Hierarchical Convolutional Attention Networks for Text Classification

Shang Gao, Arvind Ramanathan, and **Georgia Tourassi**
{gaos, ramanathana, tourassig}@ornl.gov
Computational Science and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA

Abstract

Recent work in machine translation has demonstrated that self-attention mechanisms can be used in place of recurrent neural networks to increase training speed without sacrificing model accuracy. We propose combining this approach with the benefits of convolutional filters and a hierarchical structure to create a document classification model that is both highly accurate and fast to train – we name our method Hierarchical Convolutional Attention Networks. We demonstrate the effectiveness of this architecture by surpassing the accuracy of the current state-of-the-art on several classification tasks while being twice as fast to train.

1 Introduction

Text classification is an important research area in natural language processing (NLP). Traditional text classification approaches utilize features generated from vector space models such as bag-of-words or term frequency-inverse document frequency (TF-IDF) (Sebastiani, 2005). More recently, deep learning approaches have been shown to outperform traditional approaches based on vector space models (Zhang et al., 2015; Tang et al., 2014). These newer deep learning approaches typically rely on architectures based off convolutional neural networks (CNNs) or recurrent neural networks (RNNs) (Young et al., 2017).

RNNs, which are designed to learn patterns over sequential data, have been successfully applied towards various NLP tasks (Liu et al., 2016; Irsoy and Cardie, 2014; Cho et al., 2014). In NLP, RNNs typically process one word at a time and learn features based on complex sequences of words. While RNNs are capable of capturing linguistic patterns useful for NLP tasks, especially over long segments of text, they can be slow to train compared to other deep learning architectures – in order to calculate the gradients associated with any given word in a sequence, an RNN must backpropogate through all previous words in that sequence, resulting in backpropogation functions far more complex than those in feedforward or convolutional architectures.

CNNs, traditionally used for computer vision, have also been applied to NLP tasks with notable success (Zeng et al., 2014; Dos Santos and Gatti, 2014; Wang et al., 2012). Unlike RNNs, which learn patterns across an entire sequence of text, CNNs use a sliding window that examines only a few words/characters at a time. Thus, CNNs learn features based on the most salient combinations of X words/characters where X is determined by the window size used; unlike RNNs, CNNs are less capable of capturing linguistic features across long distances. Despite this shortcoming, CNNs can often be as effective as RNNs in many basic NLP tasks (Yin et al., 2017). Furthermore, CNNs are generally faster to train than RNNs.

In this paper, we introduce Hierarchical Convolutional Attention Networks (HCANs), an architecture based off self-attention that can capture linguistic relationships over long sequences like RNNs while still being fast to train like CNNs. HCANs can achieve accuracy that surpasses the current state-of-the-art on several classification tasks while being twice as fast to train.

2 Related Work

In 2014, Kim (2014) proposed one of the first CNNs for text classification. Kim's CNN used three parallel convolutional layers; these process a sentence using a sliding window that examines three, four, and five words at a time. The three

convolutions then feed into a maxpool across the entire sentence, which selects the most potent features in each convolution and concatenates them into a single feature vector. Finally, the selected features are fed into a dense softmax layer for classification. Due to its simplicity and strong performance in many tasks, Kim's CNN architecture is still commonly used today in many text classification tasks (Qiu et al., 2017; Gehrmann et al., 2017).

One shortcoming of Kim's CNN approach is that it cannot find linguistic patterns beyond a fixed window size, which may harm performance for complex NLP tasks. Attempts have been made to mitigate this issue by increasing the CNN depth and using local maxpooling to increase the receptive field (Conneau et al., 2017). However, Le et al. (2017) showed that increasing CNN depth helps the performance of character-level CNNs but not word-level CNNs. They further demonstrated that a shallow word-level CNN similar to Kim's proposed structure can outperform much deeper and more complex CNN architectures on a wide range of text classification tasks.

The current state-of-the-art in text classification are Hierarchical Attention Networks (HANs), developed by Yang et al. (2016). Whereas the previous approaches mentioned are all based on CNNs, HANs utilize RNNs. HANs use a hierarchical structure in which each hierarchy uses the same architecture – a bidirectional RNN with gated recurrent units (GRUs) (Chung et al., 2014), followed by an attention mechanism that creates a weighted sum of the RNN outputs at each timestep. The HAN processes documents by first breaking a long document into its sentence components, then processing each sentence individually before processing the entire document. By breaking a document into smaller, more manageable chunks, the HAN can better locate and extract critical information useful for classification. This approach surpassed the performance of all previous approaches across several text classification tasks. However, compared to CNN-based approaches, HANs are much slower to train because they utilize RNNs.

In 2017, Vaswani et al. (2017) created a deep learning model for machine translation based entirely on self-attention mechanisms (Cheng et al., 2016; Lin et al., 2017; Paulus et al., 2017). Traditionally, CNN or RNN layers are used to extract meaningful features from words or images; attention is applied afterwards to the output of the CNN or RNN layers to help the network focus on features that are most salient (Luong et al., 2015; Xu et al., 2015; Hermann et al., 2015). However, Vaswani showed that self-attention could be applied directly on raw word embeddings to extract important relations and apply meaningful transformations on words. Like RNNs, this attention-based approach can capture relationships over long distances; unlike RNNs, this approach utilizes a feedforward architecture and is much faster to train. Vaswani achieved state-of-the-art results in machine translation using 10x-100x fewer trainable parameters than previous state-of-the-art models.

