sciendo INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS

Article | DOI: 10.2478/ijssis-2023-0010 Issue 1 | Vol. 16 (2023)

Hyper-parameter optimization in neural-based translation systems: A case study

Goutam Datta[1,2,*], Nisheeth Joshi[1] and Kusum Gupta[1]

[1]Deparment of Mathematical and Computer Science, Banasthali Vidyapeeth, Rajasthan, India

[2]School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India

*Emails: gdatta1@yahoo.com; jnisheeth@banasthali.in; gupta_kusum@yahoo.com

Received for publication: February 22, 2023.

Abstract

Machine translation (MT) is an important use case in natural language processing (NLP) that converts a source language to a target language automatically. Modern intelligent system or artificial intelligence (AI) uses a machine learning approach and the machine has acquired learning ability using datasets. Nowadays, in the MT domain, the neural machine translation (NMT) system has almost replaced the statistical machine translation (SMT) system. The NMT systems use a deep learning framework in their implementation. To achieve higher accuracy during the training of the NMT model, extensive hyper-parameter tuning is required. The paper highlights the significance of hyper-parameter tuning in various machine learning algorithms. And as a case study, in-house experimentation was conducted on a low-resource English–Bangla language pair by designing an NMT system and the significance of various hyper-parameter optimizations was analyzed while evaluating its performance with an automatic metric BLEU. The BLEU scores obtained for the first, second, and third randomly picked test sentences are 4.1, 3.2, and 3.01, respectively.

Keywords
Neural machine translation, LSTM, transformer, GAN, hyper-parameter

1. Introduction

NMT uses a deep neural network-based approach to translate a source language to a target language with the help of a trained corpus. The NMT has replaced the SMT by applying a neural-based approach to its language model design [1, 2]. In MT, there are two important components: the language model and the translation model. SMT uses a statistically based hidden Markov model (HMM) to design its language model component [3]. A phrase-based SMT model is better than a word-based SMT model.

NMT solves many problems that were present in SMT and its earlier versions, such as dictionary-based and rule-based approaches [4]. The NMT got its recognition gradually when Google first developed an NMT model that used an end-to-end approach where a multilayered LSTM model was used that mapped the source component to a fixed-sized vector representation. Another deep LSTM model decoded this vector to produce the target language [5–7].

There are several other variants of NMT models proposed by various researchers; for instance, Google developed another refined model that eliminates this fixed-sized vector bottleneck problem by a soft search technique that considers the entire portion of the source representation that is essential for the target representation. Researchers have conducted many surveys and reviews of the MT domain and captured in detail its paradigm shift [8, 9].

All these NMT models require hyper-parameter tuning to generate better results [10–12]. However, tuning hyper-parameters is not an easy job due to the high training time of deep learning models [13]. There are several techniques for hyper-parameter tuning. The hyper-parameters are actual models' parameters based on their architectural complexity, such as the number of hidden layers, learning rate, etc. These hyper-parameters are tuned before training any ML models [14]. When the models are being trained with an initial set of hyper-parameters, only the optimal set of parameters is computed (learned). Basic ML models such as logistic regression have regularization (L1, L2) as their hyper-parameters; support vector machine (SVM) has a kernel and penalty parameter as its hyper-parameter, and K-nearest neighbor has value K as its hyper-parameter. For advanced models such as artificial neural networks (ANN), learning rate, dropout, and activation function are some of the important hyper-parameters. The search space is the entire volume of hyper-parameters and, out of that, different techniques are applied to select those hyper-parameter values from their given range of dimensions so that the most optimal model can be designed that can return maximum accuracy and minimum error. There are various basic searching strategies, such as grid search and random search, that are widely used in ML models. Some of the advanced searching strategies, such as Bayesian optimization and evolutionary optimization, are also helpful in solving complex ML use cases. These searching strategies with hyper-parameters to train a model to achieve maximum accuracy with its new set of trained parameters are called hyper-parameter optimization techniques. However, in NMT, apart from hyper-parameter optimization, there are other approaches such as linguistic features, representation of abstract meaning on semantic graphs, and various other novel approaches used by researchers to improve the performance of NMT systems, specifically when using low-resource language pairs [15–18].

The aims of this paper are to:

- explore various strategies toward hyper-parameter tuning in various ML models;
- provide a comprehensive review of the importance of hyper-parameter tuning in achieving various state-of-the-art results in NMT systems; and
- implement a case study to evaluate an NMT model on English to Bangla parallel corpus and evaluate its performance by using automatic metric BLEU and highlight a few useful conclusions.

The remainder of the paper is organized as follows: Section 2 briefly discusses some NMT models. Section 3 reports some previous work on NMT and, based on various hyper-parameter selections, their performance on the automatic metric. Section 4 briefly describes the methodology. Section 5 covers our experimentation on the MNIST data set to validate the hyper-parameter tuning and its effect on the performance of the ML model. Furthermore, in this section, we also explored the performance of an NMT system with low-resource language as a case study. Section 6 presents some analysis of the results and discussion. Finally, Section 7 has some useful conclusions drawn from the overall research and provides future directions.

2. State-of-the-art NMT Models

NMT uses a neural network in the language model and is known as a neural language model. Unlike the traditional SMT systems where language and translation models need to be tuned separately, NMT integrates both the language model (a neural language model) and the translation model (responsible for translating the source to target), thereby easing its implementation compared to SMT models. The artificial neural network (ANN) is the basic building block of NMT.

ANN has an input layer, a hidden layer, and an output layer. In the optimization problem, specifically in gradient-based optimization, the gradient is considered one of the important parameter estimators that is used to compute the parameters of the network under different iterations [19, 20]. In the network, we have to search for the suitable values of weights so that we can reduce the error function, which is nothing but the difference between the actual and desired output. The basics of feedforward neural networks are primarily based on the computation of the derivatives of the error function [21, 22].

In a simple feedforward neural network, each unit in the network computes a weighted sum of the product of inputs, where w_{ji} is the weight associated with that connection and is the input to the unit j, as represented in Figure 1. Its mathematical representation is as follows:

$$a_j = \sum_i w_{ji} z_i \tag{1}$$

The sum represented in Eq. (1) is transformed with the help of non-linear activation function g(.) to get Z_j to unit j, as represented below:

$$z_j = g(a_j) \tag{2}$$

We can represent the error function as the sum of all the patterns in the training data set. Each error pattern can be represented separately as follows:

$$E = \sum_n E^n \tag{3}$$

In the above expression, E is the error and n represents labels of the pattern, where E^n can be differentiable with respect to output y and hence can be represented as follows:

$$E = E^n (y_1 \ldots\ldots\ldots, y_c) \tag{4}$$

Here, the primary objective of the network is to compute the derivative of the error function with respect to the weights and biases of the given network.

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{5}$$

Now, using a new notation,

$$\partial_j \equiv \frac{\partial E^n}{\partial a_j} \tag{6}$$

Here, ∂s are the errors, and using Eq. (1), we can write,

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \tag{7}$$

Substituting Eqs. (6) and (7) into Eq. (5), we finally obtain,

$$\frac{\partial E^n}{\partial w_{ji}} = \partial_j z_i \tag{8}$$

With the use of the above formulas, the final back propagation formulae can be represented as follows:

$$\partial_j = \dot{g}(a_j) \sum_k w_{kj} \partial_k \tag{9}$$

Hence, since we know the value of one of the output units, we can easily compute the values of all hidden units by recursively applying the above formulae in a feedforward neural network.

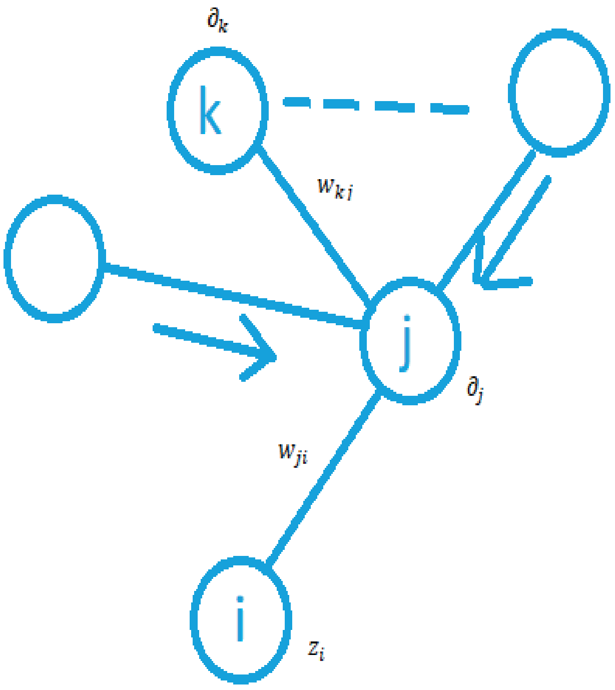


Figure 1: Schematic illustration of computation of ∂_j for hidden unit j with the help of back propagation.

ANN optimization technique general rules: We start training our network with some randomly chosen weight values. It has already been found that optimization algorithms where the error function decreases monotonically often reach local minima. It is therefore essential to select a carefully appropriate set of initial weights that can eventually produce a good set of weights that can lead to faster training of the overall network. Even stochastic algorithms such as gradient descent are very sensitive toward initial weight values. In the majority of the cases, the initial weights are randomly selected with small values. However, if the initial weights are too small, then the sigmoidal activation function would be almost linear, which results in a longer training time [21, 22]. There are various hyper-parameter searching strategies used in ML and deep learning models. Some of the most commonly used searching strategies are grid search, manual search, and random search. A nice comparative study of different types of hyper-parameter searching strategies is presented by the researchers Bergstra et al. in the paper Random Search for Hyper-parameter Optimization [23].

When the hidden layers are more than two, we consider the network a deep neural network [24]. Deep neural networks are widely exploited in NMT.

