Deep Learning and Translation Technology

Siu Sai Cheong
(siusaicheong@gmail.com)

1. Introduction

This article discusses deep learning and its application to translation technology. We first explain what deep learning is and how it works (Section 2), with an introduction to key concepts (Section 3), such as artificial neural networks, activation functions, loss functions, forward propagation, backpropagation and optimization. We then explore three approaches to machine translation with deep neural networks (Sections 4-7), which is followed by a discussion of the application of deep neural networks to speech translation, intersemiotic translation, and translation memory (Section 8) and a review of noteworthy trends (Section 9).

2. An Introduction to Deep Learning

2.1 Deep Learning and Artificial Intelligence

Deep Learning is a type of machine learning, which is in turn an approach to artificial intelligence (Goodfellow et al., 2016), with common tasks such as classification, regression, tagging, denoising, and sequence learning (A. Zhang et al., 2021). Loosely inspired by biological neurons (Goodfellow et al., 2016) and bringing together findings from such disciplines as mathematics, statistics, and computer science, deep learning can learn any functions by using neural networks composed of multiple layers of artificial neurons (A. Zhang et al., 2021).

Different from other machine learning or artificial intelligence methods, deep learning focuses on the automatic identification of features (i.e., finding the right representations of the input data for downstream algorithms (Stevens, Antiga & Viehmann, 2020)) from large amounts of training data, reducing the need for hand-crafted feature engineering (A. Zhang et al., 2021). The stacking of neuron layers allows the progressive identification of more complex features as the data or its intermediate representations pass through the layers (see, for example, Goodfellow et al., 2016).

Along with applications in computer vision (e.g., object detection (Jiao et al., 2019) and image classification (Algan & Ulusoy, 2021)), prediction (e.g., time series prediction (Lim & Zohren, 2021)), and gaming (e.g., MuZero (Schrittwieser et al., 2020), which masters Atari, Go, chess and shogi)), deep learning has brought about dramatic changes to natural language processing, such as machine translation (e.g., Yang, Wang & Chu, 2020), speech recognition (Kumar, Verma & Mangla, 2018), speech synthesis (Ning et al., 2019), sentiment analysis (L. Zhang, Wang & Liu, 2018), text classification (Qi et al., 2020), named entity recognition (Yadav & Bethard, 2018; J. Li et al., 2020), summarization (Dong, 2018; Ma et al., 2020), visual question answering (Srivastava et al., 2019), and chatbots (H. Chen et al., 2017).

2.2 Recent Surge in Popularity of Deep Learning

As discussed in Goodfellow et al. (2016), the use of artificial neural networks can be traced back to the 1940s. Over the past decades, neural network research has emerged under different names. There was a boom in the 1940s-1960s (under the name of “cybernetics”) and the 1980-1995 period (under the name of “connectionism”). It has seen a resurgence since 2006, with deep learning achieving impressive results in a number of tasks compared with other machine learning methods. The success of deep learning is attributed to the availability of larger datasets and more computational power, as well as improvements in the algorithms and software tools for the design and training of neural networks.

3. Main Tasks of Deep Learning

3.1 Key Components of Deep Learning

Similar to machine learning, to apply deep learning to a practical task, we need to have (1) data for training and evaluation, (2) a model for making predictions (a mathematical model specifies how the input is mapped to the output); and (3) an algorithm to measure the performance and enhance the results. In other words, to build a deep learning system, we need to collect data and design, train and evaluate models. Sections 3.2-3.5 further explains the tasks (see Goodfellow et al. (2016), Kelleher (2019), A. Zhang et al. (2021) for more information).

3.2 Data Collection

This refers to the collection of data and the building of datasets for training and evaluation. The data collected are often divided into the training set, the validation set (see Section 3.4 Model Training) and the test set (see Section 3.5 Model Evaluation). The validation and test sets, which are used to monitor the training process, select hyperparameters, or evaluate the model, should contain data not seen in the training set. For supervised learning tasks, which predict labels given input features (A. Zhang et al., 2021), the data comprises examples and labels (for a text classification task, for example, the text to be classified is an example, and the class it belongs to is a label).

3.3 Model Design

Model design is about defining the architecture of a model (in our case designing a neural network) for the mapping of the input to the output. The basic units of neural networks are neurons (Kelleher, 2019), simple processing units that are interconnected to handle complex tasks or model different functions. A simple example is a perceptron, which receives multiple inputs, assigns different weights to them, and computes a weighted average, which is fed into an activation function that adds non-linearity. Common activation functions include the hyperbolic tangent function and the rectified linear unit (ReLU). The output is then sent to other neurons connected to it. There are many interesting alternatives to perceptrons, and they are not necessarily analogous to biological

neurons. Examples include Long Short-term Memory and Gated Recurrent Units (see Section 5.1).

A network consists of multiple layers of neurons. Different neural network architectures (see, for example, Sections 5-7) emerge as a result of the use of various types of neurons or processing units and the diverse ways in which they are connected. A simple example is a “multilayer perceptron” (also known as a feedforward neural network), comprising layers of perceptrons as discussed above. More specifically, we have three types of layers: The layer that receives the input is called the input layer. The one that gives the final output is the output layer. The ones between the input and output layers are the hidden layers. The word “deep” in “deep learning” signifies the use of multiple hidden layers in a network for the identification of (increasingly) complex features in the data.

3.4 Model Training

This refers to the process by which the computer “learns” from data, finding the “right” set of weights (also known as parameters) in the neurons for transforming the inputs and intermediate results into good predictions.

The first step is forward propagation. The training data, which are often divided into batches (we use the term "batch size" to denote the number of examples in each batch), are fed into the model, in which the weights are initialized randomly. The input data are processed with reference to the weights and activation functions of the neurons, and the intermediate results are processed and passed through layers of neurons until the output layer is reached for the generation of the prediction.

The second step is the computation of the loss function. The predicted values are compared with the "ground truth" (i.e., the labels in the training data) with reference to a loss function (a measure of how good the model is). For numerical prediction, we can use the squared error to calculate the difference between the predicted and true values. For classification (e.g., the output layer is a softmax function that generates a list of values representing the predicted probabilities of different classes, as in the case of machine translation), we can consider the cross-entropy loss.

The third step is to find the contribution of different weights in the network to the above differences, so that we can later adjust the weights in the direction where the loss value can be reduced or minimized. This is done by backpropagation: the gradients of the loss function with respect to the weights are calculated (starting from the output layer), and by using the chain rule in multivariate calculus, they are "propagated" back to the previous layers of neurons for the computation of the corresponding gradients.

The fourth step is optimization, which is the updating of the weights based on the results of the previous step. For each weight, the actual amount of adjustment is determined by the gradient and learning rate (which indicates the size of the weight update at each step). The learning rate should be neither too large nor too small; we can have either a constant learning rate or a dynamic

learning rate (e.g., a gradual increase in the learning rate at the beginning of training and a gradual decrease afterwards). There are various optimization algorithms for the adjustment of weights, such as Stochastic Gradient Descent and ADAM (Kingma & Ba, 2014).

