Deep Learning-based Frame and Timing Synchronization for End-to-End Communications

Hengmiao Wu*, Zhuo Sun and Xue Zhou

Key Laboratory of Universal Wireless Communications, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author’s e-mail: Hengmiao_wu@yeah.net

Abstract. End-to-end learning-based communications systems are more likely to achieve the global optimal performance. In this work, we implement a complete communications system as an end-to-end deep neural network, including transmitter, channel model, synchronization and receiver. There are a lot of issues which can result to the out of synchronization between the transmitter and the receiver, including sampling timing offset, sampling frequency offset and so on. Therefore, we focus on frame and timing synchronization in the end-to-end learning-based communications systems. Our results show that such an end-to-end learning-based communications system is robust to the impairments that arise from sampling frequency offset. An extra synchronization model based on the convolutional neural networks is introduced to achieve frame synchronization and compensate for the impairments which arise from sampling time offset and sampling time error. The performances of the synchronization model are evaluated from the symbol error rate and the frame header detection results. It can correctly detect most of the actual position of frame header, and it is about 2dB better than that of the direct correlation detection in the end-to-end communications systems.

1. Introduction

The fundamental goal of communications is reproducing at one point either exactly or approximately a message transmitted at another point [1]. An autoencoder learns to compress the input layer into smaller data and then uncompressing that data into something that closely matches the original data. Compared to the aim of communications that the receiver reconstructs the data the transmitter transmitting, the autoencoder has enormous potential in the communications.

There are a lot of researches on communications based on neural networks (NNs) which comprise Joint source-channel decoding [2], detection algorithms [3], synchronization [4] and compress sensing [5,6]. Recently, the end-to-end communications systems based on NNs were first introduced in [7] which interpreted the end-to-end communications systems as an autoencoder. In [8], the end-to-end communications systems were extended to multiple inputs multiple outputs (MIMO) communications systems. Sebastian et.al [9] built the complete end-to-end communications systems solely composed of NNs over the air using unsynchronized off-the-shelf software-defined radios (SDRs) and open-source deep learning (DL) software libraries. In [10], a novel method for synthesizing new physical layer modulation and coding schemes in communications systems using a learning-based approach which does not require an analytic model of the impairments in the channel was presented. Alexander et.al [11] implemented the orthogonal frequency division multiplexing (OFDM) systems with the cyclic prefix (CP) through the deep neural networks (NNs)-based autoencoder.
However, there are still some crucial issues to be solved in the end-to-end communications systems. Due to the out of synchronization between the transmitter and the receiver, the receiver may not retrieve the message because of unawareness of the beginning symbols of each frame. And also because of the different oscillators between transmitter and receiver, the receiver side may have the sampling frequency offset (SFO). When the sampling doesn’t align at the center of the samples, it will cause sampling timing error (STE) in addition. All those timing issues will have important influences on the performances of learning-based end-to-end communications systems.

In this work, we present a concise method based on convolutional neural networks (CNNs) to solve above mentioned issues. Compared to the traditional methods on those issues, the NNs-based method is more suitable for the learning-based end-to-end communications systems, which will make training whole systems possible.

The rest of the paper is organized as follows. In Section II, the end-to-end communications systems are introduced. Then, we formulate the synchronization problems in the end-to-end communications systems in Section III. Section IV describes the synchronization model. Experiments results are provided in Section V. Conclusions are provided in Section VI.

2. Deep Learning-based communications systems model

In digital communications systems, data are modulated into a series of symbols for transmission over the channel at the transmitter and the receiver tries to make the recovered data equal to the source data as much as possible. Intuitively, if there are more symbols for the constant amount of data, the receiver is more possible to recover the data. The transmitter tries to modulate more data into one symbol for a higher transmission rate, but the receiver aims to reconstruct the source data accurately as much as possible. Therefore, there is a tradeoff between the throughput and reliability. The better way to reach the global optimal result is to design the communications systems to be autoencoder-based learning systems [9]. Considering the end-to-end communications systems consisting of a transmitter, channel, and a receiver, as shown in Figure. 1, the receiver aims to recover the data that the transmitter transmitted over the impaired channel. The autoencoder network consists of an encoder and a decoder. The encoder is used to construct a compressive representation of data, while the decoder is used to reconstruct the original data from the compressed data.

2.1. Channel Model

Another major advantage of end-to-end communications systems is that it can learn the modulation and demodulation according to properties of the channel applied in training. The channel model applied in this paper is referenced in [9]. The properties of the channel are as follows: upsampling and pulse shaping, constant sample time offset, constant phase offset and carrier frequency offset, additive white Gaussian noise. Such a stochastic channel model contains the major features of the real communications channel, to make the end-to-end communications systems more meaningful.