We hypothesize that a similar self-attention-based architecture can achieve both fast and accurate performance in text classification tasks. In the following section, we show how we adapt the attention-based architecture developed by Vaswani for machine translation into an effective approach for text classification.

3 Hierarchical Convolutional Attention Network

The overall structure of our HCAN is shown in Figure 1. The components and structure of our HCAN are described in greater detail in the following subsections.

3.1 Scaled Dot Product Attention

Suppose we have a sequence of word embeddings $E^{input} \in \mathbb{R}^{l \times d}$, where l is the length of the sequence, d is the embedding dimension, and e_i^{input} is the i-th word embedding in the sequence.

Self-attention, sometimes referred to as intra-attention, compares each entry e_i^{input} to every entry e_i^{input} in that same sequence; this allows for the discovery of relationships between entries in the sequence. Self-attention outputs a new sequence $E^{output} \in \mathbb{R}^{l \times d}$ in which each entry e_i^{output} is a weighted average of all entries e_i^{input} in the input sequence. Each entry e_i^{output} should contain within it the most pertinent information to that entry from all entries in the input sequence e_i^{input}.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \quad (1)$$

Scaled-dot-product attention (Figure 2, Equation 1) is a type of self-attention developed by Vaswani et al. that was shown to work well

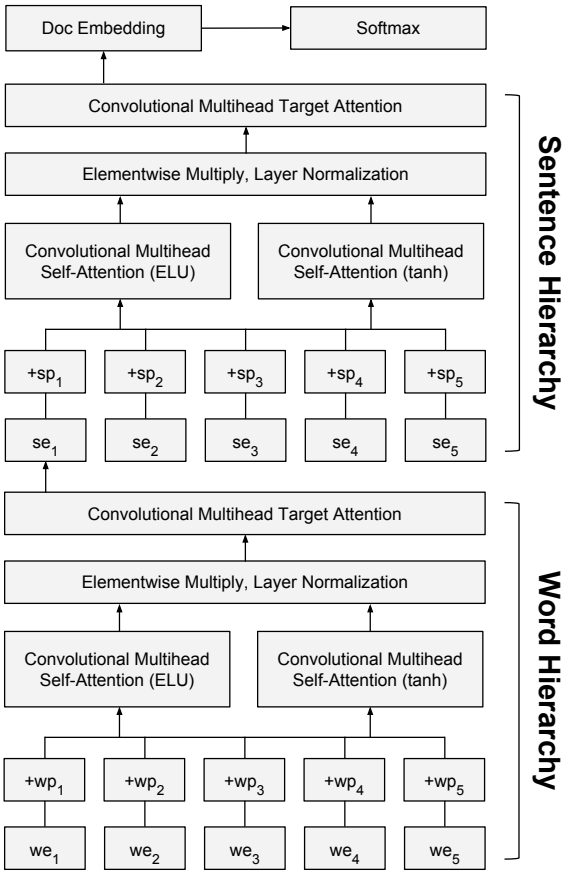

Figure 1: Architecture for our Hierarchical Convolutional Attention Network (HCAN).

in machine translation. Scaled-dot-product attention utilizes three word embedding matrices – the 'query' embeddings $Q \in \mathbb{R}^{l \times d}$, the 'key' embeddings $K \in \mathbb{R}^{l \times d}$, and the 'value' embeddings $V \in \mathbb{R}^{l \times d}$.

In the most basic implementation of self-attention, Q, K, and V can all be substituted by the same sequence of word embeddings $E^{input} \in \mathbb{R}^{l \times d}$. Q and K are used to create a weight matrix QK^T based on the similarity of entries in the sequence. Vaswani et al. found that scaling this weight matrix by a factor of $\sqrt{d}$ yields better performance for higher-dimensional word embeddings. Once this weight matrix is scaled and normalized, it is multiplied with V to create a new output sequence $E^{output} \in \mathbb{R}^{l \times d}$ in which each entry e_i^{output} is a weighted average of all entries e_i^{input} in the input sequence.

3.2 Convolutional Feature Extraction

Rather than use the same E^{input} for Q, K, and V, we can use a function to extract different features from E^{input} for each of the Q, K, and V embeddings. This allows for more expressive comparison between entries in a sequence; for example, certain features may be useful when comparing Q and K but may not be necessary when creating the output sequence from V. For our feature extractor function, we use a 1D convolution with d filter maps and a window size of three words, which provides more context for each center word when extracting important features (Equation 2).

$$\begin{aligned} Q &= \text{ELU}(\text{Conv1D}(E, W^q) + b^q) \\ K &= \text{ELU}(\text{Conv1D}(E, W^k) + b^k) \\ V &= \text{ELU}(\text{Conv1D}(E, W^v) + b^v) \end{aligned} \tag{2}$$

In the equation above, $\text{Conv1D}(A, B)$ is a 1D convolution operation with A as the input as B as the filter, $\{Q, K, V, E\} \in \mathbb{R}^{l \times d}$, $\{W^q, W^k, W^v\} \in \mathbb{R}^{3 \times d \times d}$, and $\{b^q, b^k, b^v\} \in \mathbb{R}^d$.