The above steps (i.e., forward propagation, loss calculation, gradient computation, and optimization) are repeated until a predefined training criterion is satisfied, for example, when the pre-defined number of epochs (i.e., the number of times all training data are seen) or number of weight update steps is reached, or when there are signs of overfitting (i.e., the model may start to "memorize" the training data and not generalize well to unseen data), such as a drop in training loss with a growing validation loss, which is computed with reference to the difference between the predicted and true values using the validation data that are not visible in the training data.

From the above discussion, it is clear that the final results of training (or the performance of the trained model) depend on a wide range of factors, such as the training data used, network architecture (how the model is designed), and hyperparameters (e.g., the learning rate and the number of epochs or training steps). The hyperparameters are predefined before training while the parameters are "trained" and adjusted during training with reference to the loss values and gradients.

3.5 Model Evaluation

After training, a model is ready to be used for prediction (e.g., classification of images/text or prediction of the next target text word, as in the case of machine translation). The performance of the model can be evaluated using a test dataset, with the prediction results compared with the true values of the samples in the dataset. This assessment is conducted with reference to an evaluation benchmark or metric, depending on the nature of the task in question. Common evaluation benchmarks or metrics in natural language processing include BLEU (Papineni et al., 2002; often used in machine translation) and GLUE (A. Wang et al., 2018; for a number of tasks relevant to natural language understanding).

4. An Overview of Neural Machine Translation

4.1 Popularity of Neural Machine Translation

The main application of deep learning to machine translation is neural machine translation (NMT), which encodes the source text and generates the target text by training and using deep neural networks. NMT has been popular since the mid-2010s, when it was shown to outperform its predecessors (e.g., statistical machine translation; see Section 4.2), and major machine translation developers, such as Google, Microsoft, and Baidu, began to adopt NMT in their products, along with a surging number of papers on NMT (Stahlberg, 2020). Prior to the rise of NMT, early attempts explored the possibilities of incorporating neural networks into statistical machine translation systems (e.g., language modeling (Bengio et al., 2003; Vaswani et al., 2013), word alignment (N. Yang et al., 2013), phrase reordering (P. Li et al., 2014), feature score

combination (Huang et al., 2015)), as opposed to end-to-end NMT systems mapping natural languages directly using neural networks (Deng & Liu, 2018).

4.2 NMT and conventional machine translation methods

Conventionally, common approaches to machine translation are rule-based machine translation (RBMT), example-based machine translation (EBMT), and statistical machine translation (SMT) (Bhattacharyya, 2015).

RBMT, an early approach to machine translation dating back to the 1950s, is about the design and application of manually designed rules for different tasks of automatic translation generation, such as the syntactic and semantic analysis of the source text and transfer of expressions and sentence structures from the source language to the target language. RBMT can be further divided into direct machine translation, transfer machine translation and interlingual machine translation (Hutchins, 2007), which differ in the depth of linguistic analysis.

EBMT, first proposed by Nagao (1984), focuses on automatic translation by "analogy," which involves the retrieval and recombination of bilingual expressions, similar to the way foreign language learners translate simple sentences. More specifically, instead of using rules, it puts an emphasis on the use of bilingual examples, such as expressions, segments and sentences. Key steps include the collection of bilingual examples, the retrieval of bilingual examples based on the input, and the recombination of the examples for the generation of the target text. For more information, see Hutchins (2005).

SMT, which was introduced in the late 1980s and early 1990s (Brown et al., 1988 and 1993) and later became the dominant approach until the mid-2010s, builds mathematical models for translation based on a large volume of linguistic data (e.g., parallel sentences). Featuring the use of language models and translation models together with language/translation features pre-determined by the developer, SMT aims to find sequences of target language expressions that are most likely to be the "correct" translations by maximizing the language and translation probabilities (see Ney (2005) and Koehn (2009) for more information).

Similar to other deep learning applications, NMT does not require the manual crafting of rules. Like EBMT and SMT, NMT requires data, which are however used for the training of artificial neural networks (i.e., the "learning" of a set of weights that can generate results minimizing the loss, as discussed above for deep learning applications in general), rather than for the retrieval and recombination of translation examples as in EMBT or for the building of models of language or translation features as in SMT. In NMT, as with other deep learning applications, we design the architecture of neural networks and specify hyperparameters, instead of focusing on feature engineering.

4.3 Key Concepts of NMT

To discuss the general working principle of NMT, formally, we denote the source sentence $\boldsymbol{x}$ with m elements as $x_1, \ldots, x_m$ and the target sentence $\boldsymbol{y}$ with n elements as $y_1, \ldots, y_n$. NMT computes the conditional probability of translating x into y as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{n} p(y_t|\boldsymbol{y_{<t}}, \boldsymbol{x})$$

where $\boldsymbol{y_{<t}} = y_1, \ldots, y_{t-1}$.

The dominant approach to NMT is the use of the encoder-decoder architecture. More specifically, the source elements in $\boldsymbol{x}$ are represented using word embeddings, which are high-dimensional vectors in a continuous space that are intended to capture the meaning of the tokens. Such embeddings can be obtained through an embedding matrix or pre-trained word embedding models using methods such as CBOW and skip-gram (see Mikolov et al., 2018). The word embeddings are passed on to the encoder for the computation of the intermediate representation $\boldsymbol{z} = z_1, \ldots, z_m$ of the source sentence. The target elements in $\boldsymbol{y}$ are generated (usually one after the other) by the decoder with reference to $\boldsymbol{z}$ and other inputs, such as the internal representation of the previous target element(s), until the production of a special end-of-sentence token.

The encoder-decoder is typically trained on parallel training corpora, with weight initialization and weight updates similar to the training process discussed in Section 4.3. During the training process, we compute the loss (e.g., the cross-entropy loss) by comparing the predictions with the reference translations in the training data. Training stops after reaching a fixed number of steps or epochs or meeting other predefined criteria.

There are different ways to implement the encoder-decoder architecture, and they differ in how they model and compute $p(y_j|\boldsymbol{y_{<j}}, \boldsymbol{x})$ (Stahlberg, 2020). In the next three sections (Sections 5-7), we will discuss the following three main approaches to building the encoder-decoder: recurrent neural networks, convolutional neural networks, and self-attention neural networks.

5. Recurrent Neural Networks for Translation

5.1 Recurrent Neural Networks Explained

Recurrent neural networks (RNNs) generalize feedforward neural networks to sequences (Sutskever, Vinyals & Le, 2014) with a hidden state h. Formally, given a sequence of inputs $\boldsymbol{x}$, at each time step t, the network computes the hidden state h_t based on the current input x_t and the previous hidden state h_{t-1}, which encodes the context of the previous inputs. Formally, we have

$$h_t = f(x_t, h_{t-1})$$

where f is a non-linear activation function (Cho et al., 2014) and the output y_t can be obtained by applying a linear transformation to h_t.

The function f refers to a simple linear transformation with an activation function (e.g., the logistic sigmoid function) in the vanilla RNN, which may have the problem of vanishing gradients given long input sequences requiring the backpropagation of many time steps for weight updates (Koehn, 2020).

To address this issue of long-term dependencies, we can define f as a Long Short-term Memory (LSTM) or Gated Recurrent Unit (GRU) network. LSTM networks (Hochreiter & Schmidhuber, 1997) maintain memory states and use three gates (the input gate, the forget gate, and the output gate) to regulate how much new information is accepted, how much old information (from previous time steps) is retained (or forgotten), and how the information in the memory is sent to the next layer. GRU (Cho et al., 2014), with only two gates (the reset gate and the update gate) and no separate memory states, is a "simpler alternative" to LSTM (Koehn, 2020).

