# Neural Machine Translation of Indian Languages

Karthik Revanuru
IIIT Bangalore
Bangalore, Karnataka, India
karthik.krvnr@iiitb.org

Kaushik Turlapaty
IIIT Bangalore
Bangalore, Karnataka, India
Kaushik.Turlapaty@iiitb.org

Shrisha Rao
IIIT Bangalore
Bangalore, Karnataka, India
shrao@ieee.org

## ABSTRACT

Neural Machine Translation (NMT) is a new technique for machine translation that has led to remarkable improvements compared to rule-based and statistical machine translation (SMT) techniques, by overcoming many of the weaknesses in the conventional techniques. We study and apply NMT techniques to create a system with multiple models which we then apply for six Indian language pairs. We compare the performances of our NMT models with our system using automatic evaluation metrics such as UNK Count, METEOR, F-Measure, and BLEU. We find that NMT techniques are very effective for machine translations of Indian language pairs. We then demonstrate that we can achieve good accuracy even using a shallow network; on comparing the performance of Google Translate on our test dataset, our best model outperformed Google Translate by a margin of 17 BLEU points on Urdu-Hindi, 29 BLEU points on Punjabi-Hindi, and 30 BLEU points on Gujarati-Hindi translations.

## CCS CONCEPTS

• **Computing methodologies** → **Machine translation**; **Machine learning approaches**; **Neural networks**;

## KEYWORDS

Computational Linguistics, GPU, Hindi, India, Machine Translation, Neural Networks

**ACM Reference Format:**
Karthik Revanuru, Kaushik Turlapaty, and Shrisha Rao. 2017. Neural Machine Translation of Indian Languages. In *Proceedings of the 10th Annual ACM COMPUTE Conference, India (ACM COMPUTE 2017).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3140107.3140111

## 1 INTRODUCTION

India is a multilingual country with people from different states speaking in different regional languages. Communication and information exchange between people is necessary not only for business purposes but also for the people to share feelings, thoughts, opinions and facts. As it is not feasible to have human translators everywhere, we need effective approaches which do this job with as little human effort as possible.

Machine Translation (MT) can be described as the task of translating text or speech from one natural language to another, with as little human effort as possible. MT aims to achieve quality translations which are semantically equivalent to the source sentence and syntactically correct in the target language. MT performs simple substitution of words on a ground level, but that alone is not enough, as recognition of whole phrases and their closest counterparts in the target language are necessary.

Work on MT dates back to as early as the 1950s [16], and has progressed rapidly since the 1990s due to the availability of storage, computing power and also large bilingual and multi-lingual text corpora.

Since then, different approaches have been proposed for MT: rule based translation [9, 27], knowledge based translation [26, 29], corpus based translation [24] and hybrid translation [22]. Each of these approaches has its own advantages and disadvantages. Statistical Machine Translation (SMT) [14] (which can be subcategorized under corpus based translation) was widely used as it produced better results compared to other methods. SMT also requires little human intervention, as it learns everything from the parallel corpus. Neural networks [1] in MT are also popular lately—a novel technique in MT called Neural Machine Translation [25] (NMT) has emerged.

Though a lot of work has been done on MT, it is limited to European and other foreign languages. Little has been done on Indian languages. We find no work giving benchmarks on Indian language pairs, which makes us the first to venture into this uncharted territory. There are benchmarks available for English-French and English-German on WMT'14 datasets (which are publicly available) with about 38.95 and 24.67 BLEU score respectively [30].

We have worked on six language pairs: Telugu-Hindi, Konkani-Hindi, Gujarati-Hindi, Punjabi-Hindi, Tamil-Hindi and Urdu-Hindi on datasets obtained from Indian Language Technology Proliferation and Deployment Center (TDIL-DC), C-DAC[1]. We have also evaluated our translations with their reference translations, to estimate the quality of our work using a widely used metric called BLEU [20]. We have also detailed and evaluated on other automatic evaluation metrics like Unknown Word Count (UNK), Word Error Rate (WER) [3], F-Measure [5] and METEOR [15].

The architecture of our system is relatively simple, with two bi-directional Long Short Term Memory (LSTM) [23] as encoders and two LSTMs [11] as decoders. On top of both encoders we have added *residual connections* [30], and an *attention mechanism* [17] on top of the decoder. We have experimented with this shallow network system on the given Indian language pairs, and observed how

[1] http://tdil-dc.in


*ACM COMPUTE 2017, November 16–17, 2017, Bhopal, India*

ACM ISBN 978-1-4503-5323-6/17/11…$15.00
https://doi.org/10.1145/3140107.3140111

they perform. We have used GPU computing to train our system as it accelerates the training process. We have also experimented with different variations of these models on the various language pairs.

We demonstrate with our system that good accuracy can be achieved on Indian language pairs even using a shallow network. It is also easier to train such a network as it requires fewer resources, and takes relatively less time to train, compared to deeper networks. In comparison with Google Translate[2], our system achieves a BLEU score of 46.47 between Punjabi and Hindi language pair on a dataset from TDIL-DC whereas Google Translate scores 17.46 on the same dataset. We get similar results for other language pairs, where also our network outperforms Google Translate by significant margins.

