy. To do this, we use a similar strategy to the grid search used to test the effects of each hyperparameter, as described in Section III A. However, once we find the value of a hyperparameter with the highest accuracy, we fix that

hyperparameter to that value and continue to explore all other hyperparameters, each time using the best values of the previously explored parameters. Due to limitations in computing resources and time, we only explore incremental fine-tuning with one order of hyperparameters, which is as follows: number of parallel convolutions and kernel size, feature map size, regularization strength, dropout rate and optimizer used.

C. Randomized Word Vectors

Kim uses a uniform distribution to generate randomized word vectors. [1] These randomized word vectors are used to train CNN-rand. For CNN-static and CNN-non-static, randomized word vectors are used for words, which do not already have an embedding in word2vec. We compare this strategy with one that uses a normal distribution to initialize the randomized vectors.

D. Modifications to the Model Architecture

Kim's model uses global max pooling for each filter map to get the value of the largest activation, which is then passed to the dense layer. This discards all the temporal information regarding where in the sentence the high activation for a particular filter occurs. [3] Through analysis, we found that the baseline model misclassifies sentences that require a deeper understanding of the relationship between words that are far apart in a sentence. For example, the model incorrectly classified *"a great idea becomes a not great movie"*, which comes from the movie review dataset.

Figure 2 illustrates why Kim's model potentially misclassifies this sentence. This figure assumes that the model uses one kernel of size 3 with 2 feature maps and supposes that *kernel1*, and *kernel2* have been trained to find sequences of words similar to *"great idea becomes"* and, *"not great movie"* respectively. Both of these kernels will have a high activation, when passing over their respective set of words in the sentence. When global max pooling occurs, the temporal information is eliminated. These two strong activations are passed to the fully connected layer. The strong activation of the kernel associated with the words *"great idea becomes"* tends the output towards a positive review, whereas the other activation of the kernel associated with the words *"not great movie"* does the opposite. This makes the output very unstable.

We make modifications to the baseline model, in order to hopefully address this loss of temporal information caused by global max pooling. In [3], the researchers use k-max pooling in their model MVCNN, as discussed in Section II, to address this issue. However, we come up with a different strategy to tackle this issue. Our approach looks at the relationship between activations across different feature maps. The modifications we make to the baseline model allow it to find patterns of filters that lead to strong activations temporally close to each other, by performing a 2D convolution across all feature maps simultaneously.

Figure 3 illustrates the modifications we make to the architecture. The first two steps of the modified model are the same as Kim's architecture. The modified model starts with an embedding layer, which is followed by a one-dimensional convolution that gets different activations for a particular N-gram. The next layer differs from Kim's design. Rather than performing global max pooling, our model reshapes the tensor, at this point in the model, by adding an extra dimension to it, so that a 2D convolution can be performed. The kernel of this 2D convolution has a shape of (x, number of feature maps) where x is a hyperparameter. This convolution can capture the relationship between different filter activations that happen up to x time steps apart. The output of this layer is then pooled and fed into a dense layer, as in Kim's architecture.