We found exponential linear units (ELUs) (Clevert et al., 2016) to perform better than rectified linear units (ReLUs) and other activation functions. Unlike ReLUs, ELUs can output negative values, which allows for more complex interactions between the Q and K embeddings when calculating word weights – each word can be assigned a large range of both positive and negative values before being sent into the softmax function.

3.3 Convolutional Multihead Self-Attention

For each entry in the output sequence, scaled dot product attention calculates a set of weights that is used to create a weighted average; the same weights are applied across all d dimensions of the V embeddings. To expand the capabilities of scaled dot product attention, Vaswani et al. introduced multihead attention. Rather than using a single attention function across all d dimensions of the embeddings, multihead attention uses h parallel attention functions, each of which attends to a different portion of the embedding dimension. This allows different portions of the embeddings to be combined using different weights so that the final output sequence can be constructed from a more expressive combination. Vaswani demonstrated that multihead attention performs better than regular scaled dot product attention for machine translation.

$$\begin{aligned} &\text{MultiHead}(Q, K, V) = [head_1, ..., head_h] \\ &\qquad \text{where } head_i = \text{Attention}(Q_i, K_i, V_i) \end{aligned} \tag{3}$$

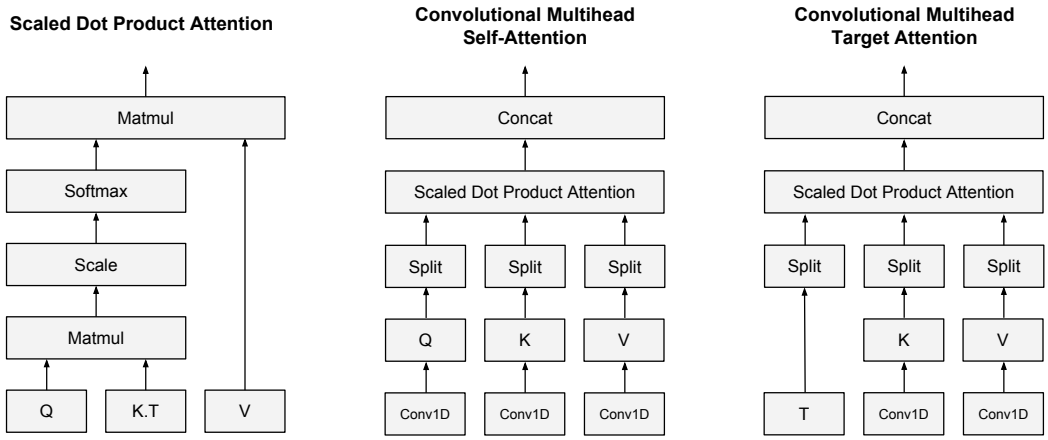

Figure 2: Scaled dot product attention, convolutional multihead self-attention, and convolutional multihead target attention.

Our implementation convolutional multihead self-attention (Figure 2, Equation 3) is based on the multihead attention developed by Vaswani. After using convolution to generate the Q, K, and V embeddings, we split each of the Q, K, and V embeddings into h sub-embeddings such that $\{Q_i, K_i, V_i\} \in \mathbb{R}^{l \times d/h}$. Each triplet of sub-embeddings is then fed into their own scaled dot product attention function. The final output is the concatenation of the outputs $head_i \in \mathbb{R}^{l \times d/h}$ from the individual scaled dot product attention functions to form an output sequence $E^{output} \in \mathbb{R}^{l \times d}$.

3.4 Capturing Complex Word Relationships

In general, attention mechanisms are designed to produce a weighted average of an input sequence. Unfortunately, when trying to capture the overall content within a linguistic sequence, a weighted average may not be sufficient. Two examples of this are negation and scaling. In negation, a word sequence such as 'was not the case' may reverse the meaning of words in another part the sequence (i.e. multiply those embeddings by -1). Similarly in scaling, a word sequence such as 'to a high degree' may increase the polarity of another part of the sequence (i.e. multiply those embeddings by some positive value). Attention mechanisms, which only create weighted averages, are not designed to capture these interactions.

To better capture these types of linguistic interactions, we test the effectiveness of using two convolutional multihead self-attentions in parallel and performing elementwise multiplication on the outputs (Figure 1, Equation 4). This allows the network to capture more complex interactions between elements in the sequence and expands the expressiveness of the final output beyond that of a simple weighted average.

$$
\begin{aligned}
\text{Parallel}(E) &= \text{MultiHead}(Q^a, K^a, V^a) \\
&\quad \odot \text{MultiHead}(Q^b, K^b, V^b) \\
\text{where } Q^a &= \text{ELU}(\text{Conv1D}(E, W^{qa}) + b^{qa}) \\
K^a &= \text{ELU}(\text{Conv1D}(E, W^{ka}) + b^{ka}) \\
V^a &= \text{ELU}(\text{Conv1D}(E, W^{va}) + b^{va}) \\
Q^b &= \text{ELU}(\text{Conv1D}(E, W^{qb}) + b^{qb}) \\
K^b &= \text{ELU}(\text{Conv1D}(E, W^{kb}) + b^{kb}) \\
V^b &= \tanh(\text{Conv1D}(E, W^{vb}) + b^{vb})
\end{aligned}
\tag{4}
$$

Because tanh outputs a value between -1 and 1, it is used to generate the V embeddings for the second self-attention to prevent the final output from becoming too small or large after multiplying the outputs from the two self-attention mechanisms.