In summary, the following are the novel accomplishments of our work:

- This is the first work to have applied Neural Machine Translation techniques on Indian language pairs. (Language translation work on Indian languages—by any method—is quite sparse; e.g., there is no tool for Konkani-Hindi, which is one of our language pairs.)
- We have trained our models using eight different configurations, and evaluated them using five different standard evaluation metrics.
- Our models are easier to train than deeper models, as they have a simpler architecture, require fewer resources, and take less time.
- We have demonstrated with our work that good accuracy can be achieved even with shallow networks, on Indian language pairs.
- Our best model outperformed Google Translate by a margin of 17 BLEU score on Urdu-Hindi, 29 BLEU score on Punjabi-Hindi and 30 BLEU score on Gujarati-Hindi translations.

The rest of the paper is organized as follows. In Section 2 we have outlined different NMT concepts we have studied, followed by the architecture used. In Section 3 we give an overview of the experiments conducted, describing details like setup, dataset, preprocessing, and training aspects. In Section 4 we describe different evaluation metrics we have used. Section 5 gives results with some sample translations, then compare our different models on the Telugu-Hindi dataset using several evaluation metrics, then extend this to other language pairs. We also compare different models based on training time, to show that our models are easier to train. Subsequently we compare our models with Google Translate to gauge the quality of their outcomes. In Section 6 we present conclusion and future ideas for work.

All the code that has been used in our work, and some documentation, can be found online.[3]

## 2 NEURAL MACHINE TRANSLATION

Neural Machine Translation is an approach to MT that uses a neural network which directly models the conditional probability of translating a given source sentence to a target sentence.

### 2.1 Simple Encoder-Decoder Architecture

A basic network in NMT consists of an encoder and decoder. The natural choice for encoder and decoder is a Recurrent Neural Network (RNN) [11] because RNNs can easily map sequences to sequences when the alignment between inputs and outputs is known ahead of time [25]. RNN is a natural extension of feed forward neural networks with the difference being RNN has a memory, i.e., it takes into account previous outputs.

Let $X$ and $Y$ be the source and target sentence pairs respectively. The encoder RNN converts the source sentence $x_1, x_2...x_n$ into vectors of fixed dimensions. The decoder outputs one word at a time using conditional probability

$$PY|X = PY|X_1, X_2, X_3, ..., X_M$$

Here $X_1, .....X_M$ in the equation are the fixed size vectors encoded by the encoder. Using chain rule the above equation is converted to the equation below where, while decoding, next word is predicted using symbols that are predicted till now and source sentence vectors. The above expression then becomes

$$PY|X = Py_i|y_0, y_1, y_2, ...., y_{i-1}; X_1, X_2, X_3, ..., X_M$$

Each term in the distribution is represented with a softmax[4] over all the words in the vocabulary.

#### 2.1.1 *Long Term Dependency Problem*

Though RNNs work well in theory, there are problems while training them with long sentences because RNNs have a "Long Term Dependency Problem." This can be explained with a simple example. If the given task is to predict the next word in a sentence using a language model, then in a sentence like "Stars are in the _____," *sky* is the obvious choice to fill in the blank, and we do not need any further context as the gap between the relevant information and its place is small. But in sentences like "I am from Andhra Pradesh. I speak fluent _____," it is not as straightforward, because from recent information we can only deduce that missing word should be name of a language, but there can be multiple choices (e.g., Telugu, Urdu, Hindi). If we want to infer any further, i.e., if we want to narrow down the possibilities, we need further context. This problem is felt when the gap between the relevant information and the place that it is needed is not small, and it is entirely possible to have gaps much bigger than in this example. In practice, it is difficult for RNNs to learn these dependencies.

#### 2.1.2 *Long Short Term Memory (LSTM)*

A variant of RNN called Long Short Term Memory (LSTM) [11] is known to learn problems with long range temporal dependencies, so RNNs are sometimes replaced with LSTMs in MT networks, because they may succeed better in this setting.

**Figure 1: LSTM**

[2] https://translate.google.com/

[3] https://github.com/KarthikRevanuru/NMT-of-Indian-Languages

[4] Softmax is the last layer of a neural network, which gives the probability for each of the class labels.

In Figure 1, the model reads an input sentence "AB" and produces "WXY" as the output sentence. The model stops making predictions after outputting the end-of-sentence (EOS) token.

#### *2.1.3 Bidirectional Encoder*

A *bidirectional encoder* [30] is based on the idea that the output at any time instant may not only depend on past data but also on future data. For example, if we are given a task to predict a missing word in a sentence, we read what is written both to the left and to the right of the missing word, to get the context. Using this idea, the LSTM is tweaked to connect two hidden layers of opposite directions to the same output. This tweaked version of LSTM is called a Bidirectional LSTM (Bi-LSTM).

Bi-LSTMs were introduced [23] to increase the amount of input information available to the network. Unlike LSTMs, Bi-LSTMs have access to the future input from the current state without need for time delays. Bi-LSTM is especially useful when the context of the input is needed. Bi-LSTMs are also used [30] as bidirectional encoders.

In MT for some language pairs, information required to translate certain words of the source side can appear on either side of the target word; for certain language pairs it can even be distributed. It makes sense to use Bi-LSTM in this scenario, because we need the best context for translation.

