Efficient Neural Machine Translation

Anshul Tomar 170070007
Anwesh Mohanty 170070009
Parth Shettiwar 170070021

December 2020

Contents

1 Introduction

Machine Translation (MT) is the task of automatically converting of written text from one natural language to another, while also preserving the meaning of the input text. The field of MT has experienced a paradigm shift in recent times; the use of count-based models (statistical MT) which were prevalent in the past have been replaced by neural machine translation (NMT) which tackle the task by deploying a neural network based model. NMT models have shown a great level of success as well as a certain level of adaptability in the field of MT and currently give state-of-the-art performances in almost all MT tasks.

However, since the field of machine learning, especially deep learning, is still a growing field, exact solutions to problem statements do not exist yet. Almost all of the models in existence face some kind of issues and it is the same with NMT models. The most prominent problem faced by most large scale NMT models is the issue of overfitting (especially in the RNN component of the models) on the training dataset i.e. though the model might give very good results on the training dataset, the performance on the test dataset is below par. This happens primarily due to the high complexity of such models as well as the discrete nature of the training inputs. This restricts the model performance during inference stages and if the models are ported to real life scenarios, we are bound to get poorer performance than expected.

Another issue, though minor, is that in NMT models to achieve a good training accuracy on the given dataset, most models take a huge amount of time to train and achieve a respectable accuracy(in the range of 3-5 days). Few models also have a high latency which is undesirable in certain real-life situations.

Hence, for our project we have decided to try to find a solution to the aforementioned issues faced by NMT models. Our goal is to build a NMT model that is efficient (doesn't take too long to train, we've tried to keep the maximum training time down to a few hours), has a small memory footprint (doesn't occupy too much space in terms of weight) and gives a good enough accuracy such that the model can be used in place of slower but more accurate models in everyday use.

Our contributions in this project are-

- We modify a state of the art model on the WMT14 French to English dataset to give decent accuracy on Multi30k German to English dataset. We also reduce the overall parameter count and introduce few components to enhance the model performance.
- We introduce an adversarial training algorithm in the decoder of our model which helps in alleviating some of the overfitting inside the model which in turn gives better test results.
- Presented a detailed characterization of model performance and how each component affects it.
- Conducted a thorough comparative study with a SOTA transformer to show how effectively we have achieved our goals.

2 Related Works

We divide this section into 2 parts: 1)Past methods to reduce the number of parameters and time taken for doing translation.2)Past approaches to avoid overfitting on the data

2.1 Reducing Parameters Approaches

In the past, only few have worked upon reducing the total parameters used and at the same time maximising the accuracy. For example, [1] proposes to use Deep Transformer of 256 Million parameters using 60 encoder layers and 12 decoder layers to achieve the state of art Bleu score on WMT14 English German dataset. A recent paper [2] has particularly tried to solve this problem by using shallow transformers where suitable transformations are made and allocation of parameters is done more efficiently to learn wider representations. However it still uses about 38 Million parameters for WMT14 dataset, as it is a transformer based model.
Techniques like compression [3], pruning[4, 5] and distillation[6] have been explored to reduce the size of language models.

Another line of models include the ones which focus on decreasing the time required for translation by using 8 and 16-bit quantization while decoding[8, 9].

2.2 Avoid Overfitting Approaches

There are mainly 2 types of ways to solve the problem of overfitting:

1)Direct Diversity Regularization: In these approaches, diversity enforcing penalty functions are added[10, 11, 12]. For example, they approach it by injecting annealed noise in softmax during each iteration. However, in language modeling, one disadvantage of the direct diversity regularization approach is that the vocabulary size V can be huge, and calculating the summation term exactly at each step is not feasible, while approximation with mini-batch samples may make it ineffective.