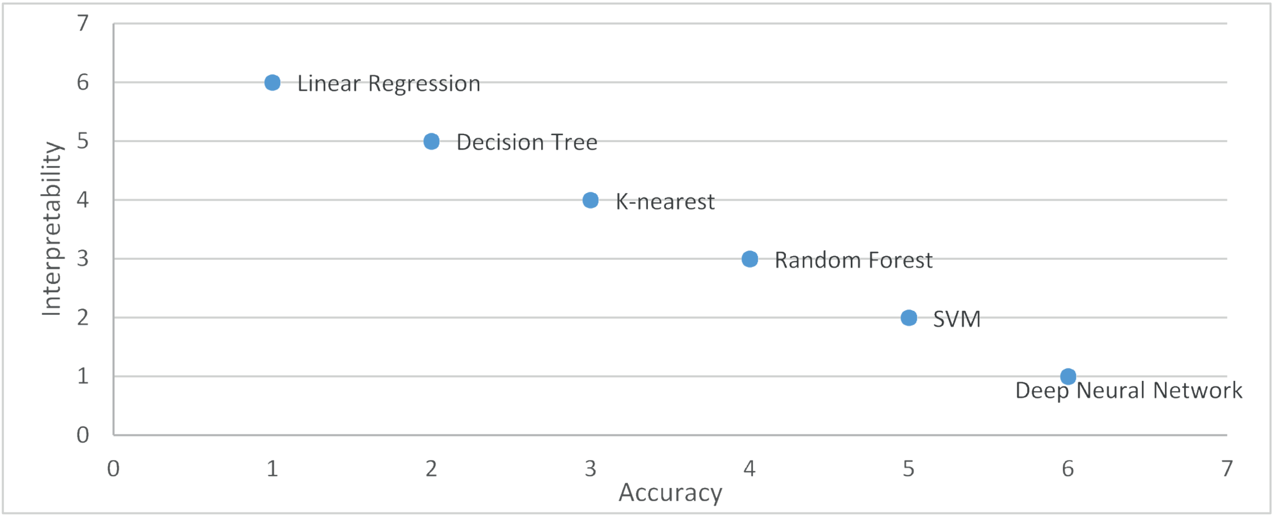


Figure 2: The accuracy of deep learning models is higher, but their interpretability is low compared to other ML models.

Deep neural networks are a very powerful machine learning model as far as accuracy is concerned. However, in terms of interpretability, these models (ANN, deep learning) are difficult to interpret compared to other classifiers (decision tree, SVM, etc.) and regression models (Fig. 2). In Figure 2, the X-axis represents the measure of accuracy, while the Y-axis represents the measure of interpretability. In terms of accuracy, deep learning models are more accurate than basic machine learning models.

The performance of various machine learning models is measured based on certain parameters such as accuracy, speed, robustness, scalability, and interpretability. As mentioned before, deep learning models are the best among other machine learning models in terms of accuracy, but these models are difficult to interpret. One of the major challenges of a deep learning model is while it processes the sequence of data. Deep learning has an essential role in solving NLP applications. As stated before, in designing NMT's language model, deep learning has a very significant role [25]. Deep learning models try to convert the sequence of data into a fixed-size vector representation [26]. These NMT models are encoder–decoder models. An encoder converts the variable-length input sentence to a fixed-length vector representation. The decoder takes this vector representation and converts it to a variable-length target representation. One of the simplest NMT models is sequence-to-to-sequence. Because the size of the NMT source and target data is not fixed, sequence-to-to-sequence models have difficulty handling such scenarios. There are two approaches proposed for handling variable-length sequences: recurrent neural networks (RNN) and gated recursive convolutional neural networks (GRCN). One of the simplest representations of the recurrent language model is shown in Figure 3. In the diagram, the input layer is represented with u, which is the weight matrix from input to hidden layer s; v is the weight matrix from the hidden layer to the output layer. If we ignore the recurrent weight w, it is a bigram neural network language model. The output layer is represented with y, and the hidden layer is represented with h.

A few terms related to RNN's language model can be mathematically represented as follows:

Hidden layer $s(t)$

$$s(t) = f\left(u.x(t) + w.s(t-1)\right) \tag{10}$$

Output layer $y(t)$ and $g(.)$ is the activation function applied

$$y(t) = g\left(vs(t)\right) \tag{11}$$

$$f(z). = 1/1 + e^{-z} \tag{12}$$

$f(z)$ is the sigmoid activation function, and the range varies from 0 to 1.

$$g(z_m) = e^{zm} \Big/ \sum_k e^{zk} \tag{13}$$

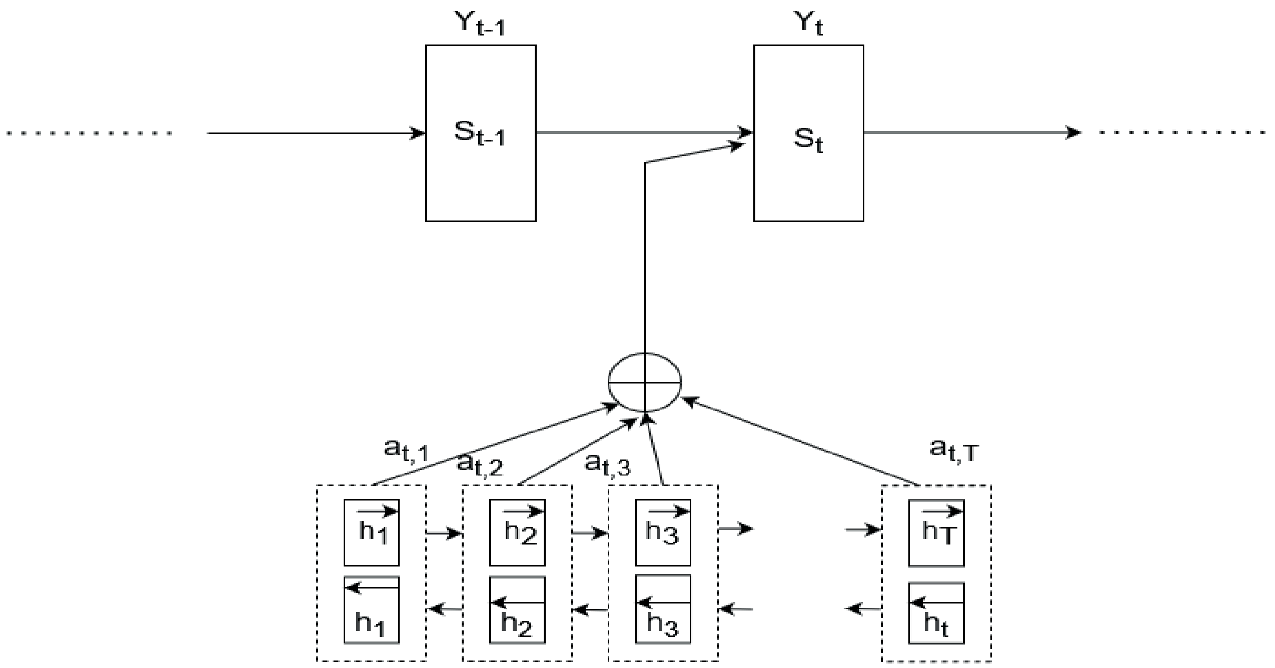


Figure 3: Schematic representation of bidirectional RNN (*Source*: Bahdanau et al. [26]).

$g(z)$ is the soft max activation function. The purpose of the soft max activation function is to ensure that all outputs are greater than zero, and that their sum is one.

RNN suffers from vanishing gradient and exploding gradient problems when dealing with longer sentence sizes. Hence, another popular approach, long short-term memory (LSTM), was introduced to overcome this problem. One of the major advantages of using the LSTM model is its ability to deal with long-term temporal dependency. NMT has the challenge of handling rare words. Researchers have also tried to address this problem and achieved a satisfactory result as well [16].

As stated before, although the LSTM model has the ability to capture long-term dependencies, in some instances, it becomes forgetful and is unable to generate the correct translations. Capturing the context is also essential in NLP. One of the most popular and widely used NMT approaches is the attention-based model. Attention-based models are suitable for handling longer sentences. The attention-based model focuses on different parts of the source sentence and tries to capture the most important word, which decides the context of the entire sentence. In Figure 3, Bahdanau et al.'s bidirectional RNNs are used for annotating text [27]. The forward RNNs are used to compute the input sequence in the order in which it is represented in the input, i.e., from X_1 to X_T and calculate all forward hidden states from h_1 to h_T, and in a similar manner, backward RNNs are used to perform the same operation in the backward direction. Furthermore, the annotation of each word is explored with the help of concatenating forward and backward hidden states.

RNN is adequately capable of representing recent word annotations of a hidden state hj that contains the overall summaries of all preceding and succeeding words. Finally, the computed sequence of annotations and alignment will be used to compute the context vector. In Bahdanau et al.'s model [27], the context vector is computed with the help of embedding all input words and representing them in hidden states. This is done by taking a weighted sum of all hidden states with the help of the following expression:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{14}$$

where c_i is the context vector.

Weighted sums are computed with the help of

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{15}$$

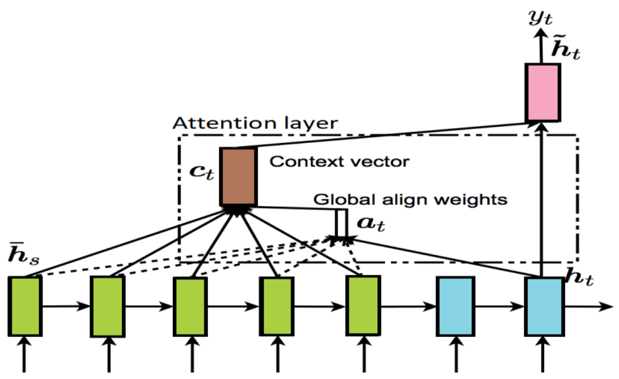


Figure 4: RNN with global attention.

