CISC7021 - Applied Natural Language Processing

Group 48: Feng Siying, Yang Hao, Zhang Mingtengyue

Review Report, 2023

Abstract

The scholarly article titled "Encoder-Decoder Methods" provides an in-depth examination of Neural Machine Translation (NMT) systems, which are predicated exclusively on neural network architectures. NMT frameworks consist of two components: an encoder that distills a fixed-length vectorial representation from an input sentence of arbitrary length, and a decoder that crafts an accurate translation predicated upon this vectorial representation.

Here provides a comparative analysis of two architectures: the RNN Encoder-Decoder and the Gated Recurrent Convolutional Neural Network (grConv). It found the efficacy of NMT in processing succinct sentences. Although it also notes that with the increase in sentence length and the occurrence of unknown lexemes, the accuracy of translation sharply declines. Furthermore, the grConv model is even capable of autonomously parsing and integrating the grammatical structures within sentences.

1 Introduction

Machine Translation (MT) is a task aimed at automatically translating text from one language to another, playing a crucial role in globalization and cross-cultural communication. With the rapid advancement of information technology and the driving force of globalization, the demand for machine translation has been increasing as it can overcome language barriers and enhance communication efficiency among people.

Before the emergence of Neural Machine Translation (NMT), traditional rule-based machine translation methods, also known as Statistical Machine Translation (SMT), were widely used. SMT employs language models to capture the probability distribution of words or phrases in both the source and target languages. These models estimate the likelihood of generating a specific translation in the context of the source sentence. SMT has achieved success in many translation tasks and has made valuable contributions to the field. However, it also has limitations, such as difficulties in handling long-range dependencies and capturing complex linguistic phenomena.

Neural Machine Translation (NMT) has emerged as the current mainstream approach in machine translation, showcasing significant progress. Unlike traditional rule-based machine translation methods, NMT utilizes neural network models to directly model the mapping relationship between source language sentences and target language sentences, enabling an end-to-end translation process.

NMT directly maps source language sentences to target language sentences using a neural network model. It learns the complex mapping relationship between the source and target languages through end-to-end training, without relying on traditional rule-based or statistical methods. The encoder-decoder structure of NMT consists of an encoder and a decoder. The encoder converts the source sentence into a fixed-length vector representation, capturing its semantics and contextual information. The decoder generates the translation based on the encoder's semantic representation. It predicts each word or character step by step, combining the context information of previous predictions. This process continues until a complete target language sentence is generated. NMT's end-to-end modeling approach enables it to capture

semantic and contextual information more effectively, resulting in more accurate and fluent translation.

In general, SMT may encounter difficulties when dealing with long sentences. Since SMT primarily relies on phrase or word-level alignments, the translation quality of SMT may not meet expectations for sentences with long-range dependencies. On the other hand, NMT performs better in handling long sentences. Due to its end-to-end modeling using neural network models, NMT can better capture long-distance semantic relationships, thus improving translation quality.

This project aims to reproduce and evaluate the Gated Recursive Convolutional Neural Network, an NMT model based on an encoder-decoder architecture with gate mechanisms for selective information storage and forgetting. The project aims to improve the model in terms of accuracy, model size, or runtime. Approaches for improvement may include incorporating attention mechanisms, introducing skip connections, and modifying the RNN model structure.

2 Methodology

Neural Machine Translation (NMT) based on pure neural networks is essentially an Encoder-Decoder architecture, typically composed of an encoder and a decoder. The encoder extracts a fixed-length vector representation from a variable-length input sentence, while the decoder generates the correct translation based on this vector. Here, two models are primarily discussed: the RNN and the grConv models.

2.1 Encoder-Decoder Approach

Encoder: This part of the network is responsible for processing the input sequence, compressing the information from the sequence into a fixed-size context vector (usually the hidden state of the last time step). The encoder can be an RNN or its variants (such as LSTM or GRU), which are capable of efficiently handling input sequences and capturing temporal dependencies. The first model discussed in this paper uses an RNN, while the second employs a grConv.