3.5 Convolutional Multihead Target-Attention

The output of our convolutional multihead self-attention is a new output sequence $E^{output} \in \mathbb{R}^{l \times d}$ in which l is based on the length of the input sequence. For classification purposes, we require that each sequence regardless of its length be represented by a single fixed-length vector $V \in \mathbb{R}^{1 \times d}$. We therefore introduce convolutional multihead target-attention, which utilizes the concepts from multihead convolutional self-attention but

operates like the traditional attention mechanism that is used on the outputs of a RNN.

$$\begin{aligned} \text{Target}(E) &= \text{MultiHead}(T, K, V) \\ \text{where } K &= \text{ELU}(\text{Conv1D}(E, W^k) + b^k) \\ V &= \text{ELU}(\text{Conv1D}(E, W^v) + b^v) \end{aligned} \quad (5)$$

In convolutional multihead target-attention (Figure 2, Equation 5), instead of comparing the entries in a sequence $E^{input} \in \mathbb{R}^{l \times d}$ to each other, we compare them to a learnable target vector $T \in \mathbb{R}^{1 \times d}$ that represents the most critical information to look for given a specific task – the content in this vector is learned through backpropogation based on the task at hand. The output is a single weighted average $E^{output} \in \mathbb{R}^{1 \times d}$ that captures the most critical content across the sequence. This final output vector may then be fed into a softmax and used for classification purposes.

3.6 Positional Embeddings

RNN-based approaches for text processing can inherently account for word order when extracting features. However, feedforward and convolution-based approaches such as our implementation of convolutional multihead self-attention do not have this capability. One way to address this problem is by adding positional embeddings $P \in \mathbb{R}^{l \times d}$ (Gehring et al., 2017; dos Santos et al., 2015). Positional embeddings are vector representations of the absolute position of an entry in a sequence. These are added directly to each word/sentence embedding in the input sequence before the sequence is fed into the convolutional multihead self-attention. We use randomly initialized embeddings that are learned during training; we found that these provide a slight boost toward classification accuracy.

3.7 Hierarchical Structure

In their work on HANs, Yang et al. attained state-of-the-art performance by utilizing a hierarchical structure that first breaks up documents into sentences. The lower hierarchy reads in word embeddings from a given sentence and outputs a sentence embedding representing the content within that sentence, and the upper hierarchy reads in the sentence embeddings created from the lower hierarchy and outputs a document embedding representing the content of the entire document; this document embedding is then used for classification. In our experiments, we test the effectiveness of our HCAN with and without this hierarchical structure. We expect that, like with RNNs, self-attention has difficulty capturing meaningful semantic relationships over very long sequences with too many entries; using a hierarchical structure to break down long sequences into more manageable chunks mitigates this issue.

Each hierarchy in our HCAN consists of two parallel convolutional multihead self-attentions followed by a convolutional multihead target attention (Figure 1). Positional embeddings are added to the inputs of each hierarchy to allow the network to identify relationships based on word/sentence positions. We tried increasing the depth within each hierarchy by using multiple layers of self-attentions but found that this did not improve model accuracy.

3.8 Regularization

To regularize our network, we apply dropout of 0.1 on the normalized attention weights (produced by scaling QK^T by $\sqrt{d}$ and then applying softmax) within every scaled dot product attention. Furthermore, we apply dropout of 0.1 on the word and sentence embeddings after the positional embeddings have been added, which has been shown to be effective in other NLP applications (Peng et al., 2015).

We also apply layer normalization (Ba et al., 2016) after the elementwise multiplication of the two parallel convolutional multihead self-attentions (Figure 1). This not only applies a regularization effect, but also speeds up the rate of convergence. Layer normalization is used instead of batch normalization because layer normalization is still effective with very small batch sizes.

4 Experiments

4.1 Datasets

We evaluate the performance of the HCAN on four classification tasks using three datasets (Table 1).

The Yelp reviews dataset [1] consists of over 4.7 million Yelp reviews of various businesses collected over 12 metropolitan areas. For our task, we use only reviews from 2016 (approximately 1 million reviews) and try to predict the rating 1-5.

The Amazon reviews dataset (McAuley and Leskovec, 2013) consists of 83.68 million Amazon product reviews from different product categories spanning May 1996 to July 2014. For

[1] https://www.yelp.com/dataset

Table 1: Dataset Descriptions

Dataset	Classes	Documents	Vocabulary	Task Description
Yelp Reviews 2016	5	1,033,037	72,880	Sentiment Analysis
Amazon Reviews Sentiment	5	500,000	67,802	Sentiment Analysis
Amazon Reviews Category	10	500,000	67,802	Topic Classification
Pubmed	8	800,000	182,167	Topic Classification

our evaluation, we selected 10 popular categories and extracted 50,000 randomly selected reviews from each: books, electronics, clothing, home and kitchen, sports and outdoors, health, video games, tools, pet supplies, and food. We use this dataset for two separate classification tasks – sentiment analysis (1-5) and product classification.

The Pubmed dataset [2] consists of more than 26 million citations, abstracts, and other metadata from biomedical literature dating back to 1964. For our experiments, we use Pubmed abstracts associated with 8 common medical subject heading (MeSH) labels: metabolism, physiology, genetics, chemistry, pathology, surgery, psychology, and diagnosis. We only use abstracts that are associated with a single label, yielding a final selection of 800,000 abstracts, 100,000 for each label.