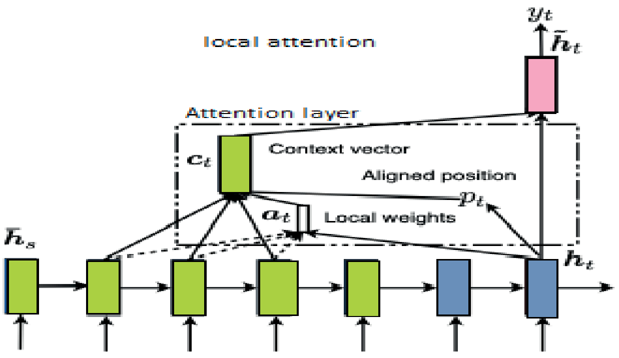


Figure 5: RNN with local attention.

This attention mechanism can be global or local. The global attention mechanism was originally introduced by Loung et al. [28]. In global attention (Fig. 4), when computing the context vector ct, it considers all hidden states that are available on the encoder side. In the global attention model, the variable-length alignment vector (at) is computed from the target side's current hidden state (ht) to each source-side hidden state (hs) as follows: at (s)= align (ht, hs). In a nutshell, global attention requires a lot of computation as it considers all hidden states. Hence, as the input size increases, the computation also increases. To overcome this problem, local attention is the solution (Fig. 5). In local attention, only a subset of hidden states is considered.

Another recent and popular approach is the transformer-based model (Fig. 6). Transformer models are faster than traditional RNN and convolutional models during training [29]. The transformer model's Bidirectional Encoder Representation of Transformers (BERT) has shown significant performance improvement in large-scale language modeling applications. BERT's pretrained model can be used to model several state-of-the-art NLP cases by adding and fine-tuning one extra output layer [30]. Some of the major drawbacks of RNN, such as its inherent sequential nature, i.e., one word at a timestamp, make it slow and inefficient in making a decision. On the other hand, the transformer model replaces this sequence nature by parallelization, i.e., it considers all sets of input sequences with their attention in a parallel manner, thus making it a fast and accurate prediction. The transformer model considers all inputs and their corresponding output sequences to be produced in an O (1) operation with the help of its powerful attention mechanism. Transformer models are encoder–decoder-based, have a strong attention mechanism, and replace the need for sluggish RNN, thereby making them extremely suitable for NMT. The attention mechanism used in the transformer or any other deep learning model can be thought of as a weight vector that decides, out of several words in a sequence/sentence, which one is more important and is responsible for deciding the context of the entire representation. The attention mechanism scans the entire sequence and tries to capture that word, and the highest weightage is assigned to it. The transformer encoder part, by default, has six layers. The decoder side also has six layers. The six layers are further divided into two sublayers, viz., multi-head self-attention mechanism and fully connected feedforward neural network. In the encoder side, at the very first instance, source-side input tokens are first embedded and then passed on to the positional encoder layer. The positional encoder is essential in transformer architecture since it has no recurrent and convolutional units; the language model scans the entire sequence of words, and the context will be decided based on the words. Hence, its position finding is essential. In the transformer model, various sine and cosine operations are performed to find the position of words. After this, the embedded vectors/tokens are fed to the self-attention layer. By default, the results of the self-attention layer are passed through a linear layer, and some dropout operations are also carried out before feeding into the subsequent layers. The attention mechanism in transformer architecture is multi-head attention. If the number of heads is n, this implies n numbers of heads are working simultaneously on the different subspaces of the entire text and are thereby able to capture better context. There is another concept in transformer architecture, i.e., self-attention. Self-attention is the process by which the entire string is scanned word by word, and the attention mechanism also checks for its surrounding

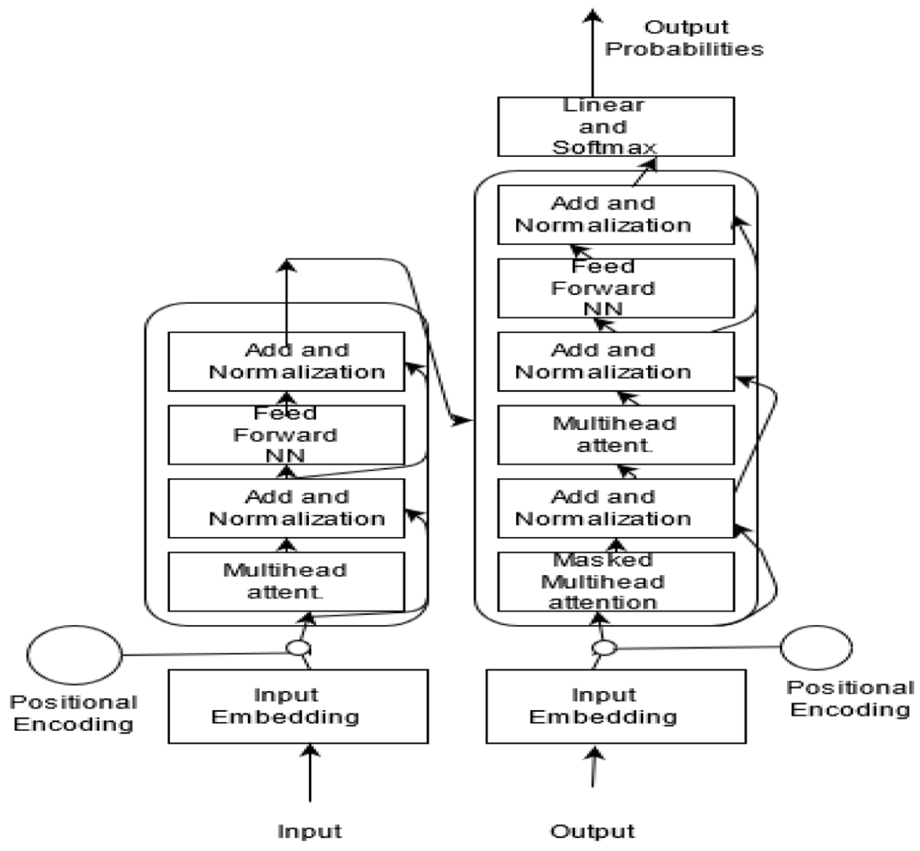


Figure 6: Transformer model.

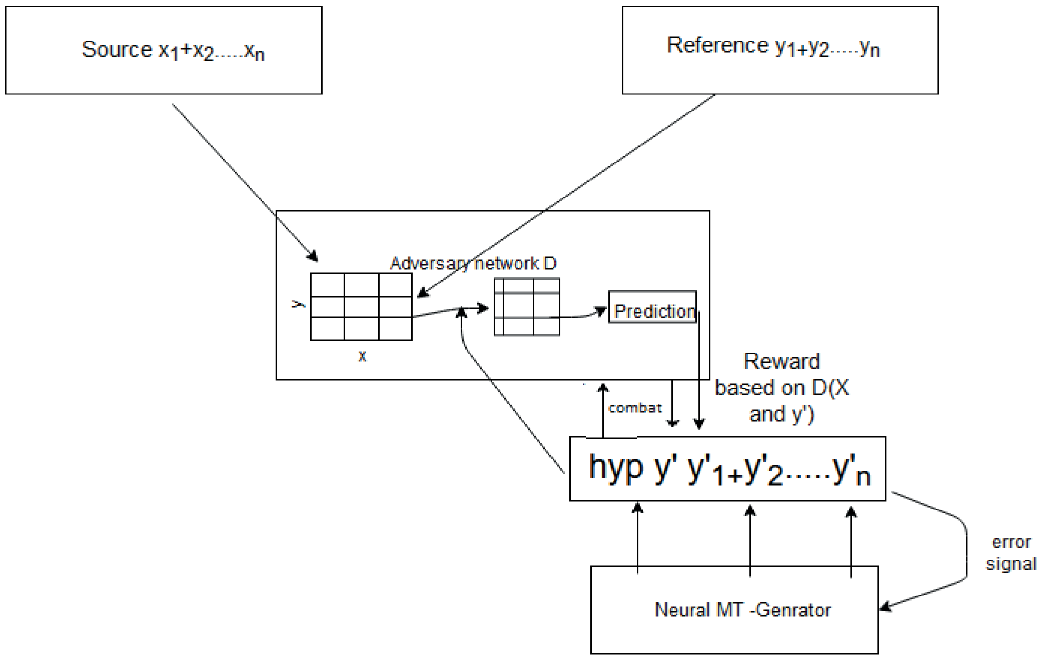


Figure 7: The generative adversarial network (GAN) model in the NMT case.

words to better understand the present word. On the other hand, the decoder predicts the target language with the help of state vector representation.

One of the new paradigms in the deep learning framework is generative adversarial network (GAN) [31]. In GAN, there are two neural networks that compete. One network is a generator, and the other is a discriminator. The generator generates plausible data. It is a model responsible for generating plausible examples of data in the given domain space with a fixed-length random vector with a Gaussian distribution. The discriminator tries to discriminate between real data and fakes (generated by generators). GAN is now widely used by researchers in a variety of NLP applications [32–35]. Figure 7 represents GAN in NMT implementation. Here, the adversary network (D) is a discriminator, which gets its input from the generator, which is our NMT model. The discriminator receives the following inputs: candidate translation from the generator (NMT model), reference translation, and source text. The discriminator tries to identify only the correct translation based on the comparison between the candidate and its reference translation for any given input source sentence. This comparison generates either a reward (if it matches the reference) or an error signal (if there is a mismatch). This error signal will be treated as feedback, and the NMT model can be further trained so that going forward, it can generate the correct translation.

3. Related work on NMT with tuned parameters for Performance optimization

There are a number of hyper-parameters under different machine learning algorithms. In deep learning, there are many hyper-parameters compared to other machine learning algorithms. The hyper-parameters in deep learning are the design parameters for the model, which include the number of hidden layers, the activation function, the optimizer, etc. Hyper-parameters are variables that we need to set to some initial default values before training. We have summarized a few hyper-parameters in Table 1. Researchers have focused on automating the hyper-parameter optimization (HPO) process, but very little work is done in the NMT space due to its large number of variants.

