Final Project: Channel Combined Transformer in Machine Translation

Xuanzhou Chen, Siyuan Wang, Guoxuan Qin
Department of Electrical & Computer Engineering
New York University
Brooklyn, NY 11201
xc2425@nyu.edu sw5227@nyu.edu paulqin@nyu.edu

Abstract

In order to improve Transformer performance on large corpus in machine translation, we explored a new architecture design based on Transformer by inserting different types of channels (BEC(p), BSC(p)) between the encoder and the decoder. Moreover, additive white gaussian noise is to simulate the noisy situation. Our results reveals that the channel combined transformer significantly shorten the training time with very little loss in accuracy in both noisy and noise free conditions. In addition, channel combined transformer better preserves the semantic meaning than the original transformer. Github Link: https://github.com/xchen793/dl_final_project

1 Introduction

Seq2Seq is a model which has been used for machine translation a few years ago. Its basic model is built on Encoder-Decoder structure. There has been numerous research conducted on Seq2Seq model, such as Sutskever et al. [2014], Bahdanau et al. [2014], and Gehring et al. [2017].

To better handle with long distance dependency, a new model Transformer (Attention is all you need) was proposed by Vaswani et al. [2017b] to first completely abandon traditional GRU or LSTM, and largely introduce attention mechanism. However, Transformer still preserves the Encoder-Decoder structure as its basic model architecture.

After the transformer came out, there has been enormous research based on this model, and some quite impressive variants has been introduced to process long text dependency issue such as Transformer-XL Dai et al. [2019], and other models such as Explicit Sparse Transformer to simplify the attention mechanism by Zhao et al. [2019]. However, it is difficult to improve the Transformer's competence for dealing with large corpus. , According to Vaswani et al. [2017a], the traditional transformer architecture has this big drawback that it has poor performance to translate a large corpus due to its high computational cost.

To solve this problem, in this project, we combined the channel and Transformer to build a new model in machine translation. Specifically, Farsad et al. [2018] firstly defined their system model associated with transmitting sentences from a transmitter to a receiver. They put forward a encoder-channel-decoder scheme. Inspired by them, we come up with the idea to use channel on transformer in neural machine translation. Since deep learning algorithms are used, the channel must satisfy the property that allows backpropagation, otherwise the model cannot be trained. There are some communication channels, including the additive Gaussian noise channel, multiplication Gaussian noise channel and the erasure channel, which can be formulated using neural network layers. So far, we have defined our own channel function as binary erasure channel and binary symmetric channel, implemented them, plugged them between encoder and decoder of the traditional Transformer, and tested their

Preprint. Under review.

performance on validation accuracy and computational time using our English to French translation dataset. We also added the gaussian noise to simulate noisy conditions.

2 Methodologies

Compared to the standard transformer, we have two modifications: 1. we add channel modeling between the encoder and decoder to decrease computation time; 2. we add additive white gaussian noise to test the model performance in noisy condition. See the figures below for model structure comparison.

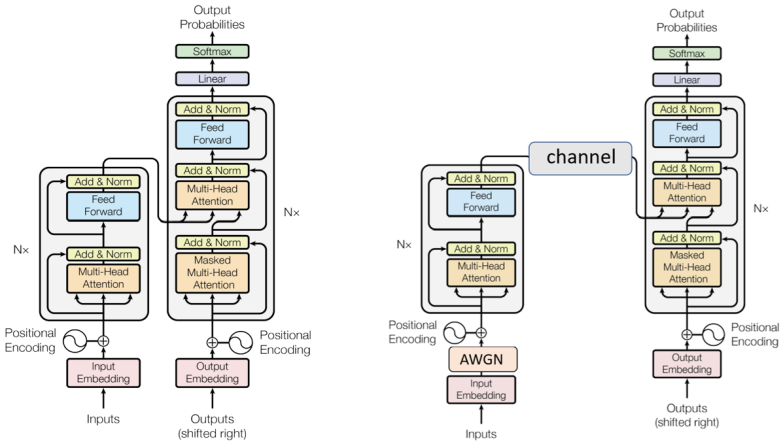

Figure: Standard Transformer

Figure: Channel combined Transformer

Figure 1: Model Structure

2.1 Channel Modeling in Transformer

There is some previous work contributed to noisy channel modeling for neural machine translation by Yee et al. [2019]. The researchers pointed out that one crucial advantage of the channel models is that they are particularly effective with large context sizes, which may potentially make up for the weakness of standard Transformer.