RNNs have been applied to NMT for encoding and decoding given their ability to model the dependencies between input tokens, especially after the application of LSTM and GRU networks, which facilitate the modeling of long-term dependencies. Examples include Sutskever, Vinyals & Le (2014), Bahdanau, Cho & Bengio (2014), Luong, Pham & Manning (2015), and Y. Wu et al. (2016).

5.2 Encoding with RNNs

An RNN encoder, comprising a stack of RNN (usually LSTM and GRU) layers, reads the source sentence $\boldsymbol{x}$ one token at a time and generates the source sentence representation $\boldsymbol{z}$. The tokens of the source sentence can be read from left to right (i.e., from the beginning to the end of the sentence using a unidirectional encoder), from right to left (i.e., from the end to the beginning using a reversed unidirectional encoder), or in both directions (using a bidirectional encoder) (see, for example, Sutskever, Vinyals & Le (2014) for the use of unidirectional encoders and Bahdanau, Cho & Bengio (2014) and Y. Wu et al. (2016) for the bidirectional encoder). For unidirectional encoders, the hidden states $\overrightarrow{h_t}$ (left-to-right; also known as the forward hidden states) and $\overleftarrow{h_t}$ (right-to-left, also known as the backward hidden states) at time step t are computed as follows:

$$\overrightarrow{h_t} = \overrightarrow{f}(x_i, \overrightarrow{h}_{t-1})$$

and

$$\overleftarrow{h_t} = \overleftarrow{f}(x_i, \overleftarrow{h}_{t+1})$$

where $\overrightarrow{f}$ and $\overleftarrow{f}$ are usually LSTM or GRU (Bahdanau, Cho & Bengio, 2014; Stahlberg, 2020). Combining the results in both directions, the bidirectional encoder computes the hidden states h_t, which consider both the past (preceding words) and future contexts, with a focus on the

surrounding parts of the t-th token in the source sequence (Bahdanau, Cho & Bengio, 2014), as follows:

$$h_t = [\overrightarrow{h_t}^\top; \overleftarrow{h_t}^\top]^\top$$

where $[;]$ is vector concatenation and $\top$ is vector/matrix transpose.

For the generation of the internal representation $\boldsymbol{z}$ of $\boldsymbol{x}$ based on h_t, we have

$$\boldsymbol{z} = q(\{h_1, \ldots, h_i\})$$

where q is a nonlinear function (Bahdanau, Cho & Bengio, 2014). Some RNN encoders use a fixed-length vector representing the entire sentence (e.g., Sutskever et al. (2014) computed $\boldsymbol{z}$ as $q(\{h_1, \ldots, h_i\}) = h_i$, i.e., the use of the hidden state of the RNN encoder at the last time step), while others use a variable-length representation that takes into account the hidden states at different time steps (e.g., Bahdanau, Cho & Bengio, 2014) using an attention mechanism, which has become popular in RNN-based NMT systems and will be further discussed in the next section.

5.3 Decoding with RNNs

RNN decoders consist of a stack of RNN layers generating a hidden state s_t based on the previous hidden state s_{t-1}, an embedding of the last predicted token y_{t-1}, and the source sentence representation $\boldsymbol{z}$ from the encoder. The hidden state is used for the prediction of the next target token through a probability distribution over all target output symbols that is computed by a softmax layer (Y. Wu et al., 2016). Depending on the form of the internal representation $\boldsymbol{z}$, the actual design of the RNN decoder varies.

Formally, the conditional probability $p(y_t|\boldsymbol{y_{<t}}, \boldsymbol{z})$ for predicting the next target token y_t at time step t given the source sentence representation $\boldsymbol{z}$ is modeled as follows:

$$p(y_t|\boldsymbol{y_{<t}}, \boldsymbol{z}) = g(s_{t-1}, y_{t-1}, \boldsymbol{z})$$

where the function g, which can also be defined as a function with only two parameters s_{t-1} and $\boldsymbol{z}$ (i.e., $p(y_t|\boldsymbol{y_{<t}}, \boldsymbol{z}) = g(s_{t-1}, \boldsymbol{z})$), comprises LSTM or GRU layers (typically the choice matches the encoder (see Koehn, 2020)) and a softmax function (and possibly other linear transformations) and computes the probability distribution.

If $\boldsymbol{z}$ is a fixed-length vector, it can be used to initialize the decoder state or fed into g for the generation of each target token (Cho et al., 2014). For variable-length representations, an attention mechanism is applied for the computation of $\boldsymbol{z}$ as a weighted sum of annotations $h_1, \ldots, h_i$, the weights of which are known as attention scores. The scores, which indicate how relevant an input token is to the production of the next word (Koehn, 2020) and allow the decoder to focus on different parts of the source sentence when generating the target sentence (Gehring

et al., 2017), can be computed by comparing each annotation with the previous hidden state, for example, using additive attention (Bahdanau, Cho & Bengio, 2014) or dot-product attention (Luong, Pham & Manning, 2015).

Formally, the first step is to compute the attention scores. Given h_m with $m = [1, i]$, for dot-product attention, we have

$$score(s_{t-1}, h_m) = s_{t-1}^{\top} h_m,$$

and for additive attention, we have

$$score(s_{t-1}, h_m) = v^{\top} \tanh(W s_{t-1} + U h_m)$$

with the weight vector v and weight matrices W and U. The second step is to normalize the attention value using the softmax function so that the scores over all tokens in the source sentence add up to one (Koehn, 2020). We denote the normalized score as a, as an element of a variable-length alignment vector of i dimensions:

$$a(s_{t-1}, h_m) = \frac{exp(score(s_{t-1}, h_m))}{\sum_{k=1}^{i} exp(score(s_{t-1}, h_k))}.$$

The third step is to compute z given the alignment vector as weights:

$$\boldsymbol{z} = q(\{h_1, \ldots, h_i\}) = \sum_{m=1}^{i} a(s_{t-1}, h_m) h_m.$$

In our case here, $\boldsymbol{z}$ is also referred to as the context vector at step t. Given the context vector, the decoder produces an attentional hidden state (or attentional vector) s_t, which is passed on to the softmax layer for the predictive distribution.

6. Convolutional Neural Networks for Translation

6.1 Convolutional Neural Networks (CNNs) Explained

Convolutional neural networks (CNNs), a special family of neural networks containing convolutional layers, have been widely used in computer vision, such as image recognition and object detection (A. Zhang et al., 2021). Consisting of convolutional layers, CNNs encode input sequences by creating representations of words with their left and right contexts using a limited window (i.e., kernel). (Gehring et al., 2017; Koehn, 2020; A. Zhang et al., 2021).