Merity et al. [36], in the NMT discussed in their paper, proposed a strategy involving dropping the weights in LSTM and introducing a new variant of averaged stochastic gradient along with some other regularizing strategies. They achieved excellent word level perplexities in their datasets. Liu et al.'s research [37] uses a deep transformer model with 60 encoder layers and 12 decoder layers, allowing a BLEU score enhancement by 2.5 to be achieved on its baseline model with six layers. The details of the same are provided in a summarized form in Tables 2 and 3. Zhang and Duh [38] proposed a look-up-based approach that may accelerate the use of automatic hyper-parameter optimization in the NMT domain. The look-up-based approach uses a library of pre-trained models with a wider range of hyper-parameters, and such an arrangement facilitates the fast, efficient, and economical execution of the HPO task in the NMT research space. There are also proposals for automatically evaluating metrics for performance evaluation. Table 4 reports the details of the hyper-parameters used to tune the different models, with their

Table 1. Generic hyper-parameters in NMT-based model.

Model	Type of MT	Hyper-parameters
Deep learning models	NMT	Hidden layers, learning rate, activation function, epochs, batch size, dropout, regularization

Table 2. WMT-14 English–French test results showed that 60L-12L ADMIN outperforms the default base model 6L-6L in different automatic metrics (Liu et al. [35]).

Model	Param	TER	METEOR	BLEU
6L-6L Default	67M	42.2	60.5	41.3
6L-6L ADMIN	67M	41.8	60.7	41.5
60L-12LDefault	262M	Diverge	Diverge	Diverge
60L-12LADMIN	262M	40.3	62.4	43.8

Table 3. WMT-14 English–German test results show that ADMIN outperforms the default base model 6L-6L in different automatic metrics (Liu et al. [35]).

Model	Param	TER	METEOR	BLEU
6L-6L Default	61M	54.4	46.6	27.6
6L-6L ADMIN	61M	54.1	46.7	27.7
60L-12LDefault	256M	Diverge	Diverge	Diverge
60L-12LADMIN	256M	51.8	48.3	30.1

Table 4. Models per data set and their best BLEU scores and respective hyper-parameter configurations (Zhang and Duh [36]).

Data set	No. of models	Best BLEU	BPE	No. of layers	No. of embedding	No. of hidden layers	No. of attention heads	Init-lr
Chinese–English	118	14.66	30k	4	512	1024	16	3e-4
Russian–English	176	20.23	10k	4	256	2048	8	3e-4
Japanese–English	150	16.41	30k	4	512	2048	8	3e-4
English–Japanese	168	20.74	10k	4	1024	2048	8	3e-4
Swahili–English	767	26.09	1k	2	256	1024	8	6e-4
Somali–English	604	11.23	8k	2	512	1024	8	3e-4

Table 5. MT models for different language pairs in a GPU-based single-node and multiple-node environment with a wider range of hyper-parameters and their BLEU scores (Lim et al. [11]).

Cell	Learning rate	ro→en P100	ro→en V100	en→ro P100	en→ro V100	de→en P100	de→en V100	en→de P100	en→de V100
GRU	le-3	35.53	35.43	19.19	19.28	28.00	27.84	20.43	20.61
	5e-3	34.37	34.05	19.07	19.16	26.05	22.16	N/A	19.01
	le-4	35.47	35.46	19.45	19.49	27.37	27.81	Dnf	21.41
LSTM	le-3	34.27	35.61	19.29	19.64	28.62	28.83	21.70	21.69
	5e-3	35.05	34.99	19.48	19.43	N/A	24.36	18.53	18.01
	le-4	35.41	35.28	19.43	19.48	N/A	28.50	Dnf	Dnf
GRU	le-3	34.22	34.17	19.42	19.43	33.03	32.55	26.55	26.85
	5e-3	33.13	32.74	19.31	18.97	31.04	26.76	N/A	26.02
	le-4	33.67	34.44	18.98	19.69	33.15	33.12	Dnf	28.43
LSTM	le-3	33.10	33.95	19.56	19.08	33.10	33.89	28.79	28.84
	5e-3	33.10	33.52	19.13	19.51	N/A	29.16	24.12	24.12
	le-4	33.29	32.92	19.14	19.23	N/A	33.44	Dnf	Dnf

Table 6. NMT models with some other range of learning rate (hyper-parameter) (Lim et al. [11]).

Cell	Learning rate	ro→en P100	ro→en t	ro→en V100	ro→en t	de→en P100	de→en t	de→en V100	de→en t
GRU	0.0	34.47	6:29	34.47	4:43	32.29	9:48	31.61	6:15
	0.2	35.53	8:48	35.43	6:21	33.03	18:47	32.55	19:40
	0.3	35.36	12:21	35.15	7:28	31.36	10:14	31.50	9:33
	0.5	34.50	12:20	34.67	17:18	29.64	11:09	30.21	11.09
LSTM	0.0	34.84	6:29	34.65	4:46	32.84	12:17	32.88	7:37
	0.2	34.27	8:10	35.61	6:34	33.10	16:33	33.89	13:39
	0.3	35.67	9:56	35.37	11:29	33.45	20.02	33.51	15:51
	0.5	34.50	15:13	34.33	12:45	32.67	20.02	32.20	13.03

performances being measured with the automatic metric bilingual evaluation understudy (BLEU).

Lim et al. [11] trained a language model of their NMT system to explore which set of hyper-parameters is able to produce better performance in terms of convergence rate and translation accuracy to produce high-performing MT systems. They tested their models in a GPU-based single-node and multiple-node environment. Translation tasks were carried out for English ⇒ Romanian, Romanian ⇒ English, English ⇒ German, and German ⇒ English. The details of their models and the range of different hyper-parameters along with the automatic BLEU score are reported in Tables 5 and 6.

There is an interesting paper on NMT where the authors achieved a significant speedup in the training time of the NMT model. Generally, NMT requires more training time when the corpus size is large. Furthermore, when the corpus size is small, NMT does not produce better results. Hence, this always remains a challenge in NMT due to its long training time. However, in this research, the authors not only achieved less training time for their model but were also able to get better performance compared to some state-of-the-art models for German-to-English and English-to-French translation tasks. In their NMT model, they used a tan hyperbolic activation function. The activation function is one of the important hyper-parameters in the NMT model. This tan hyperbolic activation function has the ability to learn the future and the past context. This important feature helps train the model faster and achieve better results [39].

4. Methodology

When we train our model, first we need to select the default set of hyper-parameters and using those hyper-parameters, our NMT model was trained. Then, based on these hyper-parameters, our model's performance was evaluated (Fig. 8). Based on the model performance, another set of hyper-parameters was explored and the performance was evaluated. The

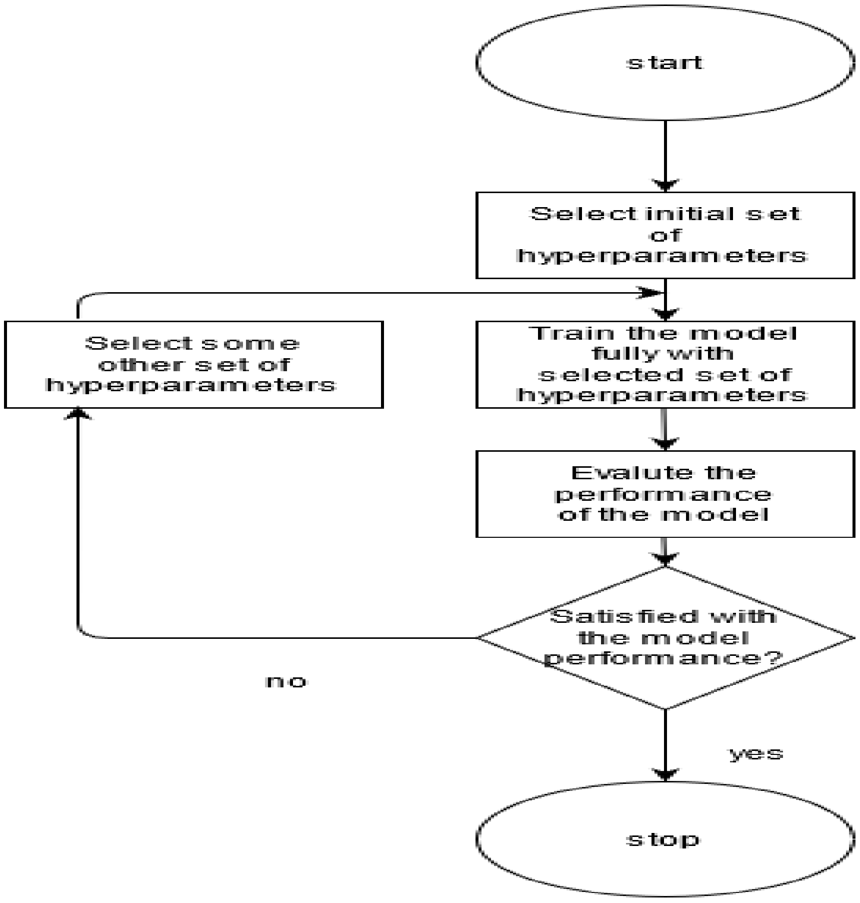

start
Select initial set of hyperparameters
Train the model fully with selected set of hyperparameters
Evalute the performance of the model
Satisfied with the model performance?
no
Select some other set of hyperparameters
yes
stop

Figure 8: Typical machine learning model building steps.

process would continue until we were satisfied with the model's performance. In ML, this hyper-parameter tuning greatly affects the model's performance in various ways. By changing the hyper-parameter configuration, we can restrict our model from overfitting and underfitting. Out of many hyper-parameters, learning rate and optimizers are a few.

We have validated our work with some basic and advanced ML algorithms with their default and two popular approaches, namely grid and random search for hyper-parameter tuning. Results are compared based on the default set of hyper-parameters and then by applying grid and random search approaches. We have also used the MNIST dataset for our hyper-parameter optimization experimentation.

In our NMT design, the overall working methodology of the model is explained with the help of the following diagram (Figure 9):

Inputs: For the given input, which is English in our case, each word in the input sequence will be encoded into one hot vector.