Our modified architecture can classify sentences like *"a*

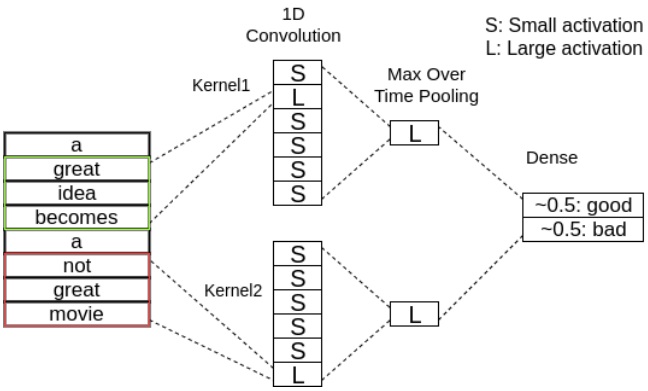

Fig. 2. Example where Kim's model struggles

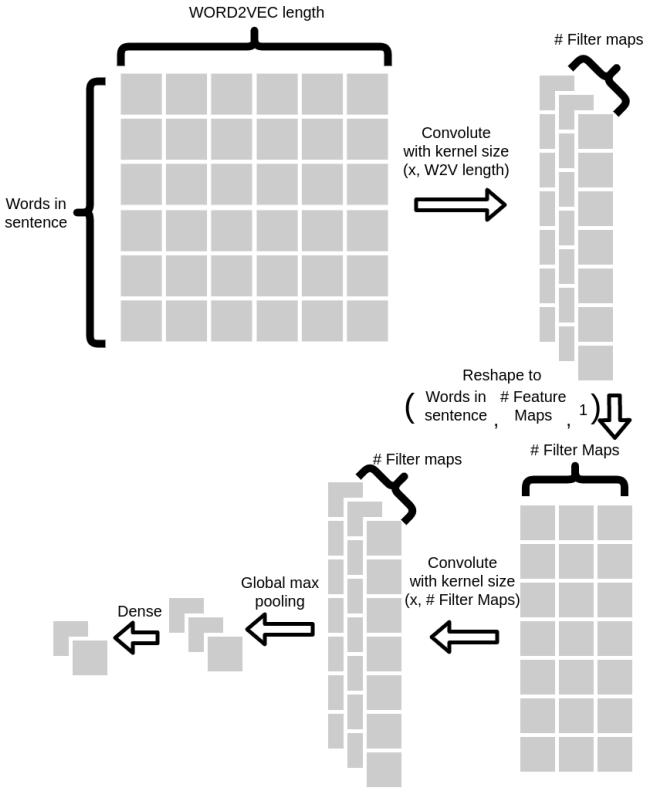

Fig. 3. Modified Model Architecture

great idea becomes a not great movie" more accurately. Figure 4 walks through this example using the modified architecture. The first step is the same as with Kim's model. Both kernels create the same feature maps with high activations, where their respective words are. After reshaping the tensor to a 2D image where each column is a feature map, a new 2D convolution is performed across the feature maps to capture their relationships. The cross feature convolution is able to capture that the high activations of *Kernel1* and *Kernel2* happen close to each other temporally and produce a high activation from that. This modified architecture is able to decrease ambiguity for cases where there is a relationship between different filter activations.

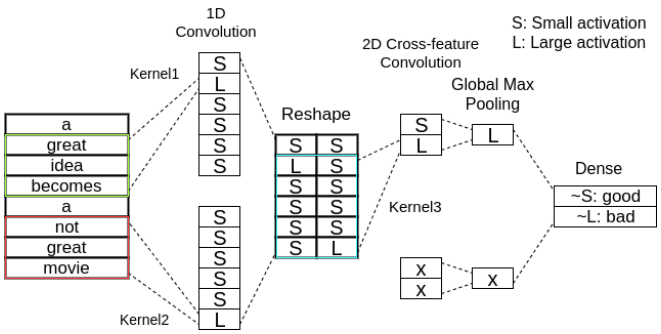

Fig. 4. Modified Architecture that addresses the issue

V. RESULTS

To save space, we use abbreviations in all tables in this section. Model CNN-rand is written as **rand**, CNN-static becomes **w2v_s** and CNN-non-static is written as **w2v_d**. All accuracy values in the tables are expressed as percentages (%).

A. Implementing the Baseline

Our implementation of the baseline achieved very similar results to those achieved in Kim's paper, as can be observed in Table I. This baseline model had identical architecture and similar hyperparameters to those described in Kim's paper. The accuracy that we achieved differs from Kim's reported accuracy by at most 0.45%. These slight differences in accuracy can be explained by the fact that the model optimizes a nonconvex function.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
Our Model	89.35	93.06	92.95	76.41	80.92	80.93
Kim's Model	89.6	93.0	93.4	76.1	81.0	81.1

TABLE I
COMPARISON OF THE ACCURACIES ACHIEVED ON OUR BASELINE MODELS WITH THE ACCURACIES OF KIM'S MODELS

B. Error Analysis

We performed error analysis for the Movie Review dataset to get a deeper understanding of the kind of text samples that our baseline model incorrectly classifies. In general, the model seemed to make three types of mistakes.