2)Adverserial Training: In these approaches, effort is made to either attack existing models by constructing adversarial examples, or train robust models to defend adversarial attacks[13, 14]. In some approaches, they have used this to regularise the network by perturbing activations of intermediate layers. We have also used adverserial training as a regulariser in our setup using the method written in [15] to avoid overfitting. We will describe our approach in more detail in sec 3.2

3 Analysis Pipeline

3.1 Model Architecture

We adopt the novel RNNSearch model presented in [7] and apply the appropriate changes to it according to our needs. The basic architecture consists of an encoder and decoder component connected via an attention mechanism (or an attention layer in-between). The attention mechanism helps the decoder search through the source sentence during decoding a translation. The overall flow diagram of the architecture can be seen in Fig.1 below.

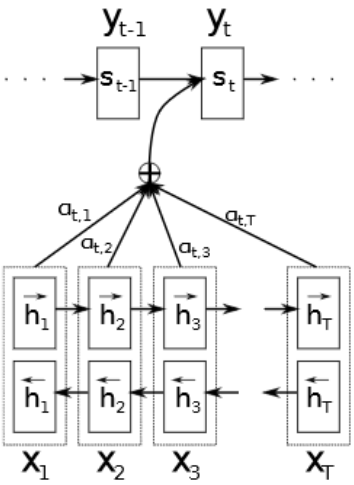

Figure 1: RNNSearch Architecture: Encoder + Attention + Decoder

The original paper suggests the use of Bi-directional RNN as the encoder such that the forward RNN $\overrightarrow{f}$ reads the input sequence as it is ordered (i.e. from X_1 to X_T) and calculates the sequence of forward hidden states $(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_T})$. Conversely, the backward RNN $\overleftarrow{f}$ reads the sequence in the opposite order and calculates the reverse hidden states $(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_T})$. While feeding the hidden states to the attention model we concatenate the hidden sequences in both the directions i.e. $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. This concatenation ensures that the final hidden state h_j contains the input text words in both directions as well as makes sure that the decoder gets a single channel input since it is not bi-directional. To give slightly faster results we have replaced the BiLSTM units by BiGRU units.

The attention layer is a simple feedforward network. It takes in the input as the previous hidden state of the decoder (s_{t-1}) as well as the stacked forward and backward hidden states from the

encoder($H = (h_1, ..., h_T)$). The final output is an attention vector, a_t, which has the same length as the source sentence and all the values lie between 0 and 1 as well as the sum of all values in the vector sums to 1. We pass the input through a linear layer followed by a tanh activation function. The output from the activation function is passed through another layer which decides how much importance each word should get and then finally we apply a softmax activation to get the final attention vector.

$$E_t = tanh(linear(s_{t-1}, H))$$

$$u_t = vE_t$$

$$a_t = softmax(u_t)$$

The decoder contains the aforementioned attention mechanism followed by a single direction GRU block which takes in the embedded input word ($e(x_t)$) at the current instant, the weighted sum of the encoder hidden states (using a_t as the weights) and the previous decoder output as input. The output of the decoder is passed through a linear layer to make a prediction of the next word in the target sentence.

$$w_t = a_t H$$

$$s_t = GRU(e(x_t), w_t, s_{t-1})$$

$$y_{t+1} = linear(e(x_t), w_t, s_t)$$

By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed-length vector. The decoder decides parts of the source sentence to pay attention to.

3.2 Adversarial Training

```
Epoch: 01 | Time: 3m 9s
        Train Loss: 3.730 | Train PPL:  41.687
         Val. Loss: 3.395 |  Val. PPL:  29.818
Epoch: 02 | Time: 3m 10s
        Train Loss: 2.639 | Train PPL:  13.993
         Val. Loss: 3.135 |  Val. PPL:  22.978
Epoch: 03 | Time: 3m 10s
        Train Loss: 2.215 | Train PPL:   9.164
         Val. Loss: 3.165 |  Val. PPL:  23.680
Epoch: 04 | Time: 3m 9s
        Train Loss: 1.952 | Train PPL:   7.041
         Val. Loss: 3.216 |  Val. PPL:  24.921
Epoch: 05 | Time: 3m 11s
        Train Loss: 1.788 | Train PPL:   5.979
         Val. Loss: 3.342 |  Val. PPL:  28.267
Epoch: 06 | Time: 3m 10s
        Train Loss: 1.681 | Train PPL:   5.370
         Val. Loss: 3.458 |  Val. PPL:  31.740
Epoch: 07 | Time: 3m 9s
        Train Loss: 1.619 | Train PPL:   5.047
         Val. Loss: 3.561 |  Val. PPL:  35.215
Epoch: 08 | Time: 3m 10s
        Train Loss: 1.577 | Train PPL:   4.841
         Val. Loss: 3.545 |  Val. PPL:  34.624
Epoch: 09 | Time: 3m 9s
        Train Loss: 1.550 | Train PPL:   4.712
         Val. Loss: 3.627 |  Val. PPL:  37.591
Epoch: 10 | Time: 3m 10s
        Train Loss: 1.547 | Train PPL:   4.699
         Val. Loss: 3.743 |  Val. PPL:  42.212
```

