Gaussian Transformer: a Lightweight Approach for Natural Language Inference

Maosheng Guo, Yu Zhang, Ting Liu*
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{msguo, yzhang, tliu}@ir.hit.edu.cn

Abstract

Natural Language Inference (NLI) is an active research area, where numerous approaches based on recurrent neural networks (RNNs), convolutional neural networks (CNNs), and self-attention networks (SANs) has been proposed. Although obtaining impressive performance, previous recurrent approaches are hard to train in parallel; convolutional models tend to cost more parameters, while self-attention networks are not good at capturing local dependency of texts. To address this problem, we introduce a Gaussian prior to self-attention mechanism, for better modeling the local structure of sentences. Then we propose an efficient RNN/CNN-free architecture named Gaussian Transformer for NLI, which consists of encoding blocks modeling both local and global dependency, high-order interaction blocks collecting the evidence of multi-step inference, and a lightweight comparison block saving lots of parameters. Experiments show that our model achieves new state-of-the-art performance on both SNLI and MultiNLI benchmarks with significantly fewer parameters and considerably less training time. Besides, evaluation using the Hard NLI datasets demonstrates that our approach is less affected by the undesirable annotation artifacts.

Introduction

Natural Language Inference (NLI), also known as Recognizing Textual Entailment (RTE), is a fundamental problem in the research field of natural language understanding, which could help tasks like questions answering, reading comprehension and summarization (Dagan et al. 2013). In NLI settings, the model is presented with a pair of sentences, namely *premise* and *hypothesis*, and asked to determine the reasoning relationship between them from a set including *entailment, contradiction and neutral*. Numerous efforts have been dedicated to this task, where the dominant trend is to build complex neural models using millions of parameters which cost lots of time to train. Although achieved impressive accuracy, previous works using recurrent models are hard to train in parallel while convolutional models often need more parameters to learn.

To address this problem, we propose an efficient RNN/CNN-free architecture named Gaussian Transformer,

*Corresponding Author

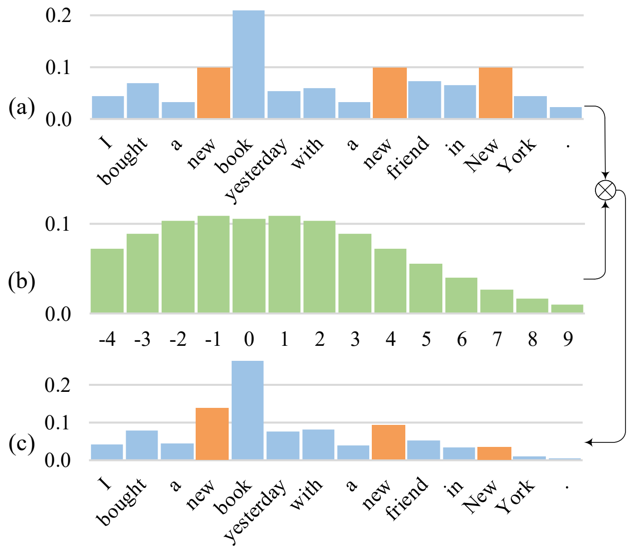

Figure 1: Probabilities of each token attending to current central word 'book': (a) illustrates the vanilla self-attention, where the word 'new' appeared in different positions obtain the same importance, which is inconsistent with our experience that adjacent words matters; (b) depicts a Gaussian distribution over distance (x-axis) that encourages focusing on neighboring tokens; (c) draws the attention corrected by the Gaussian prior, where the first 'new' is more important.

which is adapted from the Transformer model in machine translation tasks(Vaswani et al. 2017). The original Transformer is based on the self-attention mechanism, eschewing the disadvantages of recurrence and convolution. However, it cannot capture local structure of texts, which is enhanced with a Gaussian prior probability in our approach according to the following observation: adjacent words contribute more semantically to current phrase than distant ones. For example, in the sentence 'I bought a new book yesterday with a new friend in New York.', the first 'new' is more important to the word 'book' than the other two. Indeed, RNNs tend to forget distant inputs, while CNNs exclude words outside the current convolutional window. This 'chunking' phenomenon is naturally modeled in RNNs and CNNs but is neglected in the original Transformer. As shown in Figure 1(a), same words at various distances are treated almost equally when computing self-attention. We correct this problem using a Gaussian prior probability over the distance to the central word, as depicted in Figure 1(b)&(c). Although positional encoding might help to alleviate this problem, experiments show that the Gaussian prior works better. Meanwhile, long-term dependency is also taken into consideration by using multiple layers of Gaussian attention in our model.

Besides using Gaussian self-attention networks to model sentences, we also introduce the high-order interaction blocks to collect evidence of multi-step inference and an efficient comparison mechanism for NLI tasks, helping us to establish a new state-of-the-art performance using one order of magnitude fewer parameters. Thanks to the CNN/RNN-free architecture, our model is also faster than previous works.

Recently, annotation artifacts were found in existing datasets, i.e., SNLI (Bowman et al. 2015) and MultiNLI (Williams, Nangia, and Bowman 2017), and the more challenging Hard NLI benchmark was proposed (Gururangan et al. 2018). Our model outperforms previous works by about 5 percent accuracy on this benchmark, showing that the proposed Gaussian Transformer is less affected by the undesirable annotation artifacts.