Embedded layers: In the embedding layers, words are turned into vectors, and the size of the vectors now depends on the size of the vocabulary, or the number of words in that vector representation that have the same meaning or domain.

Recurrent layers: In the recurrent layer, the context of the sentence from the previous layer will be applied to the current layer and so on.

Dense layers: These are fully connected layers used in the decoder side to decode the encoded input sequence.

Outputs: The outputs are again a sequence of vectors and these vectors will be mapped to the corresponding Bengali word to generate the final translated output.

In the encoder part, we have an RNN model. Because of RNN's long-term dependency problem,

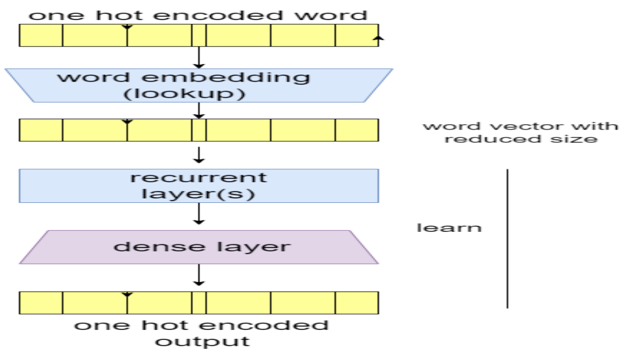


Figure 9: Schematic representation of the NMT model.

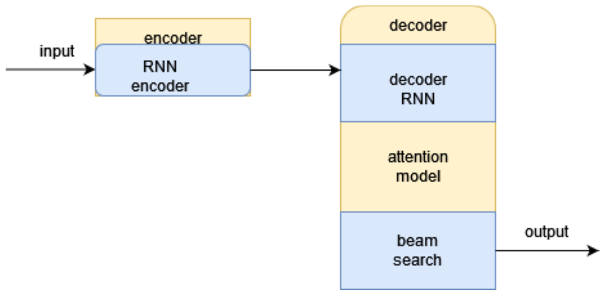


Figure 10: Encoder–decoder-based NMT model.

LSTM, GRU, bidirectional LSTM, or any advanced architecture, such as a transformer model, is widely used. Similarly, on the decoder side, we also have RNN or its other advanced variants that are used along with the attention model to have better accuracy during translation. Beam search is used to select the best target language representation for the corresponding source language (Fig. 10).

For NMT evaluation, we have used an automatic evaluation metric, i.e., BLEU [40]. BLEU takes n-gram candidate representation and n-gram reference representation. Apart from BLEU, there are other popular automatic evaluation metrics used in NMT evaluation these days [41]. Considering these two representations, BLEU then tries to count the total number of matches. The range of the BLUE metric is from 0 to 1. The highest score is 1, and the lowest is 0.

5. Experimental Setup

We have experimented with 70,000 grayscale images from the MNIST data set consisting of hand-written digits ranging from 0 to 9. We have 2D images of resolution 28 × 28. The task is to classify the unseen image of any digit. We have used 60,000 images for training and 10,000 images for testing. We used Keras's sequential model with densely connected layers. We fixed 512 units in our layer and the activation function ReLU. In the output layer, we have used the softmax function. We had 10 possible categories in the output layer, with the result that there would be 10 units in the output layer. To evaluate how well our model performs during training, we used the loss function as categorical entropy.

Our model is sequential. The total number of parameters was 669,706, the total number of trainable parameters was 669,706, and the total number of untrainable parameters was 0. We ran our model for five epochs, and within five epochs we got close to 100%

Table 7. Training and validation accuracy of our model with five epochs.

Epochs	Training accuracy	Validation accuracy
1	0.9426	0.9698
2	0.9730	0.9708
3	0.9792	0.9776
4	0.9829	0.9726
5	0.9859	0.9762

Table 8. Training and validation accuracy of our model with a higher number of epochs.

Epochs	Training accuracy	Validation accuracy
1	0.9431	0.9606
2	0.9742	0.9729
3	0.9796	0.9777
4	0.9835	0.9748
5	0.9865	0.9794
6	0.9872	0.9802
7	0.9896	0.9830
8	0.9898	0.9782
9	0.9916	0.9764
10	0.9924	0.9799

accuracy (Table 7). The graphical representation of this is shown in Figure 11.

We have run the same experiment for 10 epochs and the accuracies are as follows (Table 8):

We have observed (Table 8 and Figure 12) that there is a slight increase in training and validation accuracy with increasing the number of epochs.

We then reduced the number of units in the

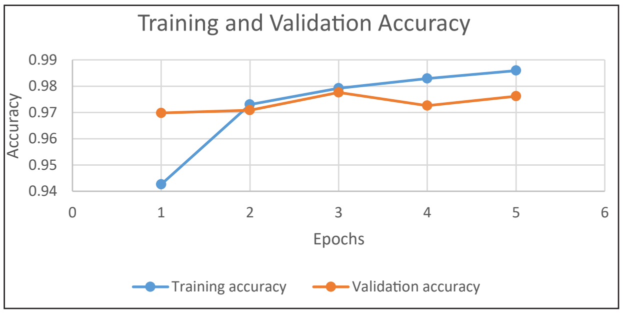


Figure 11: Training and validation accuracy with five epochs.

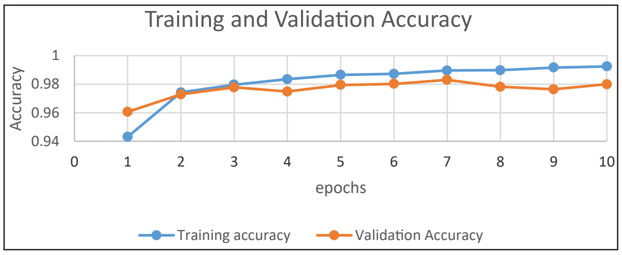
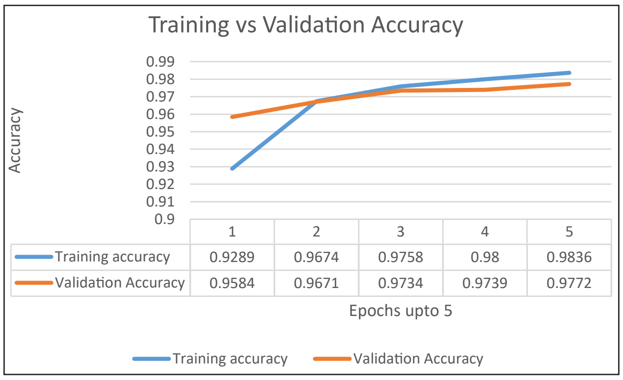


Figure 12: There is a slight improvement in training and validation accuracy with increasing the number of epochs up to 10.

Table 9. Training and validation accuracy with 100 units in different layers with five epochs.

Epochs	Training accuracy	Validation accuracy
1	0.9289	0.9584
2	0.9674	0.9671
3	0.9758	0.9734
4	0.9800	0.9739
5	0.9836	0.9772



	1	2	3	4	5
Training accuracy	0.9289	0.9674	0.9758	0.98	0.9836
Validation Accuracy	0.9584	0.9671	0.9734	0.9739	0.9772

Figure 13: Graphical representation of training and validation accuracy with reduced units in different layers and up to five epochs.

different layers from 512 to 100, and report the resultant training and validation accuracy in Table 9. A graphical illustration of the same is shown in Figure 13. We can see that the training and validation accuracy under different epochs was reduced a little bit. However, in the last epoch, the validation accuracy was marginally high. Reducing the number of units may cause an underfitting problem.

Table 10. Training and validation accuracy with 100 units in different layers with 10 epochs.

Epochs	Training accuracy	Validation accuracy
1	0.9293	0.9631
2	0.9674	0.9730
3	0.9763	0.9751
4	0.9807	0.9729
5	0.9829	0.9724
6	0.9852	0.9780
7	0.9882	0.9773
8	0.9890	0.9756
9	0.9908	0.9784
10	0.9913	0.9793

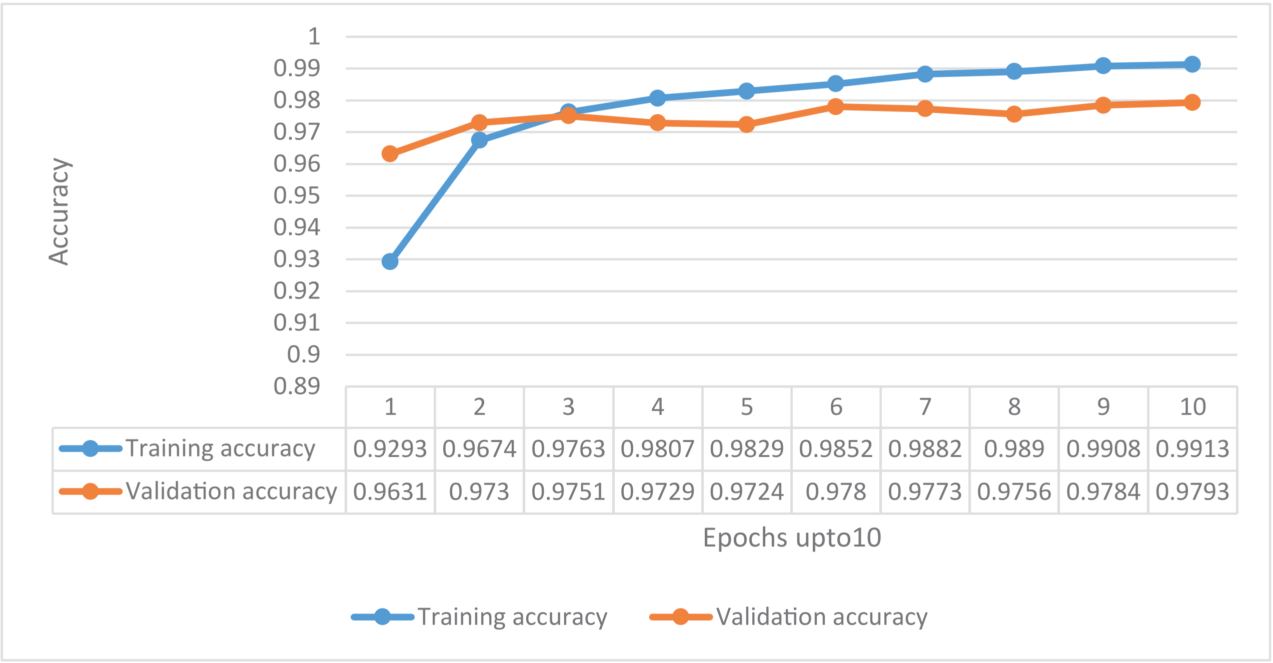
We have now increased the number of epochs employed and kept the number of units under different layers the same, i.e., 100 units. Now our total parameter is 89,610, the total trainable parameter is 89,610, and the non-trainable parameter is 0. The resultant training and validation accuracy is reported in Table 10 and the associated graphical representation in Figure 14. We can observe that the training and validation accuracy increased marginally with increasing the number of epochs.