Figure 2: Overfitting in Baseline Model as the validation perplexity increases with epoch count

The inspiration to add this method to our model was essentially to counter the overfitting observed in Fig.2. The basic idea is to add an adversarial noise to the output embedding vectors in case of language modelling or the output hidden state for machine translation while maximum likelihood training. The formulation presented by the authors yields a simple closed form solution which helps in deriving a simple algorithm in which the parameters of the model and the adversarial noise are optimized alternately.

Algorithm 1 Adversarial MLE Training

Input Training data $\mathcal{D} = \{x^{\ell}_{1:T}\}$, model parameters θ, w

while not converge **do**

Sample a mini-batch $\mathcal{M}$ from the data $\mathcal{D}$.

For each sentence $x^{\ell}_{1:T}$ in the minibatch and $t \leq T$, set the adversarial noise on $p(x^{\ell}_t | x^{\ell}_{1:t-1})$ to be

$$\delta_{j;t,\ell} = \begin{cases} -\epsilon h^{\ell}_t / ||h^{\ell}_t||, & \text{for } j = x^{\ell}_t \\ 0, & \text{for } j \neq x^{\ell}_t, \end{cases}$$

where h^{ℓ}_t is the RNN hidden state related to $x^{\ell}_{1:t-1}$, define in (2).

Update $\{\theta, w\}$ using gradient ascent of log-likelihood (4) on minibatch $\mathcal{M}$,

end while

Figure 3: Adversarial MLE Training algorithm from original paper [15]

The adversarial MLE training can be seen as a technique which increases the "diversity" among the hidden states of the output. Variations in the hidden states are encouraged by adding noisy perturbations which tries to keep the model generalized and prevents over-fitting to some extent. The algorithm also ensures that the hidden states generated are separated from each other by atleast ϵ and there is no conflict while choosing representations in the hidden space which might lead to erroneous predictions.

3.3 Datasets

Originally we had planned to use the WMT14 De to En dataset but due to GPU resource constraints (training on WMT14 requires a lot of GPU resources) we were unable to proceed with our experiments on that dataset. Hence we resorted to using the Multi30K De to En dataset. The relevant features of the Multi30K dataset are as follows:

- Training:

  En: 29000 sentences, 377534 words, 13.0 words/sent

  De: 29000 sentences, 360706 words, 12.4 words/sent

- Validation:

  1014 sentences, 13308 words, 13.1 words/sent

  1014 sentences, 12828 words, 12.7 words/sent

- Test:

  1000 sentences, 12968 words, 13.0 words/sent

  1000 sentences, 12103 words, 12.1 words/sent

Since the Multi30K is a much smaller dataset compared to the WMT14 dataset (which has around 4.5 million sentence pairs in the training dataset), hence it will have a limited vocabulary and might not give optimal results for sentences containing common but rare (in Multi30K) words.

3.4 Implementation Details

The input and output dimensions of the encoder and the decoder are the length of the source and target vocabulary respectively. We use embedding layers of size 128 in both the encoder and decoder (originally we had started with 256 in both but observed that the performance was almost same with 128). The hidden state dimension in both the encoder and decoder GRU is 256. We had initially started with 512 but that was suffering from a lot of overfitting. Reducing it to 256 gave us better results as well as reduced the parameter count and computation time. The dimensions of the NN in the attention layer depend upon the dimensions of the hidden states in the encoder and decoder. We

use the Adam optimizer to train the model with the default learning rate of 0.001. The model is run for 20 epochs.