The contributions of this paper are listed as follows:

- We propose the novel Gaussian self-attention inspired by the 'chunking' phenomenon, which could better capture both local structure and global dependency of sequences without introducing recurrence or convolution.
- We propose an efficient NLI model, i.e, Gaussian Transformer, consisting of Gaussian encoding blocks, high-order interaction blocks and efficient comparison blocks, obtaining new state-of-the-art performance with significant fewer parameters and considerably less training time.

Gaussian Self-attention

As shown in Figure 1, the original transformer treats the same words at various distances almost equally[1], which is inconsistent with our experience of natural language texts that adjacent words contribute more semantically to central words. While CNNs / RNNs model this chunking phenomenon internally, the vanilla self-attention mechanism in Transformer could not capture the local structure of texts.

Supposing that x_i represents the central word from the sentence x, the vanilla dot-product self-attention works as Figure 2(a): It soft-aligns each token x_j from x to x_i, according to the compatibility function computed by the softmax of dot products, i.e., $Comp_{i,j} = Softmax_j(x_i \cdot x_j)$, and then sums the attended values together, i.e.,

$$\tilde{x}_i = \sum_j Comp_{i,j} x_j. \tag{1}$$

For better modeling the local structure of texts, we increase the importance of neighboring tokens and decay the weight of distant ones. We hypothesize that the semantic

[1]For simplicity, we omit the positional encoding here and discuss it later.

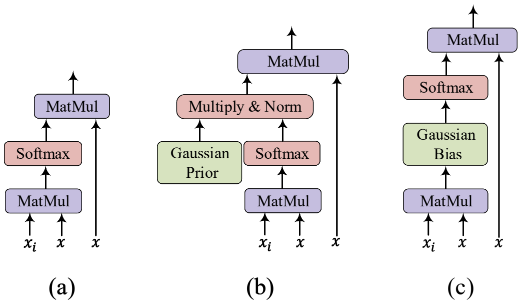

Figure 2: Illustration of attentions: (a) depicts the original dot-product attention of Transformer, (b) & (c) are Gaussian prior extended self-attention.

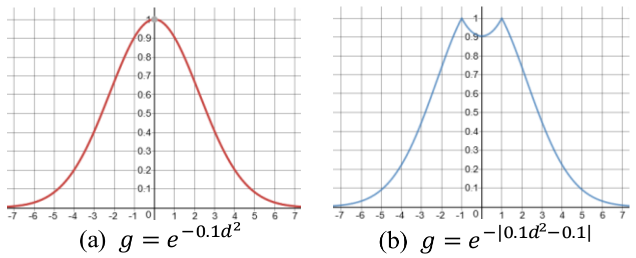

Figure 3: Examples of Gaussian multiplier: (a) illustrates the vanilla version; (b) shows the variant introduced in Eq. (4).

contribution to central word from tokens at different distance obey a normal distribution, and then use a variant of Gaussian prior to correct the importance of tokens aligning to the current central word. The reason to choose normal distribution is that it is hard to statistically measure the semantic importance of a word according to another one, and comparison experiments with different decaying mechanisms, e.g., linear decaying according to distance (Im and Cho 2017), and other distributions such as Zipf Distribution, demonstrate that the Gaussian assumption works better.

For simplicity, we use the (Stigler 1982) definition of a standard normal distribution with variance $\sigma^2 = 1/(2\pi)$, whose probability density function is: $\phi(d) = e^{-\pi d^2}$, where d is the random variable, i.e., the distance between tokens. Then, we insert $\phi(d_{i,j})$ to Eq. (1) just like Figure 2(b) to correct the importance of tokens at various distances:

$$\begin{aligned}\tilde{x}_i &= \sum_j \frac{\phi(d_{i,j}) comp_{i,j}}{Z_1} x_j = \sum_j \frac{e^{-d_{i,j}^2} \cdot e^{(x_i \cdot x_j)}}{Z_2} x_j \\ &= \sum_j \frac{e^{-d_{i,j}^2 + (x_i \cdot x_j)}}{Z_2} x_j \\ &= \sum_j Softmax(-d_{i,j}^2 + (x_i \cdot x_j)) x_j,\end{aligned} \tag{2}$$

where $Z_1 = \sum_k \phi(d_{i,k}) comp_{i,k}$, $Z_2 = \sum_k e^{-d_{i,k}^2 + (x_i \cdot x_k)}$ are normalization factors. Eq. (2) converts the normal distribution to a Gaussian bias term (Figure 2(c)), which saves additional operations of multiplication. Since the Gaussian variance (σ^2) might not incidentally equal $1/(2\pi)$, we introduce a scalar variable w to Eq. (2) to loose the restriction:

$$\tilde{x}_i = \sum_j Softmax(-wd_{i,j}^2 + (x_i \cdot x_j))x_j. \quad (3)$$

Instead of applying the vanilla Gaussian prior, we found it beneficial to introduce a punishment term b to the weight of the central word attending itself, as depicted in Figure 3(b):

$$\tilde{x}_i = \sum_j Softmax(-|wd_{i,j}^2 + b| + (x_i \cdot x_j))x_j, \quad (4)$$

where $|\cdot|$ represents the absolute value, $w > 0$, $b \leq 0$ are scalar parameters. This policy is inspired by (Shen et al. 2017), which disables the attention to itself, while we only reduce the weight instead to keep information of the central word which is also important to current chunk.