Inspired by the channel modeling in neural machine translation. We define our discrete memoryless binary channels as below. Since we use the layer normalization, for each token in each sentence $X^{N_{embdim}} = \{X_1, ..., X_{N_{embdim}}\}$, $X_i \in [0, 1]$, $X_i \sim i.i.d.$, is a vector with dimension equal to the embedding dimension. We denote N_{embdim} as N. The channel is defined as a function $g : \mathcal{X}^N \to \mathcal{Y}^N$. For different channels, function g is defined below:

Definition 2.1 (Function g for BSC(p))**.** For a discrete random variable $X \in [0, 1]$ and probability value $p \in [0, 1]$ function g is

$$g(X) = \begin{cases} X, & X \geq p \\ -X, & X < p \end{cases}$$

For a sequence X^N, $N \in \mathcal{N}$, the function $g(X^N) := (g(X_1), g(X_2), ..., g(X_N))$.

Remark. We also denote this channel with BSC(p), since we assume X_i is uniformly distributed. In this case, BSC(p) is actually a binary symmetric channel, which turns input X into $-X$ with cross probability equal to p.

Definition 2.2 (Function g for BEC(p))**.** For a discrete random variable $X \in [0, 1]$ and probability value $p \in [0, 1]$ function g is

$$g(X) = \begin{cases} X, & X \geq p \\ dropout, & X < p \end{cases}$$

For a sequence X^N, $N \in \mathcal{N}$, the function $g(X^N) := (g(X_1), g(X_2), ..., g(X_N))$.

Remark. We also denote this channel with BEC(p), since we assume X_i is uniformly distributed. In this case, BEC(p) is actually a binary erasure channel, which drops out input X with probability of erasure equal to p.

The channel is plugged between the encoder and the decoder. The number of paralleled channels equals to input sequence length * batch size.

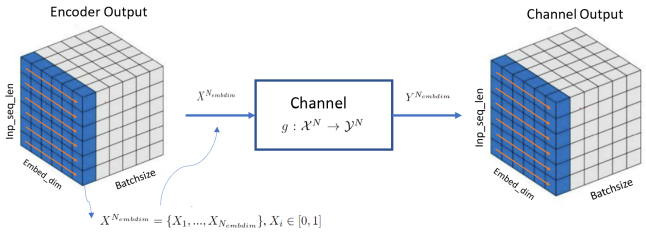

Figure 2: Channel Modeling

2.2 Additive Gaussian Noise Model

We choose additive white gaussian noise and put it after the input embedding step AWGN is an ideal noise signal that is easy to analyze. Generally, it can be assumed that the noise generated by the system or the interference of the noise signal is Gaussian white noise under a certain frequency band or limited condition.

3 Experiments

3.1 Experiment setup and data preprocessing

We use this Eng-Fre translation corpus (https://download.pytorch.org/tutorial/data.zip) for our dataset. The data is a collection of thousands of English-to-French translation pairs. Since this data is clean (i.e. without noise), we manually add gaussian noise using AWGN channel. In addition, each sentence pair, which consists of one English sentence and its corresponding target sentence in French, is independent from others. This means that there was no information loss between sentence pairs when we used the multi-head attention. In our training process, we set the maximal sentence length to be 50 tokens with padding for short sentences and truncation for long sentences. One-hot coding was used and the total word count is limited in thousands. We also used "teacher forcing" to faster the convergence of the model, and set the teacher forcing ratio to be 0.5.The data preparation is following: first, split the text files into lines,cand then split lines into pairs; second, normalize the text, and filter it by length and content; finally, make a words list from pairs of sentences.

3.2 Noise Free Experiments

In the Noise Free Experiments, we tested original Transformer and Transformers with BEC and BSC channel at batch size 64 and epoch 30. Table1 shows our Noise free experiment tested series.