3.5 Loss Functions and Metrics

We have used perplexity as the loss function in our model and use the model which gives the least validation perplexity. Though perplexity metric is generally used for language modelling tasks, a good machine translation model should also ideally have a low perplexity. For reporting the model performance, we use the popular Bilingual Evaluation Understudy (BLEU) score. We were aiming for a BLEU score of 30+ when we started this experiment as a SOTA transformer gave a BLEU score of around 34.

4 Results, Error Analysis and Ablation Study

4.1 Translation of sentences with epoch

We first analysed how our network was performing over epochs. We took a sentence and found its translation after every epoch as predicted by our model.

Input sentence: *Eine junge Dame macht Yoga am Strand .*
Here are the epoch wise translation by our model:
Epoch 1: a man in a a a .
Epoch 2: a man is a a a a .
Epoch3: a young woman is a a a .
Epoch 4: a young woman is a a the ocean .
Epoch 5: a young lady is a on the beach .
Epoch 6: a young lady is on the beach on the beach .
Epoch 7: a young lady is swinging on the beach .
Epoch 8: a young lady is on the on the beach .
Epoch 9: a young lady is doing yoga on the beach .

Almost all sentences got converged in about 9-10 epochs. In the above example, we see that related words like ocean and man(related to beach and lady) show up in starting epochs, but our network learns the correct translation eventually.

4.2 Comparison with SOTA Transformer

In this section, we compare our model with a transformer trained on same dataset and compare it upon various parameters. The transformer is used from [16]

	Our Model	**Transformer**
Parameters	8.8 Million	25.96 Million
Time per Epoch	28 sec	40 sec
Val. Perplexity(PPL)	27.2	5.773
BLEU Score	31.34	34.25

We observe that Transformer takes much more time and parameters to achieve a BLEU score of 34.25. Whereas our model takes about 3 times less parameters and about 1.4 times less time per epoch to achieve a decent Bleu score of 31.34.
Our model clearly is more favourable and with proper hyperparameter tuning, we can further increase the Bleu score.

4.3 BLEU Score vs Sentence Length

We find out how the model performs on sentences of varying lengths. This will give us an idea where the model lacks and hence we will be able to carry out a more structured error analysis.

Length of Sentence (L)	Number of Sentences in Test Dataset	BLEU Score
L ≤ 5	7	19.64
$6 \leq L \leq 10$	390	33.33
$11 \leq L \leq 15$	432	32.40
$16 \leq L \leq 25$	162	28.32
$26 \leq L$	14	23.54

Table 1: BLEU Score variation with sentence length

It can be clearly seen from the table that since the number of sentences with length less than 5 or more than 26 is much less compared to the other categories, the performance is much lower compared to the other categories. Also as the length of sentence keeps on increasing (from a certain length) there is a drop in BLEU score as evidenced by the low BLEU score for $16 \leq L \leq 25 compared to the previous 2 ranges$.

4.4 Attention Maps

We produced the attention map for our translations, in order to analyse and understand whether the model was learning proper attention between words. Following are the attention maps for various sentences from dataset. The German input sentence is on x-axis and translated English sentence is on y-axis. The tags ¡eos¿ is the end tag.