Formally, as discussed in Stahlberg (2020), given an input sequence $\boldsymbol{x} = x_1, \ldots, x_i$ comprising i vectors in M dimensions (i.e., $x_d \in R^M$ for $d = [1, i]$, with the d-axis and M referred to in the literature as spatial dimension and channels respectively), a standard one-dimensional

convolutional layer for the generation of a sequence of N-dimensional vector y of the same length i is defined as follows:

$$y_{d,n} = \sum_{m=1}^{M} \sum_{k=0}^{K-1} w_{kM+m,n} x_{d+k,m}$$

where K is the width of the kernel, $w_{kM+m,n}$ refers to the element in the $(kM + n)$-th row and n-th column of the weight matrix $W \in R^{KM \times N}$, and $x_{d+k,m}$ and $y_{d,n}$represent the m-th element ($m \in [1, M]$) of the $(d + k)$-th input vector and the n-th element ($n \in [1, N]$) of the d-th output vector respectively. Note that the inner sum and the outer sum represent spatial dependency and cross-channel dependency respectively, and they can be factored out as pointwise and depthwise convolutions.

CNNs enable parallelization over the elements in the input sequence as they reduce sequential computation (Gehring et al., 2017; Stahlberg, 2020). For the generation of representations of input sequences, we can increase the effective context size by stacking multiple layers, where nearby input elements interact at lower levels and distant elements at higher levels, and this hierarchical structure provides a shorter path for capturing long-term dependencies, compared with chain modelling using RNNs. (Gehring et al., 2017).

6.2 NMT with CNNs

CNN-driven NMT systems perform encoding and decoding with CNNs. According to Koehn (2020), Kalchbrenner and Blunsom (2013) proposed the first modern end-to-end NMT model using CNNs instead of RNNs. Their CNN model encodes the input by merging the representations of input tokens into a single vector through the iterative use of convolutional kernels and reverses the process when generating the output.

Gehring et al. (2017) proposed an alternative architecture for CNN-driven NMT. The model adopts CNNs and the attention mechanism. For encoding, given the word embeddings of an input sequence, position embeddings are added so that the model is aware of the position of each of the tokens. After that, the CNN encoder, which is a stack of blocks comprising a one-dimensional convolution and Gated Linear Units (GLU, a nonlinear function implementing a gating mechanism (Dauphin et al., 2017)), computes intermediate states based on a fixed number of elements in the input. For decoding, the CNN decoder, which is another stack of blocks with a linear transformation and a softmax layer for the computation of a probability distribution over the possible target tokens, features (1) the combination of the encoder states and input word embeddings, (2) the use of masks to avoid the network's access to the future context, and (3) the introduction of a separate attention mechanism to each decoder layer (Koehn, 2020; Stahlberg, 2020).

F. Wu et al. (2019) proposed lightweight convolutions (LightConv) and dynamic convolutions (DynamicConv) for NMT. Instead of using standard convolutions, both methods adopt depthwise

convolutions, which perform convolutions independently over every channel and reduce the number of parameters. LightConv uses kernels with context element weights that remain unchanged across time steps and shares certain output channels (with weights normalized across the temporal dimension using the softmax function). Built on LightConv, DynamicConv uses kernels that vary across time steps. The weights assigned to context elements change over time and do not depend on the entire context. The architecture of the CNN encoder-decoder is similar to that of self-attention models to be discussed in the next section, except that the self-attention modules are replaced by LightConv or DynamicConv modules, which (1) apply an input projection to map the input from dimension d to $2d$; (2) use a GLU, similar to Gehring et al. (2017); (3) perform LightConv or DynamicConv; and (4) apply an output projection to the results of LightConv or DynamicConv.

7. Self-attention Neural Networks for Translation

7.1 Self-attention NMT Explained

As discussed in Section 5, the attention mechanism can bridge the encoder and the decoder. Here, in self-attention NMT, the mechanism is also used (1) within the encoder for the generation of $\boldsymbol{z}$ for the building of context-aware word embeddings, with reference to other words in the source sentence, and (2) within the decoder for the consideration of the current translation history (Stahlberg, 2020). Such self-attention neural models, which introduce short paths between distant words and reduce the amount of sequential computation (Stahlberg, 2020), were first used in the Transformer (Vaswani et. al, 2017) and have become the standard architecture for NMT because of its translation quality (Stahlberg, 2020; see, for example, Barrault et al., 2019, 2020) and facilitated the generation of contextualized word embeddings like BERT (Devlin et al., 2019).

7.2 Multi-head Attention with Queries, Keys and Values

To better understand the extension of attention to encoding and decoding, we can consider attention in terms of queries, keys and values, using the terminology of Vaswani et al. (2017) and Stahlberg (2020). We can view attention as a mapping of n query vectors q to their output vectors through a memory of m pairs of key vectors k and value vectors v, where q, k and v are all assumed to have the same dimension d here and can be stacked into matrices $Q \in R^{n \times d}$, $K \in R^{m \times d}$ and $V \in R^{m \times d}$ respectively. For each q, we compute a weighted sum of the value vectors in V, and the weights are based on the similarity scores between q and the key vectors in K. Formally, the attention given Q, K and V is calculated as follows:

$$attention(Q, K, V) = (scoremat(Q, K))V$$

where $scoremat(Q, K) \in R^{n \times m}$ is the matrix of similarity scores, normalized by the softmax function for each column. Using this notation, if we consider the dot-product attention we discussed above, we can compute the score matrix as follows:

$$scoremat(Q,K) = softmax(QK^{\top} \cdot scale)$$

with $scale = 1$ for Luong, Pham & Manning (2015, with $q = s_j$ and $k = v = h_i$) and $scale = \frac{1}{\sqrt{d}}$ for Vaswani et. al (2017), which uses scaled dot-product attention.

A noteworthy feature of the self-attention model is the use of multi-head attention: instead of having only one attention operation, we perform H attention operations (H is known as the number of attention heads). For each attention head (mathematically, $head_h \in R^{n \times \frac{d}{H}}$, where $h \in [1, H]$), we apply linear transformations to Q, K and V and compute attention:

$$head_h = attention(W_Q Q, W_K K, W_V V) = attention(Q_h, K_h, V_h)$$

where $Q_h \in R^{n \times \frac{d}{H}}$, $K_h \in R^{m \times \frac{d}{H}}$ and $V_h \in R^{m \times \frac{d}{H}}$. The output of the multi-head attention is a linear transformation of the concatenation of the heads:

$$multihead(Q,K,V) = W_{multihead}[head_1; head_2; \ldots; head_H]$$

with the weight matrix $W_{multihead} \in R^{d \times d}$. Based on the above discussion of multi-head attention with queries, keys and values, we will discuss the design of self-attention encoders and decoders in Sections 7.3 and 7.4.

7.3 Self-attention Encoder

Similar to CNNs, we use positional encodings to make word embeddings sensitive to position. Here trigonometric functions of different frequencies are used:

$$posenc(pos, dim) = f(pos \cdot 10000^{-\frac{dim}{d}})$$

where pos is the absolute position of a word in the sentence, d is the size of the word embedding with $dim \in [1, d]$, and f is either a sine function (when dim is even) or a cosine function (when dim is odd).

The self-attention encoder is a stack of encoding blocks. Given an input matrix X_{input} of size $n \times d$, which is either the positional encodings (for the first encoding block) or the output of the previous encoding block (for the blocks thereafter), each block first performs multi-head attention with the query, key and value projections of X_{input}. The block computes $X_{intermediate}$ with (1) the layer normalization of the result of multi-head attention and (2) a residual connection:

$$X_{intermediate} = layernorm(multihead(X_{input}, X_{input}, X_{input}) + X_{input}).$$

The intermediate result is then passed to a feedforward network comprising two position-wise feedforward layers F_{x1} and F_{x2} (in the form $Wx + b$) with a ReLU activation in between, followed

by layer normalization and a residual connection for the generation of the output of the encoding block. Formally, we have

$$X_{output} = layernorm(F_{x2}(ReLU(F_{x1}(X_{intermediate})) + X_{intermediate}).$$

The output X_{output} is then passed on to the next encoding block or the decoder for further processing.