It is worth noting that, besides the local structures, long-range dependency also matters to the semantics of sentences. RNNs, e.g., LSTMs, use gates and memory to capture the long-term dependency, while CNNs build deep models with multiple convolution layers for this phenomenon. In our case, we choose the CNNs-style, that is, stacking Gaussian attention layers to track global dependency.

Gaussian Transformer

In this section, we provide a detailed description of the Gaussian Transformer. The input is a pair of sentences $\{p_i\}_{i=1}^{l_p}$ / $\{h_j\}_{j=1}^{l_h}$, where p_i / $h_j \in R^V$ is the one-hot vector representing i-th / j-th word of the *premise* / *hypothesis* with length l_p / l_h, and V is the vocabulary size. The goal is to predict labels from $\{entailment, contradiction, neutral\}$. Figure 4 presents an overview of the architecture, including the following components:

Embedding Blocks

The embedding blocks convert each word of a sentence to a dense vector and combine them to construct a matrix representation. We employ pre-trained word vectors, random initialized character n-grams embeddings and positional encoding to project sentences to high-dimensional space.

Besides commonly used pre-trained word vectors, we utilize a lightweight character-level representation of tokens. Unlike using trainable character embeddings and additional CNNs/LSTMs in previous work, we choose static random initialized character n-gram embeddings to save parameters, and then represent each token as the max-over-time pooling of its embedded character n-grams. This allows us to handle typos and out-of-vocabulary words at minimum computational cost without any additional parameters. The sinusoidal positional encoding is used to exploit the information about the order of sequences since Transformer contains no recurrence and convolution (Vaswani et al. 2017).

All the three embeddings remain fixed during training. The only trainable parameter here is the projection matrix that reduces the dimension of token representation to d_{model},

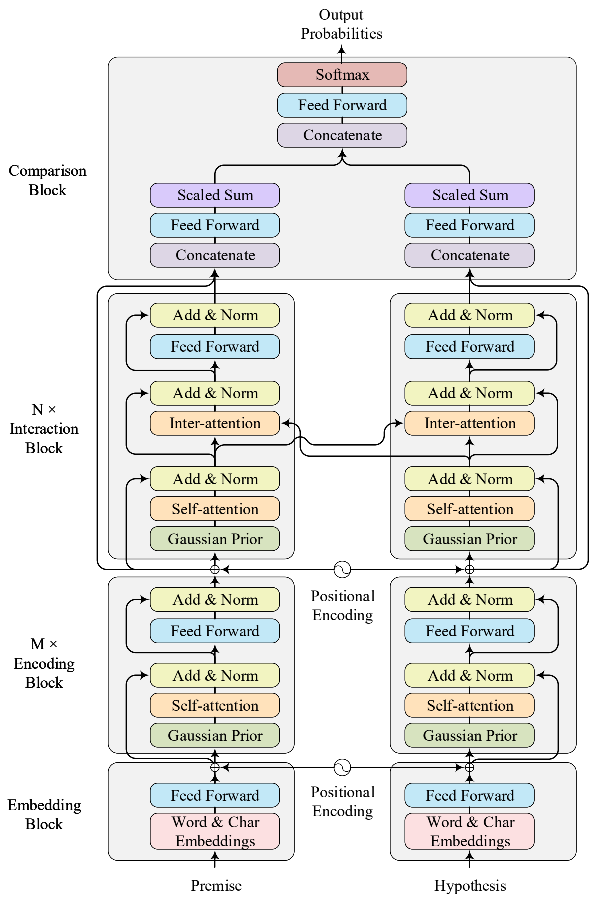

Figure 4: The overall architecture of Gaussian Transformer.

which is the dimension number used internally in the rest of model.

The process of the embedding blocks could be formally described as follows:

$$x_i^{(w)} = x_i E_w, \quad (5)$$

$$x_i^{(c)} = Max_t(\{x_{i,t}^{(c)} E_c\}_{t=1}^{l_{x_i}}), \quad (6)$$

$$x_{i,2k}^{(p)} = sin(i/10000^{2k/d_{model}}), \quad (7)$$

$$x_{i,2k+1}^{(p)} = cos(i/10000^{2k/d_{model}}), \quad (8)$$

$$x_i^{(e)} = x_i^{(p)} + [x_i^{(w)} : x_i^{(c)}]W_e, \quad (9)$$

where x is a placeholder of sentence p or h, $x_i \in R^V$ is the one-hot representation of i-th word of x, which consists of l_{x_i} character n-grams, $x_{i,t}^{(c)} \in R^{V_c}$ indicates the t-th one, $x^{(w)}$, $x^{(c)}$, $x^{(p)}$ and $x^{(e)}$ are the word-level representation, character-level representation, positional encoding, and the final representation of token x_i in this block respectively, $E_w \in R^{V \times d_w}$ is initialized from pre-trained word vectors, $E_c \in R^{V_c \times d_c}$ is a randomly initialized character n-gram embeddings, $2k, 2k+1 \in [0, d_{model})$ indicates the corresponding dimension, $[\cdot : \cdot]$ represents the operation of concatenation, V is the vocabulary size of tokens, V_c is the vocabulary size of character n-grams, $W_e \in R^{(d_w+d_c)\times d_{model}}$ is the trainable projection matrix of encodings.

Encoding Blocks

In this part, we employ a stack of M encoding blocks to capture the local and global dependency of words in sentences, which is similar with the encoder of the original Transformer except that we introduce the Gaussian prior probability according to the chunking phenomenon.