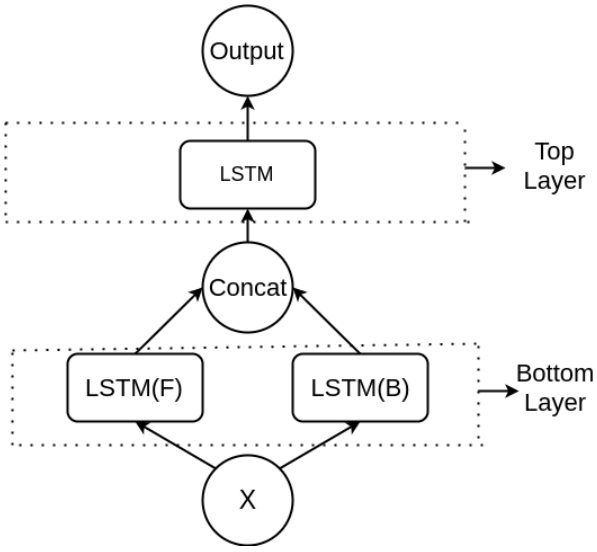


**Figure 2: Bidirectional Encoder**

Figure 2 details our use of Bi-LSTMs. Bi-LSTMs are used only in the bottom layer. The encoder in the bottom layer consists of two independent encoders, the one labelled LSTM(F) encoding a normal sequence, and the LSTM(B) encoding the sequence in the reverse direction. The outputs from both these layers are concatenated and sent to the next layer labeled LSTM.

#### *2.1.4 Deep Bidirectional Encoder*

This is a variant of a bidirectional encoder in which the output of every layer is summed up prior feeding to the next layer.

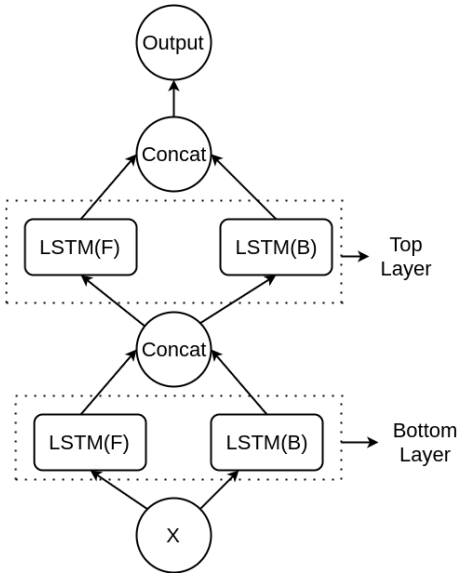


**Figure 3: Deep Bidirectional Encoder**

From Figure 3, we can see that concatenated output from the bottom LSTM(F) and LSTM(B) is sent again to LSTM(F) and LSTM(B), which encode them again in the forward and reverse directions respectively, $X$ is the input. This makes it a *deep bidirectional encoder*.

### 2.2 Residual Connections

There is plenty of theoretical and empirical evidence to suggest that the depth of neural networks is a crucial ingredient for their success. However as the depth of a network increases, it becomes more and more difficult to train, due to vanishing and exploding gradients [21]. This problem has been addressed in the past [10] using the idea of modeling differences between an intermediate layer's output and the targets. These are called *residual connections*.

With residual connections, the input of a layer is added element-wise to the output before feeding to the next layer.

In Figure 4, the output of LSTM1 is added to input $X$ and sent as an input to LSTM2. Residual connections are known to greatly improve the gradient flow in the backward pass, which allows us to train very deep networks.

### 2.3 Bridges

A *bridge* is an additional layer between an encoder and decoder that defines how information is passed from encoder to decoder. Instead of just copying the encoder states to the decoder, encoder states can be passed through a dense layer. To extend this further, it can also use non-linearity in the processing of state information.

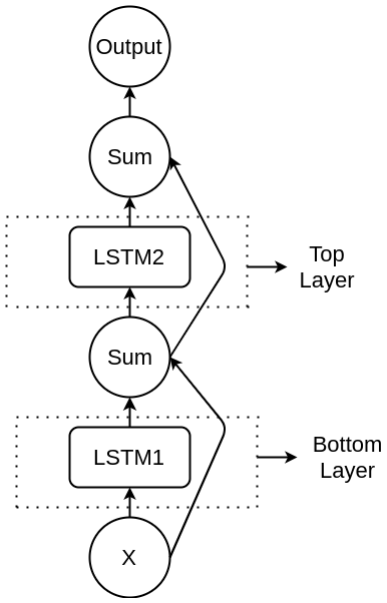


**Figure 4: Residual Connections**

## 2.4 Attention Mechanism

Even though LSTMs are used to deal with the problem of long term dependencies, they do not solve the problem completely solved by them, because the model can still suffer from failures in long sentence translation due to its incapability in capturing long term dependencies. This is largely due to the fact that the encoder of this basic approach needs to compress a whole source sentence into a single vector.

We use an *attention mechanism* [17] to solve this problem. The basic idea of attention mechanisms is that instead of encoding a sentence into a single vector, we encode each word in the sentence into a vector [4], and reference these vectors while decoding. Since the number of vectors available while decoding is equivalent to the number of words in the sentence, long sentences have many vectors and short sentences have few vectors. This makes it feasible to represent sentences in an efficient way, avoiding the problems arising because of the inefficient representation if we encode the whole sentence by a single vector.

We now encode each word in the sentence by passing it into an RNN in both directions. The output from both the directions is concatenated. The notations used below are from [17]

$$h_j = h_j^f \quad h_j^b$$

Here $h_j^f$ and $h_j^b$ are outputs of a word from an RNN in both forward and backward directions respectively.