Table 11. Statistics of English to Bangla tourism corpus (text) collected from TDIL.

Corpus (English to Bangla)	Size in terms of sentence pairs
Tourism	11,976

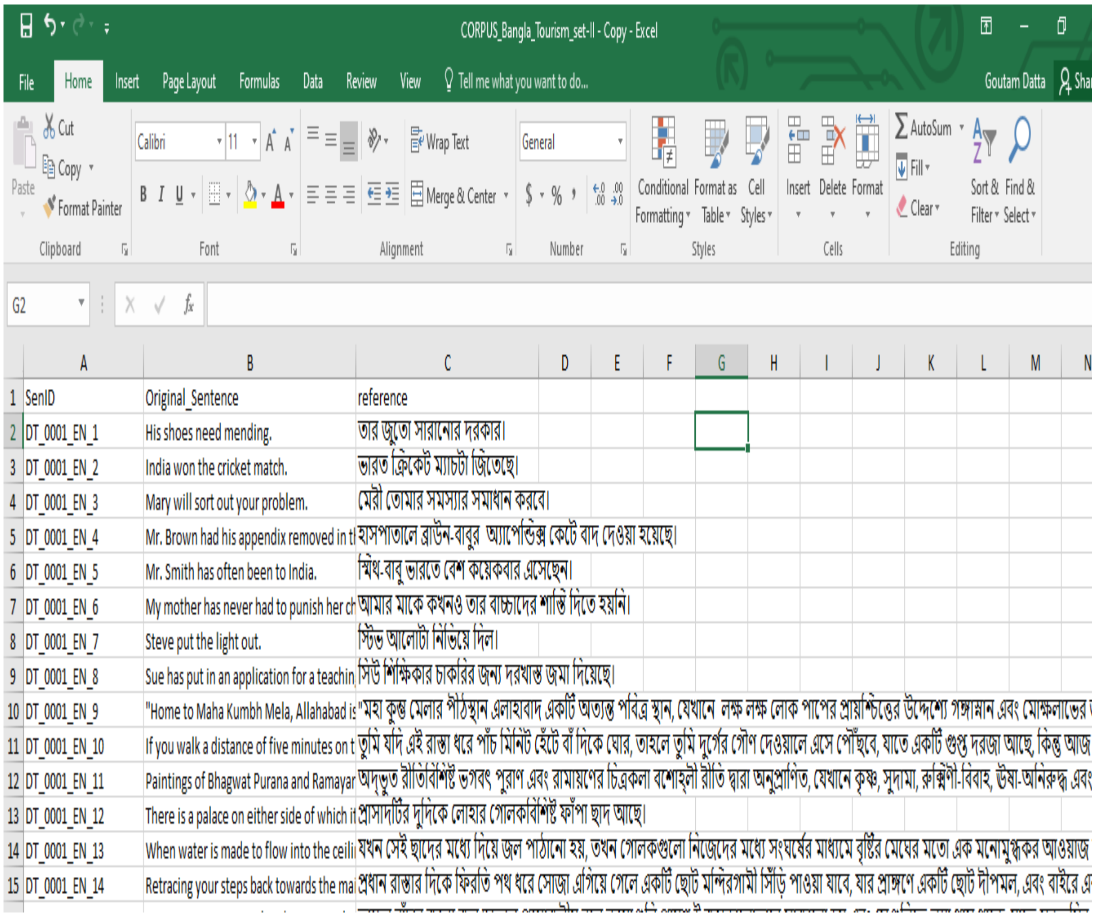
Our second experiment was on NMT. In our experiment, we used the low-resource language pair English to Bangla. Bangla is mostly spoken in India and Bangladesh. There are machine translation systems developed for various Indian languages [42]. We used the tourism dataset, taken from TDIL (https://tdil.meity.gov.in/). Statistics and a snapshot of the corpus are shown in Table 11 and Figure 15, respectively.

We used the bidirectional LSTM model. We adopted the BiLSTM network-based technique and the design is depicted in Figure 15. It is made up of an encoder and a decoder. The encoder learns to turn the input source text with embeddings into a hot vector. The decoder learns to translate this vector to generate the output translation throughout the training phase. BiLSTMs are used in the encoder and decoder. LSTMs are a kind of recurrent neural



	1	2	3	4	5	6	7	8	9	10
Training accuracy	0.9293	0.9674	0.9763	0.9807	0.9829	0.9852	0.9882	0.989	0.9908	0.9913
Validation accuracy	0.9631	0.973	0.9751	0.9729	0.9724	0.978	0.9773	0.9756	0.9784	0.9793

Figure 14: Graphical representation of training and validation accuracy with reduced units per layer and increased number of epochs, i.e., up to 10.



	A	B	C
1	SenID	Original_Sentence	reference
2	DT_0001_EN_1	His shoes need mending.	তার জুতো সারানোর দরকার।
3	DT_0001_EN_2	India won the cricket match.	ভারত ক্রিকেট ম্যাচটা জিতেছে।
4	DT_0001_EN_3	Mary will sort out your problem.	মেরী তোমার সমস্যার সমাধান করবে।
5	DT_0001_EN_4	Mr. Brown had his appendix removed in t	হাসপাতালে ব্রাউন-বাবুর অ্যাপেন্ডিক্স কেটে বাদ দেওয়া হয়েছে।
6	DT_0001_EN_5	Mr. Smith has often been to India.	স্মিথ-বাবু ভারতে বেশ কয়েকবার এসেছেন।
7	DT_0001_EN_6	My mother has never had to punish her ch	আমার মাকে কখনও তার বাচ্চাদের শাস্তি দিতে হয়নি।
8	DT_0001_EN_7	Steve put the light out.	স্টিভ আলোটা নিভিয়ে দিল।
9	DT_0001_EN_8	Sue has put in an application for a teachin	সিউ শিক্ষিকার চাকরির জন্য দরখাস্ত জমা দিয়েছে।
10	DT_0001_EN_9	"Home to Maha Kumbh Mela, Allahabad is	"মহা কুম্ভ মেলার পীঠস্থান এলাহাবাদ একটি অত্যন্ত পবিত্র স্থান, যেখানে লক্ষ লক্ষ লোক পাপের প্রায়শ্চিত্তের উদ্দেশ্যে গঙ্গাস্নান এবং মোক্ষলাভের
11	DT_0001_EN_10	If you walk a distance of five minutes on t	তুমি যদি এই রাস্তা ধরে পাঁচ মিনিট হেঁটে বাঁ দিকে ঘোর, তাহলে তুমি দুর্গের গৌণ দেওয়ালে এসে পৌঁছবে, যাতে একটি গুপ্ত দরজা আছে, কিন্তু আজ
12	DT_0001_EN_11	Paintings of Bhagwat Purana and Ramayar	অদ্ভুত রীতিবিশিষ্ট ভগবৎ পুরাণ এবং রামায়ণের চিত্রকলা বশোহ্লী রীতি দ্বারা অনুপ্রাণিত, যেখানে কৃষ্ণ, সুদামা, রুক্মিণী-বিবাহ, উষা-অনিরুদ্ধ এবং
13	DT_0001_EN_12	There is a palace on either side of which it	প্রাসাদটির দুদিকে লোহার গোলকবিশিষ্ট ফাঁপা ছাদ আছে।
14	DT_0001_EN_13	When water is made to flow into the ceili	যখন সেই ছাদের মধ্যে দিয়ে জল পাঠানো হয়, তখন গোলকগুলো নিজেদের মধ্যে সংঘর্ষের মাধ্যমে বৃষ্টির মেঘের মতো এক মনোমুগ্ধকর আওয়াজ
15	DT_0001_EN_14	Retracing your steps back towards the ma	প্রধান রাস্তার দিকে ফিরতি পথ ধরে সোজা এগিয়ে গেলে একটি ছোট মন্দিরগামী সিঁড়ি পাওয়া যাবে, যার প্রাঙ্গণে একটি ছোট দীপমল, এবং বাইরে এ

Figure 15: Snapshots of the English–Bangla parallel corpus collected from TDIL.

network (RNN) that is often used to capture long-term dependencies.

RNNs record all past information in a memory cell to forecast the output of an input sequence, which limits their ability to predict the output of a very long sentence. To address this restriction, LSTMs were developed, which are composed of input, output, and forget gates and are capable of recording long-term relationships.

The hyper-parameter of our model and the BLEU score are reported in Table 12. We randomly picked three sentences from the test set for

Table 12. Performance of BiLSTM, Google Translate, and Bing in terms of the automatic metric BLEU.

Model	Hyper-parameter	BLEU score
BiLSTM (for English to Bangla; 1st sentence)	Optimizer = Adam;	4.1
BiLSTM (for English to Bangla; 2nd sentence)	Learning rate = 0.001;	3.2
BiLSTM (for English to Bangla; 3rd sentence)	No. of encoder and decoder layers = 6	3.01

Table 13. Translations generated by Google and Bing.