- The model was unable to pick up on sarcastic comments, such as *"the humor is hinged on the belief that knees in the crotch , elbows in the face and spit in the eye are inherently funny"*, which was classified as a positive review. This kind of sentence requires a more complex understanding of the tone, to understand that the sentence is not meant to be literal.
- The model also struggled to correctly classify sentences with complex relationships between far apart words. An example of such a sentence would be *"only two fifths of a satisfying movie experience"*. Because the model performs 1D convolutions for specific N-grams, it has a strong activation for the sequence of words *"satisfying movie experience"*, which indicates a positive review. However, it is not able to capture that *"only two fifths"* is referring to the *"satisfying movie experience"*, which actually makes it a bad review. Many other examples from this class of error were discovered, which partially inspired our modification to Kim's architecture discussed in Section IV D.
- Comments that do not make sense without more context were also commonly misclassified. These sentences are also difficult to classify for a human. For example, the review *"norton is magnetic as graham"* is very tricky to classify without knowing who/what *norton* and *graham* are. Another example of this would be the review that reads *"full of surprises"*, which could be both interpreted as good or bad depending on the context.

This error analysis enabled us to see where Kim's baseline model was lacking. We attempted to address one of these types of errors, by modifying the model architecture as described in Section IV D.

C. Hyperparameter Fine-Tuning

In order to improve the baseline, we attempted to find the best hyperparameter values by fine-tuning each hyperparameter individually. To ensure the experiments and comparisons were as fair as possible, only the one hyperparameter being examined was changed, while the others remained constant. The constant values used were the same hyperparameter values as stated in Kim's paper (described in Section IV A). We recorded the results of the accuracies achieved for each experiment in Tables II to VI. The highest accuracy we achieved for each dataset and model is bolded in each table.

1) Feature maps: We tested the effects of using a different number of feature maps. As seen in Table II, we managed to improve on all baselines for all datasets, and were able to achieve an accuracy that is higher than the reported accuracies of Kim's model, which used 100 feature maps. For instance, using 400 feature maps resulted in a 94.05% accuracy for the subjectivity dataset with CNN-static, which is an increase of 1.05% from Kim's model. It can be observed that using 400 feature maps provided the best results over all 6 combinations of models and datasets.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
10	89.55	92.1	92.45	73.93	78.72	80.5
25	89.45	91.95	93.3	76.09	**81.2**	80.83
50	90.2	92.15	**93.65**	75.43	80.73	**81.67**
100	89.35	93.06	92.95	76.41	80.92	80.93
200	88.65	93.45	93.6	74.96	79.37	80.97
400	**91.2**	**94.05**	93.15	**76.56**	79.98	79.98
500	89.25	93.15	93.2	76.37	81.06	80.92

TABLE II

ACCURACIES OBTAINED WHEN VARYING FEATURE MAPS

2) Number of Convolutions and Kernel Sizes: We varied the number of parallel convolutions used as well the kernel size in each. Table III shows the results of these tests. The best result achieved for each number of convolutions is in bold. With only one convolution, a kernel size of 1 performed the best for randomly initialized word vectors (CNN-rand). For the CNN-non-static model, it seemed that using a bigger kernel size, such as 3 or 5, tended to increase accuracy. A similar trend can be observed as the amount of parallel convolutions increases, where CNN-non-static performs slightly better with larger kernel sizes, in general.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
[1]	**90.25**	93.03	92.02	**76.74**	**81.03**	79.79
[2]	89.8	**93.38**	92.72	75.24	80.57	80.455
[3]	89.33	92.62	92.73	75.6	80.65	**80.99**
[4]	89.58	93.02	92.81	75.52	79.67	80.84
[5]	88.95	92.7	**93.08**	75.09	80.49	80.79
[6]	88.85	92.33	92.4	76.08	80.29	80.59
[7]	89.47	92.47	92.13	75.26	80.1	80.59
[1,2]	89.67	93.33	93.28	76.52	80.57	80.48
[2,3]	89.82	**93.63**	**93.33**	**76.65**	80.79	**81.32**
[3,4]	**89.95**	92.98	**93.33**	76.4	**80.93**	80.93
[4,5]	89.33	92.87	92.75	76.63	80.53	81.28
[5,6]	89.68	92.755	92.9	75.79	80.2	80.99
[6,7]	89	92.52	92.68	75.7	80.23	81.26
[1,2,3]	89.98	**93.25**	93.32	75.85	**81.09**	80.45
[2,3,4]	**90.15**	92.68	93.33	**76.57**	80.81	**81.13**
[3,4,5]	89.95	92.9	**93.38**	75.56	80.51	80.68
[4,5,6]	89.17	93.02	92.75	76.34	79.84	80.84
[5,6,7]	89.9	92.35	92.23	**76.57**	80.99	80.79
[1,2,3,4]	**89.87**	92.88	**93.37**	76.56	80.78	81.09
[2,3,4,5]	89.65	92.97	93.15	76.85	**80.99**	80.74
[3,4,5,6]	89.58	**92.98**	93.13	**77.22**	80.87	**81.31**
[4,5,6,7]	89.1	92.5	93.33	75.87	80.46	80.84