These vectors are concatenated into a single matrix for all the words.

$$H^F = \text{concat\_col}(h_1^F .... h_F^F)$$

Every column in the matrix represents one word. This matrix is of variable number of columns but the decoder accepts a vector of fixed dimension while decoding to get the context. So that implies context vector should be of fixed length. So we multiply this matrix with the attention vector.

$$c_t = H^F \alpha_t$$

$H^F$ is the matrix, $c_t$ is the context vector and $\alpha_t$ is the attention vector.

The basic idea behind the attention vector is that it indicates how much we are "focusing" on a particular source word at a particular instant of time. The larger the value of $\alpha_t$, the more impact a word has when predicting the next word in the output sentence.

To calculate the attention vector, we first calculate attention scores. These attention scores can be computed with an arbitrary function that takes two vectors as inputs, and outputs a score between 0 and 1 indicating how much to focus on this particular input word encoding $h_j^F$ at the time step $h_t^e$.

We then normalize this to get the actual attention vector itself by taking a softmax over the scores.

$$\alpha_t = softmax a_t$$

This attention vector is then used to weight the encoded representation $H^F$ to create a context vector $c_t$ for the current time step.

We calculate attention scores using the following formulae. Here $h_s$ is source hidden state and $h_t$ is target state.

$$a_t s = \frac{expscore h_t, \bar{h}_s}{_{s'} expscore h_t, \bar{h}_s}$$

$$score h_t, \bar{h}_s = h_t^T W_a \bar{h}_s$$

These context vectors are used while decoding. They give better translations by selectively focusing on words in the source sentence during translation.

## 2.5 System Architecture

Figure 5 depicts the architecture of one of our models with an encoder, decoder and a bridge. The encoder has two layers in which the bottom layer has a Bi-LSTM and the other layer has an LSTM whereas in decoder both are LSTMs. It also depicts the residual connections where output from each layer is added to the input before feeding it to the next layer.

# 3 EXPERIMENTS

We applied the methods described above to the MT task with Indian languages and report the accuracy of these methods with some sample translations in Section 6.

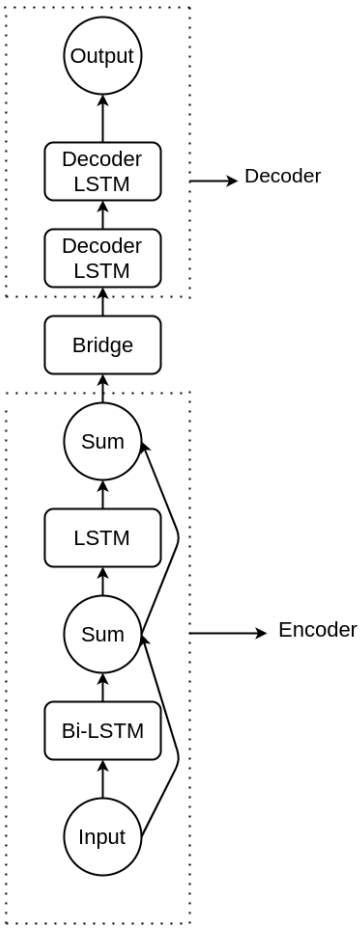


**Figure 5: Architecture**

### 3.1 Setup

A Keras [6] implementation of our basic encoder decoder model with the configuration described in previous sections on a Dual Core CPU with 8 GB of RAM is seen to achieve a throughput speed of approximately 100 words per second. This was too slow for our purposes because one epoch [5] took nearly two hours. So we parallelized our model using a machine with an NVIDIA Geforce GTX 980 GPU. This machine processes at a throughput speed of approximately 1600 words per second.

### 3.2 Dataset Details

We used data obtained on request from the Indian Language Technology Proliferation & Deployment Centre (TDIL-DC). TDIL-DC has linguistic data for different domains like Agriculture, Entertainment, Health and Tourism. We combined data across all these domains to build our final parallel corpus[6]. We allocated 70,15,15 percent of the data for training, validation and testing respectively. The total training data had 64000 sentences in Telugu and Hindi for training, 14184 for validation and 14000 for testing. The data was encoded in UTF-8 format.

[5] An epoch is defined as a single pass through your entire training set while training a machine learning model.

[6] A parallel corpus is a corpus that contains a collection of original texts in source language and their translations into one or many target languages.

### 3.3 Data Pre-Processing

The data from each domain were not given in a single file. They were in separate files which had to be merged. The data had a unique id for each sentence and it was tagged with Parts Of Speech. These had to be removed since we required only source and target sentence. The construction of final corpus was not easy and took time because in some cases translations were missing as they were intermediate versions. So it required a manual check which was very time consuming.

### 3.4 Training Details

We trained this data on the models described above. We found that shallow LSTMs were fairly easy to train as they take less training time. We use LSTMs with 2 layers for encoding and decoding with 500 cells at each layer and 500 dimensional word embeddings, with an input vocabulary of 50,004 and an output vocabulary of 50,004. We used softmax over 50,004 words at each output.

We build a deeper network by using LSTMs with 4 layers. Instead of SGD [18] we also experimented with Adagrad [7] with a learning rate of 0.1. We used Adagrad because sparse features can be very useful and Adagrad makes it such that features that are more sparse in the data have a higher learning rate which translates into a larger update for that feature.

We replace LSTM with Bi-LSTM and also experimented with Deep Bi-LSTM. We also add residual connections and attention mechanism. We also add a dense layer which acts like a bridge between encoder and decoder and compared it's performance with other methods.