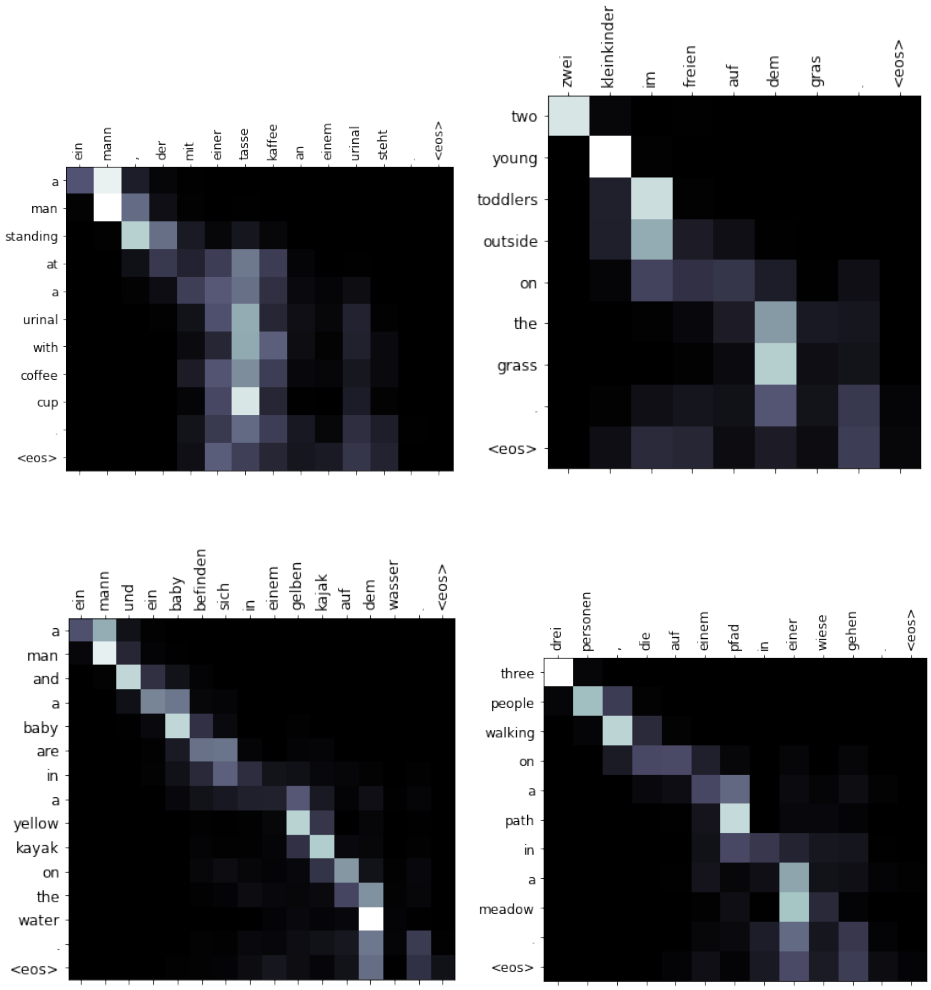

Figure 4: Attention maps generated for various sentences

We clearly observe, that our model pays more attention if there is a direct translation word in target vocab (For example, in first attention map, *mann* in German is directly converted to *man* in English

and hence shown white in attention map. The attention maps are mostly diagonal, implying that English-German languages are quite related and word by word translation is possible with little reordering of words.

4.5 Error Analysis

We did a holistic error analysis our models predicted translations. First we translated all sentences our test and train dataset. We then tried to identify a pattern of which our model gives inaccurate translations by comparing with the groundtruth sentences. We observed 3 main types of error analysis:

4.5.1 Repetition of words

Firstly, many a times our translation gives repeated words as translation. A striking example for this is:

Input Sentence: *Zwei Männer reiten über eine Farm und führen einen von einem Esel gezogenen Wagen .*
Translated Sentence: **Two men are riding a a a a a of a donkey .**
Groundtruth Sentence: Two men ride across a farm and lead a wagon pulled by a donkey.

In this example, we clearly see that "a" has been repeated many times in our output. We speculate that there is some collapse happening which triggers a chain reaction and our model outputs same word many times. Or it might be possible that we didn't run the model for enough epochs.

4.5.2 Erroneous Translations for less frequent words in vocab

Sometimes it happens that even though the word is present in the vocabulary (generated from training data), the translated sentence shows out of vocabulary word (we have used tag <unk>) or some other rare word. This happens specifically when the number of times the word has appeared in training data is very less. For example:

Input Sentence: *Ein Reh springt über einen Zaun .*
Translated Sentence: **A free athlete is jumping over a fence .**
Ground Truth Sentence: A deer jumps a fence .

