Unsupervised multimodal machine translation using visual signals as rewards for reinforcement learning

Gargeya Sharma
School of Science & Engineering (EECS)
Queen Mary University of London
London, United Kingdom
gargeya.sharma@gmail.com

***Abstract*—Unsupervised machine translation, a rapidly evolving field within natural language processing, aims to develop translation models without relying on parallel corpora. This paper explores the novel approach of using visual signals as rewards within a reinforcement learning framework for unsupervised multimodal machine translation. By leveraging visual cues, the proposed method avoids the need for paired source and target language sentences, making it a more scalable and cost-effective solution. The research employs an on-policy policy gradient method in an attempt to optimize the translation policy and utilizes adversarial training techniques to improve the model's performance. Evaluation metrics such as BLEU and METEOR are utilized to assess the quality of the generated translations. The experimental setup includes the integration of visual reasoning scores from a multimodal transformer model, acting as rewards for the reinforcement learning agent. The baseline has been set with the pre-trained Large Language Model before fine-tuning. This work contributes to the diversity of approaches for unsupervised machine translation and provides direction for the research community to further explore this multimodal setup to facilitate cross-lingual communication and understanding.**

***Index Terms*—Unsupervised Learning, Machine Translation (MT), Reinforcement Learning (RL), Natural language Processing (NLP), Visual Signals, Policy Gradient**

I. Introduction

In today's interconnected global landscape, linguistic diversity remains a significant challenge. Language is a cornerstone of human cognition, enabling intricate expression and communication. Despite the digital age's connectivity, linguistic barriers persist, often hindering mutual understanding. Modern machine learning and artificial intelligence have been able to overcome several of these hurdles by developing methods that automatically translate a text written in one language to another, also termed Machine Translation (MT). Traditional MT approaches rely heavily on parallel data for supervised learning, making data collection, and its expert curation a challenging and time-consuming task. Due to such processes, the availability of highly curated data becomes a bottleneck in further widening the scope of machine translation in order to include languages that are not readily available in correspondence to the available data. This limitation among others gives rise to the field of unsupervised machine translation (UMT), a subfield of Natural Language Processing (NLP) that doesn't depend on parallel corpora of bi/multilingual source and target language sentences; in order to learn the inter-syntactic and semantic mapping between them. For several years, researchers have been exploring alternative methods to eradicate the use of target language during the training of MT models. More on this is discussed in the later section. A pivotal approach, which ignited the idea of this paper and heavily influenced the direction of the author's research, is to leverage unsupervised rewards in Machine Translation via reinforcement learning by Julia Ive et.al [1] Their paper demonstrated the potential of the unsupervised reward function using soft actor-critic (SAC) algorithm (a type of reinforcement learning's adversarial training setup) for the cases when they had to choose between possible translations for an ambiguous word (i.e., better exploration of the search space).

Reinforcement learning (RL), a subfield of machine learning, plays a crucial role in this paper. In general, RL focuses on training agents to make sequential decisions in an environment to maximize cumulative rewards. In the context of unsupervised machine translation, RL algorithms, such as Proximal Policy Optimization (PPO), are employed to train translation models. These models, often based on neural network architectures, take a source sentence as input and generate a target sentence as output. By interacting with an environment and receiving either supervised or unsupervised rewards, the translation model learns to improve the quality of its generated translations.

The use of on-policy policy gradient methods, a popular approach in reinforcement learning, is fundamental to optimizing the translation policy. These methods directly optimize the translation policy using gradient-based optimization algorithms. The translation model's performance is enhanced through iterative training processes that aim to maximize the cumulative rewards obtained from the environment. Adversarial training techniques, incorporating a discriminator model to distinguish between generated translations and human translations, can further improve the translation model's performance and fluency.

This paper introduces a novel approach to training a UMT model via RL using unsupervised multimodal rewards. The author is utilizing the power of trained and pretrained transformers to build an RL pipeline where the rewards are calculated using a trained vision-text-based multimodal transformer. Those rewards facilitate the fine-tuning of a pretrained Large Language Model (LLM) to perform unsupervised machine

translation with respect to the training dataset. Training an MT model using PPO itself is a challenging task due to the erratic nature of RL, There is hardly any literature available on machine translation tasks using PPO, so the conclusive approach mentioned in this paper was finalized after countless experiments and training. The finalized unsupervised MT model performs German to English translations on a competitive level to the baseline, while their difference is statically not significant with the p-value of 0.01 on a Permutation test.

The integration of visual signals into the unsupervised machine translation process serves as a means of guiding the translation model during training. Visual signals, obtained from image-caption datasets or image-based language tasks, provide reinforcement signals that indicate the quality and correctness of the generated translations. By leveraging visual signals, the author aims to bypass the dependency on parallel corpora and explore the feasibility of developing scalable, cost-effective and adaptable machine translation systems.