Each encoding block consists of two sub-layers: a multi-head Gaussian self-attention sub-layer and a position-wise feedforward layer. Sub-layers are stacked using residual connections and layer normalizations (Ba, Kiros, and Hinton 2016), so that the output of a sub-layer becomes $y = LayerNorm(x + Sublayer(x))$, where x, y respectively represents the input and output, $Sublayer$ is either the H-head Gaussian self-attention or position-wise feedforward networks.

The multi-head attention mechanism projects representations into H sub-spaces of dimension $d_h = d_{model}/H$ and computes attention in parallel. Then a position-wise feed-forward network (FFN) is applied to the output of attention sub-layer at each position separately and identically:

$$FFN(x) = Dense_2(Relu(Dense_1(x))), \tag{10}$$

$$Dense_k(x) = xW_k + b_k,\ k \in \{1, 2\}, \tag{11}$$

$$Relu(x) = max(0, x), \tag{12}$$

where $W_k \in R^{d_{model}\times d_{model}}$, $b_k \in R^{d_{model}}$.

In CNNs, the concept of receptive field is defined as the region in the input space that affects a particular CNN's feature, because inputs outside the filter window are excluded. Similarly, we could informally define the concept of 'focus field' of Gaussian attention since most values drawn from a normal distribution are within one or two standard deviations (σ) away from the mean, so that the Gaussian attention mainly focuses on that field, although σ is not explicitly provided. Like multilayer CNNs, the bottom Gaussian attention captures the local structure while higher-level attentions track the 'global' dependency. This is why we stack M encoding blocks. Thanks to residual connections, both local and global information could easily flow to upper modules.

Interaction Blocks

Unlike encoding blocks extract semantics from a single sequence, the interaction blocks take the alignment between sentences into account. Besides Gaussian self-attentions to model current sentence and the position-wise feedforward nets to perform non-linear projection, the interaction block inserts a multi-head inter-attention sub-layer between them to align the sentences and collect the evidence of inference.

The inter-attention works similarly with the self-attention. Supposing that vector p_i represents the current central word from the *premise* p, while q_j indicates the j-th token of the *hypothesis*, the (single-head) dot-product inter-attention between p_i and h soft-aligns each token h_j from h to p_i, according to a compatibility function computed by the softmax of dot products, i.e., $Comp(p_i, h_j) = Softmax_j(p_i \cdot h_j)$, and then sums the attended values together, i.e., $\tilde{p}_i = \sum_j Comp(p_i, h_j)h_j$. Similarly, we have $\tilde{h}_j = \sum_i Comp(h_j, p_i)p_i$, where $Comp(h_j, p_i) = Softmax_i(h_j \cdot p_i)$.

The attended $\tilde{p}_i$, as a representation of p_i using its soft-aligned tokens in h, contains information of the relationship between the central token p_i and its most compatible phrases from h, which could be viewed as the evidence of local inference (Parikh et al. 2016), i.e., a single-step inference.

(Liu, Duh, and Gao 2018) point out that some sentence pairs needs more than one-step inference to determine their reasoning relationship, and took the average of multiple softmax layers as their prediction. In this work, we employ a stack of N interaction blocks to perform high-order inter-attention, i.e., attention over attention, to collect evidence of multi-step inference. Thanks to the residual connections between sub-layers, evidence of both single-step alignment and multi-step inference could easily flow to upper modules.

Since the rest parts of interaction blocks, e.g., multi-head Gaussian self-attention, are already introduced in previous sections, we will not repeat them here.

Comparison Block

The comparison block consists of aggregation layers and prediction layers. The aggregation layers combine the output of encoding and interaction blocks, aggregate the alignments of words / phrases and then compares the difference between the two sentences to form a fixed-length feature vector for classification. The prediction layer projects that feature vector to the target space, i.e., $\{entailment, contradiction, neutral\}$, and predicts the probability distribution over them.

As depicted in Figure 4, the comparison block uses a Siamese architecture – two identical parameter-tied aggregation networks are applied to *premise* and *hypothesis* respectively, and then a multilayer perceptron classifier is employed to determine the relationship between them:

Aggregation Networks The encoder blocks build a semantic representation of sentence $x \in \{p, h\}$, denoted as $\{x_i\}_{i=1}^{l_x}$, according to the local and global dependency of tokens modeled by multilayer Gaussian self-attention; the interaction blocks, on the other hand, output the cross-sentence information, denoted as $\{\tilde{x}_i\}_{i=1}^{l_x}$, according to high-order alignments and multi-step inferences collected by the stack of inter-attention. The aggregation networks collect these information to extract the feature-space representation $\bar{x}$ as follows:

$$v_i = Dense_4(Relu(Dense_3([x_i : \tilde{x}_i]))), \tag{13}$$

$$Dense_k(x) = xW_k + b_k,\ k \in \{3, 4\}, \tag{14}$$

$$\bar{x} = \frac{1}{\sqrt{l_x}} \sum_{i=1}^{l_x} (v_i), \tag{15}$$

where $W_3 \in R^{2d_{model}\times d_{model}}$, $W_4 \in R^{d_{model}\times d_{model}}$, and $b_3, b_4, x_i, \tilde{x}_i, \bar{x} \in R^{d_{model}}$.