In this example, the input sentence word "Reh" (which means "deer" in english) has come only 3 times in our training data. For such cases we have observed either erroneous predictions or ¡unk¿ being predicted in the final translated sentence.

4.5.3 Verb forms Interchange

It was also observed that, our model interchangeably gives the continuous form and 3rd Person Singular form of Verb while translating the sentence. For example:

Input Sentence: *Ein kleines Mädchen hält einen kleinen Jungen auf ihrem Schoß .* Translated Sentence: A little girl **holds** a little boy on her lap .
Ground Truth Sentence: A little girl is **holding** a little boy on her lap .

Also observed in previous section example where predicted is "is jumping" but groundtruth is "jumps". We speculate this may be due to the inaccuracies in the training set. Multi30k is a manually translated dataset, for which it might happen that these 2 forms of verb are interchangeably used while preparing dataset.

4.6 Ablation-1: Effect of Temperature on Attention

In this experiment, we change the softmax temperature , which would in turn control the amount of attention and weightage we would be giving to each word while translating

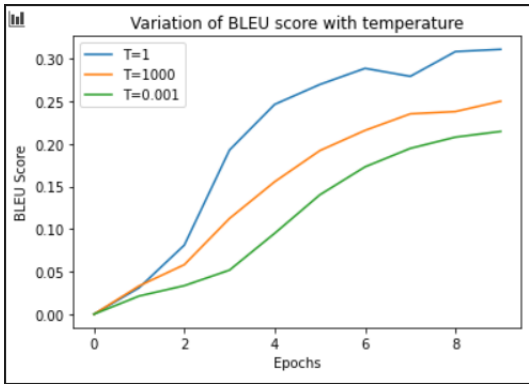

Figure 5: Variation of BLEU score vs Epochs with varying Softmax temperature

- We see that a very high temperature, which means no attention or effectively, equal weightage for all words, achieves a Bleu score of just 25.
- At the same time, a very low temperature, which means very high attention to most probable word, also has a very low Bleu score of about 20.
- Hence a moderate temperature of T = 1, in our case gave us the highest BLEU score and shows the effect of attention.

4.7 Ablation-2: Effect of Attention and Adversarial MLE Training

In this setup, we analyse the effect of Adding attention and adding attention and adversarial training on our base model.

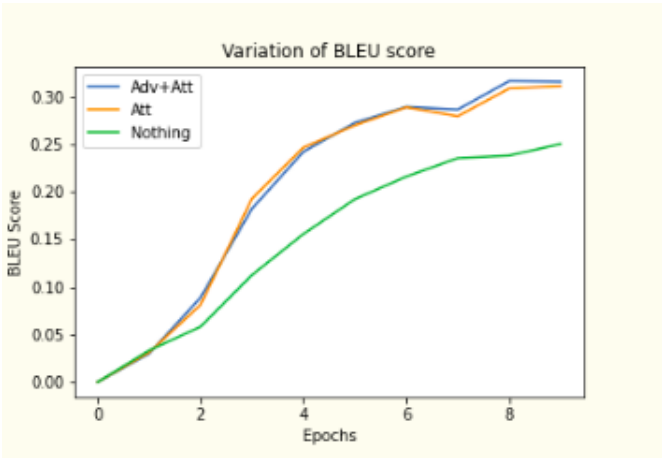

Figure 6: Variation of Bleu score vs Epochs with varying Attention and Adverserial training setups

- We clearly observe the baseline model with no attention and no adversarial training performs poorly and only reaches Bleu score of about 25.
- We further observe, our adversarial training setup helped and has produced a little increase in the Bleu score as compared to the model with only attention.
- Hence this confirms adding Adverserial training and Attention to our model has improved its performance over baseline.

4.8 Ablation-3: Training with random noise and/or Adversarial MLE method

In this we analysed the effect of Noise and Adversarial training on our base model.
There are 4 variations:1)Base,2) Adversarial Training, 3)Noise only, 4) Adversarial Training + Noise.

Noise was added to encoder side hidden state, whereas the noise due to adversarial training was implemented at decoder side.