In addition to unsupervised training, the author has also performed supervised training to confirm that the PPO system works just fine and that the model performance is not being manipulated by the training framework. After any training, evaluation metrics play a crucial role in assessing the quality and effectiveness of any model, especially an unsupervised one. Commonly used metrics, such as BLEU (Bilingual Evaluation Understudy) and METEOR (Metric for Evaluation of Translation with Explicit Ordering) are being used in this paper to evaluate the performance of the model. These metrics compare the generated translations against reference translations and measure their similarity, precision, recall, and alignment. These also provide quantitative measures to evaluate the translation model's progress and facilitate comparisons between different approaches.

This paper presents the literature review, methodologies, experimental setup and results of the author's research, highlighting the challenges faced, the approaches that were undertaken, and the potential of unsupervised machine translation using visual signals as rewards for reinforcement learning. The hypothesis is that visual cues can provide a reliable form of 'universal language' that can help bridge the gap between different languages in the absence of parallel text. The author aims to contribute to advancing the understanding and capabilities of this research topic and broaden the field of machine translation to facilitate cross-lingual communication with ease.

II. Literature Review

A. Reinforcement learning in machine translation

Everything changed when OpenAI released their paper introducing ChatGPT, With the latest version (2023) of this paper "Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models" [2], their findings reveal a significant and increasing interest in ChatGPT/GPT-4 research, predominantly centered on direct natural language processing applications. This step boosted the relationship between NLP and RL. Before that in terms of MT, RL has been used to make training and testing better by directly focusing on the main goal as shown by Yu et al., (2017) [3]; Ranzato et al., (2015) [4]; Bahdanau et al., (2017) [5]. But, most of the time, they've used the REINFORCE (Williams, 1992) [6] algorithm and its variants (Ranzato et al., 2015 [4]; Kreutzer et al., 2018 [7]). These methods are basic and can handle the many options in language, but they don't give many rewards.

To make the model steadier, Actor-Critic (AC) models check the reward at every step and use the Critic model to decide what to do next (Konda and Tsitsiklis, 2000) [8]. This method has been tried in MT too as shown by Bahdanau et al., (2017) [5] and He et al., (2017) [9]. But, more complex AC models with Q-Learning aren't used much for creating language because it's tough to guess the Q-function with so many choices. Choshen et al., (2020) [10] show that having too many choices is a big challenge for using RL in making text. It's crucial to start the agent's training close to the real outcome for RL to work well.

Additionally, The use of reinforcement learning in machine translation has been explored by several researchers. "Deep Reinforcement Learning for Dialogue Generation" by Li, J. et al. (2016) [11] proposes a framework that uses reinforcement learning to generate more diverse and interesting responses in dialogue systems. This work demonstrates the potential of RL in improving the quality of generated text in various NLP tasks. Nguyen et.al. (2017) [12] proposed that MT is a natural candidate problem to be solved with reinforcement learning from human feedback, having users provide quick, dirty ratings on candidate translations to guide a system to improve. [12] introduces a reinforcement learning algorithm that improves MT from simulated human feedback using the combination of advantage actor-critic (A2C) with an attention-based neural encoder-decoder system. Their work resulted in better performance on problems with large action space and delayed rewards, effective optimization and increased robustness to skewed, high variance and granular feedback just like how humans behave.

A new distributed policy gradient approach: MAD by Domenic Donato et.al., (2022) [13] not only utilizes the training stability and efficient performance backed by PPO for fine-tuning LLMs for machine translation as a reference but also outperforms it with their approach (MAD). Through the paper "Proximal Policy Optimization Algorithms" by OpenAI, (2017) [14], it can be concluded that PPO offers stable training in reinforcement learning by limiting policy updates, ensuring consistent performance. Its sample efficiency makes it ideal for tasks with limited data. PPO effectively handles sparse rewards, common in NLP, and balances exploration and exploitation. Its flexibility allows adaptability across diverse tasks without extensive hyperparameter tuning.

The author realized that there is a significant gap in research while solving unsupervised machine learning problems with PPO, especially UMT. This gap provided a great opportunity and motivation for the author to move forward with the novel research on unsupervised multimodal machine translation us-

ing PPO. The later section will shed some light on why multimodal learning was opted to tackle this task.