Figure 1. Block diagram for the end-to-end communications systems

3. Modeling of frame and timing offset

3.1. Frame Model

In digital communications systems, a word is usually composed of a number of symbols, and a number of words form a sentence, i.e., constituting a "frame" for transmission. And out of synchronization on the frame location may lead to the misunderstanding at the receiver. Therefore, it is necessary to detect
the beginning and end of the frame. As for the continuous transmission, once the beginning of the frame is detected, the end of the frame will be also detected. And the frame message is uncertain and random, so it’s nearly impossible to detect the beginning of the frame based on the frame message. For the successful frame transmission, it is indispensable for the transmitter to transmit extra specific data at the beginning of each frame which is called frame header. The data frame structure is shown in Figure 2. The process of detecting the frame header is how to achieve frame synchronization.

3.2. Timing Offset
Sampling Time Offset (STO)
Because the transmitter and the receiver clocks are out of perfect synchronization, there is a certain sampling time offset (STO) at the receiver. And the STO can be caused by two parts; sampling frequency offset (SFO) and sampling timing error (STE). SFO means the clock bias between the transmitter and the receiver, and STE implies that the sampling doesn’t align at the center of the samples.

Sampling time error.
Taking the sampling process in the communications systems into account, the continuous signals are multiplied by the sample function $P(t)$ ($P(t)=\sum_{n=-\infty}^{\infty} \delta(t - nT)$) to obtain the discrete signal. $T$ is the sampling period. When there is a constant STE $\tau_{\text{off}}$, it means there is a constant offset $\tau_{\text{off}}$ in the sample function $P(t)$, e.g., the continuous signal is multiplied by $P(t-\tau_{\text{off}})$ to obtain the discrete signal. In this work, we use two-phase sampling processes to model the STE during the pulse-shaping step at the transmitter side. We first use a higher up-sampling rate $\gamma'$ than the expected $\gamma$ during the pulse-shaping, which means we can get more samples than the expected (phase I). When the $\gamma'$ is large enough, we can approximate the discrete signal up-sampled by $\gamma'$ as a continuous-time signal. And in order to get the expected number of samples, it is necessary to down-sampling the pulse-shaping samples and during the down-sampling, there is a constant STE (phase II). According to the relationship of $\gamma$ and $\gamma'$, we can obtain the down-sampling rate $\gamma'/\gamma$. Only if set the appropriate up-sampling and down-sampling rate, can the impairments of STE be perfect to be modelled.

Sampling frequency offset
SFO occurs when the oscillators at the transmitter and the receiver run at slightly different frequencies. Over a long transmission period, it will result that there will be more or fewer IQ-samples recorded at the receiver than those that have been transmitted, e.g., for an oscillator offset of 100 parts per million (ppm) between the transmitter and the receiver and a sampling frequency of 2 MHz, there are 200 more or fewer IQ-samples per second recorded at the receiver than the transmitted by the transmitter. In order to model the SFO in the end-to-end communications systems, we randomly choose to add or delete one IQ-sample every N IQ-samples which results in that there is a different number between IQ-samples transmitted at the transmitter and those recorded at the receiver.

4. Deep learning-based frame and timing synchronization

4.1. Synchronization model
As described in section 3, the impairments of STO arise from SFO and STE. To compensate for the impairments of STE, the straightforward method is to estimate the $\tau_{\text{off}}$. And we have tried to add extra NNs to estimate the $\tau_{\text{off}}$ during the end-to-end communications systems training, but there is no extra performance. In other words, the decoder network is robust enough to compensate for the impairments of STE so we do not need such extra NNs to deal with STE.

Due to the impairments of SFO, the number of IQ-samples recorded at the receiver may be larger or smaller than that of transmitted, leading to the locations of frame headers between the transmitter and the receiver mismatch. The decoder needs the certain number of IQ-samples to decode one complete frame, but the locations of IQ-samples for one complete frame recorded at the receiver may be not the expected ones at the transmitter because of the SFO. Therefore, the decoder needs to search the specific locations of IQ-samples for per frame to decode, which implies that some IQ-samples
recorded at the receiver may be repeatedly selected and some may be ignored during decoding without resampling at the receiver. What’s more, the time for visible impairments of SFO is much greater than the transmitting time for one frame, so we do not take the SFO into account within one frame. From the above analysis, we can conclude that we just need to search the location of frame headers to compensate for the impairments of SFO. The task is very similar to the goal of frame synchronization, searching the frame header of each frame. Therefore, we try to handle it in the frame synchronization model instead of adding extra NNs for the equalization of SFO to simplify the model.