TABLE III

ACCURACIES OBTAINED WHEN VARYING KERNEL SIZES

Overall, the best scoring accuracy between different amounts of parallel convolutions did not change by much. Therefore, a much simpler version of Kim's baseline, which only uses one convolution layer rather than three parallel layers, could achieve very similar results.

Furthermore, the improvements in accuracy from our baseline were very minimal, in general, and were all less than 1%. The highest improvement in accuracy was for CNN-rand for the subjectivity dataset, which achieved 90.25%, which is 0.9% more than the 89.35% achieved on our baseline.

3) Regularization strength: Table IV shows the effects of changing the regularization strength of the models. Online documentation suggested λ be set to a value ranging between 0 and 0.01 [11]. Therefore, we decided to run grid search over parameter values within this range. We managed to obtain the highest accuracy thus far for the subjectivity dataset with CNN-non-static of 93.85%, with a regularization strength of 0.0001.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
0	89.3	91.7	92.85	76.23	80.59	81.52
0.0001	88.3	92.75	**93.85**	75.9	81.34	81.2
0.0003	89.65	**93.15**	93.35	75.95	79.65	80.45
0.0005	**90.25**	91.8	93.15	75.34	80.26	80.68
0.001	89.35	93.06	92.95	76.41	80.92	80.93
0.0015	89.2	92.95	93.65	**77.03**	80.5	**81.58**
0.002	88.85	91.85	92.65	75.76	79.51	81.06
0.005	89.85	92.15	92.6	75.34	81.53	80.83
0.01	89.25	92	91.95	74.64	**81.62**	81.43

TABLE IV

ACCURACIES OBTAINED WHEN VARYING REGULARIZATION STRENGTH

4) Dropout rate: We varied the dropout rate between 0 and 0.9 and reported the results in Table V. The best accuracies for each dataset were all obtained with a dropout rate above 0.2. It seems that the model performed better when the dropout rate was high. The best accuracy out of all experiments for the movie review dataset with CNN-non-static was 82.42%, which was achieved with a dropout rate of 0.4. We also obtained the best accuracy for the movie review with CNN-static, at 82.04%, which is more than a 1% improvement on both our and Kim's baseline. Generally, the accuracies obtained from performing grid search on the dropout rate outperformed those from performing grid search on the regularization strength. We therefore can conclude that the value chosen for the dropout rate hyperparameter has a more significant impact on the model's performance than the one chosen for regularization strength.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
0	89.1	91.55	92.9	76.09	79.56	81.62
0.1	89.9	92.7	93.05	75.62	81.15	81.25
0.2	90.1	92.75	93.1	74.68	79.98	80.54
0.3	90.15	92.85	93.1	**77.64**	80.5	82
0.4	89.65	92.05	93.5	74.5	80.08	**82.42**
0.5	89.35	93.06	92.95	76.41	80.92	80.93
0.6	**90.5**	93.05	93.2	76.23	**82.04**	80.4
0.7	88.95	**93.25**	93.15	76.28	80.45	81.39
0.8	88.75	93.05	**93.85**	77.17	80.87	81.01
0.9	88.85	92.4	92.7	76.98	80.59	81.48

TABLE V

ACCURACIES OBTAINED WHEN VARYING DROPOUT RATE

5) Optimizer: The accuracies deviated the most when we ran grid search on the optimizer used, which can be observed in Table VI. For example, the movie review dataset had an accuracy 78.43%, for CNN-rand, with the rmsprop optimizer, but an accuracy of 67.98% with the sgd optimizer. We also

achieved the best accuracy out of all the trials for the CNN-rand with the subjectivity dataset, which was 92% using adam as the optimizer. Kim's baseline uses the adadelta optimizer. For our models, this optimizer did not have have the highest accuracy for any of the configurations.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
adadelta	89.35	93.06	92.95	76.41	80.92	80.93
adam	**92**	93.2	93.1	76.42	81.01	80.59
rmsprop	91.8	92.65	92.55	**78.43**	81.43	79.89
sgd	83.55	91.75	92	67.98	77.97	78.57
adagrad	91.4	92.75	93.15	77.07	79.89	79.65
adamax	90.2	91.8	92.5	76.84	**81.67**	**81.62**
nadam	89.8	**93.55**	**93.55**	76.51	80.03	79.89