We use Keras [6] with Tensor Flow [2] as the backend for basic encoder and decoder model. We used OpenNMT [13] for Bi-LSTM, Residual Connections and Attention Mechanism. Evaluation of all these models is tabulated in the results section below.

## 4 EVALUATION METRICS

We evaluate the different experiments using automatic evaluation metrics such as UNK Count, METEOR, F-Measure, and BLEU.

### 4.1 BLEU Score

The BLEU score [20] is a metric calculated for translated sentences in comparison with human generated reference translations. It computes the $n$-gram precision with respect to the reference translation, but does not take grammatical correctness into account. A Brevity Penalty is added to account for shorter translations. The BLEU score metric does poorly if used to judge individual translated sentences, as it is designed to approximate human judgment at corpus level. The BLEU score is formally a number between 0 and 1, but is usually shown as a percentage score, by multiplying by 100. The higher the value, the better the translation. The BLEU score is one of our evaluation methods, and it is widely used in the MT community.

Therefore, we chose BLEU as a primary evaluation metric. Our results for various experiments are tabulated in Section 6. The code used for the calculation of BLEU scores can be found online [7].

### 4.2 UNK Count

Out-of-vocabulary (OOV) words are usually represented by Unknown Word (UNK) in our machine translated output. The translated sentences are usually said to be better if there are fewer UNK words in the translated output.

### 4.3 WER

Word Error Rate (WER) [3] is a metric used for automatic evaluation of MT system. It compares human translated output to machine translated output. The lower the WER, the better the translation. The code used for the calculation of Word Error Rate can be found online [8].

### 4.4 F-Measure

F-measure [5] is the harmonic mean of precision and recall. Precision is a measure of exactness or quality whereas recall is a measure of completeness and quantity. The higher the F-measure, better is the translation. The code used for calculating the F-Measure can be found online [9].

### 4.5 METEOR

METEOR [15] is based on harmonic mean of unigram precision and recall. This seeks good correlation with human translated output at sentence level whereas BLEU seeks good correlation at corpus level. The higher the METEOR value, better the translation. The `nlp-metrics` package (*op. cit.*) has been used for the calculation of METEOR values.

## 5 RESULTS

### 5.1 Sample Translations

**Telugu to Hindi**

(1) **Telugu:** నందాకోట్ - పీక్ - ఒకవైపు పిండారగంగా - నదీలోయ మరియు రెండో వైపు గౌరీగంగా - నదీలోయ ఉన్నది.

**Hindi:** 'कैथेलिक पीक के एक ओर सूर्य - पश्चिम की ओर तथा दूसरी ओर की खाड़ी का संग्रह होता है ।

(2) **Telugu:** ఈ లోయల్ని కలిపేవాటిని దర్రా దానా దూరా పాస్ అంటారు .

**Hindi:** कहा जाता है कि इस घाटी को ' रोहतंग दर्रा ' कहते हैं।

(3) **Telugu:** నందాకోట్కు - నలువైపులా నాలుగు గ్లేషియర్లు - కూడా ఉన్నాయి.

**Hindi:** नैनीताल के चारों ओर चार शहरों भी हैं ।

From the above translations it can be inferred that the translations are fairly good but there seems to be a problem with translation of nouns. These translations may be handy for day to day tasks but may need improvement for business purposes.

**Tamil to Hindi**

(1) **Tamil:** தியாகி பிராமணர்களின் நலனுக்கு மற்றும் அவர்களை பாதுகாக்கும் பொறுப்பு சமூகம் ஏற்றுக்கொண்டிருப்பதற்கு இது தான் காரணமாகும் .

**Hindi:** इसका कारण यह है कि शहीद - नानषा करने के लिए जिम्मेदार समाज। <unk> और उनकी रक्षा करने के लिए जिम्मेदार समाज है ।

(2) **Tamil:** பத்திரிகையாளர் லட்சாதிபதியாக இருக்க விரும்புவதில்லை ஆனால் தனக்கும் தன்னுடைய குழந்தைகளுக்கும் உடுத்த துணியும் உணவும் தேவைப்படுகின்றது .

**Hindi:** पत्रकार चाहे पत्रकार नहीं बन रही थीं , लेकिन प्रियंका ने अपने बच्चों को भी अपने बच्चों के साथ मारने का भी एक साथ भोजन किया ।

(3) **Tamil:** அத்தியாவசியப் பொருட்கள் தேவையாகத் தான் இருக்கிறது அதனால் வாழ்க்கை பயணம் இயக்கப்படுகிறது .

**Hindi:** जिस प्रकार का सामान ऐसी वेश से होता है जिससे जीवन की यात्रा संचालित होती है ।

**Konkani to Hindi**

(1) **Konkani:** ही एकदम स्पश्टशी गजाल की कॅमेराचे तकनीकेचो विकास आनी फोटोग्राफीच्या आविश्कारा शिवाय सिनेमाय आसचो नाशिल्लो आनी टॅलिवीजनूय बी .

**Hindi:** यह एकदम मेहनत की बात है कि वैक्सिंग की वफादारी का विकास और न ही कपूर के साथ भी हो जायेंगे और न ही गर्भधारणों कैमरे और लय की <unk>की कहानी भी बहुत बड़ी और काफी पुरानी है - लगभग कई सैकड़ों काल में फैली थी ।

(2) **Konkani:** कॅमेराच्या आनी फोटोग्राफीच्या आविश्काराची काणी सुद्दा खूब व्हड आनी बरीच पोरणी आसा - सुमार जायत्या शेंकड्यांनी पातळिल्ली .