B. Multimodal learning

Multimodal deep learning focuses on developing and training models capable of handling, processing, and connecting data from diverse modalities Baltrušaitis et al., (2017) [15]. Several reviews highlight the advantages of multimodal methods over unimodal ones for tasks like retrieval, matching, and categorization (Atrey, P.K. et.al. 2010 [16], Bhatt, C.A. and Kankanhalli, M.S. (2010) [17]). [15] emphasize the inherently multimodal nature of our world, where experiences span across visual, auditory, tactile, olfactory, and gustatory senses. The primary sources for multimodal approaches include text, images, videos, and sound. Their comprehensive survey underscores the significance of multimodal machine learning, which seeks to interpret these diverse signals cohesively.

A significant stride in this direction is the work by Jiang et al., (2021) [18], which introduces a novel multimodal deep learning architecture, TechDoc. This architecture processes technical documents that often contain both text and images. By synthesizing convolutional neural networks, recurrent neural networks, and graph neural networks, TechDoc achieves superior classification accuracy for technical documents based on the hierarchical International Patent Classification system. Their results underscore the potential of leveraging multimodal information for enhanced document classification.

The application of multimodal learning in healthcare has been particularly promising. A study by Tiulpin et al.,(2019) [19] leveraged multimodal machine learning to predict the progression of knee osteoarthritis. By integrating raw radiographic data, clinical examination results, and patients' medical histories, their model achieved a commendable classification accuracy of 80.33%, outperforming traditional AI-based emotion classification methods.

Even for the financial sector, the latest contribution by Cho et al.(2023) [20], proposes an intelligent document processing framework for the financial sector. This framework combines traditional robotic process automation (RPA) with a pre-trained deep learning model to process real-world financial document images. The proposed solution demonstrates the efficacy of a multimodal approach in understanding financial documents, even with limited training data, and can handle multilingual documents with ease.

In conclusion, multimodal machine learning is proving to be indispensable in tackling complex challenges across various domains, showcasing its versatility and high potential.

C. Unsupervised machine translation

In the field of machine translation, unsupervised learning has emerged as a promising approach to train models without the need for parallel corpora. In the paper "Unsupervised Neural Machine Translation" by Artetxe et al. (2018) [21] the authors proposed an initial unsupervised training step that uses a shared encoder and a shared decoder for both languages which was competitive with supervised methods on low-resource language pairs, highlighting the effectiveness of unsupervised methods in scenarios where paired translation data is scarce, followed by a language modelling step that refines the model using monolingual corpora. This approach has been influential in the field, leading to further research and development in unsupervised machine translation.

Another significant contribution to this field leveraging monolingual data is the work of Lample et al. (2018) [22], who proposed a method for unsupervised machine translation that uses a shared latent space and a cycle consistency loss. Both of their approaches, which use back-translation and denoising autoencoders, show competitive results with supervised methods on several language pairs. Their work has been widely cited and has inspired subsequent research in the area.

A few major inspirations for this research and methodology are derived from these couple of papers: The paper "SURF: Semantic-level Unsupervised Reward Function for Machine Translation" by Julia Ive et.al.(2022) [23] presents a novel approach that leverages semantic-level rewards for reinforcement learning in machine translation. The authors propose a reward function that is based on semantic similarity between the source and target sentences, which is computed using pre-trained language models. This approach allows the model to focus on preserving the meaning of the sentence during translation, rather than just matching the exact words or phrases. As mentioned in the Introduction of the paper [1] "Exploring supervised and Unsupervised rewards for machine translation" (2021), the authors investigate the use of both supervised and unsupervised rewards in reinforcement learning for machine translation. They found that combining both types of rewards can lead to better performance than using either type alone.

D. Multimodal Transformers

The field of machine translation has also seen the integration of multi-modal learning, where models leverage information from multiple modalities, such as text and images, to improve performance. The paper "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks" (2019) [24] presents a model that learns joint representations of image and text data. The authors demonstrate that this approach can significantly improve performance on several vision-and-language tasks.

Along the same lines, the novelty of this paper arises by utilizing visual reasoning as the unsupervised multimodal reward criteria for UMT. To design the reward, the author is using The Vision-and-Language Transformer (ViLT) model, introduced in the paper "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision" (2021) [25]. The ViLT model leverages visual cues along with the text to provide a score that helps bridge the gap between different languages in the absence of parallel text. The authors demonstrate that this approach can achieve competitive performance on several benchmarks, while being simpler and more efficient than predecessor methods (ViLBERT [24] (2019),

VisualBERT (2019) [26] etc.), thereby addressing another key challenge in unsupervised MT.