It first concatenates token x_i and its attentive alignments $\tilde{x}_i$, and then applies position-wise feedforward networks to compare the aligned phrases, and finally aggregates the comparison vectors v_i by summation. We use the reciprocal square root of sentence length to alleviate the problem that longer sequences have larger features vectors.

We also explored other aggregation mechanisms. For example, max-pooling and average-pooling over the concatenation $[x_i : \tilde{x}_i : x_i - \tilde{x}_i : x_i \odot \tilde{x}_i]$ in (Chen et al. 2017b) could not further improve over our model but almost consumes as 4 times parameters as our aggregation network.

Prediction Layers We employ the vanilla MLP classifier to predict the final label of sentence pairs as follows:

$$y = Softmax(Dense_6(Relu(Dense_5([\bar{p} : \bar{h}])))), \quad (16)$$

$$Dense_k(x) = xW_k + b_k, \; k \in \{5, 6\}, \quad (17)$$

where $W_5 \in R^{2d_{model} \times d_{model}}$, $W_6 \in R^{d_{model} \times 3}$, $b_5 \in R^{d_{model}}$, $b_6 \in R^3$.

We also tried more complex prediction layers, such as replacing $[\bar{p} : \bar{h}]$ with $[\bar{p} : \bar{h} : \bar{p} + \bar{h} : \bar{p} - \bar{h}]$ (Kim et al. 2018), and found it makes no significant difference but costs more additional parameters. According to the law of Occam's razor, we choose the simple comparison block as above.

Experiments and Analyses

We conduct experiments on SNLI and MultiNLI datasets, which consists of 570k / 433k English sentence pairs, to train and evaluate the proposed model. We first use the MultiNLI validation datasets to find the optimal settings of the proposed Gaussian Transformer and conduct ablation test to verify the effectiveness of each component of our model. Then we report the performance on both the vanilla and hard version of SNLI / MultiNLI, of models trained from scratch or by leveraging external resources, followed by the details of our implementation. Besides the accuracy of classification, we also compare the number of parameters, training time and inference time with previous work.

Effectiveness of Each Component

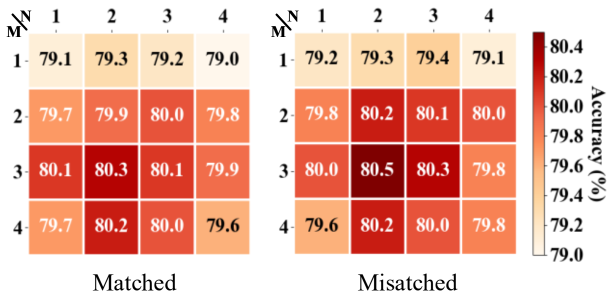

Figure 5: Heatmap of accuracy (%) on MultiNLI validation sets. Multiple encoding / interaction blocks ($M > 1$ / $N > 1$) performing better demonstrates the usefulness of global dependency and multi-step inference.

Global Dependency and Multi-step Inference Figure 5 shows the accuracy on MultiNLI validation sets w.r.t. the various combination of numbers of encoding (M) and interaction (N) blocks, i.e., the depth of model, where the label 'matched' indicates the accuracy of in-domain data, while 'mismatched' represents out-domain. Performance becoming better when increasing M and N initially and peaking at $M = 3$ and $N = 2$, demonstrates the effectiveness of the global dependency, which is captured by multiple Gaussian self-attentions ($M > 1$), and the superiority of multi-step inference modeled by high-order interactions ($N > 1$).

Although continuing deepening the network does not result in a significant accuracy regression, we keep the values $(3, 2)$ in the rest of experiments, since the training time rises linearly as the depth increases.

Model	Matched	Mismatched
Vanilla Transformer (i)	79.3	79.3
+ Zipf prior (ii)	79.7	79.5
+ trainable distance bias (iii)	79.5	79.6
+ linear decay bias (iv)	79.9	79.8
+ Conv. layers (v)	80.1	80.0
+ Gaussian prior (vi)	<u>80.2</u>	<u>80.3</u>
+ Gaussian prior variant (vii)	**80.3**	**80.5**

Table 1: Accuracy (%) of different variants of Gaussian Transformer on the MultiNLI development datasets.

Effectiveness of Gaussian Assumption Since the sinusoidal positional encoding in the original Transformer could already represent the relative position of tokens, it is necessary to evaluate the usefulness of the Gaussian assumption, which encourages self-attention to focus on locally close tokens. We evaluate several models with different prior probabilities and present the results in Table 1. We could observe that adding Gaussian prior to the vanilla Transformer improves about 1 percent accuracy on MultiNLI validation datasets, and using a punishment bias to reduce the importance of central word attending to itself brings additional enhancement, where (vi) and (vii) indicate the model using Eq. (3) and (4) respectively. Next, we found that depthwise separable convolutions (v) which is used in QANet (Yu et al. 2018) is competitive with Gaussian prior but extra parameters are introduced by convolutions. Besides Gaussian prior, we also evaluate the Zipf distribution (ii), since many linguistics statistics, such as word / phrase frequency, obey the famous Zipf's Law. To follow the Zipf's law, we assume the semantic contribution of a token to current phrase is inversely proportional to its distance to the central word, and use Eq. (18) to compute self-attention:

$$\tilde{x}_i = \sum_j \frac{1}{|d_{i,j}| + 1} \frac{comp_{i,j}}{Z_z} x_j; \quad (18)$$

model (iii) uses trainable distance bias term $b_{d_{i,j}}$ to model the relative distance (Parikh et al. 2016) :

$$Comp_{i,j} = Softmax_j(x_i \cdot x_j + b_{d_{i,j}}); \quad (19)$$

while model (iv) adds a linear function of the distance to the compatibility function (Im and Cho 2017):

$$Comp_{i,j} = Softmax_j(x_i \cdot x_j + w_l|d_{i,j}|), \qquad (20)$$

where $w_l(< 0)$ and $b_{d_{i,j}}$ are scalar parameters, $Z_z = \sum_k comp_{i,k}/(|d_{i,k}| + 1)$ is the normalization factor.