Before the decoder decodes the received data, it needs to detect the location of the frame header. And the frame header is a series of specific constant data, the task is to detect them despite the impairments of the channel which is similar to the application of CNNs on ImageNet classification [12]. And the application of CNNs motivates us to apply CNNs to detect the frame header. We define \( N_{\text{header}} \) as the number of symbols per frame header and \( N_{\text{message}} \) as the number of symbols per frame message. After pulse-shaping, there will be \( \gamma^* (N_{\text{header}} + N_{\text{message}}) \) symbols per frame at the receiver, i.e., there is one and only one frame header every \( \gamma^* (N_{\text{header}} + N_{\text{message}}) \) symbols. The symbols with IQ-samples can be seen as two-channel data, I-samples for one, and Q-samples for another. Intuitively, the accuracy of detection depends on the content of the frame header, e.g., if the frame header equals to some frame messages, it will mislead the synchronization model to find the wrong location of the frame header. The design principle of the frame header is to try to be different from the frame message as much as and to make the synchronization model easy to detect. Also, having tried a lot of different frame header, we use the sequence with alternating positive and negative 1s as the frame header at last.

The ANNs model for frame synchronization is composed of only one CNNs layer with softmax activation function with a negligible number of trainable parameters to achieve the frame synchronization and SFO compensation. Synchronization model is embedded in the end-to-end communications systems to evaluate its performance.

First of all, referring to the architecture in [9], the end-to-end communications systems model is built based on autoencoder. But at the output layer of the encoder NNs, activation function is not applied. According to the property of relu activation function, its outputs are greater than zero or equal to zero which leads to the symbols only located in the first quadrant of the constellation diagram. In this case, it is not an efficient encoder method from the signal space perspective. Therefore, there is no activation function in the output layer of the encoder to make the symbols possibly locate any quadrant. Indeed, the experiments also show that the autoencoder without activation function applied in the output layer of encoder performs better than that with activation function. The architecture of the end-to-end communications systems with a synchronization model is showed in Figure.3.

**Figure 3.** The architecture of the synchronization model in the end-to-end communications systems

5. Experiments
The systems parameters used in the training are described as follows: sequence length of one message is 16; frame header size is 8; frame message size is 110; up-sampling factor is 4; down-sampling factor is 10; roll-off factor is 0.35; filter span is 31; number of messages is 256 and complex symbols
The autoencoder is trained over the stochastic channel model using BP with Adam optimizer with learning rate exponential decay and Es/N0=8dB, and the batch normalization is also used during training. The synchronization model was trained together with the end-to-end communications systems to obtain the global optimal result.

5.1. Training of Synchronization Model
The synchronization model is to solve the issues of SFO and frame synchronization, therefore, the loss function should take the influences of SFO and frame synchronization into account. The location of frame header can be any possible value, but the offset caused by SFO can be only one of {-1,0,1} for one frame, because the transmitting time of one frame is much smaller than the time required for SFO to generate impairments, implying that there are no impairments caused by SFO within one frame, e.g., with an SFO of 100 ppm and a sample frequency 2MHz, one expects to repeat/neglect one sample every 20,000 samples. And γ*(N_{header}+ N_{message}) is much smaller than this value, the aforementioned conditions hold.

\[ L = -\alpha_1 \log(y_{ih-1}) - \alpha_2 \log(y_{ih}) - \alpha_3 \log(y_{ih+1}) \]  \hspace{1cm} (1)

\( y \) is the output of the softmax layer, and \( ih \) is the ideal location of the frame header, \( \alpha_i \) is the hyperparameters should be manually set. And the \( y_{ih} \) is much more likely to appear than the other two, so we make the \( \alpha_1 \) larger than other two. We have searched many different models with deeper CNNs and combination of CNNs and NNs, but that doesn’t get any better performance than the above simple CNNs architecture. The loss function \( L \) is the cost to be minimized through Adam optimizer during the training, and the Moving-Average Model is applied to all trainable parameters. We set the learning rate as an exponentially decaying variable

![Figure 4](image-url). The performance of the synchronization model

![Figure 5](image-url). SER performance of the different detection methods in end-to-end communications systems

It can be seen from the Figure. 4 that most of the predicted positions equal the actual positions and some have offsets. Due to the impairments of SFO, the number of IQ-samples recorded at the receiver do not equal to the number of transmitted, resulting to some frame headers offset. It’s surprising that some of the positions predicted by synchronization model also have offset, and we infer that these offsets in predicted values are caused by SFO, e.g., the synchronization model can not only achieve the frame synchronization but also solve the SFO issues.