TABLE VI
ACCURACIES OBTAINED WHEN VARYING OPTIMIZERS

Overall, the best accuracy achieved during individual hyperparameter fine-tuning for the subjectivity dataset was 92% for CNN-rand (with the adam optimizer), 94.05% for CNN-static (with 400 feature maps), and 93.85% for CNN-non-static (with regularization of 0.0001). For the movie review dataset, the highest accuracies were 78.43% for CNN-rand (with the rmsprop optimizer), 82.04% for CNN-static (with a dropout of 0.6), and 82.42% for CNN-non-static (with a dropout of 0.4). Through each hyperparameter search, we were able to consistently obtain accuracies well above Kim's baseline, which proves that fine-tuning the baseline model is a reasonable way to improve accuracy. It is clear to see that modifying even just one hyperparameter can have drastic effect on the resulting accuracy. Furthermore, all of our models' best performances were obtained from a different hyperparameter experiment. This shows that the optimal selection of hyperparameters is linked to the dataset and classification task being handled.

Furthermore, we found that Kim's models could be simplified, by using less parallel convolutional layers, and still achieve very similar results for all datasets.

D. Incremental Hyperparameter Fine-Tuning

The obtained results for our incremental hyperparameter fine-tuning can be seen in Table VII. All results are improvements on the baseline, though they did not beat the best accuracies for each dataset obtained by individually fine-tuning the hyperparameters. It can also be noted that the best hyperparameter values obtained with the incremental strategy differs from the best hyperparameter values found during fine-tuning. For example, 400 feature maps generally had the best performance during individual hyperparameter fine-tuning, whereas 100 feature maps performed the best during incremental fine-tuning. These discrepancies are due to the fact that there exists certain dependencies between the parameters [10]. Furthermore, due to limits in computational resources and time, we only had the chance to run incremental hyperparameter fine-tuning with one order of hyperparameters. If we had changed the order in which the parameters were chosen and set, we may have ended up with a drastically different combination of parameter values and accuracies. Since the first hyperparameters selected were the kernel sizes, they are the only parameter values that match the individual hyperparameter fine-tuning results. The values obtained through our incremental hyperparameter fine-tuning can therefore be considered as local maximums. Ideally, if we had the necessary computing resources, we would have run multiple incremental searches, each with different sequence of parameters to explore, and compared their performance and outcomes.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
Accuracy	91.88	93.97	93.48	78.38	81.57	81.54
Feature maps	500	100	100	50	100	100
Kernel Size	[1]	[2, 3]	[3, 4, 5]	[3, 4, 5, 6]	[1, 2, 3]	[2, 3]
Reg. Strength	0.0001	0.001	0.001	0.001	0.0005	0.001
Dropout rate	0	0.5	0.5	0.3	0.5	0.5
Optimizer	adadelta	adadelta	rmsprop	rmsprop	adadelta	adadelta

TABLE VII
HYPER-PARAMETER VALUES AND THEIR ASSOCIATED ACCURACY AFTER INCREMENTAL TUNING

E. Normal vs. Uniform Distribution

We compared the effects of using a uniform distribution to initialize the randomized word vectors, with those of using a normal distribution. To do this, we used the hyperparameters for Kim's baseline model (as described in Section IV A). The results of this experiment are in Table VIII.