The majority of multimodal transformer applications revolve around solving NLP tasks using images or vice versa, another example is "AI Ekphrasis: Multi-Modal Learning with Foundation Models for Fine-Grained Poetry Retrieval" (2022) [27] paper which discussed a deep learning approach for the automatic retrieval of poetry suitable to the input images. The approach takes advantage of strong pre-training of the CLIP model (OpenAI, 2021) [28] and overcomes its limitations by introducing shared attention parameters to better model the fine-grained relationship between both modalities. Similarly, but without transformers, The paper "Learning Cross-modal Embeddings for Cooking Recipes and Food Images" (2017) [29] presents a method for learning joint embeddings of recipes and food images. This work demonstrates the potential of cross-modal learning in tasks that involve multiple types of data.

All these research evidences clearly points towards a high potential direction of utilizing multimodal data and learning to enhance the rate of success with highly complex tasks, and unsupervised machine translation is clearly one of them.

The author began the research with a powerful multitasking LLM: T5 (Text-to-Text Transfer Transformer) (2020) [30], a state-of-the-art model for NLP tasks, that is pre-trained on a large corpus of text and then fine-tuned for specific tasks. Its ability to handle different NLP tasks by converting them into a text generation problem makes it a versatile tool for this research. The model's impressive performance on various benchmarks further attests to its efficacy. During the experimentation, mentioned in the section below, T5 gets replaced by another LLM: Opus (2020) [31], a model that is trained on a collection of translated texts from the web, providing a rich resource for training and testing translation models. Its extensive multilingual and multi-domain dataset capabilities allow us to train our model on a diverse range of texts, thereby enhancing the model's robustness and generalization capabilities.

III. Methodology

A. Neural Machine Translation (NMT)

Neural Machine Translation (NMT) commonly employs a sequence-to-sequence (seq2seq) structure (Sutskever et al., (2014) [32]; [5]). In this setup, a source sentence $x = (x_1, x_2, \ldots, x_n)$ is transformed by the encoder into a set of hidden states. During each decoding phase t, a target word y_t is produced based on $p(y_t|y_{<t}, x)$, which is conditioned on the input sequence x and the decoded sequence $y_{<t} = (y_1, \ldots, y_{t-1})$ up to the t-th step. Given a corpus of source and target sentence pairs $\{x_i, y_i\}_{i=1}^{N}$, the training goal, or maximum likelihood estimation (MLE), is expressed as (equation 1):

$$L_{MLE} = -\sum_{i=1}^{N}\sum_{t=1}^{T} p(y_t^i|y_1^i, \ldots, y_{t-1}^i, x_i)$$

(1)

B. Reinforcement Learning for NMT

In the Reinforcement Learning (RL) context, NMT is viewed as a sequential decision-making task. Here, the state is represented by previously generated words $y_{<t}$ and the action is the upcoming word to be produced. Given state s_t, an agent selects an action a_t (equivalent to y_t in seq2seq) based on a policy π_θ and receives a reward r_t for that action, which can be determined using metrics like BLEU.

The RL training's objective is to optimize the expected reward (equation 2):

$$L_{RL} = E_{a_1,\ldots,a_T\sim\pi_\theta(a_1,\ldots,a_T)}[r(a_1, \ldots, a_T)]$$

(2)

Under policy π, the values of the state-action pair $Q(s_t, y_t)$ and the state $V(s_t)$ can be defined as (equation 3):

$$Q_\pi(s_t, a_t) = E[r_t|s = s_t, a = a_t]$$

$$V_\pi(s_t) = E_{a\sim\pi(s)}[Q_\pi(s_t, a = a_t)]$$

(3)

Conceptually, the value function V gauges the potential of a model in a specific state s_t. The Q function evaluates the worth of selecting a particular action in that state.

From these definitions, an advantage function, represented as A_π, which relates V and Q, can be defined (equation 4):

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$$

(4)

The goal then becomes to optimize one of these objectives (equation 5):

$$\max_a A_\pi(s_t, a_t) \rightarrow \max_a Q_\pi(s_t, a_t)$$

(5)

Different RL methods employ varied strategies to find the best policy. Techniques like REINFORCE and its variant MIXER [4], which are prevalent in linguistic tasks, seek the best policy using Eq. 2 through the Policy Gradient. Actor-Critic (AC) models typically enhance the Policy Gradient models' performance by addressing Eq. 5's left side [5]. Q-learning models focus on maximizing the Q function (Eq 5, right side) to surpass both the Policy Gradient and AC models (Dai et al., 2018) [33].

Proximal Policy Optimization (PPO) is another advanced reinforcement learning algorithm that comes under the category of on-policy policy gradient methods and has gained significant popularity due to its stability and efficiency [14], this research takes advantage of that. Unlike traditional policy gradient methods that can have large policy updates, potentially leading to suboptimal policies, PPO aims to take the largest step possible while ensuring the new policy doesn't stray too far from the old policy. This is achieved by adding a constraint to the policy update. There are two primary variants of PPO: PPO-Penalty and PPO-Clip. In this paper, the words PPO-Clip and PPO are used interchangeably because that is the primary variant used by both OpenAI and the author.