For channel-Transformer's training and validation, we adopted Adam as optimizer (betas=(0.9, 0.98), eps=1e-9) with scheduled learning rate shown in Figure3. Noise free experiments ran on a Windows 11 machine with 32.00 GB (3200 MHz) RAM, NVIDIA GeForce RTX 3060 Laptop GPU (4095MB) and 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz CPU.

3.3 AWGN Experiments

In the experiment with Additive white Gaussian noise, based on different epoch, batch size , channel channel parameters, tests were conducted. The test combinations are shown in Table2. Same Adam optimizer and learning rate scheduler were implemented here. Different from noise free experiments, AWGN experiment ran on a Windows 10 machine with NVIDIA GeForce RTX 3070 Laptop GPU (8192MB), same CPU and RAM as Noise free experiments. The computation work for AWGN experiment is much heavier than Noise free experiments, therefore a graphic card with larger capacity was utilized.

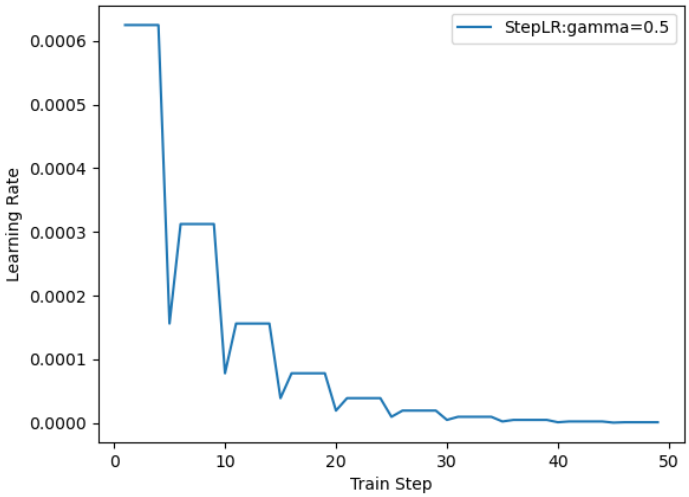

Figure 3: Scheduled Learning Rate

Table 1: Noise Free Experiments Tested Series

Model	Batch Size	Epochs
Transformer	64	30
Transformer + BSC(0.5)	64	30
Transformer + BSC(0.2)	64	30
Transformer + BEC(0.2)	64	30

4 Results

4.1 Validation Accuracy and Computational Time

According to model performance in Table 3, we found that, in noise free condition, BSC(0.5) has the highest accuracy, BSC(0.2) has the second highest accuracy, and the BEC(0.2) has the lowest accuracy. In gaussian noise condition, BSC(0.5) has the highest accuracy, BSC(0.2) has the second highest accuracy, and the BEC(0.5) has the lowest accuracy. From above observation, we can draw the conclusion that the model with highest accuracy in our experiment is BSC(0.5).

The validation accuracies are plotted for the baseline (transformer), transformer + BSC(0.2), transformer + BEC(0.2), transformer + BSC(0.5) in noise free condition(batch size = 64& epochs = 30)

Table 2: AWGN Experiments Tested Series

Model	Noise	Batch Size	Epochs
Transformer	AWGN	128	30
Transformer + BSC(0.5)	AWGN	128	30
Transformer + BEC(0.2)	AWGN	128	30
Transformer	AWGN	128	80
Transformer + BSC(0.5)	AWGN	128	80
Transformer + BSC(0.2)	AWGN	128	80
Transformer + BEC(0.5)	AWGN	128	80
Transformer	AWGN	256	30
Transformer + BSC(0.5)	AWGN	256	30
Transformer + BEC(0.5)	AWGN	256	30
Transformer + BEC(0.2)+ BSC(0.5)	AWGN	256	30

in Fig.7. The differences of accuracies of each variant compared to the baseline is shown in Fig.8. Similarly, the Fig.9 and Fig.10 are shown for the AWGN condition. From these four figures, we noticed that the validation accuracy for the BSC model is actually very close to that of the standard transformer after some certain epoch, in both conditions (noise free & AWGN).