7.4 Self-attention Decoder

The structure of the self-attention decoder is similar to that of the encoder in terms of the use of positional encodings. As a stack of multiple decoding blocks, the decoder differs from the encoder in the following ways: First, the positional encodings here are based on the target language tokens generated. Second, each of its decoding blocks comprises two self-attention modules instead of one as in the encoding block.

More specifically, given an input matrix Y (either the positional encodings (for the first decoding block) or the output of the previous decoding block (for the blocks thereafter)), each decoding block first performs masked multi-head self-attention with future decoder states masked to prevent computation conditioned on future tokens. The block computes the first intermediate result $Y_{intermediate(1)}$ with (1) the layer normalization of the output of the masked self-attention and (2) a residual connection:

$$Y_{intermediate(1)} = layernorm(multihead(Y_{input}, Y_{input}, Y_{input}) + Y_{input}).$$

To utilize the information from the encoder, the block performs the second multi-head self-attention using the first intermediate result $Y_{intermediate(1)}$ (for the computation of Q_h) together with the output of the self-attention encoder $X_{output(encoder)}$ (for the computation of K_h and V_h). For the computation of the second intermediate output, the second self-attention is followed by layer normalization and a residual connection:

$$Y_{intermediate(2)} = layernorm(multihead(Y_{intermediate(1)}, X_{output(encoder)}, X_{output(encoder)}) + Y_{intermediate(1)}) .$$

Similar to the encoding block, the second intermediate result $Y_{intermediate(2)}$ is then passed to a two-layered position-wise feedforward network (F_{y1} and F_{y2} below; again, in the form $Wx + b$) with an ReLU activation in between:

$$Y_{output} = layernorm(F_{y2}(ReLU(F_{y1}(Y_{intermediate(2)}))) + Y_{intermediate(2)}.$$

The output Y_{output} is then passed to the next decoding block. For the last decoding block, similar to RNN and CNN decoders, a linear transformation is applied to the output, which is then passed to a softmax layer for the generation of the output probabilities.

7.5 Other Transformer Architectures

Many variants of the self-attention model have been proposed since the emergence of the Transformer. For example, Transformer-XL (Dai et al., 2019) uses relevant positions for the computation of positional encodings. Other approaches include the use of fixed patterns, learnable patterns, memory, and low-rank approximations or kernels; see Tay et al. (2020) for more information.

8. Deep Learning for Other Translation-related Applications

In addition to text translation, application of deep learning to areas such as speech and image processing has also facilitated the development of other translation-related tools. The following are three noteworthy areas:

8.1 Deep Learning and Speech Translation

Speech translation refers to the translation of speech in the source language into text or speech in the target language, with three modes of delivery: the batch mode (translating a recorded speech as a whole), the consecutive mode (translating utterances like consecutive interpreting), and the simultaneous mode (translating an input audio stream in real time like simultaneous interpreting) (Sperber and Paulik, 2020). Generally, there are two approaches to building speech translation systems: (1) cascaded systems and (2) direct systems.

Cascaded systems usually consist of three parts: (1) automatic speech recognition (ASR) for speech-to-text conversion, (2) text-to-text machine translation, and (3) speech synthesis for text-to-speech conversion (applicable to speech-to-speech translation systems). Deep learning has contributed not only to machine translation as discussed above, but also to ASR and speech synthesis. For ASR, Baevski et al. (2020), for example, proposed a model comprising CNN and the Transformer for self-supervised learning on unlabeled raw audio data, followed by fine-tuning on a small set of labelled data instead of thousands of hours of transcribed speech. There have also been encouraging results in speech synthesis, including the use of RNNs (e.g., Shen et al., 2018), the Transformer (Ren et al., 2019), and Generative Adversarial Networks (GANs, proposed by Goodfellow et al. (2014), consist of a generative model capturing the data distribution and a discriminative model estimating whether a sample is from the distribution; see, for example, Kong et al., 2020 for their use in speech synthesis).

Cascade systems may be susceptible to error propagation (e.g., ASR errors leading to translation errors), modeling mismatches between components (e.g., spoken language modeling for ASR but written language modeling for machine translation), and loss of speech information (e.g., unawareness of prosody by achine translation) (Sperber and Paulik, 2020). To help address these issues, direct systems perform end-to-end speech translation. Examples include (1) the sequence-to-sequence model proposed by Jia et al. (2019), which consists of an attention-based

network generating target spectrograms and a vocoder converting the spectrograms to time-domain waveforms, and (2) the Tandem Connectionist Encoding Network proposed by C. Wang et al. (2020), which comprises two connected encoders and a decoder with an attention module. However, direct systems often require large-scale end-to-end datasets, which are not always readily available, and the use of augmented data may help (e.g., the use of automatic translation results or synthesized speech) (Sperber and Paulik, 2020).

8.2 Deep Learning and Intersemiotic Translation

According to Jakobson (1959/2012), intersemiotic translation refers to “an interpretation of verbal signs by means of signs of nonverbal sign systems”. Considering this to be “a change of medium” (Munday 2009), we can extend its scope by including the conversion of nonverbal signs to verbal signs, which allows us to explore the contribution of deep learning to cross-modal translation from the following perspectives:

Multimodal Machine Translation (MMT) refers to machine translation with multimodal inputs (e.g., given an image and its source language description, an MMT system translates the description into the target language) (Barrault et al., 2018). Calixto, Liu and Campbell (2017), Ive et al. (2019), and Yao and Wan (2020) proposed the use of RNNs (with pre-trained CNNs), the Transformer, and a multimodal Transformer (with an “image-aware” attention mechanism instead of the direct encoding of image features), respectively. The use of cross-modal contextual input in text translation is beneficial for the translation of subtitles and manga (see, for example, Hinami et al., 2021).

Image captioning is the verbal description of the content of an image (Cornia et al., 2020). RNN, CNN and Transformer-based architectures have been used for image-to-text conversion with (1) a combination of models for image understanding and text generation (Hossain et al., 2018) or (2) visual-language pre-training models, featuring the use of shared Transformer networks for encoding and decoding (e.g., Zhou et al., 2019). See X. Li et al. (2020), Cornia et al. (2020), and P. Zhang et al. (2021) for more examples.

Video captioning refers to the automatic explanation of the content of video frames using natural language (Islam et al., 2021). Inspired by deep image captioning, deep learning-driven video captioning adopts sequence learning methods that comprise two phases similar to NMT: the encoding of visual features (e.g., using CNNs) and the decoding/generation of captions (e.g., using RNNs) (Z. Zhang et al., 2020; Singh et al., 2020).

Text-to-image conversion synthesizes images from textual descriptions. Since Mansimov et al. (2015), there has been much work on the creation of images conditioned on text captions using, for example, GANs (e.g., Reed et al. (2016)). More recently, Ramesh et al. (2021) trained a Transformer with 12 billion parameters on 250 million image-text pairs for the autoregressive modeling of text and image tokens as a single data stream.