The objective function for PPO is given by (equation 6):

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

(6)

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio of the new policy to the old policy, and $\hat{A}_t$ is an estimator of the advantage function at time t. The clip function ensures that the ratio $r_t(\theta)$ remains within a specified range, defined by ϵ, thus preventing overly large policy updates.

In essence, PPO strikes a balance between making significant policy updates to improve performance and ensuring these updates don't destabilize the learning process. This balance has made PPO a preferred choice for many reinforcement learning tasks, including some NMT challenges.

C. Working of the Models used

We shall briefly discuss the working of the transformer models that are used in this research to shed some light on how the system comes together to achieve the task.

LLM: The input sentence in the source language is tokenized and passed through an **embedding layer**, converting each token into a high-dimensional vector. This embedding captures semantic information about the words.

$$e_i = E(x_i)$$

where e_i is the embedded vector for token x_i and E is the embedding function. The embedded vectors are processed by the **encoder**, in our case, a Transformer architecture. The encoder captures the sequential dependencies of the source language.

$$h_i = f_{\text{encoder}}(e_i, h_{i-1})$$

where h_i is the hidden state at step i and f_{encoder} is the encoder function. After encoding the signal, the fundamental mechanism that governs the superiority of transformers: **Attention**; allows the model to focus on different parts of the source sentence when producing the translation. For each target word, a context vector c_t is computed as a weighted sum of all source hidden states.

$$c_t = \sum_i \alpha_{t,i} h_i$$

where $\alpha_{t,i}$ are the attention weights, calculated using a compatibility function between the source hidden states and the current target state. After the attention layer, we have the **decoder**, also typically a Transformer structure, that generates the target sentence. It takes the last hidden state of the encoder (or the context vector in attention models) and produces the target tokens sequentially.

$$s_t = f_{\text{decoder}}(y_{t-1}, s_{t-1}, c_t)$$

$$y_t = g(s_t)$$

where s_t is the decoder's hidden state at step t, y_t is the output token, y_{t-1} is the previously generated token, and g is a function producing the token from the state. Last but not the least, we have an **output** layer at the end of the architecture. It is a softmax over the vocabulary of the target language, producing a probability distribution for the next word.

$$P(y_t|y_{<t}, x) = \text{softmax}(W_o s_t + b_o)$$

where W_o and b_o are the weight matrix and bias for the output layer, respectively. The model that is described above is trained using a suitable loss function, often the cross-entropy loss between the predicted probability distribution and the actual target sequence. The gradients are then back-propagated through the model to adjust the weights.

In essence, an LLM for machine translation encodes the source sentence into a dense representation, optionally applies attention to weigh source tokens, and decodes this representation into the target language, all while optimizing for translation accuracy through iterative training.

ViLT: A multimodal transformer model [25] which incorporates NLP tasks in alliance with visual data such as Images into a unified framework. It is designed to handle a variety of tasks related to visual and linguistic modalities such as Classification tasks (visual question answering, natural language for visual reasoning) and Retrieval tasks (image-to-text and text-to-image retrieval). The major reason for involving a multimodal model such as VilT is its unified embedding space. ViLT processes both images and text to represent them in a shared embedding space. This means it transforms both visual and textual data into a format where they can be compared, related, or jointly processed; which further allows a transformer architecture to capture intricate relationships and dependencies between visual and textual elements.

D. Reward Formulation

The heart of any reinforcement learning system is its rewards. In this paper, we utilize two different reward functions for two models (unsupervised and supervised). For the main aim of this paper (unsupervised model), Images and texts are used in unison to generate rewards via ViLT transformer with a classifier head on top for image-to-text and text-to-image retrieval. This particular functionality produces classification logit scores that validate how well the text and image align with each other. The range of this score lies between (-inf, inf) where the higher the score, the better the alignment of the text with the image and vice versa, the lower the score, the poorer the reasoning. Hugging face, an open-source library mentioned in the later section, implemented ViLT and its functionalities including the one utilized in this paper and made it publicly available through their 'transformers' python package. There are 5 steps to retrieve this ViLT score using hugging face implementation:

1. Make a processor object by loading the ViltProcessor class with the same pretrained configuration as used by the model in step 2.

2. Make a model object by loading the ViltforImageAndTextRetrieval class with a certain pretrained configuration ("dandelin/vilt-b32-finetuned-coco" in our case).

3. Pass the input image and text into the processor with the required output configuration such as max_length, truncation, return type of the output etc.