Decoder: The decoder part receives the context vector generated by the encoder and begins to produce the output sequence. In both models discussed in this article, the decoder uses an RNN, which may refer to the context vector at each time step. It generates the output sequence step by step, with each time step's output depending on the output and hidden state of the previous step.

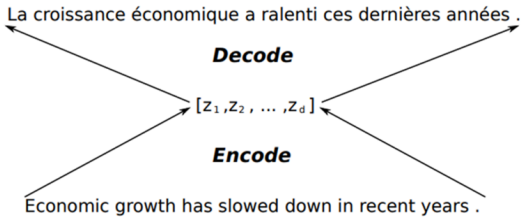

Figure 1: The Encoder-Decoder architecture

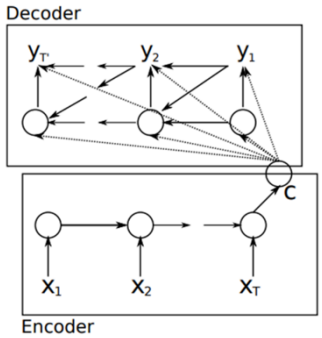

Figure 2: An illustration of RNN Encoder-Decoder

General workflow

1. Input Processing:

The input sequence is fed into the encoder, where each element is processed in turn.

2. Context Vector Generation:

The encoder processes the input sequence through the states of its hidden layers, producing a context vector.

3. Sequence Generation:

The decoder begins to generate the output sequence based on the context vector.

4. Step-by-Step Decoding:

At each time step, the decoder predicts the next output symbol and uses it as the input for the next time step.

5. End of Sequence Generation:

The generation of the sequence is terminated once the decoder produces a special end symbol or reaches the maximum length limit.

2.2 Model 1: Recurrent Neural Network with Gated Hidden Neurons

Preliminary: Recurrent Neural Networks

RNN processes variable-length sequences $x = (x_1, x_2, \ldots, x_T)$ by maintaining a hidden state h over time. The hidden state h is updated at each time step using the following formula.

$$h^{(t)} = f\left(h^{(t-1)}, x_t\right) \tag{1}$$

f is an activation function which often involves a linear transformation of the input vectors followed by a sum and then the application of an element-wise logistic sigmoid function.

The RNN is capable of effectively learning distributions over variable-length sequences by learning the distribution of the next input in the sequence $[p(x_{t+1}|x_t, \ldots, x_1)]$. A multinomial distribution (1-of-K coding) can be output using asoftmax activation function as:

$$p(x_{t,j} = 1 \,|\, x_{t-1}, \ldots, x_1) = \frac{exp(w_j h_{<t>})}{\sum_{j'=1}^{K} exp(w_{j'} h_{<t>})} \tag{2}$$

For every potential symbol within the range of $j = 1, \ldots .K$, where w_j represents the individual row vectors of the weight matrix W, the following expression gives the joint distribution.

$$p(x) = \prod_{t=1} Tp(x_t \,|\, x_{t-1}, \ldots, x_1) \tag{3}$$

Hidden Unit that Remembers and Forgets

This is a new variant of RNN that employs a gating mechanism, inspired by the LSTM unit, has been simplified so that can be easy to compute and implement, and it is better suit our needs and more effectively handle long-term dependencies.

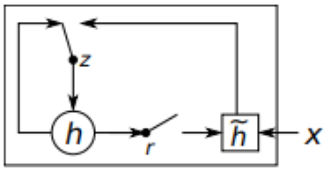

Figure 3: An illustration of hidden activate function

In this model, the state update of each hidden unit at every time step depends on two gating mechanisms:

Reset gate r: This determines how much of the previous hidden state will be considered. When r approaches 0, the previous hidden state is almost ignored. The activation of the j-th hidden unit r_j is computed by:

$$r_j = \sigma\left([W_r x]_j + [U_r h_{(t-1)}]_j\right) \tag{4}$$

Update gate z: This controls the extent of the update to the hidden state. When z is close to 1, the old state is nearly retained; when z is close to 0, the new candidate state will completely influence the hidden state. The update gate z is computed by:

$$z_j = \sigma\left([W_z x]_j + [U_z h_{(t-1)}]_j\right) \tag{5}$$

Actual activation $\widehat{h}$: The new hidden state is calculated through a combination of the update gate z,

reset gate r, the current input, and the previous hidden state. The actual activation of the proposed unit h is then computed by:

$$h_j^{(t)} = z_j h_j{}^{(t-1)} + (1 - z_j)\tilde{h}_j^{(t)} \tag{6}$$

where

$$\tilde{h}_j^{(t)} = \phi\left([Wx]_j + [U\left(r \odot h_{(t-1)}\right)]_j\right) \tag{7}$$

This approach essentially allows the hidden state to retain only that information, which is deemed relevant for the future, enabling a more compact representation. On the other hand, the update gate controls how much information from the previous hidden state should be carried over to the current hidden state. This helps the RNN to remember long-term information and acts in a role similar to the memory cells in an LSTM network. It could be seen as an adaptive form of a leaky integrator unit.

Since each hidden unit has its own reset and update gates, each unit will learn to capture dependencies on different time scales. Those units with frequently activated reset gates will tend to capture short-term dependencies, while those with active update gates will tend to capture long-term dependencies. In our preliminary experiments, we found that without any gating, we were unable to achieve meaningful results[1].

2.3 Model 2: Gated Recursive Convolutional Neural Network

Another natural method to handle variable-length sequences is to use a recursive convolutional neural network, in which the parameters of each layer are shared throughout the entire network. The paper introduces a network structure that combines the characteristics of Recursive Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), whose weights are recursively applied through the input sequence until a single fixed-length vector is outputted. In addition to the usual convolutional architecture, the model also introduces a gating mechanism, which allows the network to learn the structure of sentences while processing the input sequence.

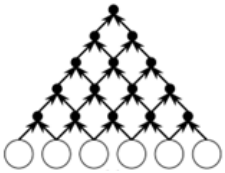

Figure 4: The recursive convolutional neural network

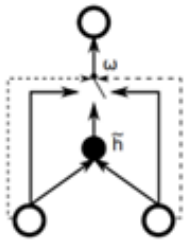

Figure 5: Gated unit for RCNN

Network Architecture

This network also maintains a hidden state h over time to process variable-length sequence $x = (x_1, x_2, \ldots, x_T)$.

The network architecture include four weight matrices W^l, W^T, G^l, G^T, which are shared across different levels of the network. For each recursion level $t \in [1, T-1]$, the activation of j-th hidden unit will be computed by:

$$h_j^{(t)} = w_c \tilde{h}_j^{(t)} + w_l h_{j-1}^{(t-1)} + w_r h_j^{(t-1)} \tag{8}$$

Noted that w_c, w_l, w_r sum to 1 which are the values of a gater.

And the hidden unit is initialized by:

$$h_j^{(0)} = U x_j \tag{9}$$

where U is a matrix to project the input into a hidden space.

Working principle of the gating unit:

The new activation $\tilde{h}_j^{(t)}$ is computed by:

$$\tilde{h}_j^{(t)} = \phi\left(W^l h_{j-1}^{(t)} + W^r h_j^{(t)}\right) \tag{10}$$

where ϕ is an element-wise nonlinear function.

The gating coefficient w are computed by:

$$\begin{bmatrix} w_c \\ w_l \\ w_r \end{bmatrix} = \frac{1}{Z}\left[\exp\left(G^l h_{j-1}^{(t)} + G^r h_j^{(t)}\right)\right]_k \tag{11}$$

Where G^l, $\mathrm{G}^r \in R^{3\times d}$ and Z is the normalization factor, ensuring that the sum of w equals 1, which is computed by:

$$Z = \sum_{k=1}^{3}[exp(G^l h_{j-1}^{(t)} + G^r h_j^{(t)}]_k \tag{12}$$

Based on this activation, the activation of a node at the recursive level t can be seen as selecting new components of activation, which can be obtained either from the left child node or the right child node or maintaining the activation from the left child node or obtaining activation from the right child node. This selection allows the overall structure of the recursive convolution to adaptively change with the input[2].