In summary, it is necessary to encourage self-attention to focus on local structures (model ii - vii), and the proposed Gaussian prior (variant) performs the best.

Experimental Results

In this section, we present the experimental results of Gaussian Transformer comparing with not only models trained from scratch but also approaches leveraging external resources. Then, we evaluate our model on the more challenging Hard NLI benchmarks.

Comparison with Models Training from Scratch We first evaluate the capability of the Gaussian Transformer itself to resolve NLI problems, by training our model from scratch, i.e., using only NLI training sets without any external resources except word embeddings, and report the accuracy on SNLI and MultiNLI, in Table 2 and 3 respectively.

It can be observed that Gaussian Transformer achieves state-of-the-art performance on both SNLI and MultiNLI benchmarks, improving 1 percent accuracy at most. However, our Gaussian Transformer only costs 665k parameters, which is one order of magnitude fewer than previous approaches. Among them, ESIM(Chen et al. 2017b), BiMPM(Wang, Hamza, and Florian 2017), MwAN(Tan et al. 2018), SAN(Liu, Duh, and Gao 2018), DR-BiLSTM(Ghaeini et al. 2018), CAFE(Tay, Tuan, and Hui 2017), DRCN(Kim et al. 2018) are recurrent models, while DIIN(Gong, Luo, and Zhang 2017) is a convolutional model and DecAtt(Parikh et al. 2016) is a self-attention network.

Model	$\lvert\theta\rvert$	Acc.(S./E.)
200D DecAtt	**580k**	86.8 / -
BiMPM	1.6m	87.5 / 88.8
600D ESIM	4.3m	88.0 / 88.6
448D DIIN	4.4m	88.0 / 88.9
150D MwAN	14m	88.3 / 89.4
SAN	3.5m	88.5 / -
450D DR-BiLSTM	7.5m	88.5 / 89.3
300D CAFE	4.7m	88.5 / 89.3
DCRN	6.7m	<u>88.9</u> / <u>90.1</u>
120D Gaussian Transformer	<u>665k</u>	**89.2 / 90.3**

Table 2: Accuracy (%) on SNLI test set of approaches training from scratch by single / ensemble models, where $|\theta|$ indicates the number of parameters. Gaussian Transformer outperforms previous state-of-the-arts with one order of magnitude fewer parameters.

In addition, we compare the time cost of Gaussian Transformer with that of ESIM, a classic recurrent NLI model with publicly available codes, on the same hardware. We evaluate them by two metrics: the time to train one epoch on the SNLI training data and the inference time to iterate through the test set once. Thanks to its RNN/CNN-free architecture, Gaussian Transformer gains 3.6x / 7.8x speedup versus ESIM, as shown in Table 4.

Model	Matched	Mismatched
ESIM	76.8 / -	75.8 / -
MwAN	78.5 / 79.8	77.7 / 79.4
DIIN	78.8 / 80.0	77.8 / 78.7
CAFE	78.7 / 80.2	77.9 / 79.0
DCRN	79.1 / <u>80.6</u>	78.4 / 79.5
SAN	<u>79.3</u> / <u>80.6</u>	<u>78.7</u> / <u>80.1</u>
Gaussian Transformer	**80.0 / 81.6**	**79.4 / 80.7**

Table 3: Performance on MultiNLI test set of single / ensemble approaches training from scratch. Gaussian Transformer improves the state-of-the-art accuracy by 1 percent at most.

	ESIM	Gauss. Trans.	Speedup
Training time	29 mins	8 mins	**3.6x**
Inference time	31s	4s	**7.8x**

Table 4: Time consumed for training / predicting on the SNLI datasets for one epoch.

Comparison with Approaches using External Knowledge Reasoning between sentences requires common sense, which might not be learned only using the training set. To fill the knowledge gap, approaches directly leveraging external knowledge, such as KIM (Chen et al. 2017a) using WordNet (Miller 1995), or indirectly transfer knowledge from other tasks, e.g., language modeling [ELMo (Peters et al. 2018), OpenAI Transformer (Radford et al. 2018)], machine translation [CoVe (McCann et al. 2017)] and discourse marker prediction [DMAN (Pan et al. 2018)], were proposed. We explore the feasibility of extending Gaussian Transformer with a pre-trained language model, i.e., ELMo. We follow the instructions (Peters et al. 2018), replacing word and character embeddings with ELMo and concatenating the output of encoding blocks with ELMo, and present the results in Table 5.

The ensemble version of Gaussian Transformer achieves state-of-the-art results among those approaches. However, compared with pre-trained word embeddings, the improvement by ELMo on SNLI (0.2) is less than that on MultiNLI (1.0/1.2). The reason might be that the training data in SNLI is larger than MultiNLI, where each genre contains only about 80k sentence pairs, while SNLI could be treated as a single genre including 550k examples. The best performing individual model, OpenAI Transformer, was trained from datasets containing billions of words on 8 GPUs for 1 month. In contrast, our single ELMo-integrated model obtains competitive results using a more lightweight architecture.