4. Get the output from step 3 and pass it as input to the model

5. The output from the model comprises various elements depending on the model's configuration and inputs, the scores are the 'logits' values included in the output.

The raw score is one of the ways to set rewards for the PPO, but section 4 (Experimental setup), will determine the paper's best practice along with other experimental approaches.

Supervised Reward: To prove the proper functioning of the PPO pipeline, another fine-tuning was performed on the pre-trained model as a separate system. This time, the reward function utilized the target language text descriptions to calculate a supervised reward. This paper utilizes one of the best practices of the community and sets a cumulative BLEU score between the predicted text and the target text as the reward. For more clarity think of it this way, there are multiple evaluation pairs for each target and predicted text where the target text remains the same in every evaluation, but the predicted text is dissected into a cumulative combination of words ([first_word, first_word second_word, first_word, second_word, third_word ..., entire predicted text]). The BLEU score generated from each pair is summed up to get an aggregate BLEU score for the specific pair of sentences, which in this paper acts as the supervised reward.

E. Implementation Details

Software and Tools:

- **Jupyter Notebook**: Jupyter Notebook was used as the primary development environment, facilitating interactive development and real-time data visualization.
- **Python**: The entire implementation was carried out using Python 3.9, ensuring compatibility and leveraging the latest language features for efficient coding.

Libraries:

- **PyTorch**: The author's primary deep learning framework, PyTorch provided the necessary tools to design, train, and evaluate neural network models with ease.
- **Transformers**: From Hugging Face, the 'transformers' library was instrumental in accessing pre-trained models and tokenization utilities, streamlining the integration of state-of-the-art NLP models into our pipeline.
- **TRL**: The TRL (Transformer Reinforcement Learning) library was employed to integrate reinforcement learning techniques with transformer architectures, enabling advanced training strategies.
- **NLTK (Natural Language Toolkit)**: NLTK was utilized to create a custom function that calculates the METEOR score, a type of evaluation metric for NLP tasks, on the entire test dataset.
- **SacreBLEU**: For evaluating the quality of machine translations, SacreBLEU library was used, which provides a standardized version of the BLEU metric, ensuring consistent and reliable evaluation.
- **WandB (Weights & Biases)**: WandB was used for experiment tracking, hyperparameter optimization, and model visualization. It allowed for monitoring the training in real-time, comparing different model versions, and ensuring reproducibility.

Hardware: The Jupyter notebook was accessed through Queen Mary University of London (QMUL)'s resources available for master's students. Here are some of the system configurations, **CPU:** Intel(R) Xeon(R) Gold 5222 @ 3.80GHz, **RAM:** 19GB and the model was trained on a 24 GB NVIDIA Quadro RTX 6000 **GPU** for an average of 6 hours. Training time varied with different experimental choices for the batch size.

IV. Experimental Setup

A. Dataset

A multimodal system requires a multimodal dataset. The author used the Multi30K dataset (Elliott et.al., 2016) [34] that consists of images along with their multilingual descriptions in English, German, French and Czech language. This research was performed with images and their corresponding English and German descriptions. The textual dataset was imported using the official GitHub repository for the Multi30K dataset: 'multi30k/dataset' while the raw images had to be requested from the University of Illinois. Here is the link to request the Flickr30K & Denotation Graph data. The dataset contains 29,000 instances for training, 1,010 for validation/development, and 1,000 for testing. This paper use **flickr2016 (2016)**, **flickr2017 (2017)**, **flickr2018 (2018)** and **coco2017 (COCO)** test sets for model evaluation.

2016 is the most **in-domain** test set as it is taken from the same pool of descriptions used for training and validation, whereas **2017**, **2018** and **COCO** are from a different pool of dataset and thus considered as **out-of-domain**.

Preprocessing: The name of the image, German descriptions and English descriptions are available in separate text files. The author made a function that takes 3 arguments: input language, output language and split. The split decides which dataset to use to perform the action. Inside this function, 3 files are simultaneously loaded so that image name ("image"), input language descriptions ("from") and target language descriptions ("to") can be easily mapped to each other in the form of a list of dictionaries where each dictionary is an instance of the dataset. This list of dictionaries is further updated with the help of a mapping function, which adds "input_ids" and "query" key-value pairs in each dictionary. "input_ids" are tokenized input language descriptions used as the LLM's inputs and "query" is detokenized "input_ids" value which validates the input description passed to the model. We are using the 'PPOTrainer' class from the 'trl' library to instantiate the RL model, which requires its datasets to be of type 'torch.utils.data.Dataset'. To fulfil this condition, the last preprocessing step for the dataset is to get transformed from a list of dictionaries to 'torch.utils.data.Dataset'.