What surprised us is that all channels greatly reduces the computation time to some extent (See Table 3). BEC decreases the computational time most significantly at the cost of dropping the model accuracy. However, BSC(0.5) has also effectively reduce computational time, but it still remain at high accuracy as the baseline.

The heat map (Fig. 11) shows that with multi-head attention, the model demonstrates its ability to focus on different locations in different sentence pairs. Moreover, the multi-head attention enables the model to focus on several nonconsecutive locations even they are far away.

Table 3: Computational Time

Model	Noise	Batch Size	Epochs	Time(s)
Transformer	AWGN	128	80	9650
Transformer + BSC(0.5)	AWGN	128	80	8949
Transformer + BSC(0.2)	AWGN	128	80	9101
Transformer + BEC(0.5)	AWGN	128	80	8906

Table 4: Model Performance

Model	Noise	Batch Size	Epochs	Accuracy
Transformer	/	64	30	0.819
Transformer + BSC(0.5)	/	64	30	0.816
Transformer + BSC(0.2)	/	64	30	0.813
Transformer + BEC(0.2)	/	64	30	0.808
Transformer	AWGN	128	80	0.822
Transformer + BSC(0.5)	AWGN	128	80	0.817
Transformer + BSC(0.2)	AWGN	128	80	0.819
Transformer + BEC(0.5)	AWGN	128	80	0.797