Text-to-music conversion is music generation conditioned on lyrics. For example, Dhariwal et al. (2020) proposed JukeBox, adopting the VQ-VAE architecture (Razavi et al., 2019) and Transformer with sparse attention (Child et al., 2019). Given inputs such as lyrics, genres and artist style, this model generates music as raw audio, rather than piano rolls specifying the pitch and length of each note as in conventional symbolic generation.

8.3 Deep Learning and Translation Memories

Translation Memories (TMs) (Simard, 2020; Ranasinghe et al., 2020) consist of translation units (TUs), which are pairs of text segments (usually sentences) in the source and target languages, to facilitate the reuse of previously translated segments. First proposed in the late 1970s, TMs have become popular among translators since their commercialization in the early 1990s (Arthern, 1978; Zaretskaya et al., 2017). TMs compute the similarity between the segments to be translated and the translation units in the database and identify the following cases: exact matches (TUs identical to the source segments are found), context matches (exact matches with identical contexts are found, often indicated by the presence of the same neighboring sentences), fuzzy matches (similar TUs are found, with similarity scores higher than the threshold), and no matches (no similar TUs are found).

Deep learning offers new ways to compute the similarity score. Conventionally, we calculate the similarity using edit distance measures, which may not be able to identify semantically similar but syntactically different segments (Ranasinghe et al., 2020). The generation of sentence encoding using neural networks provides an alternative: instead of directly comparing the segments to be translated with the source language segments in TUs, we compare their sentence embeddings generated by the NMT encoder.

Ranasinghe et al. (2020), for example, proposed the use of Universal Sentence Encoder (Cer et al., 2018). One of the models provided by the sentence encoder uses the Transformer. The output of the Transformer encoder after multiple rounds of attention computation is a list of context-sensitive word embeddings, which are averaged for the generation of sentence-level embeddings for sentence comparison. Formally, given two source sentences A and B with sentence embeddings $a \in R^n$ and $b \in R^n$ respectively, the similarity score is computed using cosine similarity:

$$cossim\ (a, b)\ = \frac{a \cdot b}{|a||b|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} {a_i}^2}\sqrt{\sum_{i=1}^{n} {b_i}^2}}$$

Their experiment showed that sentence encodings outperformed simple edit distance.

9. Other Noteworthy Trends

In addition to the design and use of neural network architectures and the exploration of new application areas discussed above, there has been much research on ways to enhance natural

language processing, model training and inference, thus contributing to the advancement of deep learning for translation and cross-lingual communication. The following are notable examples:

Use of subwords: The division of the input sequence into subword units rather than words can help reduce the size of a translation model (by reducing the vocabulary size) and deal with rare words or out-of-vocabulary items. Possible approaches include word-piece modeling (Y. Wu et al., 2016), byte-pair encoding (Sennrich, Haddow & Birch, 2016b), sentence-piece modeling (Kudo & Richardson, 2018), and byte-level subwords (C. Wang, Cho & Gu, 2020).

Multilingual NMT: Instead of building translation engines between one source language and one target language, multilingual NMT focuses on the development of a single model between "as many languages as possible," featuring exposure to diverse languages, which may result in better generalization and enhance machine translation for low-resource languages (Dabre, Chu & Kunchukuttan, 2020). Recent examples include Y. Liu et al. (2020) and Fan et al. (2020).

Decoding methods: When decoding, NMT requires an effective method to search for target sentences (i.e., the "best" combinations of target language tokens) to maximize the translation probability (see Gu, Cho & Li, 2017; Stahlberg, 2020). Common ways include greedy search (using the candidate with the highest probability at each time step), beam search (keeping multiple highest scoring candidates at each step, with variants such as dynamic beam allocation (Post & Vilar, 2018)), and other methods like re-ranking (Y. Liu et al., 2018).

Data augmentation: There are ways to enhance NMT models using monolingual data, which are more readily available than bilingual documents (Edunov et al., 2018). One of these methods is back translation (Sennrich, Haddow & Birch, 2016a), which augments parallel training corpora by translating monolingual target language data into the source language, probably with the addition of noise (Edunov et al., 2018) or tags (Caswell, Chelba & Grangier, 2019). This method has been used in the development of different systems, such as Ng et al. (2019), Meng et al. (2020), L. Wu et al. (2020) and S. Wu et al. (2020). Other ways of using monolingual data are discussed in Edunov (2018) and Meng (2020).

Use of multiple models: The use of multiple translation models, as opposed to a single set of trained weights, can help improve the accuracy of translation (see, for example, Y. Liu et al., 2018). Two common approaches are model ensembling and checkpoint averaging. The former involves the use of multiple neural networks by the decoder, which averages their predictions of different networks (e.g., Sutskever, Vinyals & Le, 2014; Cromieres et al., 2016; Ng et al., 2019), and the latter is the combination of several recent checkpoints into a single model (e.g., Popel & Bojar, 2018; Ng et al., 2019) to reduce small fluctuations in the training process (Stahlberg, 2020).

Non-autoregressive NMT: As opposed to the generation of output words one by one (i.e., $p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{n} p(y_t|\boldsymbol{y}_{<\boldsymbol{t}}, \boldsymbol{x})$, also known as autoregressive NMT) as discussed above, non-autoregressive NMT aims to generates target sentence tokens in parallel to reduce inference time (i.e., $p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{n} p(y_t|\boldsymbol{x})$) (see Gu et al., 2017). Recent examples include the Levenshtein Transformer (Gu, Wang & Zhao, 2019), which is a sequence generation model comprising

insertion and deletion operations, and EDITOR (Xu & Carpuat, 2021), which generates new sequences by introducing a reposition operation and editing hypotheses iteratively.

Deeper networks: Increasing the number of layers of the encoder and decoder may lead to better results as in many other deep learning tasks, but there may be bottlenecks for architectures like the Transformer because of issues such as gradient vanishing, which can be mitigated by decreasing parameter variance in the initialization phase, reducing the output variance of residual connections, using merged attention sublayers, and adopting adaptive initialization methods for the stabilization of early-stage training (see B. Zhang, Titov & Sennrich, 2019; L. Liu et al., 2020; X. Liu et al., 2020).

Document-level NMT: Unlike sentence-based translation systems, which may degrade the coherence and cohesiveness of the translation as a result of neglecting the discourse connections between sentences, document-level NMT systems take into account the document context by employing methods such as sentence concatenation (Tiedemann & Scherrer, 2017), hierarchical attention networks (Miculicich et al., 2018), capsule networks (Z. Yang et al., 2019), and discourse structure modeling (J. Chen et al., 2020).

Domain Adaptation: Given that general-purpose translation systems tend to perform poorly when translating domain-specific documents (Chu and Wang, 2017), domain adaptation, which focuses on the adaptation of machine translation systems to new domains (Freitag and Al-Onaizan, 2016), aims to identify ways to enhance the performance of in-domain translation by leveraging out-of-domain and in-domain datasets, the latter often being less readily available in both the source and target languages (Chu and Wang, 2017). A common method is fine-tuning, which trains out-of-domain models on in-domain data or on a combination of general and specialized data (see, for example, Luong & Manning, 2015; Chu, Dabre & Kurohashi, 2017).