2.4 Experimental results

Finally, the performance is tested using the trained network for French-to-English translation. According to the analysis, as the length of the source sentence increases, the performance of the NMT model rapidly declines. Additionally, we find that the size of the vocabulary has a significant impact on translation performance. Nevertheless, in terms of quality, both models are mostly capable of generating correct translations most of the time. Moreover, the newly proposed grConv model can learn some syntactic structure of the source language in an unsupervised manner.

3 Related Work

Tomas Mikolov et al. presented several improvements that make the Skip-gram model more expressive and enabled it to learn higher quality vectors more rapidly. They showed that by subsampling frequent words they obtained significant speedup, and also learned higher quality representations as measured by their tasks[3].

Will Y. Zou et al. introduced bilingual word embeddings: semantic embeddings associated across two languages in the context of neural language models. They proposed a method to learn bilingual embeddings from a large unlabeled corpus, while utilizing MT word alignments to constrain translational equivalence. The new embeddings significantly outperformed baselines in word semantic similarity[4].

Stanislas Lauly et al. investigated an autoencoder model for learning multi-lingual word representations that does without word-level alignements. Their autoencoder was trained to reconstruct the bag-of-word representation of given sentence from an encoded representation extracted from its translation. In their experiments, they observed that their method compared favorably with a previously proposed method that exploitd word-level alignments to learn word representations[5].

Jean Pouget-Abadie et al. proposed a way by automatically segmenting an input sentence into phrases that could be easily translated by the neural network translation model to address an issue that there was a significant drop in translation quality when translating long sentences. Once each segment had been independently translated by the neural machine translation model, the translated clauses were concatenated to form a final translation. Empirical results showed a significant improvement in translation quality for long sentences[6].

Dzmitry Bahdanau et al. conjectured that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and proposed to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that were relevant to predicting a

target word, without having to form these parts as a hard segment explicitly. With this new approach, they achieved a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation[7].

Xinchi Chen et al. proposed a gated recursive neural network (GRNN) to model sentences, which employed a full binary tree (FBT) structure to control the combinations in recursive structure. By introducing two kinds of gates, their model could better model the complicated combinations of features. Experiments on three text classification datasets showed the effectiveness of their model[8].

4 Experimental Design

In this section, we will describe the experimental design that we have currently established, along with the rationale behind it and the expected outcomes.

4.1 Dataset

The original dataset used in the paper is a French-English dataset. It was compiled by combining data from multiple sources, including Europarl (61 million words), news commentaries (5.5 million words), the United Nations (4.21 billion words), and two web-crawled corpora (90 million words and 78 million words), resulting in a total of 348 million sentence pairs.We will replicate the training and testing process using the same French-English dataset as mentioned in the original paper. This step aims to validate the results reported in the paper using the identical experimental setup. Following the replication, we will acquire a Chinese-English dataset from the First Conference on Machine Translation (WMT16). This dataset will be used to train and test the model, allowing us to evaluate its generalizability and applicability to Chinese-English translation tasks. By including both the replication of the original dataset and the expansion to a Chinese-English dataset, we aim to assess the model's performance across different language pairs and evaluate its suitability for both French-English and Chinese-English translation tasks.

4.2 Replication

In this step, we will replicate the Gated Recursive Convolutional Neural Network mentioned in the paper. It utilizes a recursive recurrent neural network and introduces gated GRU (Gated Recurrent Unit) units.

Recursive Neural Network is effective in processing the structure and grammatical information of sentences. By recursively combining word and phrase representations, they can capture the hierarchical structure and dependency relationships within sentences, thereby enhancing the understanding of sentence meaning.

GRU (Gated Recurrent Unit) introduces update gates and reset gates, which enable better capturing of long-term dependencies. The update gate determines whether the current input should be remembered, while the reset gate determines how to utilize the historical information. This enhances the modeling capability of GRU in handling long sequence tasks and helps alleviate the vanishing gradient problem, making training more stable and efficient on long sequences. The structure of GRU is illustrated in Figure 6.