4.2 Baselines and Hyperparameters

As a baseline, we test the performance of two traditional machine learning classifiers that do not utilize deep learning: Naive Bayes (NB) and logistic regression (LR). For logistic regression, we use L1 regularization with a penalty strength of 1.0.

We also compare the performance of our HCAN to that of two other deep learning models. First, we test a word-level shallow-and-wide CNN using an architecture similar to that developed by Kim (2014) for sentence classification. We use three parallel convolution layers with 3-, 4-, and 5-word windows, all with 100 feature maps. These feed into a temporal maxpool across the entire document and the result is concatenated. We apply 50% dropout on the concatenated vector and feed this vector into a softmax classifier. This simple architecture has been shown to outperform many deeper and more complex CNN-based models (Le et al., 2017).

We also test the performance of HANs (Yang et al., 2016), which are the current state-of-the-art. For our HAN, we use the same optimized hyperparameters as those used by Yang – each hierarchy is composed of a bi-directional GRU with 50 units and an attention mechanism with a hidden layer of 200 neurons.

For the HCAN, we tuned the hyperparameters on the Yelp 2013 dataset. We tuned the attention embedding size d and the number of attention heads h used in our scaled dot-product attention. We use embedding size 512 and 8 heads for our final implementation.

4.3 Setup Details

For each dataset, we lowercase all characters and remove non-alphanumerics other than periods, exclamation marks, and questions marks (used to split documents into sentences). For the traditional machine learning approaches that utilize TFIDF features, we generate unigrams and bigrams with a minimum document frequency of 5. For deep learning models that utilize word embeddings, we train Word2Vec embeddings using a minimum word frequency of 5 and a dimension size of 512.

The deep learning models are all trained on a single document at a time with no batching. This configuration allows for variable length input so that long documents do not need to be cut short and short documents do not need to be padded. All models are trained using the Adam optimizer (Kingma and Ba, 2015) with learning rate 2E-5, beta1 0.9, and beta2 0.99.

We split each dataset into train, validation, and test sets using stratified 80/10/10 splitting. The TFIDF-based models are fitted on the train sets and evaluated on the test sets. The deep learning models are trained on the train set, and every 50,000 documents we evaluate on the validation set until the model converges. We save the model parameters with the highest validation accuracy and use those parameters to evaluate on the test set.

4.4 Results

The results of our experiments are displayed in Table 2. For each deep learning model, we record the final test accuracy, average time to train on a single

[2] https://www.ncbi.nlm.nih.gov/pubmed/

Table 2: Test set accuracy, mean training time for a single document, and total training time on each task

Classifier	Yelp 2016	Amazon Sentiment	Amazon Category	Pubmed
Naive Bayes	63.12 –, 1.8s	61.66 –, 0.8s	88.14 –, 1.3s	75.81 –, 4.2s
Logistic Regression	71.31 –, 306s	67.57 –, 101s	88.69 –, 173s	78.57 –, 463s
Word shallow-and-wide CNN	74.44 17ms, 9hr	70.75 15ms, 5hr	88.20 15ms, 5hr	78.15 35ms, 22hr
Hierarchical Attention Network	76.30 96ms, 67hr	72.56 97ms, 35hr	89.68 113ms, 37hr	79.89 167ms, 110hr
Conv Attention Network (One Self-Attention, Maxpool)	75.01 19ms, 13hr	71.24 19ms, 8hr	89.27 19ms, 8hr	79.21 38ms, 25hr
Conv Attention Network (One Self-Attention, Target Attention)	75.17 21ms, 14hr	71.45 21ms, 9hr	89.35 21ms, 9hr	79.70 39ms, 26hr
Conv Attention Network (Two Self-Attentions, Maxpool)	75.21 23ms, 15hr	71.45 22ms, 9hr	89.41 22ms, 9hr	79.86 41ms, 27hr
Conv Attention Network (Two Self-Attentions, Target Attention)	75.25 25ms, 17hr	71.78 24ms, 10hr	89.71 24ms, 10hr	79.95 43ms, 29hr
Hiearchical Conv Attention Network (One Self-Attention, Maxpool)	75.00 24ms, 16hr	71.09 23ms, 9hr	88.85 23ms, 9hr	79.31 42ms, 28hr
Hierarchical Conv Attention Network (One Self-Attention, Target Attention)	75.75 34ms, 23hr	72.33 29ms, 12hr	89.55 29ms, 12hr	79.91 47ms, 31hr
Hiearchical Conv Attention Network (Two Self-Attentions, Maxpool)	75.54 31ms, 21hr	72.43 31ms, 13hr	89.34 31ms, 13hr	80.09 50ms, 33hr
Hierarchical Conv Attention Network (Two Self-Attentions, Target Attention)	**76.51** 49ms, 32hr	**72.85** 38ms, 16hr	**89.89** 38ms, 16hr	**80.13** 53ms, 35hr

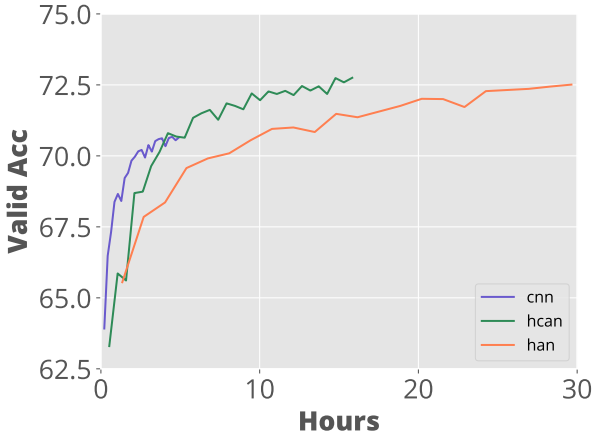

Figure 3: Validation accuracy vs time on Amazon sentiment analysis task.

document, and total time to converge. For timing, all models were trained on a single NVIDIA TITAN X GPU.