Pre-trained language models: Since the advent of the Transformer, there has been a growing number of pre-trained language models such as BERT (Devlin et al., 2019) based on this architecture, and they provide contextual embeddings and can be fine-tuned for downstream natural language processing tasks, such as classification and question answering. The success of such models has been followed by research on their incorporation into NMT for encoding or decoding. Zhu et al. (2020), for example, explored BERT-generated representations that are used by each layer of the NMT encoder and decoder with attention.

Refer to Stahlberg (2020) and J. Zhang & Zong (2020) for other research directions.

10. Conclusion

Recent rapid advances in deep learning, as an approach to machine learning and artificial intelligence with an emphasis on automatic feature extraction through the training and use of multilayer artificial neural networks, have stimulated the development of translation technology in different aspects, including not only NMT using recurrent neural networks, convolutional neural

networks, or self-attention networks, but also cascaded or end-to-end speech translation, intersemiotic translation (e.g., multimodal machine translation, image and video captioning, and text-to-image/music translation), and translation memories based on sentence embeddings. We expect more progress in the future with the availability of more data, the design of novel neural architectures, the development of new approaches to training, inference and natural language processing, and the innovative use of deep learning models.

References

Algan, G., & Ulusoy, I. (2021). Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, *215*, 106771.

Arthern, P. J. (1978, November). Machine translation and computerized terminology systems: a translator's viewpoint. In *Translating and the Computer: Proceedings of a Seminar, London* (Vol. 14, pp. 77-108).

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *Advances in Neural Information Processing Systems*, *33*.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015, January). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Barrault, L., Biesialska, M., Bojar, O., Costa-jussà, M. R., Federmann, C., Graham, Y., ... & Zampieri, M. (2020, November). Findings of the 2020 conference on machine translation (wmt20). In *Proceedings of the Fifth Conference on Machine Translation* (pp. 1-55).

Barrault, L., Bojar, O., Costa-Jussa, M. R., Federmann, C., Fishel, M., Graham, Y., ... & Zampieri, M. (2019, August). Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)* (pp. 1-61).

Barrault, L., Bougares, F., Specia, L., Lala, C., Elliott, D., & Frank, S. (2018, October). Findings of the third shared task on multimodal machine translation. In Proceedings of the Third Conference on Machine Translation: Shared Task Papers. (Vol. 2, pp. 308-327).

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, *3*, 1137-1155.

Bhattacharyya, P. (2015). *Machine translation*. CRC Press.

Bojar, O., Chatterjee, R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., ... & Verspoor, K. (2018, October). Proceedings of the Third Conference on Machine Translation: Shared Task Papers. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Mercer, R. L., & Roossin, P. (1988). A statistical approach to language translation. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, *19*(2), 263-311.

Calixto, I., Liu, Q., & Campbell, N. (2017, July). Doubly-Attentive Decoder for Multi-modal Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1913-1924).

Caswell, I., Chelba, C., & Grangier, D. (2019, August). Tagged Back-Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)* (pp. 53-63).

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Kurzweil, R. (2018, November). Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 169-174).

Chen, H., Liu, X., Yin, D., & Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, *19*(2), 25-35.

Chen, J., Li, X., Zhang, J., Zhou, C., Cui, J., Wang, B., & Su, J. (2020, July). Modeling Discourse Structure for Document-level Neural Machine Translation. In *Proceedings of the First Workshop on Automatic Simultaneous Translation* (pp. 30-36).

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, October). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724-1734).

Chu, C., & Wang, R. (2018). A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258*.

Chu, C., Dabre, R., & Kurohashi, S. (2017). An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Cornia, M., Stefanini, M., Baraldi, L., & Cucchiara, R. (2020). Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10578-10587).

Cromieres, F., Chu, C., Nakazawa, T., & Kurohashi, S. (2016, December). Kyoto university participation to WAT 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)* (pp. 166-174).

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., & Salakhutdinov, R. (2019, July). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 2978-2988).

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017, July). Language modeling with gated convolutional networks. In *International conference on machine learning* (pp. 933-941). PMLR.

Deng, L., & Liu, Y. (Eds.). (2018). *Deep learning in natural language processing*. Springer.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171-4186).

Dong, Y. (2018). A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*.

Edunov, S., Ott, M., Auli, M., & Grangier, D. (2018). Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 489-500).

Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., ... & Joulin, A. (2020). Beyond english-centric multilingual machine translation. *arXiv preprint arXiv:2010.11125*.

Freitag, M., & Al-Onaizan, Y. (2016). Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017, July). Convolutional sequence to sequence learning. In *International Conference on Machine Learning* (pp. 1243-1252). PMLR.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial networks. arXiv preprint arXiv:1406.2661.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*. Cambridge: MIT press.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., & Socher, R. (2017). Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Gu, J., Cho, K., & Li, V. O. (2017, September). Trainable Greedy Decoding for Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 1968-1978).

Gu, J., Wang, C., & Zhao, J. (2019). Levenshtein transformer. *arXiv preprint arXiv:1905.11006*.

Hinami, R., Ishiwatari, S., Yasuda, K., & Matsui, Y. (2021). Towards Fully Automated Manga Translation. *arXiv preprint arXiv:2012.14271*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hossain, M., Sohel, F., Shiratuddin, M. F., & Laga, H. (2018). A Comprehensive Survey of Deep Learning for Image Captioning. arXiv preprint arXiv:1810.04020.

Huang, S., Chen, H., Dai, X., & Chen, J. (2015, July). Non-linear Learning for Statistical Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 825-835).

Hutchins, J. (2005, September). Towards a definition of example-based machine translation. In *Machine Translation Summit X, Second Workshop on Example-Based Machine Translation* (pp. 63-70).

Hutchins, J. (2007). Machine translation: A concise history. *Computer aided translation: Theory and practice*, *13*(29-70), 11.

Islam, S., Dash, A., Seum, A., Raj, A. H., Hossain, T., & Shah, F. M. (2021). Exploring Video Captioning Techniques: A Comprehensive Survey on Deep Learning Methods. SN Computer Science, 2(2), 1-28.

Ive, J., Madhyastha, P. S., & Specia, L. (2019, July). Distilling Translations with Visual Awareness. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 6525-6538).

Jakobson, R. (2012). 14. On Linguistic Aspects of Translation. In *Theories of Translation* (pp. 144-151). University of Chicago Press.

Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., & Wu, Y. (2019). Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model}}. *Proc. Interspeech 2019*, 1123-1127.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R. (2019). A survey of deep learning-based object detection. *IEEE Access*, *7*, 128837-128868.

Kalchbrenner, N., & Blunsom, P. (2013, October). Recurrent continuous translation models. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1700-1709).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

Kong, J., Kim, J., & Bae, J. (2020). HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. *Advances in Neural Information Processing Systems*, *33*.

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *EMNLP 2018*, 66.

Kumar, A., Verma, S., & Mangla, H. (2018, October). A survey of deep learning techniques in speech recognition. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)* (pp. 179-185). IEEE.

Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Li, P., Liu, Y., Sun, M., Izuha, T., & Zhang, D. (2014, August). A neural reordering model for phrase-based translation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 1897-1907).

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., ... & He, L. (2020). A Text Classification Survey: From Shallow to Deep Learning. *arXiv preprint arXiv:2008.00364*.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., ... & Gao, J. (2020, August). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision* (pp. 121-137). Springer, Cham.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200209.