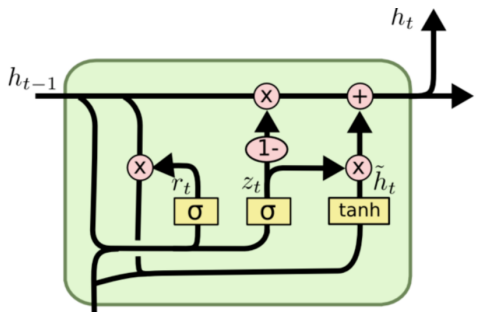

Figure 6: The GRU

4.3 Model Modification and Enhancement

4.3.1 Attention Mechanism

The attention mechanism allows the model to focus on relevant parts of the input sequence while ignoring

irrelevant parts. This helps the model capture important information and improve overall performance. For long sequence data, the attention mechanism can automatically learn and emphasize key information, reducing the model's reliance on the entire sequence and improving its ability to handle long sequences. By using the attention mechanism, the model can dynamically adjust attention weights based on the relevance of the context, leading to a better understanding of the input sequence. This enhances the model's ability to comprehend semantics and context, thereby improving its performance in natural language processing tasks. The principle of the attention mechanism is illustrated in Figure 7.

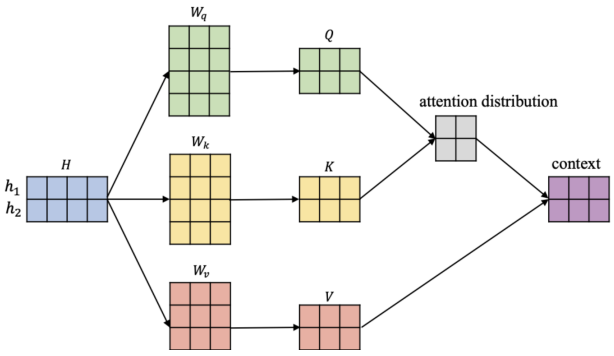

Figure 7: Attention mechanism

Regarding the attention mechanism, we considered conducting ablation experiments from three perspectives: encoder self-attention, decoder self-attention, and encoder-decoder attention, to determine which one is more suitable for the model. Introducing self-attention mechanism in the encoder helps the model capture semantic relationships within the input sentence more effectively. Using self-attention mechanism in the decoder assists the model in better focusing on the generated parts during the target language sentence generation process. Introducing attention mechanism between the encoder and decoder aids the model in aligning and attending to the encoder's outputs more effectively during the decoding process[9].

4.3.2 Skip Connections

Skip connections provide several benefits in neural networks for natural language processing (NLP). They facilitate feature reuse by directly connecting shallow and deep layers, allowing the shallow layers to access the outputs of deep layers and leverage the higher-level features learned by them. This enhances the model's expressive power, generalization ability, and reduces the risk of overfitting. Skip connections also help alleviate the challenges of gradient propagation in deep neural networks. Deep networks often face issues such as vanishing or exploding gradients during training. By providing a direct gradient path, skip connections enable faster gradient propagation to the shallow layers, thus accelerating the training process. The principle of the skip connection is illustrated in Figure 8.

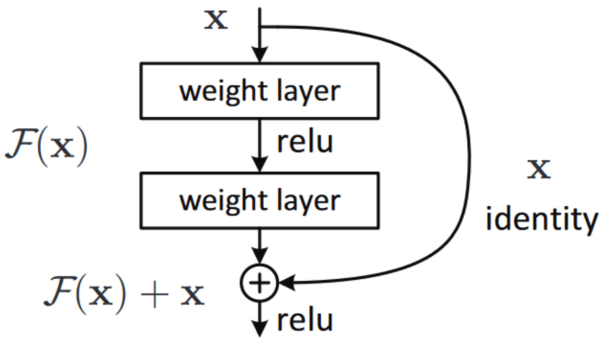

Figure 8: Skip connection

We plan to conduct experiments to select the most suitable approach for adding skip connections in the Gated Recurrent Unit (GRU). We will experiment with adding skip connections between the input and output of the GRU and between different time steps of the GRU.Adding skip connections between the input and output of the GRU allows for more direct information propagation between the recurrent units, promoting the flow of gradients and features. On the other hand, adding skip connections between different time steps of the GRU facilitates information propagation across the temporal dimension, en-

abling the model to capture long-term dependencies.By conducting these experiments, we aim to determine which approach better enhances the GRU model's performance[10].