5.2. Symbol Error Rate
The goal of communications is to reconstruct the transmitted data at the receiver, therefore the symbol error rate (SER) performance is crucial metrics for any communications blocks. In order to evaluate the SER performance of the synchronization model, we embed the synchronization model in end-to-end communications systems. Figure.5 shows the SER results comparing three cases: synchronization model detection, perfect detection and direct correlation detection over the same stochastic channel model in end-to-end communications systems. Although the parameters of the whole systems are the
result of training at a fixed $E_s/N_0=8$dB, the communications systems are able to generalize to a wide range of different $E_s/N_0$. Figure 5 shows that the performances of synchronization model detection and the perfect detection are almost the same whether the $E_s/N_0$ is large or small. It implies that the synchronization model achieves near-perfect performance and shows great robustness to $E_s/N_0$. What’s more, it can be seen that the performance of synchronization model detection is better than the direct detection. And when the $E_s/N_0$ becomes larger, the synchronization model detection performs much better compared the direct detection which implies the CNNs-based synchronization model can learn more from the end-to-end communications systems and obtain better performance. when the $E_s/N_0$ becomes larger and larger, the performance difference between the synchronization model detection and the direct detection is becoming more and more obvious. When the SER reaches $10^{-2}$ or smaller, the synchronization model detection performs about 2dB better than the direct.

6. Conclusions

In this work, a synchronization model for end-to-end communications systems is introduced. The model can not only achieve frame synchronization but also compensate for the impairments of STO. Compared to the direct detection in end-to-end communications systems, the synchronization model performs about 2dB better when the $E_s/N_0$ reaches to 8dB or larger. However, the differences between the stochastic channel model and the real channel may lead to a different performance. In the future work, we will try to build a channel model composed of NNs whose properties are closer to the real channel.

References

[1] C. E. Shannon 1948 A mathematical theory of communications Bell Syst.Tech. Journal Volume. 27 pp. 379–423 623–656.
[2] Nariman Farsad, Milind Rao, Andrea Goldsmith Deep Learning for Joint Source-Channel Coding of Text unpublished.
[3] Nariman Farsad, Andrea Goldsmith Detection algorithms for communications systems using deep learning unpublished.
[4] Timothy J O’Shea, Latha Pemula, Dhruv Batra, T. Charles Clancy 2016 Radio transformer networks: attention models for learning to synchronize in wireless systems 50th Asilomar Conf. on Signals, Systems and Computers.
[5] Mark Borgerding, Philip Schniter 2016 Onsager-corrected deep learning for sparse linear inverse problems IEEE Global Conference on Signal and Information Processing (GlobalSIP).
[6] Ali Mousavi and Richard G. Baraniuk 2017 Learning to invert: Signal recovery via deep convolutional networks IEEE International Conf. on Acoustics, Speech and Signal Processing (ICASSP).
[7] Timothy J O’Shea, Kiran Karra, T. Charles Clancy 2016 Learning to communicate:Channel auto-encoders, domain specific regularizers, and attention IEEE Int. Symp. Signal Process. Inform. Tech. (ISSPIT), pp. 223–228.
[8] Timothy J. O’Shea, Jakob Hoydis 2017 An Introduction to Deep Learning for the Physical Layer IEEE Transactions on Cognitive Communications and Networking Volume 3 p563-575.
[9] Sebastian Döörner, Sebastian Cammerer, Jakob Hoydis, Stephan ten Brink 2017 Deep learning-based communications over the air IEEE Journal of Selected Topics in Signal Processing Volume 12 132-143.
[10] Timothy J. O’Shea, Tamoghna Roy, Nathan West, Benjamin C. Hilburn Physical Layer Communications Systems Design Over-the-Air Using Adversarial Networks unpublished.
[11] Alexander Felix, Sebastian Cammerer, Sebastian Döörner, Jakob Hoydis, Stephan ten Brink OFDM-Autoencoder for End-to-End Learning of Communications Systems unpublished.
[12] Alex Krizhevsky, Ilya Sutskever, Geoffrey E.Hinton 2012 ImageNet Classification with Deep Convolutional Neural Networks Neural Information Processing Systems Conference.