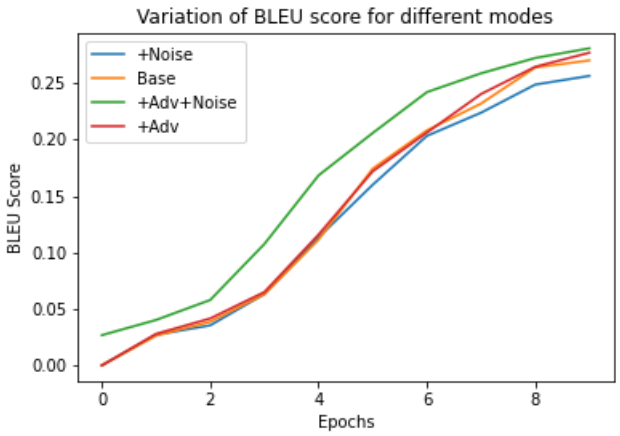

Figure 7: Variation of Bleu score with Epochs with random noise and Adverserial training setups

- We can clearly observe the trend that, Adversarial training + Noise happens to be best followed by only Adversarial, then Base and finally by only Noise. In short adding only Noise won't necessarily reduce the overfitting problem, we need adaptive noise too, like done in adversarial training.

5 Future Work and Conclusion

In this project we built a NMT model based on the popular RNNSearch model and showed how with very less paramters, we can do an Efficient Neural Machine Translation. We further added adverserial training on top of the model to further improve the Bleu score and avoid overfitting of the model on dataset MUlti30k. We showed a comprehensive Results, Error analysis and ABlation study on various setups to analyse our models performance and observe the effect of various components. With further hyperparameter tuning and addition of latest state of art techqniues on top of our model, we can further improve its Bleu score with almost same parameters. Following are the pausible future works which we could do to further analyse and improve our model:

- To train our model and compare results on more bigger and standard language datasets.(like WMT14)
- The epsilon used in adversarial training can be made adaptive to embeddings. We have kept it fixed in our experiments.
- As the above described adversarial training works on RNNSearch it should also work with Transformers. We could apply the same method to improve the BLEU scores of SOTA Transformers.
- Handle unknown or OOV words in a better way. Specifically to handle those sentences where the frequency of words in corpus is less, as was seen in error analysis. Also solve the problem of multiple same words appearing in translations.
- Generate Attention maps and analyse their effect more coherently

References

[1] Xiaodong Liu, Kevin Duh, Liyuan Liu, Jianfeng Gao *Very Deep Transformers for Neural Machine Translation*

[2] Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, Hannaneh Hajishirzi *DeLighT: Very Deep and Light-weight Transformer*

[3] Chen, Patrick and Si, Si and Li, Yang and Chelba, Ciprian and Hsieh, Cho-Jui *GroupReduce: Block-Wise Low-Rank Approximation for Neural Language Model Shrinking*

[4] Song Han, Huizi Mao, and William J Dally. Deep compression *Compressing deep neural networks with pruning, trained quantization and huffman coding*

[5] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. *Analyzing multi-head selfattention: Specialized heads do the heavy lifting, the rest can be pruned*

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean *Distilling the knowledge in a neural network. g*

[7] Dzmitry Bahdanau,KyungHyun Cho Yoshua Bengio *Neural Machine Translation by Jointly Learning to Align and Translate*

[8] Jerry Quinn, Miguel Ballesteros *Pieces of Eight: 8-bit Neural Machine Translation*

[9] Jacob Devlin *Fast and accurate neural machine translation decoding on the cpu*

Pieces of Eight: 8-bit Neural Machine Translation

[10] Binghui Chen , Weihong Deng , Junping Du *Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation*

[11] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. *Large-Margin Softmax Loss for Convolutional Neural Networks*

[12] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Li *Additive Margin Softmax for Face Verification*

[13] Anish Athalye, Nicholas Carlini, and David Wagner *Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples*

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy *Explaining and harnessing adversarial examples*

[15] Dilin Wang, Chengyue Gong, Qiang Liu *Improving Neural Language Modeling via Adversarial Training*

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin *Attention Is All You Need*