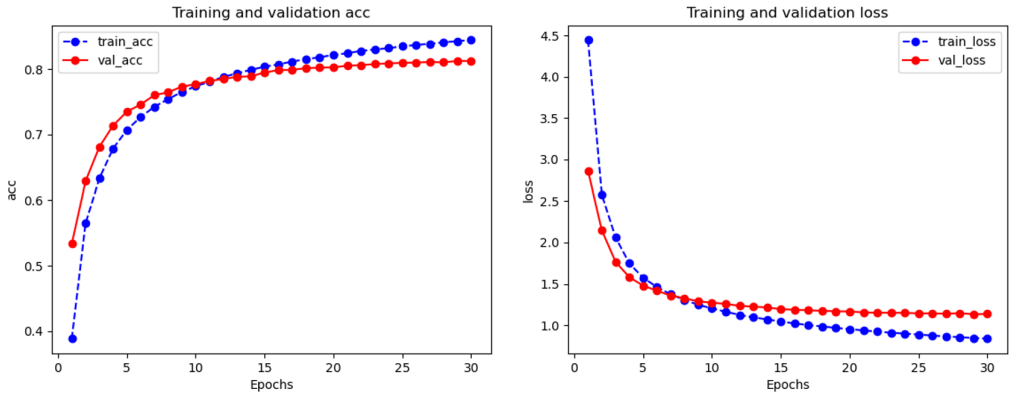

Figure 4: TF+BSC(0.2), Noise free, Accuracies Loss, Epoch 30

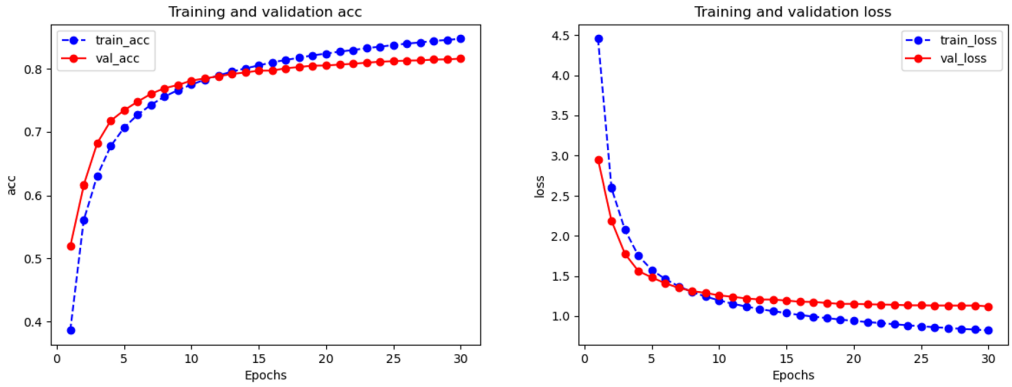

Figure 5: TF+BSC(0.5), Noise free, Accuracies Loss, Epoch 30

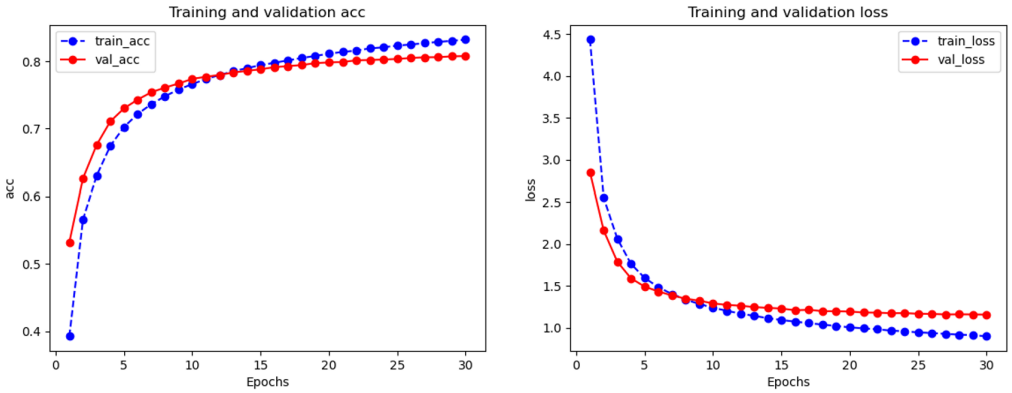

Figure 6: TF+BEC(0.2), Noise free, Accuracies Loss, Epoch 30

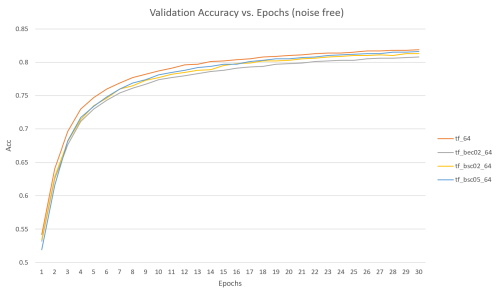

Figure 4: Val-Acc, Noisefree, Epoch 30

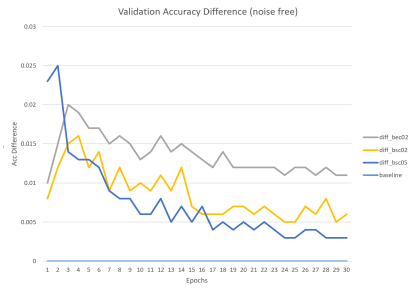

Figure 5: Val-Acc Difference, Noisefree, Epoch 30

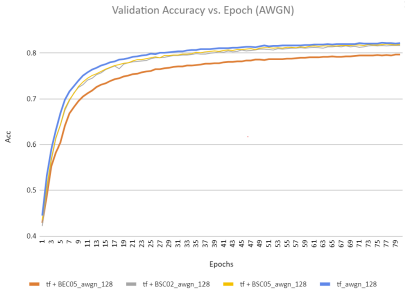

Figure 6: Val-Acc, awgn, Epoch 30

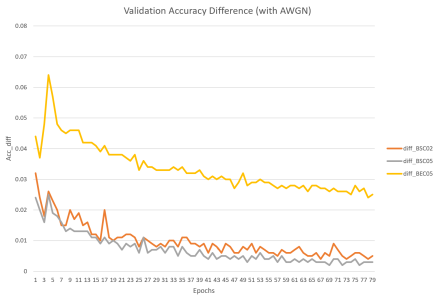

Figure 7: Val-Acc Difference, awgn, Epoch 30

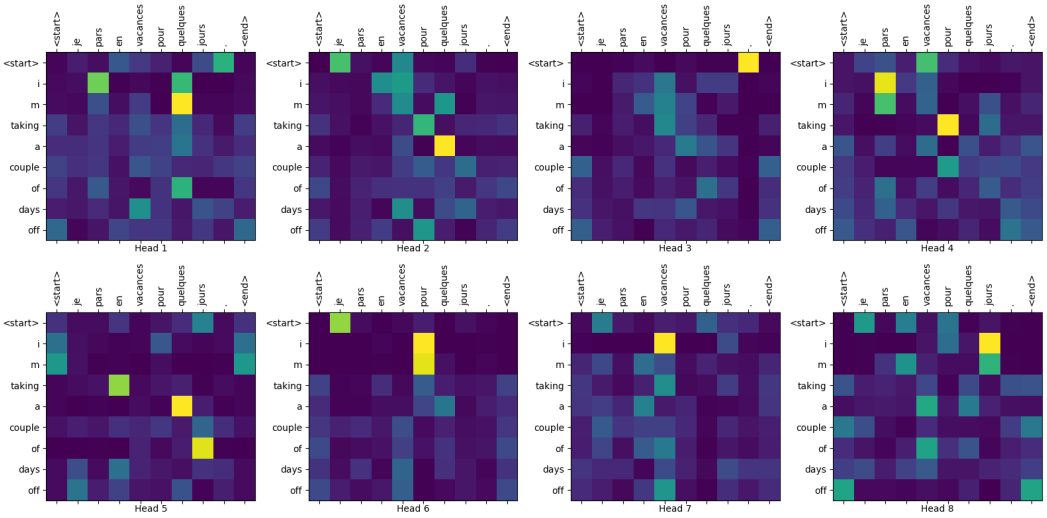

Figure 11: Transformer + BSC(0.5) Attention Visualization

4.2 Semantic Meaning Preservation

The loss and accuracy cannot accurately measure the semantic meaning preservation of the translation result and the best metric is a a human judge. In the previous results, the noisy channel model obtained very close testing accuracy to the original model. In this section, we compare the translation results of different settings of parameters manually human. Bellow are 10 samples of translation. We denote the prediction results of different parameter settings as pred(bec probability, awgn mean, bsc probability).

< Sample 1 >
input: je pars en vacances pour quelques jours .

target: i m taking a couple of days off .
pred(original): <start>i m taking a couple of days off .
pred(0.1, 0.1, 0.8): <start> i m taking a couple of days off .
pred(0.1, 0.0, 0.5): <start> i m taking a vacation .

< Sample 2 >
input: je ne me panique pas .
target: i m not panicking .