TABLE I
BLEU SCORE OF PRETRAINED LLMS ON MULTI30K

LLM Model	BLEU Score
T5 (en-de)	34.368
Opus (de-en)	34.687

B. Model Selection

The experimental setup for unsupervised machine translation using visual signals as rewards for reinforcement learning encompassed several key steps. To begin, the author explored the use of traditional language models like Text-to-Text Transfer Transformer (T5), Bidirectional Auto-Regressive Transformers (BART), and Generative Pre-trained Transformers (GPT)-2, which are commonly employed in machine translation pipelines. The objective was to assess their performance in machine translation tasks and determine their applicability to the research at hand.

The experimentation phase commenced by utilizing the T5 model on the Multi30K dataset to evaluate its performance for machine translation. As shown in Table 1., The researcher calculated the Bilingual Evaluation Understudy (BLEU) score, a widely adopted metric for assessing the quality of machine translation outputs. The T5 model demonstrated satisfactory results, providing a solid foundation for further investigation.

In order to focus on unsupervised machine translation using reinforcement learning, the researcher proceeded to construct a pipeline that integrated the T5 model with the ViLT (Vision-and-Language Transformer) model for reward generation. This integration facilitated the evaluation of predictions and visual reasoning outputs together, enabling the formulation of the unsupervised reward function mentioned in section 3.D. Leveraging the visual reasoning output from ViLT as input allowed the reinforcement learning process to have a solid base for guidance. However, the training process encountered a significant challenge. The T5 model, primarily designed for English (en)-to-other-language translation, produced outputs in German (de) that do not align with the input language (English) for visual reasoning by ViLT. This misalignment hindered the successful merging of these two components in an unsupervised manner, leading to issues during training.

To overcome this challenge, the author made a decision to switch from the T5 to the Opus model. Unlike the T5 model, Opus supported machine translation from multiple languages, including German (de) to English (en). In terms of model evaluation, both T5 and Opus got a similar BLEU score, which suggests that model performance will not suffer and degrade due to changing the LLM (Table 1). At the same time, this switch facilitated a more suitable alignment between the Opus model's English outputs and the visual reasoning's English inputs, ensuring the successful integration of visual signals and language translation to be able to perform unsupervised machine translation.

C. Working of the System & Training

In section 4. B, Opus was finalized as the LLM model to perform this research. In this section, the entire working pipeline of the system as well as model training is explained sequentially with the help of Fig.1. to provide a detailed understanding of what is happening and how. The author developed the figure in such a way that makes the implementation self-explanatory in the form of a visual algorithm.

In Fig. 1. each arrow holds the sequence number and direction of the data from one component to another. Here is what they mean:

1) The model store is the open source model repository made available by hugging face through their library transformers and trl. The necessary models (Opus and PPOTrainer) and their tokenizers are imported from this repository to be further used in the code. A copy of the pretrained opus model is being made to act as a reference model, which during training will be used to calculate the Kullback-Leibler (KL) divergence between the in-training model and the pretrained model.
2) Unzip the raw images and make them ready to be loaded by setting their directory path in the preprocessing function to be further appended with the image name to access the image data as mentioned in section 4. A.
3) Pull the GitHub repository for Multi30K textual data and pass image names, English descriptions and German descriptions text files into the data processor function to prepare any split of the dataset to be used by PPOTrainer as mentioned in Section 4. A.
4) Wrap the preprocessed dataset along with the Opus model, Opus reference model and Opus tokenizer around the PPOTrainer class.
5) Before instantiating the PPOTrainer object, we also need to pass the model configuration data which carries the hyperparameters of the model (model name, steps, batch size, learning rate, logging with wandb). Once steps 4 and 5 are performed, we get a PPOTrainer object ready to be trained as well as monitored in rum-time on the Weights & Biases portal.
6) Time to create reward function for the unsupervised approach, see section 3.D. Once the pretrained ViltProcessor and ViLT's visual reasoning model are loaded, the reward function is ready to receive the image name and predicted description as the inputs to output the relevance score for the entire batch of data during training.
7) There is another set of parameters that governs the output type from the LLM, these parameters control the minimum length of the input in the Opus model, performing sampling of tokens during training or not, tokens to pad the input if mentioned in the tokenizer settings, the maximum number of tokens allowed to be generated to produce the output translation and end of statement token to add if necessary by the tokenizer.
8) The red arrows signify the continuous flow of data