Evaluation on Hard NLI Benchmarks Annotation artifacts were found in the original NLI datasets and more challenging benchmarks, i.e., Hard SNLI and MultiNLI were proposed. As shown in Table 6, Gaussian Transformer improves about 5 percent on average on the Hard NLI datasets, suggesting that our model is less affected by the undesirable

Model	SNLI	M-ma	M-mis
KIM	88.6	77.2	76.4
CoVe	88.1	-	-
ESIM(ELMo)	88.7	-	-
DMAN	88.8	78.9	78.2
DMAN*	89.6	80.3	79.4
OpenAI Transformer	89.9	82.1	81.4
Gaussian Transformer(ELMo)	89.4	81.0	80.6
Gaussian Transformer*	**90.5**	**83.0**	**82.5**

Table 5: Experimental results of approaches using external resources. '*' indicates ensemble models. The ensemble Gaussian Transformer improves the state-of-the-art accuracy by 0.6 / 1.0 percent on SNLI / MultiNLI.

annotation artifacts. Integration with ELMo brings about 1.0 percent additional enhancement, showing the usefulness of knowledge transferred from language models.

Model	SNLI	M-ma	M-mis
DecAtt	69.4	55.8	56.2
ESIM	71.3	59.3	58.9
DIIN	72.7	64.1	64.4
Gaussian Transformer	**78.1**	**69.8**	**68.5**
Gaussian Transformer(ELMo)	**79.2**	**70.7**	**71.3**
Gaussian Transformer*	**79.9**	**73.0**	**72.8**

Table 6: Experimental results on the Hard NLI benchmarks. '*' indicates ensemble models. Gaussian Transformer gains an improvement of 5.0 percent accuracy on average.

Implementation Details

We implement our model using Tensorflow (Abadi et al. 2016), with the library tensor2tensor (Vaswani et al. 2018). All experiments are conducted on a single Nvidia Titan Xp GPU. The best performing individual model consists of $M = 3$ encoding blocks, $N = 2$ interaction blocks, using $H = 4$ heads attention with $d_{model} = 120$, $d_w = 300$, $d_c = 30$. Word embeddings are initialized from the pre-trained fasttext word vectors (Bojanowski et al. 2016), while character-level 5-grams embeddings are randomly initialized, and all embeddings remain fixed during training. We share the parameters of encoding and interaction blocks between *premise* and *hypothesis*, where the parameters at various depth, however, are different. Dropout (Srivastava et al. 2014) (rate $= 0.1$) is applied to all sub-layers. We employ the AdamWR algorithm (Loshchilov and Hutter 2017) to train our model on SNLI and MultiNLI separately, with batch size 64, learning rate range $[4E-5, 3E-4]$, normalized weight decay $w_{norm} = 1/600$, restarting term $T = 10$.

Related Work

Neural models became popular in NLI field since large annotated datasets, i.e., SNLI (Bowman et al. 2015) and MultiNLI (Williams, Nangia, and Bowman 2017), were released. The dominant trend of NLI models is employing deep complex neural models including RNNs (Chen et al. 2017b; Wang, Hamza, and Florian 2017; Tan et al. 2018; Liu, Duh, and Gao 2018; Ghaeini et al. 2018; Tay, Tuan, and Hui 2017; Kim et al. 2018), CNNs (Gong, Luo, and Zhang 2017), SANs (Parikh et al. 2016; Shen et al. 2017; Im and Cho 2017). Although obtained state-of-the-art results, those models often cost millions of parameters and lots of time to train.

Transformer was proposed by Vaswani et al. for machine translation as an encoder-decoder architecture, which based solely on attention mechanisms, eschewing recurrence and convolution. QANet (Yu et al. 2018) was the first model trying to improve the capability of Transformer to model local structure of texts, which utilizes convolution layers before self-attention to capture local dependency. However, a convolution layer could not model words outside current filter window and costs additional parameters. Besides QANet, Gaussian transformer is also inspired by the following two works: Parikh et al. first introduced a distance-sensitive bias term to attention mechanisms to capture sequential information of sentences. Im and Cho further restricted the distance-based bias to a linearly decaying manner. We also re-implemented those strategies as our baselines and found that Gaussian prior probability performs the best among these enhanced self-attention networks.

Besides training a neural model using only NLI datasets from scratch, approaches leveraging external resources (Chen et al. 2017a; Peters et al. 2018; Radford et al. 2018; McCann et al. 2017; Pan et al. 2018) emerged recently and further improved the performance of NLI. We also explored the feasibility of extending the Gaussian Transformer via a pre-trained language model, i.e., ELMo (Peters et al. 2018).

Gururangan et al. demonstrated that annotation artifacts in SNLI and MultiNLI inflated NLI models' performance, and proposed the Hard NLI benchmarks. We employ their datasets to evaluate our models besides the original ones.

Lastly, the lightweight character-level n-grams word encoding used in this work is inspired by (Tomar et al. 2017). They used the sum of character n-grams to represent tokens. We employ max-over-time pooling instead because in this way our model performs better in preliminary experiments.