4.3.3 Variants of GRU

We plan to experiment with different variants of GRU to address the translation model's performance. For instance, when encountering a low BLEU score, we can consider using stacked GRU, which involves stacking multiple GRU layers to form a deep network and enhance the model's expressive power and learning capacity. On the other hand, when facing low computational efficiency, we can explore efficient GRU approaches such as matrix decomposition and low-rank approximation, and analyze their impact on GRU performance and computational efficiency.

4.3.4 Depthwise Separable Convolution

By replacing convolutional layers with depthwise separable convolutions in the model, the number of parameters can be reduced, thereby improving the model's computational efficiency. Depthwise separable convolution first applies independent convolutions to each input channel, known as depthwise convolution. Then, pointwise convolution is applied to the convolutional results of each channel, using a 1x1 convolutional kernel to fuse information between channels. Ultimately, by combining these two steps, the final convolutional output is obtained. The structure is shown in Figure 9[11].

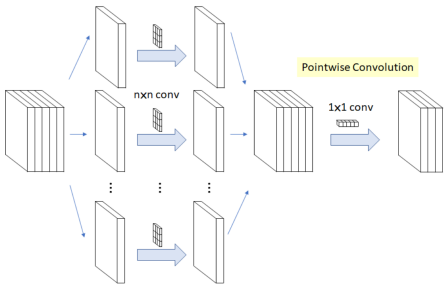

Figure 9: Depthwise Separable Convolution

4.3.5 Model Architecture

It is within our consideration to replace the model architecture. The original paper used a recursive neural network (RNN) as the main structure of the model, following an encoder-decoder pattern. We are also considering other encoder-decoder based models such as U-Net[12] and V-Net[13].

4.4 Model Evaluation

We will conduct testing using both the original paper's French-English dataset and a new Chinese-English dataset. While the original paper employed BLEU as the evaluation metric, the authors expressed some dissatisfaction with it. Therefore, we will utilize BLEU, TER, METEOR, as well as subjective ratings for a comprehensive evaluation of the model.

5 Schedule

We have outlined a provisional project schedule in Table 1, which can be adjusted according to the difficulty of the task when actually completing the project.

Tasks	Date
Reading related work papers	11.1-11.5
Complete the Review Report	11.6-11.7
Collect and collate data sets	11.8-11.9
Data preprocessing	11.10-11.11
Model training	11.12-11.16
Model optimizing	11.17-11.20
Collate experimental results	11.21-11.22
Complete the Final Report	11.23-11.25
Pre-record Presentation	11.26-11.28

Table 1: Provisional project schedule

References

[1] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[2] K. Cho, B. V. Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and hrases and their compositionality," *advances in neural information processing systems*, vol. 26, pp. 3111–3119, 2013.

[4] W. Y. Zou, R. Socher, D. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation," 2013.

[5] S. Lauly, A. Boulanger, and H. Larochelle, "Learning multilingual word representations using a bag-of-words autoencoder," 2014.

[6] J. Pouget-Abadie, D. Bahdanau, B. V. Merrienboer, K. Cho, and Y. Bengio, "Overcoming the curse of sentence length for neural machine translation using automatic segmentation," *Eprint Arxiv*, 2014.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, 2014.

[8] X. Chen, X. Qiu, C. Zhu, S. Wu, and X.-J. Huang, "Sentence modeling with gated recursive neural network," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 793–798, 2015.

[9] Z. N. A, G. Z. A, and H. Y. B, "A review on the attention mechanism of deep learning," *Neurocomputing*, 2021.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645, Springer, 2016.

[11] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," 2017.

[12] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. Mcdonagh, N. Y. Hammerla, and B. Kainz, "Attention u-net: Learning where to look for the pancreas," 2018.

[13] F. Milletari, N. Navab, and S. A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," *IEEE*, 2016.