In all four tasks, the HCAN achieves the highest test accuracy. Furthermore, HCANs process documents and converge more than twice as fast as the HAN (Figure 3). Within the HCAN, using a hierarchical structure achieves better accuracy than not using a hierarchical structure, using two parallel self-attentions achieves better accuracy than using a single self-attention, and using target attention outperforms using maxpool, especially when using a hierarchical structure.

We note that the difference in accuracy between the deep learning approaches and traditional machine learning approaches is greater in the sentiment classification tasks than in the topic classification tasks. We expect that this is because sentiment classification requires more semantic nuance, which can be difficult to capture via TFIDF features. On the other hand, topic classification may require the presence of only a few specific words to indicate a specific topic.

5 Discussion

Based on our results, we see that the two best performing architectures are the HCAN and the HAN. Unlike the shallow-and-wide CNN,

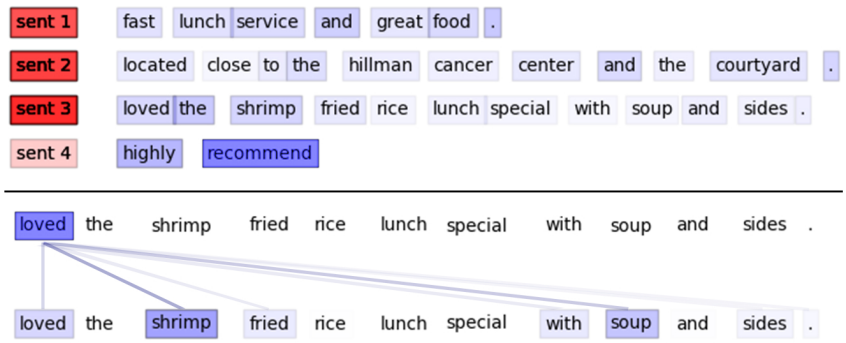

Figure 4: Attention weights assigned to a sample Yelp review by one attention head. The top portion shows the attention weights assigned to each word and sentence by the convolutional multihead target-attention. The bottom portion shows the attention weights assigned to the word "loved" in sentence 3 by the convolutional multihead self-attention.

HCANs and HANs utilize a hierarchical structure that first breaks a document down into its constituent sentences. Using this structure, the networks first locate the most critical information within each sentence and then establish the relationships between the critical information found from each sentence. Our results suggest that this approach works better for text classification than scanning the entire document in one single pass to look for key features.

On our tasks, we see that the HCAN achieves similar performance with the HAN but trains much faster. We attribute this to the fact that HCANs utilize a self-attention-based architecture instead of an RNN-based architecture to extract features. Self-attention utilizes a feed-forward structure, whereas an RNN must backpropagate onto itself over time. When computing gradients, this means that much more calculation is required for RNN-based architectures. For our HCAN, we utilized a self-attention mechanism with a width of 512 neurons and were able to perform faster than our HAN that used an RNN with only 50 GRUs.

Another important implication of self-attention is that it is easier to parallelize than RNNs. Self-attention utilizes a fixed number of feed-forward steps that remain the same regardless of the length of the input sequences. This makes it simple to split the model parameters associated with self-attention across multiple GPUs even when processing multiple documents of different lengths. On the other hand, the number of operations for an RNN is dependent on the length of the input sequence. This makes it challenging to efficiently split RNN parameters across multiple GPUs when dealing with a mini-batch of documents that vary in length (Huang et al., 2013).

Utilizing an attention-based approach also increases the interpretability of the model. By examining the attention weights assigned to each word/sentence by the target attention mechanisms in each hierarchy of the HCAN, we can locate the words/sentences in a document that contribute most to its final label (Figure 4). Furthermore, we can also examine the attention weights assigned to each word/sentence in the self-attention mechanisms to establish how the HCAN is finding relationships between individual words/sentences when extracting important features (Figure 4).

Our results show that using two parallel self-attention mechanisms results in higher accuracy than using a single self-attention mechanism. Upon analyzing the attention weights assigned by the self-attention mechanisms, we found that using two parallel self-attentions captures more relationships involving modifier words than one single self-attention mechanism alone (Figure 5). Furthermore, we analyzed the documents that two parallel self-attentions classified correctly but one single self-attention did not. In sentiment analysis tasks, we found that many of these documents (1) begin positively but conclude negatively or vice versa, (2) contain a mix of positive and negative words, or (3) contain words that scale the meaning of another word or phrase (Supplementary A). This supports our hypothesis that two parallel self-attentions better distinguishes complex word relationships like scaling and negation.