**Hindi:** अन्य कई ध्रुवपदों की तरह इन दोनों के स्वरों का श्रेय भी किसी एक व्यक्ति को नहीं दे सकता ।

(3) **Konkani:** हेर खुबश्या आविश्कारां भशेन ह्या दोनांयच्या आविश्काराचो श्रेय सुद्दा खंयच्याय एका मनशाक दिवंक शकना .

**Hindi:** इस प्रकार , कैमरे की भाषा का श्रेय भी बहुत से लोग करियर को भी <unk> देते हैं , जिन्होंने <unk> ने इस साल पहला डिब्बा कैमरे को तैयार किया ।

[7] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl
[8] https://github.com/zszyellow/WER-in-pytflohon
[9] https://github.com/harpribot/nlp-metrics

**Table 1: Comparison of different models using BLEU score**

| Model | BLEU Score |
|---|---|
| 2 Layer LSTM + SGD | 12.67 |
| 4 Layer LSTM + SGD | 4.94 |
| 2 Layer (Bi-dir) LSTM +SGD + Res + Attention | 13.89 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention | 11.55 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention + Bridge (non-linear) | 14.16 |
| 4 layers (Bi-dir) LSTM(Concat)+SGD + Res + Attention | 13.57 |
| 4 layers (deep Bi-dir) LSTM +SGD+Res + Attention | 8.46 |
| 4 layers (Bi-dir) LSTM + AdaGrad+ Res + Attention | 0.65 |

**Table 2: Comparison of different models based on training time**

| Configuration | Start Time | End Time | Time taken |
|---|---|---|---|
| 2 layers (Bi-dir) LSTM + SGD+ Res + Attention | 15:54:56 | 18:23:42 | 2:28:46 |
| 2 layers (Bi-dir) LSTM + SGD+ Res +Attention | 15:54:56 | 20:33:59 | 4:39:03 |

## 5.2 Comparison Of Various Models

In this section we compare various models applied on various datasets based on training time, BLEU score and other evaluation metrics.

#### *5.2.1 Using BLEU Score*

It has been suggested in earlier works that deeper networks have better accuracy compared to shallow networks. But we can see from Table 1 that BLEU score has gone down when we have made our network deeper, i.e., when we increased the number of layers in encoder and decoder from 2 to 4. This is due to vanishing and exponential gradients—so simply stacking layers does not work beyond a point. Residual connections are used to solve this problem. Even with residual connections and attention mechanism, there is a slight dip in the accuracy when the number of layers are increased. This can be attributed to limited training data. It can also be seen that the BLEU score has decreased when the optimization method is changed from SGD to Adagrad. Ideally, Adagrad should give better accuracy because it learns sparse features also, but here it does not. The reason is again the limited training data.

Adding Bi-LSTM increased the accuracy whereas Deep Bi-LSTM decreased the accuracy. This is again because of limited data. Interestingly residual connections on a shallow network increased the accuracy and performed better compared to a deeper network by a margin of two BLEU score. Four layer Bi-directional LSTM with residual connections, attention mechanism and a non linear bridge between encoder and decoder gives the best accuracy.

#### *5.2.2 Based On Training Time*

We compared our models based on training time which is tabulated below for Telugu - Hindi translation.

**Table 3: Comparison of different models using BLEU score on Konkani-Hindi language pairs**

| Model | BLEU Score |
|---|---|
| 2 Layer LSTM + SGD | 24.07 |
| 4 Layer LSTM + SGD | 17.01 |
| 2 Layer (Bi-dir) LSTM +SGD + Res + Attention | 24.35 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention | 25.14 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention + Bridge (non-linear) | 25.39 |
| 4 layers (Bi-dir) LSTM(Concat)+SGD + Res + Attention | 25.25 |
| 4 layers (deep Bi-dir) LSTM +SGD+Res + Attention | 16.56 |
| 4 layers (Bi-dir) LSTM + AdaGrad+ Res + Attention | 0 |

If we correlate the above results and results from Table 1, we can see that there is no big difference in accuracy when we move from a two-layer to a four-layer network, whereas the training time almost increased by two hours. So we can conclude that good accuracy can be achieved even with shallow networks and a lot of training time can be saved, in turn also reducing resource utilization. If this trend can be extended further, it may open up the possibility of training a neural network for translation even on embedded devices.

## 5.3 Extension To Several Indian Language Pairs

We also experimented with our methods on a few other language data sets and evaluated them.

#### *5.3.1 Konkani-Hindi*

The total training data had 75000 sentences in Konkani and Hindi for training, 2000 for validation and 1000 for testing. The data was encoded in UTF-8 format.

From Table 3, we observe that even with Konkani-Hindi same trends repeat, the only difference being that unlike Telugu-Hindi, the four-layer model with residual connections and attention mechanism performs slightly better compared to the two-layer model. In Bi-LSTM, concat works slightly better compared to sum. A four-layer Bi-directional LSTM with residual connections, attention mechanism and a non-linear bridge gives the best BLEU score and AdaGrad gives the lowest BLEU score.

#### *5.3.2 Gujarati-Hindi*

The total training data had 80000 sentences in Telugu and Hindi for training, 2000 for validation and 1000 for testing. The data was encoded in UTF-8 format.