Conclusion

In this paper, we propose a novel attention mechanism inspired by the 'chunking' phenomenon, i.e., Gaussian self-attention, which could better capture both local structure and global dependency of sequences without introducing recurrence or convolution.

Then we present an efficient NLI model named Gaussian Transformer, consisting of Gaussian encoding blocks, high-order interaction blocks and efficient comparison blocks, outperforming previous state-of-the-art approaches on both SNLI and MultiNLI benchmarks with significantly fewer parameters and considerably less training time.

Additional evaluation using the Hard NLI datasets demonstrates that the proposed approach is less affected by the undesirable annotation artifacts than previous works.

We also explore the feasibility of extending Gaussian Transformer using external resources, and obtain further improvement brought by the transferred knowledge from a pre-trained language model.

Acknowledgments

The paper is supported by the 973 Program (No. 2014CB340503) and National Natural Science Foundation of China (No. 61472105 and No. 61502120).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; and Isard, M. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, 265–283.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. *arXiv:1607.06450 [cs, stat]*. arXiv: 1607.06450.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching Word Vectors with Subword Information. *arXiv:1607.04606 [cs]*. arXiv: 1607.04606.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Chen, Q.; Zhu, X.; Ling, Z.-H.; Inkpen, D.; and Wei, S. 2017a. Natural Language Inference with External Knowledge. *arXiv:1711.04289 [cs]*. arXiv: 1711.04289.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; and Jiang, H. 2017b. Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference. *arXiv:1609.06038 [cs]*. arXiv: 1609.06038.

Dagan, I.; Roth, D.; Sammons, M.; and Zanzotto, F. M. 2013. Recognizing Textual Entailment: Models and Applications. *Synthesis Lectures on Human Language Technologies* 6(4):1–220.

Ghaeini, R.; Hasan, S. A.; Datla, V.; Liu, J.; Lee, K.; Qadir, A.; Ling, Y.; Prakash, A.; Fern, X. Z.; and Farri, O. 2018. DR-BiLSTM: Dependent Reading Bidirectional LSTM for Natural Language Inference. *arXiv:1802.05577 [cs]*. arXiv: 1802.05577.

Gong, Y.; Luo, H.; and Zhang, J. 2017. Natural Language Inference over Interaction Space. *arXiv:1709.04348 [cs]*. arXiv: 1709.04348.

Gururangan, S.; Swayamdipta, S.; Levy, O.; Schwartz, R.; Bowman, S. R.; and Smith, N. A. 2018. Annotation Artifacts in Natural Language Inference Data.

Im, J., and Cho, S. 2017. Distance-based Self-Attention Network for Natural Language Inference. *arXiv:1712.02047 [cs]*. arXiv: 1712.02047.

Kim, S.; Hong, J.-H.; Kang, I.; and Kwak, N. 2018. Semantic Sentence Matching with Densely-connected Recurrent and Co-attentive Information. *arXiv:1805.11360 [cs]*. arXiv: 1805.11360.

Liu, X.; Duh, K.; and Gao, J. 2018. Stochastic Answer Networks for Natural Language Inference. *arXiv:1804.07888 [cs]*. arXiv: 1804.07888.

Loshchilov, I., and Hutter, F. 2017. Fixing Weight Decay Regularization in Adam. *arXiv:1711.05101 [cs, math]*. arXiv: 1711.05101.

McCann, B.; Bradbury, J.; Xiong, C.; and Socher, R. 2017. Learned in Translation: Contextualized Word Vectors. *arXiv:1708.00107 [cs]*. arXiv: 1708.00107.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11):39–41.

Pan, B.; Yang, Y.; Zhao, Z.; Zhuang, Y.; Cai, D.; and He, X. 2018. Discourse Marker Augmented Network with Reinforcement Learning for Natural Language Inference. In *the 56th Annual Meeting of the Association for Computational Linguistics*, 11.

Parikh, A. P.; Tckstrm, O.; Das, D.; and Uszkoreit, J. 2016. A Decomposable Attention Model for Natural Language Inference. *arXiv:1606.01933 [cs]*. arXiv: 1606.01933.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv:1802.05365 [cs]*. arXiv: 1802.05365.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving Language Understanding by Generative Pre-Training. Technical report.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; and Zhang, C. 2017. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. *arXiv:1709.04696 [cs]*. arXiv: 1709.04696.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Stigler, S. M. 1982. A modest proposal: a new standard for the normal. *The American Statistician* 36(2):137–138.

Tan, C.; Wei, F.; Wang, W.; Lv, W.; and Zhou, M. 2018. Multiway Attention Networks for Modeling Sentence Pairs. In *IJCAI*, 4411–4417.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2017. A Compare-Propagate Architecture with Alignment Factorization for Natural Language Inference. *arXiv:1801.00102 [cs]*. arXiv: 1801.00102.

Tomar, G. S.; Duque, T.; Tckstrm, O.; Uszkoreit, J.; and Das, D. 2017. Neural Paraphrase Identification of Questions with Noisy Pretraining. *arXiv:1704.04565 [cs]*. arXiv: 1704.04565.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.

Vaswani, A.; Bengio, S.; Brevdo, E.; Chollet, F.; Gomez, A. N.; Gouws, S.; Jones, L.; Kaiser, \.; Kalchbrenner, N.; and Parmar, N. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. *arXiv:1702.03814 [cs]*. arXiv: 1702.03814.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *arXiv:1804.09541 [cs]*. arXiv: 1804.09541.