Translators	Language pair	BLEU
Google	English ⇒ Bangla (1st sentence)	36.84
	English ⇒ Bangla (2nd sentence)	6.42
	English ⇒ Bangla (3rd sentence)	4.52
Bing	English ⇒ Bangla (1st sentence)	36.11
	English ⇒ Bangla (2nd sentence)	6.01
	English ⇒ Bangla (3rd sentence)	4.05

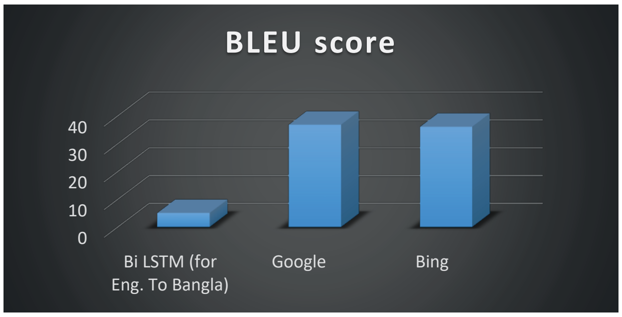


Figure 16: BLEU scores produced by different NMT models for the first test data.

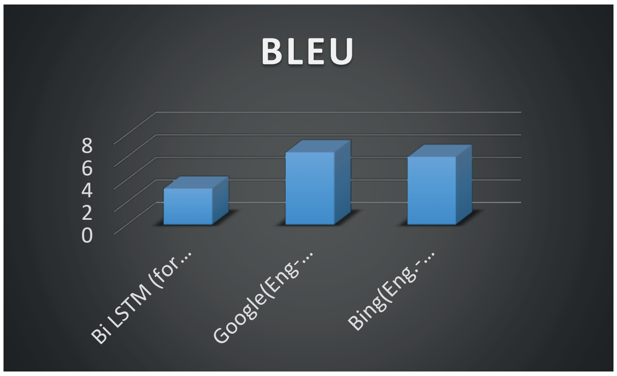


Figure 17: BLEU scores generated by different NMT models for the second test data.

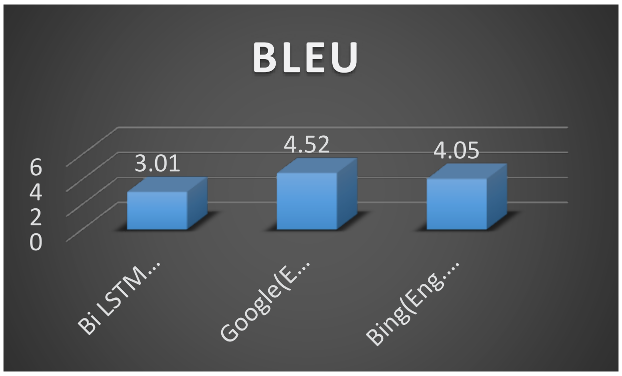


Figure 18: BLEU score produced by various NMT models on third test data.

inferencing. Our model's score as well as Google and Bing translators' scores are presented in Tables 12 and 13, respectively. The graphical illustrations of test set, sentences 1, 2, and 3, are, respectively, shown in Figures 16–18.

We randomly picked the first sentence from the test data set:

English: Ask your tribal guide and he will show you the traditional fishing gears, hunting gadgets, and indigenous medicinal plants – an important part of their culture and more.

Gold Standard Output: তোমার উপজাতীয় পথপ্রদর্শককে জিজ্ঞেস করে দেখো আর সে তোমাকে প্রথাগত মাছ ধরার উপকরণ, শিকার করার সামগ্রী ও দেশজ ঔষধি উদ্ভিদ দেখাবে,যা তাদের সংস্কৃতি

Google Translate: আপনার উপজাতীয় গাইডকে জিজ্ঞাসা করুন এবং তিনি আপনাকে ঐতিহ্যগত মাছ ধরার সরঞ্জাম, শিকারের গ্যাজেট এবং দেশীয় ঔষধি গাছ দেখাবেন - তাদের সংস্কৃতির একটি গুরুত্বপূর্ণ অংশ এবং আরও অনেক কিছু৷

Bing Translate: আপনার উপজাতীয় গাইডকে জিজ্ঞাসা করুন এবং তিনি আপনাকে ঐতিহ্যবাহী মাছ ধরার গিয়ার, শিকারী গ্যাজেট এবং দেশীয় ঔষধি গাছপালা দেখাবেন - তাদের সংস্কৃতির একটি গুরুত্বপূর্ণ অংশ এবং আরও অনেক কিছু৷

We randomly picked the second sentence from the test data set:

English: Mr. Smith has often been to India.

Gold Standard output: স্মিথ-বাবু ভারতে বেশ কয়েকবার এসেছেন৷

Google Translate: মিস্টার স্মিথ প্রায়ই ভারতে গেছেন৷

Bing Translate: স্মিথ প্রায়ই ভারতে যেতেন৷

We randomly picked the third sentence from the test data set:

English: Take a tour of Jaipur to know the city which is fairly young, less than three centuries old.

Gold standard output: তিনশো বছরেরও কম পুরনো নবীন শহর জয়পুর ঘুরে দেখ৷

Google Translate: তিন শতাব্দীরও কম পুরানো শহরটি মোটামুটি তরুণ, শহরটি জানতে জয়পুরে ঘুরে আসুন৷

Bing Translate: জয়পুরের একটি ভ্রমণ করুন শহরটি জানতে যা মোটামুটি তরুণ, তিন শতাব্দীরও কম পুরানো৷

6. Analysis and Discussion

We provide a detailed analysis of our experimentation in this section. First, in the MNIST dataset, we started with varying different hyper-parameters such as the number of units in different layers, number of epochs, etc., and observed their effect on training and validation accuracy. In ANN/deep neural networks, it is difficult to predict in advance the number of iterations that would be needed for the model's convergence. In a deep neural network, the number of hidden layers and the total number of neurons in each layer are two very important hyper-parameters [43]. Here, our study was mainly focused on the following hyper-parameters of the sequential model: activation function: ReLU, number of units in layers: 512 (initially), number of epochs, etc. In our case, we have found that the model converges after 10 epochs. When we reduced the number of units in different layers, we found there is a slight decline in both accuracies. However, reducing more units in different layers may cause underfitting. In the ML models, the number of epochs refers to the number of times for which the training data are shown to the network during training. We need to keep on increasing the number of epochs until the validation accuracy starts decreasing while the training accuracy is increasing. This overfits the model. To overcome this overfitting problem, dropout is one of the techniques. With dropout, the model can be regularized. The general rule is to apply dropouts between 20% and 50%. In dropout, the number of neurons in layers is reduced. Hence, too high a dropout results in the model coming to be characterized by the behavior of an underlearner. Another important hyper-parameter is the number of hidden layers. Several hidden layers are to be increased until we don't get any further improvement in the test data set (error). The model's accuracy depends on the number of units in different layers.

Second, for the NMT system, we have used the following hyper-parameters along with others: optimizer, learning rate, encoder, and decoder layers. The selected values of our hyper-parameters are reported in Table 12. We have used Adam as an optimizer. Adam is an appropriate selection when the data sets and parameters are large in a model. It combines gradient descent with momentum and RMSProp [44]. The second hyper-parameter is the learning rate, which is one of the important hyper-parameters for training the neural network model. The learning rate value ranges from 0.0 to 1.0. We have selected the learning rate value as 0.001. If the learning rate value is too high, then the model may converge faster but to a suboptimal level of solution. And if the value is too low, then it will take longer to train the model, i.e., will require more epochs. As stated before, several hidden layers need to be increased till we don't get any further improvement in test data. In our model, we have taken the number of encoder and decoder layers to six. With this hyper-parameter setting of our bidirectional LSTM model and with randomly picked three English sentences from the test data set, we achieved BLEU scores of 4.1, 3.2, and 3.01 for sentences 1, 2, and 3, respectively. We compared the results of our model with the output from the translation services offered by Google and Bing. Referring to the results shown in Tables 12 and 13 and their graphical representation in Figures 16–18, we observe that our model's BLEU score is much lower than those of the Google and Bing translation services for the three randomly picked English sentences from the test data set, although we are not aware of the optimized hyper-parameters used in either of the translation engines, Bing Translator and Google Translate. However, there are several possible reasons for this. Some of the reasons are as follows: we used a corpus that is not large enough as demanded by the deep learning model. A sufficient and appropriate corpus has a significant impact on NMT performance [45]. We experimented with a limited corpus, and that too from the low-resource category. Also, our language pairs are highly morphological and syntactically different. These are some of the linguistic challenges along with our model constraint. However, neither the BLEU scores that are generated for our model nor those for Google Translate and Bing Translator are completely reliable. BLEU's score generation process solely depends on matching criteria between the translated results (sentences) and the gold standard output (reference sentences). It matches n tokens at a time, i.e., 1-gram (uni-gram), 2-gram (bigram), and n-gram from candidate tokens and reference tokens. Candidate tokens are the tokens from the translated sentences, and reference tokens are the tokens from the reference sentences. Based on the translated results as reported in Tables 12 and 13 as well as their graphical representations in Figures 16–18, it can be observed that the scores for the three sample test sentences are quite low. However, the translations generated by our model, as well as those by the Google and Bing translation services, are comprehensible and can be understood by the human evaluator. In MT evaluation, human evaluation is considered to be the best but is a time-consuming process. These generated results were shown to five human evaluators, and according to the opinions of each of these five, all the translated results are comprehensible and their average score comes out to be more than 60 and in a range of 60–85. The maximum scores are for Google Translate and Bing Translator, and the lower scores, ranging from 60 to 65, are for our model.