From Table 4, we can observe that Gujarati-Hindi has behavior similar to Konkani-Hindi, the only difference being that a slightly higher BLEU score is achieved without bridge than with one. It is noteworthy that using a bridge gives the greatest accuracy previously, in a four-layer model with Bi directional encoder, residual mechanism and attention mechanism.

#### *5.3.3 Punjabi-Hindi*

We allocated 70,15,15 percent for training, validation and testing. The total training data had 55000 sentences in Telugu and Hindi for training, 15088 for validation and 14802 for testing. The data was encoded in UTF-8

**Table 4: Comparison of different models using BLEU score on Gujarati-Hindi language pairs**

| Model | BLEU Score |
|---|---|
| 2 Layer LSTM + SGD | 34.14 |
| 4 Layer LSTM + SGD | 30.67 |
| 2 Layer (Bi-dir) LSTM +SGD + Res + Attention | 35.26 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention | 35.69 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention + Bridge (non-linear) | 35.18 |
| 4 layers (Bi-dir) LSTM(Concat)+SGD + Res + Attention | 33.91 |
| 4 layers (deep Bi-dir) LSTM +SGD+Res + Attention | 32.62 |
| 4 layers (Bi-dir) LSTM + AdaGrad+ Res + Attention | 0 |

**Table 5: Comparison of different models using BLEU score on Punjabi-Hindi language pairs**

| Model | BLEU Score |
|---|---|
| 2 Layer LSTM + SGD | 43.75 |
| 4 Layer LSTM + SGD | 31.56 |
| 2 Layer (Bi-dir) LSTM +SGD + Res + Attention | 45.97 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention | 46.47 |
| 4 layers (Bi-dir) LSTM +SGD+ Res + Attention + Bridge (non-linear) | 45.48 |
| 4 layers (Bi-dir) LSTM(Concat)+SGD + Res + Attention | 41.14 |
| 4 layers (deep Bi-dir) LSTM +SGD+Res + Attention | 0.89 |
| 4 layers (Bi-dir) LSTM + AdaGrad+ Res + Attention | 3.11 |

**Table 6: Comparison of different language pairs using different evaluation metrics.**

| Language Pair | BLEU | UNK | WER(%) | F-Measure | METEOR |
|---|---|---|---|---|---|
| Punjabi-Hindi | 45.97 | 631 | 37.280 | 0.819 | 0.431 |
| Gujarati-Hindi | 35.26 | 986 | 49.170 | 0.755 | 0.367 |
| Konkani-Hindi | 24.35 | 574 | 52.883 | 0.784 | 0.367 |
| Tamil-Hindi | 7.56 | 747 | 87.276 | 0.702 | 0.309 |
| Urdu-Hindi | 22.47 | 0 | 72.848 | 0.730 | 0.347 |

format.

From Table 5, we can observe that Punjabi-Hindi behaves similar to Gujarati-Hindi.

We evaluated our methods on metrics other than BLEU and the details are tabulated below. Results in the following table are obtained using a 2 layer Bi-LSTM with SGD, Residual Connections and Attention Mechanism.

The above table shows that the language pair with high METEOR score also has high BLEU score and the least word error rate, which implies that the translation model is good overall for that language pair.

**Table 7: Comparison of our model with Google translate**

| Language Pair | Our Best Model | Google Translate |
|---|---|---|
| Punjabi-Hindi | 46.47 | 17.46 |
| Gujarati-Hindi | 35.69 | 4.87 |
| Urdu-Hindi | 22.47 | 5.79 |
| Tamil-Hindi | 7.56 | 2.65 |

## 5.4 Comparison With Google Translate

To determine the effectiveness of our translations we translated our test data on Google Translate and calculated BLEU scores for those and compared it with our scores which are tabulated below.

**Note: Values based on TDIL-DC dataset, Google Translate accessed on $15^{th}$ June 2017**

**It is evident from the table above that our best models have outperformed Google Translate consistently. Since we do not know the data on which Google has trained its model, we are unable to come up with an exact reason why our best models are doing better. We trained our model on sentences relevant to the agriculture, entertainment, health and tourism domains. In these, our models have consistently outperformed Google Translate.**

# 6 CONCLUSION

**In this work we have applied neural machine translation techniques on several Indian Language Pairs and evaluated them on some automatic evaluation metrics. We were surprised with the accuracy with which our model was translating given that we had limited data and a shallow network of two layers. It was even more surprising that our model for Urdu-Hindi, Punjabi-Hindi and Gujarati-Hindi outperformed Google Translate with a margin of over 17, 29 and 30 BLEU score respectively. Most importantly, we demonstrated that good translation accuracy can be achieved even with simple approaches using shallow networks on some Indian language pairs.**

**With the recent success of Generative Adversarial Networks we can experiment GAN's [28] on translation for Indian Languages. We can also try using Convolutional Neural Networks since convolutional approach allows us to discover compositional structure in the sequences more easily since representations are built hierarchically [8]. Since they can work at faster speeds and have better accuracies they can help in our idea of having a real time translator with human level translation quality.**

**We can unleash the potential of this MT system by developing some cloud based applications which gives us the luxury of getting real-time translations in about a fraction of the cost of hiring a translator. Going further, if we can optimize our architecture, our models can be run on embedded devices which opens up new possibilities and gives us the freedom for offline translation. This is particularly useful in countries like India where internet penetration is low especially in rural areas.**

We can extend this to a real time speech to speech translator, i.e., what ever we speak in one language gets translated into target language and it would be read aloud as speech. This would be particularly useful in customer service as majority of population in India is not well versed in English and since customer service and satisfaction are on top priority for many companies they can use this type of speech to speech translation services to serve their customer base. Zero shot translation [12] comes handy to create a single translation system which can translate multiple languages.