pred(original): <start> i m not panicking .
pred(0.1, 0.1, 0.8): <start> i m not panicking .
pred(0.1, 0.0, 0.5): <start> i m not panicking .

< Sample 3 >
input: je recherche un assistant .
target: i am looking for an assistant .
pred(original): <start> i am looking for an assistant .
pred(0.1, 0.1, 0.8): <start> i am looking for an assistant .
pred(0.1, 0.0, 0.5): <start> i m looking for a sweater .

< Sample 4 >
input: je suis loin de chez moi .
target: i m a long way from home .
pred(original): <start> i m a long way from home .
pred(0.1, 0.1, 0.8): <start> i m a long way from home .
pred(0.1, 0.0, 0.5): <start> i m tired of all this nagging .

< Sample 5 >
input: vous etes en retard .
target: you re very late .
pred(original): <start> you are very late .
pred(0.1, 0.1, 0.8): <start> you are late .
pred(0.1, 0.0, 0.5): <start> you re really absent minded .

< Sample 6 >
input: j ai soif .
target: i am thirsty .
pred(original): <start> i am thirsty .
pred(0.1, 0.1, 0.8): <start> i m thirsty .
pred(0.1, 0.0, 0.5): <start> i m done .

< Sample 7 >
input: je suis fou de vous .
target: i m crazy about you .
pred(original): <start> i m crazy about you .
pred(0.1, 0.1, 0.8): <start> i m crazy about you .
pred(0.1, 0.0, 0.5): <start> i m lucky to have you as a friend .

< Sample 8 >
input: vous etes vilain .
target: you are naughty .
pred(original): <start> you re bad .
pred(0.1, 0.1, 0.8): <start> you re bad .
pred(0.1, 0.0, 0.5): <start> you re grumpy .

< Sample 9 >
input: il est vieux et laid .
target: he s old and ugly .
pred(original): <start> he s old and ugly .
pred(0.1, 0.1, 0.8): <start> he s old and ugly .
pred(0.1, 0.0, 0.5): <start> he s old .