For CNN-rand, we found that initializing the word vectors using a uniform distribution resulted in a higher accuracy, than using a normal distribution to initialize the randomized word vectors. Furthermore, we noticed that there was not a significant difference between using the two different distributions for the models that use word2vec. This is due to the fact that randomized vectors are only used in these models, when a word is not present in the word2vec embeddings, which does not occur very frequently.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
Normal	88.7	92.92	92.93	76.07	80.52	81.11
Uniform	89.35	93.06	92.95	76.41	80.92	80.93

TABLE VIII
ACCURACIES OBTAINED WHEN DISTRIBUTION OF RANDOMIZED VECTORS WAS MODIFIED

F. Modifications to the Model Architecture

Table IX compares the accuracies obtained from our baseline model with those obtained from our modified model. Both models use the hyperparameters described in Kim's paper. Unfortunately, our modification led to a decline in performance for both datasets. While our modifications addressed the loss of temporal information during global max pooling, it also likely introduced different problems. For

example, we noticed that the convergence of this modified model was much faster than the baseline model, which could indicate that it was slightly overfitting due to the extra degrees of freedom that were added.

Dataset	SUBJ			MR		
	rand	**w2v_s**	**w2v_d**	**rand**	**w2v_s**	**w2v_d**
Baseline	89.35	93.06	92.95	76.41	80.92	80.93
Modified	88.55	91.43	92.05	73.86	78.93	80.29

TABLE IX

ACCURACIES OBTAINED WITH MODIFIED ARCHITECTURE

VI. DISCUSSION AND CONCLUSION

In this paper, we successfully reproduced the baseline model, a single-layered CNN (CNN-rand) for sentence classification, used in Kim's research. [1] We also implemented two other models discussed in Kim's paper, CNN-static and CNN-non-static, which train ontop of the pretrained word embeddings word2vec. We assessed the performance of our models, with two of the seven datasets used in Kim's model, and achieved very similar results to those published by Kim. Then, we analyzed the types of errors made by the model, in order to understand where the model was lacking. We performed extensive grid search on the hyperparameters used in the models and found that each dataset and model has a different set of optimal hyperparameters. By fine-tuning our models, we were able to achieve accuracies higher than those reported by Kim. Furthermore, we found that using less parallel convolutional layers achieves very similar results. Therefore, we conclude that a simpler model with only one convolution could serve as a baseline with less computational complexity. Furthermore, we used incremental hyperparameter fine-tuning to find a set of hyperparameters that perform well together. We also experimented with using different distributions to generate the randomized word vectors and found that the uniform distribution performed the best overall, which is what Kim used in his paper. Finally, we modified the architecture of the baseline model, in order to address an issue with global max pooling. However, our modifications likely introduced errors and we weren't able to beat the accuracies achieved by the baseline models.

VII. FUTURE WORK

The incremental hyperparameter fine-tuning approach discussed in Section V D was able to deliver results that are higher than the baselines. However, because we only considered hyperparameters in one order, it is very likely that the best selection of parameters we obtained can still be improved. If we had more time and computational resources, a proper grid search could be performed by considering all possible orders of hyperparameters. To save computational time for this task, we could also eliminate values for which we already know that they do not perform very well (for example the sgd optimizer).

As discussed in Section V F, the modifications we made to Kim's model unfortunately were not able to outperform the original model. The modifications were meant to fix a targeted class of errors, but introduced other mistakes in the process. Further investigations could be performed here to identify what new mistakes were introduced by the changes. We could implement mitigation strategies to limit the negative side effects introduced by the modifications. It is also possible that the hyperparameters need to be fine-tuned to fit this new model.

REFERENCES

[1] Y. Kim, "Convolutional Neural Networks for Sentence Classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014.

[2] R. Collobert and J. Weston, "A unified architecture for natural language processing," Proceedings of the 25th international conference on Machine learning - ICML 08, 2008.

[3] W. Yin and H. Schütze, "Multichannel Variable-Size Convolution for Sentence Classification," Proceedings of the Nineteenth Conference on Computational Natural Language Learning, 2015.

[4] Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, pages 90–94. Association for Computational Linguistics.

[5] Sam Thompson. "Encoding and Decoding Graph Representations of Natural Language". Ph.D thesis, Carnegie Mellon University, March 2019.

[6] D. Tang, B. Qin, and T. Liu, "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," In Proceedings of NIPS 2013, 2013.

[8] B. Pang, and L. Lee, "Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales," Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), 2005.

[9] B. Pang, and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," In Proceedings of ACL 2004. 2004.

[10] P. Probst, B Bischl and A-L. Boulesteix, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," 2018.

[11] M. Kuhn and K. Johnson, "Applied predictive modeling,", page 144, New York: Springer, 2016.