Digital literacy in India is low, one of the reasons for this is language barrier. Using our work, applications can be developed to translate content on web. This will reduce the language barrier and would also help the Government with Digital India initiative by bringing more people on to digital platforms.

## ACKNOWLEDGMENTS

We would like to thank Indian Language Technology Proliferation and Deployment Centre(TDIL-DC) a subsidary of CDAC for providing us the data required. We would also like to thank Jaya Nair and Dinesh Babu Jayagopi for permitting us to use their equipment for our work, James Bradbury for his advice, and Graham Neubig for his tutorial [19].

## REFERENCES

[1] 1997. *Proceedings 1997 International Conference on Image Processing, ICIP '97, Santa Barbara, California, USA, October 26-29, 1997*. IEEE Computer Society. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4998

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.

[3] International Speech Communication Association. 2004. *INTERSPEECH 2004 - ICSLIP: Proceedings, 8th International Conference on Spoken Language Processing : October 4 - 8, 2004, International Convention Center Jeju, Jeju Island, Korea*. Sunjin Printing Company. https://books.google.co.in/books?id=nL9lHwAACAAJ

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). http://arxiv.org/abs/1409.0473

[5] Nancy Chinchor. 1992. MUC-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding, MUC 1992, McLean, Virginia, USA, June 16-18, 1992*. 22–29. https://doi.org/10.1145/1072064.1072067

[6] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

[7] John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12 (2011), 2121–2159. http://dl.acm.org/citation.cfm?id=2021068

[8] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. *CoRR* abs/1705.03122 (2017). http://arxiv.org/abs/1705.03122

[9] Siddhartha Ghosh, Sujata Thamke, and Kalyani U. R. S. 2014. Translation Of Telugu-Marathi and Vice-Versa using Rule Based Machine Translation. *CoRR* abs/1406.3969 (2014). http://arxiv.org/abs/1406.3969

[10] Sepp Hochreiter. 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 2 (1998), 107–116. https://doi.org/10.1142/S0218488598000094

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[12] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *CoRR* abs/1611.04558 (2016). http://arxiv.org/abs/1611.04558

[13] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *CoRR* abs/1701.02810 (2017). http://arxiv.org/abs/1701.02810

[14] P. Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press. https://books.google.ch/books?id=4v_Cx1wIMLkC

[15] Alon Lavie and Michael J. Denkowski. 2009. The Meteor metric for automatic evaluation of machine translation. *Machine Translation* 23, 2-3 (2009), 105–115. https://doi.org/10.1007/s10590-009-9059-4

[16] W.N. Locke and A.D. Booth. 1955. *Machine translation of languages: fourteen essays*. Published jointly by Technology Press of the Massachusetts Institute of Technology and Wiley, New York. https://books.google.co.in/books?id=dK8iAAAAMAAJ

[17] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *CoRR* abs/1508.04025 (2015). http://arxiv.org/abs/1508.04025

[18] Deanna Needell, Rachel Ward, and Nati Srebro. 2014. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1017–1025.

[19] Graham Neubig. 2017. Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. *CoRR* abs/1703.01619 (2017). http://arxiv.org/abs/1703.01619

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*. 311–318. http://www.aclweb.org/anthology/P02-1040.pdf

[21] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR* abs/1211.5063 (2012). http://arxiv.org/abs/1211.5063

[22] P. Salunkhe, A. D. Kadam, S. Joshi, S. Patil, D. Thakore, and S. Jadhav. 2016. Hybrid machine translation for English to Marathi: A research evaluation in Machine Translation: (Hybrid translator). In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. 924–931. https://doi.org/10.1109/ICEEOT.2016.7754822

[23] M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (Nov 1997), 2673–2681. https://doi.org/10.1109/78.650093

[24] J. Su, Z. Wang, Q. Wu, J. Yao, F. Long, and H. Zhang. 2017. A Topic-Triggered Translation Model for Statistical Machine Translation. *Chinese Journal of Electronics* 26, 1 (2017), 65–72. https://doi.org/10.1049/cje.2016.10.007

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural

Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks

[26] G. R. Tahir, S. Asghar, and N. Masood. 2010. Knowledge Based Machine Translation. In *2010 International Conference on Information and Emerging Technologies*. 1–5. https://doi.org/10.1109/ICIET.2010.5625695

[27] F. Wong, M. Dong, and D. Hu. 2006. Machine translation using constraint-based synchronous grammar. *Tsinghua Science and Technology* 11, 4 (Aug 2006), 295–306. https://doi.org/10.1016/S1007-0214(06)70193-6

[28] Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial Neural Machine Translation. *CoRR* abs/1704.06933 (2017). http://arxiv.org/abs/1704.06933

[29] W. Wu and L. Li. 2013. Automated Chinese-English translation scoring based on answer knowledge base. In *2013 IEEE 12th International Conference on Cognitive Informatics and Cognitive Computing*. 341–346. https://doi.org/10.1109/ICCI-CC.2013.6622264

[30] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144 (2016). http://arxiv.org/abs/1609.08144