< Sample 10 >
input: je suis terrifiee .
target: i m terrified .
pred(original): <start> i m hit .
pred(0.1, 0.1, 0.8): <start> i m a desperate .
pred(0.1, 0.0, 0.5): <start> i m wet .

In the above prediction results, pred(original) achieved a 0.822 testing accuracy after 100 training epochs, and pred(0.1, 0.1, 0.8), pred(0.1, 0.0, 0.5) achieved 0.821 and 0.723 respectively. We can see the trade off between the semantic meaning preservation and the accuracy when the degree of channel interruption is high. The observation of the testing result of pred(0.1, 0.0, 0.5) indicates that when the model is trained with stronger noise, it translates with substitution of words in the target sentence (like in sample 8) or even substitution of phrases and paraphrase of the sentence (like sample 1 and 7). In the meantime it mostly preserved the semantic meaning. However, strong noise may corrupt the training sample and cause the model to learn from wrong information or fraction of information (like sample 3).

To further examine the performance, we compare different models using sentences with more complex grammar and semantic meaning.

< Sample 1 >
input: Vous allez le briser si vous ne faites pas attention .
target: You ll break it if you re not careful .
pred(original): <start> you re not going to do it .
pred(0.1, 0.1, 0.8): <start> you re not going to keep it .

< Sample 2 >
input: Je n aurais jamais pensé qu il y ait là un tel endroit tranquille dans cette ville bruyante .
target: I never dreamed of there being such a quiet place in this noisy city .
pred(original): <start> i m not thinking it was in a new ideas .
pred(0.1, 0.1, 0.8): <start> i m not thinking about this until now.

< Sample 3 >
input: Votre voix me semble très familière .
target: Your voice sounds very familiar to me .
pred(original): <start> i m really stranger to your voice .
pred(0.1, 0.1, 0.8): <start> i m very happy to hear your voice .

< Sample 4 >
input: Presque tout le monde est déjà rentré à la maison .
target: Almost everybody has already gone home .
pred(original): <start> i m almost home all the time .
pred(0.1, 0.1, 0.8): <start> we re almost to the same house .

According to the observation, the original model did better on preserving the correct grammar of the target language, but it did not necessarily preserves the correct meaning of the sentence. It could negate the meaning or refer to unrelated information. Although the noisy model did not preserve

grammar of the target language, it preserved the semantic meaning in a subtle way. The reader can infer the meaning from the prediction like solving a riddle.

5 Conclusion

In this project, we inserted Transformer with different channels (BEC(p), BSC(p)) between the encoder and the decoder. In addition, we added additive white gaussian noise in the training environment. Our results reveals that the binary symmetric channel combined transformer significantly shorten the training time with very little loss in accuracy. In the future, we will explore the channel combined Transformer's application in large text dataset and unpaired dataset.

For semantic meaning preservation, although the translation results are usually problematic when the sentence is complex, we demonstrated that the noisy model outperforms the original model, as recovery of the exact transmitted sentence may not be important as long as the semantic information of the sentence is conveyed.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL https://arxiv.org/abs/1409.0473.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019. URL http://arxiv.org/abs/1901.02860.

Nariman Farsad, Milind Rao, and Andrea Goldsmith. Deep learning for joint source-channel coding of text. *CoRR*, abs/1802.06832, 2018. URL http://arxiv.org/abs/1802.06832.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning, 2017. URL https://arxiv.org/abs/1705.03122.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017a. URL http://arxiv.org/abs/1706.03762.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017b. URL https://arxiv.org/abs/1706.03762.

Kyra Yee, Nathan Ng, Yann N. Dauphin, and Michael Auli. Simple and effective noisy channel modeling for neural machine translation. *CoRR*, abs/1908.05731, 2019. URL http://arxiv.org/abs/1908.05731.

Guangxiang Zhao, Junyang Lin, Zhiyuan Zhang, Xuancheng Ren, Qi Su, and Xu Sun. Explicit sparse transformer: Concentrated attention through explicit selection. *CoRR*, abs/1912.11637, 2019. URL http://arxiv.org/abs/1912.11637.