Liu, L., Liu, X., Gao, J., Chen, W., & Han, J. (2020). Understanding the Difficulty of Training Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 5747-5763).

Liu, X., Duh, K., Liu, L., & Gao, J. (2020). Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., ... & Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, *8*, 726-742.

Liu, Y., Zhou, L., Wang, Y., Zhao, Y., Zhang, J., & Zong, C. (2018, August). A comparable study on model averaging, ensembling and reranking in nmt. In *CCF International Conference on Natural Language Processing and Chinese Computing* (pp. 299-308). Springer, Cham.

Luong, M. T., & Manning, C. D. (2015, December). Stanford neural machine translation systems for spoken language domains. In *Proceedings of the international workshop on spoken language translation* (pp. 76-79).

Luong, M. T., Pham, H., & Manning, C. D. (2015, September). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1412-1421).

Ma, C., Zhang, W. E., Guo, M., Wang, H., & Sheng, Q. Z. (2020). Multi-document Summarization via Deep Learning Techniques: A Survey. *arXiv preprint arXiv:2011.04843*.

Mansimov, E., Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2015). Generating images from captions with attention. arXiv preprint arXiv:1511.02793.

Meng, F., Yan, J., Liu, Y., Gao, Y., Zeng, X., Zeng, Q., ... & Zhou, H. (2020, November). WeChat Neural Machine Translation Systems for WMT20. In *Proceedings of the Fifth Conference on Machine Translation* (pp. 239-247).

Mikolov, T., Grave, É., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018, May). Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Munday, J. (2009). Issues in translation studies. *The Routledge companion to translation studies*, 1-19.

Nagao, M. (1984). A framework of a mechanical translation between Japanese and English by analogy principle. *Artificial and human intelligence*, 351-354.

Ney, H. One decade of statistical machine translation: 1996-2005. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.* (pp. 2-11). IEEE.

Ng, N., Yee, K., Baevski, A., Ott, M., Auli, M., & Edunov, S. (2019, August). Facebook FAIR's WMT19 News Translation Task Submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)* (pp. 314-319).

Ning, Y., He, S., Wu, Z., Xing, C., & Zhang, L. J. (2019). A review of deep learning based speech synthesis. *Applied Sciences*, *9*(19), 4050.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

Popel, M., & Bojar, O. (2018). Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, (110), 43-70.

Post, M., & Vilar, D. (2018, June). Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1314-1324).

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., ... & Sutskever, I. (2021). Zero-shot text-to-image generation. arXiv preprint arXiv:2102.12092.

Ranasinghe, T., Orasan, C., & Mitkov, R. Intelligent Translation Memory Matching and Retrieval with Sentence Encoders. In *22nd Annual Conference of the European Association for Machine Translation* (p. 175).

Razavi, A., Oord, A. V. D., & Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016, June). Generative adversarial text to image synthesis. In International Conference on Machine Learning (pp. 1060-1069). PMLR.

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., & Liu, T. Y. (2019). Fastspeech: Fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., ... & Silver, D. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604-609.

Sennrich, R., Haddow, B., & Birch, A. (2016a, August). Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 86-96).

Sennrich, R., Haddow, B., & Birch, A. (2016b, August). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1715-1725).

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., ... & Wu, Y. (2018, April). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4779-4783). IEEE.

Simard, M. (2020). Building and using parallel text for translation. *The Routledge Handbook of Translation and Technology*, 78-90.

Singh, A., Singh, T. D., & Bandyopadhyay, S. (2020). NITS-VC System for VATEX Video Captioning Challenge 2020. arXiv preprint arXiv:2006.04058.

Sperber, M., & Paulik, M. (2020, July). Speech Translation and the End-to-End Promise: Taking Stock of Where We Are. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7409-7421).

Srivastava, Y., Murali, V., Dubey, S. R., & Mukherjee, S. (2019). Visual question answering using deep learning: A survey and performance analysis. *arXiv preprint arXiv:1909.01860*.

Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69, 343-418.

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep learning with PyTorch*. Manning Publications Company.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, *27*, 3104-3112.

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2020). Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Tiedemann, J., & Scherrer, Y. (2017, September). Neural Machine Translation with Extended Context. In *Proceedings of the Third Workshop on Discourse in Machine Translation* (pp. 82-92).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017, December). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 6000-6010).

Vaswani, A., Zhao, Y., Fossum, V., & Chiang, D. (2013, October). Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1387-1392).

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2018, November). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In

Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (pp. 353-355).

Wang, C., Cho, K., & Gu, J. (2020, April). Neural machine translation with byte-level subwords. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 05, pp. 9154-9160).

Wang, C., Wu, Y., Liu, S., Yang, Z., & Zhou, M. (2020, April). Bridging the gap between pre-training and fine-tuning for end-to-end speech translation. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 05, pp. 9161-9168).

Werlen, L. M., Ram, D., Pappas, N., & Henderson, J. (2018). Document-Level Neural Machine Translation with Hierarchical Attention Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 2947-2954).

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., & Auli, M. (2019). Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Wu, L., Pan, X., Lin, Z., Zhu, Y., Wang, M., & Li, L. (2020, November). The Volctrans Machine Translation System for WMT20. In *Proceedings of the Fifth Conference on Machine Translation* (pp. 305-312).

Wu, S., Wang, X., Wang, L., Liu, F., Xie, J., Tu, Z., ... & Li, M. (2020, November). Tencent neural machine translation systems for the WMT20 news translation task. In *Proceedings of the Fifth Conference on Machine Translation* (pp. 313-319).

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, W., & Carpuat, M. (2021). EDITOR: An Edit-Based Transformer with Repositioning for Neural Machine Translation with Soft Lexical Constraints. *Transactions of the Association for Computational Linguistics*, *9*, 311-328.

Yadav, V., & Bethard, S. (2018, August). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 2145-2158).

Yang, N., Liu, S., Li, M., Zhou, M., & Yu, N. (2013, August). Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 166-175).

Yang, Z., Zhang, J., Meng, F., Gu, S., Feng, Y., & Zhou, J. (2019, November). Enhancing Context Modeling with a Query-Guided Capsule Network for Document-level Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 1527-1537).

Yao, S., & Wan, X. (2020, July). Multimodal transformer for multimodal machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4346-4350).

Zaretskaya, A., Pastor, G. C., & Seghiri, M. (2017). User perspective on translation tools: Findings of a user survey. *Trends in E-tools and Resources for Translators and Interpreters*, 37-56.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). *Dive into deep learning*.

Zhang, B., Titov, I., & Sennrich, R. (2019, November). Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 897-908).

Zhang, J., & Zong, C. (2020). Neural machine translation: Challenges, progress and future. *Science China Technological Sciences*, 1-23.

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1253.

Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., ... & Gao, J. (2021). VinVL: Revisiting Visual Representations in Vision-Language Models. *arXiv preprint arXiv:2101.00529*.

Zhang, Z., Shi, Y., Yuan, C., Li, B., Wang, P., Hu, W., & Zha, Z. J. (2020). Object relational graph with teacher-recommended learning for video captioning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13278-13288).

Zhou, L., Palangi, H., Zhang, L., Hu, H., Corso, J., & Gao, J. (2020, April). Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 07, pp. 13041-13049).

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., ... & Liu, T. Y. (2020). Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*.