To better understand how the HCAN functions in comparison with the HAN, we compared the attention weights assigned to each word/sentence by the target attention mechanisms in the two net-

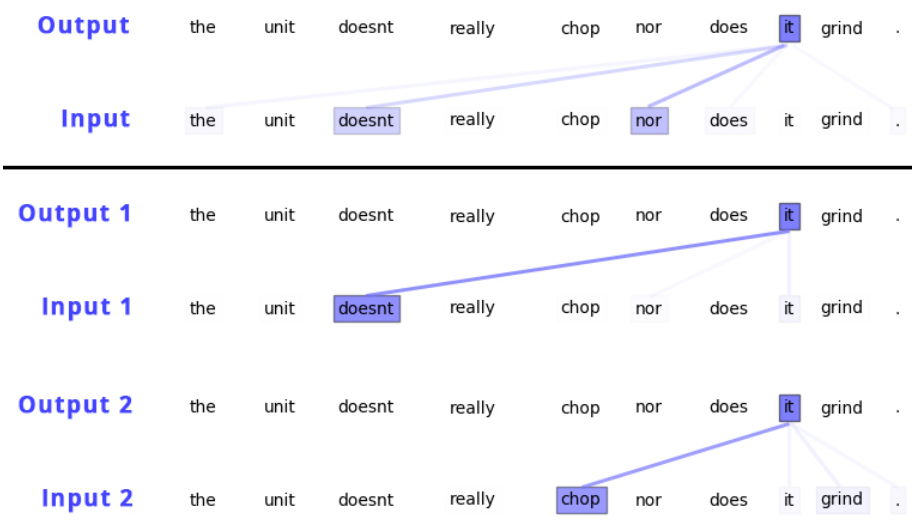

Figure 5: Self-attention weights assigned to a sample word 'it' by (top) HCAN with a single self-attention and (bottom) HCAN with two parallel self-attentions. With two self-attentions, the first self-attention captures the relationship between 'it' and 'doesnt' and the second self-attention captures the relationship between 'it' and 'chop'. This is a more nuanced negation relationship that isn't captured when using a single self-attention.

works. We found the HCAN weight assignments to be more spread out than those from the HAN (Supplementary B). Further analysis revealed that the self-attention mechanisms in the HCAN distribute the meaning of important keywords across many other words before the sequence is fed into the target attention mechanism, thus resulting in the wider distribution of attention weights.

6 Conclusion

In this work, we introduced a new self-attention-based text classification architecture, HCANs, and compared its performance with the current state-of-the-art, HANS, in four classification tasks: Yelp review sentiment, Amazon review sentiment, Amazon review product category, and Pubmed abstract topic. In all four tasks HCANs achieved slightly better performance than HANs while being more than twice as fast to train. Our results show that in time-critical NLP tasks, self-attention-based architectures may be able to replace RNN-based architectures to reduce training time without sacrificing accuracy. Moving forward, we plan to explore efficient implementations of data and model parallelism for self-attention-based architectures such as the HCAN. The code for our experiments is available online at `https://code.ornl.gov/v33/HCAN/`.

Acknowledgments

This work has been supported in part by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) program established by the U.S. Department of Energy (DOE) and the National Cancer Institute (NCI) of the National Institutes of Health. This work was performed under the auspices of the U.S. Department of Energy by Argonne National Laboratory under Contract DE-AC02-06-CH11357, Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, Los Alamos National Laboratory under Contract DE-AC5206NA25396, and Oak Ridge National Laboratory under Contract DE-AC05-00OR22725. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. pages 551–561.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Djork-Arn Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). *ICLR* .

Alexis Conneau, Holger Schwenk, Loc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *ACL-EACL*. pages 1107–1116.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*. pages 69–78.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks pages 626–634.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *ICML* pages 1243–1252.

Sebastian Gehrmann, Franck Dernoncourt, Yeran Li, Eric T Carlson, Joy T Wu, Jonathan Welt, John Foote Jr, Edward T Moseley, David W Grant, Patrick D Tyler, et al. 2017. Comparing rule-based and deep learning models for patient phenotyping. *arXiv preprint arXiv:1703.08705* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*. pages 1693–1701.

Zhiheng Huang, Geoffrey Zweig, Michael Levit, Benoit Dumoulin, Barlas Oguz, and Shawn Chang. 2013. Accelerating recurrent neural network training via two stage classes and parallelization. In *Proc IEEE Workshop Autom Speech Recognit Underst*. IEEE, pages 326–331.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*. pages 720–728.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*. pages 1746–1751.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *ICLR* .

Hoa T. Le, Christophe Cerisara, and Alexandre Denis. 2017. Do convolutional networks need to be deep for text classification? *CoRR* abs/1707.04108.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *IJCAI* pages 2873–2879.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation pages 1412–1421.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM conference on Recommender systems*. ACM, pages 165–172.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. In *EMNLP*. pages 2106–2111.

J Qiu, HJ Yoon, PA Fearn, et al. 2017. Deep learning for automated extraction of primary sites from cancer pathology reports. *IEEE J Biomed Health Inform* .

Fabrizio Sebastiani. 2005. Text categorization. In *Encyclopedia of Database Technologies and Applications*, IGI Global, pages 683–687.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*. pages 1555–1565.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Jakob Uszkoreit, Aidan N Gomez, and ukasz Kaiser. 2017. Attention is all you need. *NIPS* .

Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. 2012. End-to-end text recognition with convolutional neural networks. In *ICPR*. IEEE, pages 3304–3308.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. pages 2048–2057.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*. pages 1480–1489.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923* .

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709* .

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*. pages 2335–2344.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*. pages 649–657.