7. Conclusions and Future Work

In this research, we have systematically examined the significance of hyper-parameter adjustment in several machine learning algorithms. By altering the hyper-parameters and evaluating the overall impact of this alteration on the model's performance, we have also confirmed the phenomenon of this adjustment having a significance in terms of the MNIST dataset. For every machine learning model, hyper-parameter optimization plays a critical role in the avoidance of overfitting and underfitting. We also looked into a case study where NMT was applied to see how well it performed on language pairings with limited resources. The effectiveness of NMT on several low-resource language pairings with various sets of hyper-parameters was thoroughly examined. As previously mentioned, we employed the English to Bangla tourism language pair (corpus), a low-resource language, for a case study and evaluated the performance of our NMT model using a chosen set of hyper-parameters; and resultantly, we have derived the following insightful conclusions:

- Optimal hyper-parameter optimization is crucial in attaining optimal model behavior and resulting in improved accuracy.
- A substantial and relevant corpus is highly vital in NMT for better performance.
- Owing to data-centricity, it is crucial to choose the right model and hyper-parameters in conjunction with suitable linguistic preprocessing such as tokenization, word embedding, named entity, true-casing, stemming, lemmatization, etc.

To improve accuracy when models are trained with low-resource languages, many techniques and methods have been proposed by researchers in terms of model setting with hyper-parameter tuning, model selection, and linguistic aspects, such as transfer learning, exploiting advanced NLP models with multi-level attention, byte-pair encoding (BPE), etc. [46, 47].

In the future, we intend to develop an NMT system while looking for the right hyper-parameters, which might aid us in speeding up the training of advanced deep learning models and enhancing accuracy, particularly for low-resource, highly morphological languages.

Data Availability

The source of the data and its URL are already provided in the manuscript.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] I. J. Unanue, E. Z. Borzeshi, and M. Piccardi, "Regressing Word and Sentence Embeddings for Low-Resource Neural Machine Translation," *IEEE Trans. Artif. Intell.*, vol. 00, no. 0, pp. 1–15, 2022, doi: 10.1109/TAI.2022.3187680.

[2] H. Wang, H. Wu, Z. He, L. Huang, and K. W. Church, "Progress in Machine Translation," *Engineering*, vol. 18, pp. 143–153, 2022, doi: 10.1016/j.eng.2021.03.023.

[3] S. A. Wang Na, Zhang Xiaohong, "A Research on HMM based Speech Recognition in Spoken English," *Recent Adv. Electr. Electron. Eng.*, 2021.

[4] A. Banerjee et al., "BENGALI-ENGLISH RELEVANT CROSS LINGUAL INFORMATION ACCESS USING FINITE AUTOMATA," 2010, pp. 595–599, doi: 10.1063/1.3516373.

[5] P. Koehn, F. J. Och, and D. Marcu, "Statistical Phrase-Based Translation," no. June, pp. 48–54, 2003.

[6] P. Koehn et al., "Moses: open source toolkit for statistical machine translation," *Proc. 45th Annu. Meet. ACL Interact. Poster Demonstr. Sess. - ACL '07*, no. June, p. 177, 2007, doi: 10.3115/1557769.1557821.

[7] Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," pp. 1–23, 2016, [Online]. Available: http://arxiv.org/abs/1609.08144.

[8] F. Stahlberg, "Neural machine translation: A review," *J. Artif. Intell. Res.*, vol. 69, pp. 343–418, 2020, doi: 10.1613/JAIR.1.12007.

[9] Z. Tan et al., "Neural machine translation: A review of methods, resources, and tools," *AI Open*, vol. 1, no. October 2020, pp. 5–21, 2020, doi: 10.1016/j.aiopen.2020.11.001.

[10] E. Salesky, A. Runge, A. Coda, J. Niehues, and G. Neubig, "Optimizing segmentation granularity for neural machine translation," *Mach. Transl.*, vol. 34, no. 1, pp. 41–59, 2020, doi: 10.1007/s10590-019-09243-8.

[11] R. Lim, K. Heafield, H. Hoang, M. Briers, and A. Malony, "Exploring Hyper-Parameter Optimization for Neural Machine Translation on GPU Architectures," pp. 1–8, 2018, [Online]. Available: http://arxiv.org/abs/1805.02094.

[12] R. Rubino, B. Marie, R. Dabre, A. Fujita, M. Utiyama, and E. Sumita, *Extremely low-resource neural machine translation for Asian languages*, vol. 34, no. 4. Springer Netherlands, 2020.

[13] Y. Li, J. Li, and M. Zhang, "Deep Transformer modeling via grouping skip connection for neural machine translation," *Knowledge-Based Syst.*, vol. 234, p. 107556, 2021, doi: 10.1016/j.knosys.2021.107556.

[14] N. Tran, J.-G. Schneider, I. Weber, and A. K. Qin, "Hyper-parameter Optimization in Classification: To-do or Not-to-do," *Pattern Recognit.*, vol. 103, p. 107245, Jul. 2020, doi: 10.1016/j.patcog.2020.107245.

[15] L. H. B. Nguyen, V. H. Pham, and D. Dinh, "Improving Neural Machine Translation with AMR Semantic Graphs," *Math. Probl. Eng.*, vol. 2021, 2021, doi: 10.1155/2021/9939389.

[16] G. X. Luo, Y. T. Yang, R. Dong, Y. H. Chen, and W. B. Zhang, "A Joint Back-Translation and Transfer

Learning Method for Low-Resource Neural Machine Translation," *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/6140153.

[17] J. G. Carbonell, R. E. Cullingford, and A. V. Gershman, "Steps Toward Knowledge-Based Machine Translation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 4, pp. 376–392, 1981, doi: 10.1109/TPAMI.1981.4767124.

[18] C. K. Wu, C. C. Shih, Y. C. Wang, and R. T. H. Tsai, "Improving low-resource machine transliteration by using 3-way transfer learning," *Comput. Speech Lang.*, vol. 72, no. February 2020, p. 101283, 2022, doi: 10.1016/j.csl.2021.101283.

[19] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.

[20] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7700 LECTU, pp. 437–478, 2012, doi: 10.1007/978-3-642-35289-8_26.

[21] M. Y. Mikheev, Y. S. Gusynina, and T. A. Shornikova, *Building Neural Network for Pattern Recognition*. 2020.

[22] C. M. Bishop, *Neural Networks for Pattern Recognition*. USA: Oxford University Press, Inc., 1995.

[23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[24] G. Dhiman, V. Vinoth Kumar, A. Kaur, and A. Sharma, "DON: Deep Learning and Optimization-Based Framework for Detection of Novel Coronavirus Disease Using X-ray Images," *Interdiscip. Sci. – Comput. Life Sci.*, vol. 13, no. 2, pp. 260–272, 2021, doi: 10.1007/s12539-021-00418-7.

[25] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *arXiv Prepr. arXiv1707.05589*, 2017.

[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.

[27] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[28] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *Conf. Proc. - EMNLP 2015 Conf. Empir. Methods Nat. Lang. Process.*, pp. 1412–1421, 2015, doi: 10.18653/v1/d15-1166.

[29] A. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[30] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[31] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020, doi: 10.1145/3422622.

[32] Z. Yang, W. Chen, F. Wang, and B. Xu, "Generative adversarial training for neural machine translation," *Neurocomputing*, vol. 321, pp. 146–155, 2018, doi: 10.1016/j.neucom.2018.09.006.

[33] L. Wu et al., "Adversarial Neural Machine Translation," 2017, [Online]. Available: http://arxiv.org/abs/1704.06933.

[34] Z. Zhang, S. Liu, M. Li, M. Zhou, and E. Chen, "Bidirectional generative adversarial networks for neural machine translation," *CoNLL 2018 - 22nd Conf. Comput. Nat. Lang. Learn. Proc.*, no. CoNLL, pp. 190–199, 2018, doi: 10.18653/v1/k18-1019.

[35] C. H. Lin, C. J. Lin, Y. C. Li, and S. H. Wang, "Using generative adversarial networks and parameter optimization of convolutional neural networks for lung tumor classification," *Applied Sciences (Switzerland)*, vol. 11, no. 2. pp. 1–17, 2021, doi: 10.3390/app11020480.

[36] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, 2018.

[37] X. Liu, K. Duh, L. Liu, and J. Gao, "Very Deep Transformers for Neural Machine Translation," 2020, [Online]. Available: http://arxiv.org/abs/2008.07772.

[38] X. Zhang and K. Duh, "Reproducible and Efficient Benchmarks for Hyperparameter Optimization of Neural Machine Translation Systems," *Trans. Assoc. Comput. Linguist.*, vol. 8, pp. 393–408, 2020, doi: 10.1162/tacl_a_00322.

[39] X. Liu, W. Wang, W. Liang, and Y. Li, "Speed Up the Training of Neural Machine Translation," *Neural Process. Lett.*, vol. 51, no. 1, pp. 231–249, 2020, doi: 10.1007/s11063-019-10084-y.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "{B}leu: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Jul. 2002, pp. 311–318, doi: 10.3115/1073083.1073135.

[41] G. Datta, N. Joshi, and K. Gupta, "Analysis of Automatic Evaluation Metric on Low-Resourced Language: BERTScore vs BLEU Score," in *Speech and Computer*, 2022, pp. 155–162.

[42] A. Pathak and P. Pakray, "Neural machine translation for Indian languages," *J. Intell. Syst.*, vol. 28, no. 3, pp. 465–477, 2019, doi: 10.1515/jisys-2018-0065.

[43] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: Investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," *J. Cheminform.*, vol. 9, no. 1, pp. 1–13, 2017, doi: 10.1186/s13321-017-0226-y.

[44] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[45] H. Gete and T. Etchegoyhen, "Making the most of comparable corpora in Neural Machine Translation: a case study," *Lang. Resour. Eval.*, 2022, doi: 10.1007/s10579-021-09572-2.

[46] B. Zoph, D. Yuret, J. May, and K. Knight, "Transfer learning for low-resource neural machine translation," *EMNLP 2016 - Conf. Empir. Methods Nat. Lang. Process. Proc.*, pp. 1568–1575, 2016, doi: 10.18653/v1/d16-1163.

[47] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *54th Annu. Meet. Assoc. Comput. Linguist. ACL 2016 - Long Pap.*, vol. 3, pp. 1715–1725, 2016, doi: 10.18653/v1/p16-1162.