A Yelp Reviews Misclassified by Single Self-Attention

The following are examples of Yelp reviews that were misclassified when using a single self-attention mechanism but correctly classified when using two parallel self-attention mechanisms. Note in many of these reviews, one section of the review negates or scales the meaning in another section.

> i got this at a grocery store thinking it would be great since i only drink a little bit of wine or sake at a time . i ended up giving it away to goodwill after a few months because it doesnt really help the wine or sake at least not for weeks like im prone to need between glasses and it is annoying to use the plastic thingy trying to get it tight and worrying that youre going to break the bottle . i think a nice reusable cork kind of gadget would do just as good a job take up less drawer space and look prettier in the bottle .

> for those of you who criticized this book for lack of a plot i can only assume that you are much more suited to books in the mystery thriller genre . i loved it and found the characters very real and compelling . if you are a reader who likes books about relationships you are going to love it too .

> i hesitated buying this grill because there were so many negative reviews . im glad i decided to buy the grill . weve used it 5 times so far . to address some of the negative reviews . you can cook with the grill on both high and low with the cover closed . in the instructions you are actually directed to clean the grill for the first time with the burners on high and the cover closed . the stand is excellent . weve been using this at the beach . the stand and fold out tables save packing additional cargo in the car . as far as cleaning i dont know what people are expecting . its a bbq it gets dirty . the grates clean up real nice with brillo . the chrome area under the grill plates cleans up with a fantastic type cleaner .

> a feel good read . dean koontz does this type of book very very well . no horrid monsters except for the unscrupulous government people so dont expect nightmares from this one . it does have its suspense however .

> its fun in the begining . but the levels get harder and game play is not as fun . it got so hard it was not much fun to play . and has not much varity in it .

B Comparing Attention Weights from HAN and HCAN

The attention weights assigned by the target-attention for the HAN (Figure 6) are more focused on keywords than for the HCAN (Figure 7). This is because the self-attention in the HCAN redistributes the content of important keywords across other words before the sequence is sent into the target-attention (Figure 8).

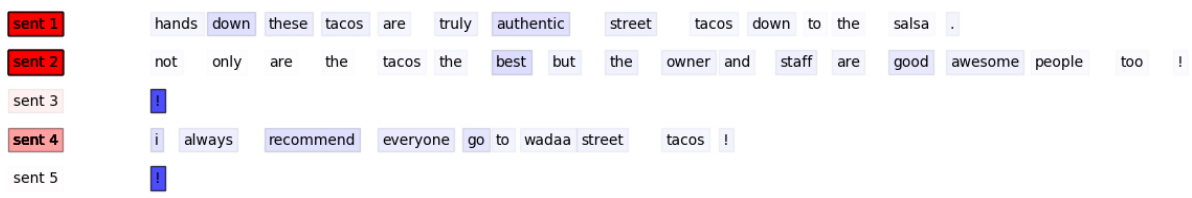

Figure 6: HAN target-attention weights assigned to a sample Yelp review. We see that the weights are primarily focused on sentiment keywords.

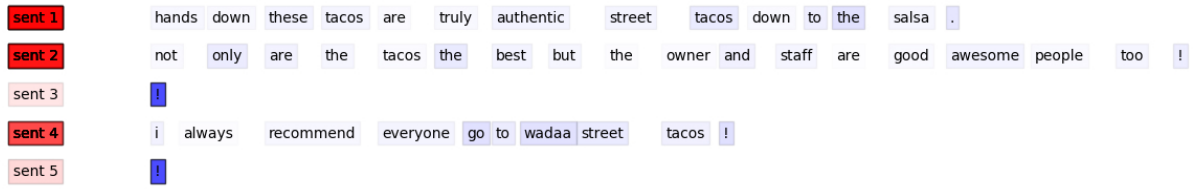

Figure 7: HCAN target-attention weights assigned to a sample Yelp review. We see that the weights are more spread out than in the HAN target-attention.

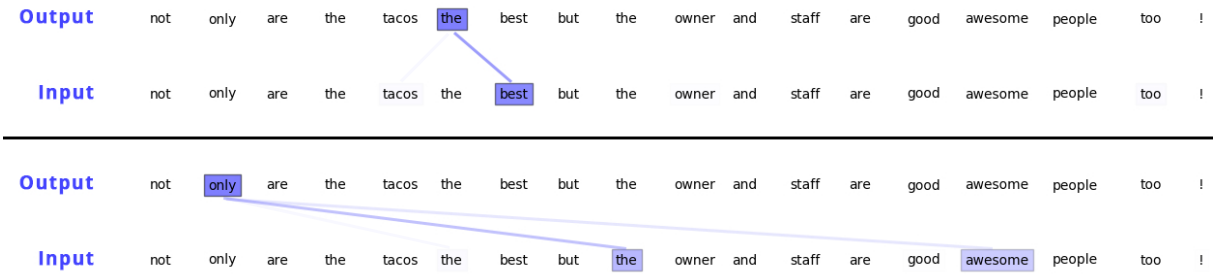

Figure 8: HCAN self-attention weights assigned to the words ”the” and ”only” in a sample Yelp review sentence. We see that meaning from sentiment keywords are redistributed among other words. In the two example shown above, we see that ”best” is reassigned to ”the” and ”awesome” is reassigned to ”only”. Therefore, the HCAN target-attention weighs the words ”the” and ”only” higher because they contain content from sentiment keywords.