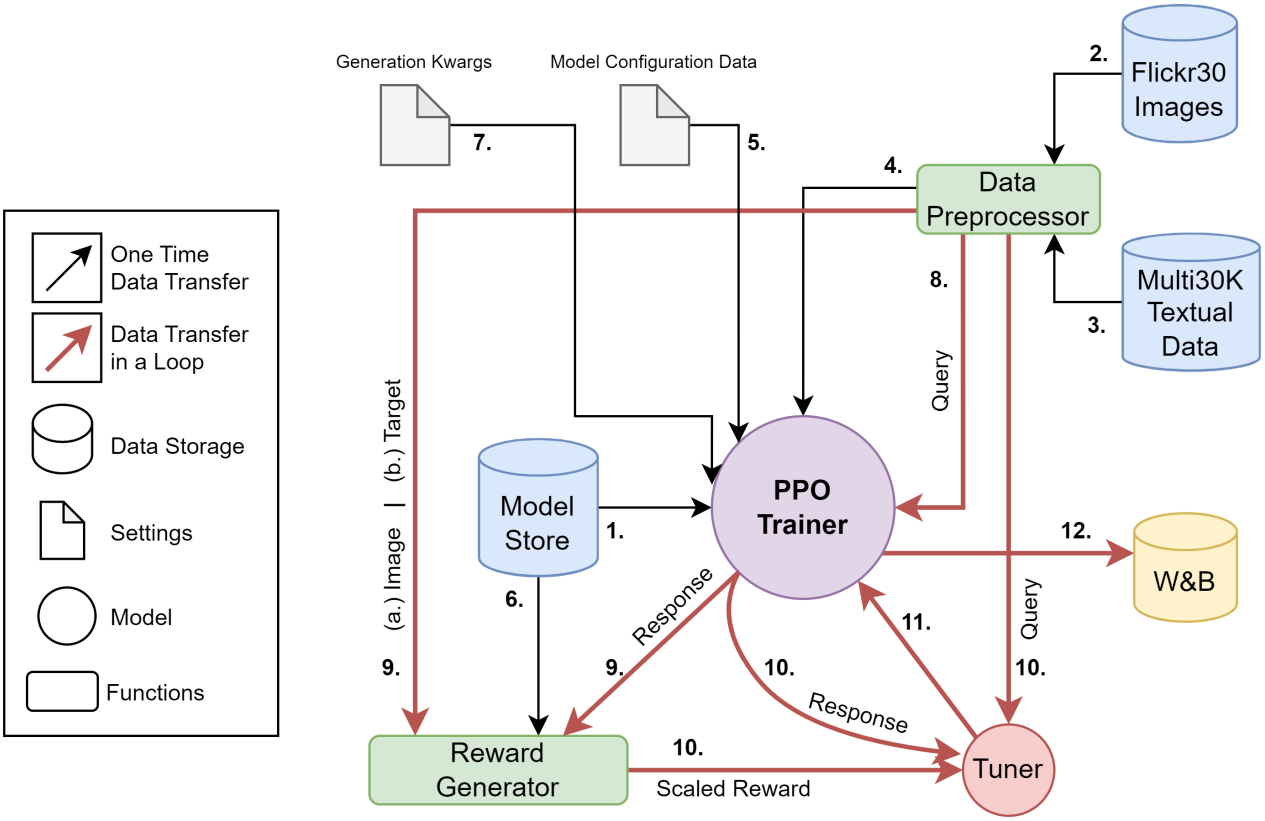

Fig. 1. End-to-end working of the system discussed in this paper. The figure also works as a visual algorithm, where each component's action is marked with a sequence number that explains the functionality of the code on a surface level. The working is as follows: 1. Loading the pretrained opus model and tokenizer into a PPOTrainer object, 2,3. Preprocess the Multi30K images and text data together, 4. Load the data into the PPOTrainer object, 5. Load the hyperparameters for the PPO Training, 6. Load the ViLT reward and tokenizer to build a reward function, 7. Set the PPO generation kwargs before the training, 8. A batch of queries is passed to the model, 9.a. Image, 9. b. The target text is passed into the reward function along with the prediction text generated by the model, 10. A batch of Rewards is calculated and passed to the model along with the original batch of queries and response tokens, 11. The model performs optimization and fine-tunes itself as a single step, 12. The logs are this step are saved on W&B for live monitoring and storage.

from one element to another, which implies that the model is undergoing training and has started fine-tuning itself, whereas the black arrows signify single-time data transfer. The model begins training from this step onwards. The dataset registered with the PPOTrainer object gives away batches of datasets with the size mentioned in the configuration file. The model object takes the batch of queries (tokenized descriptions) along with the parameters mentioned in step 7 to generate a batch of response tensors. These are the model outputs in the form of tokens.

9) This particular step handles the heart of this training procedure. The response tokens are detokenized using the opus tokenizer with certain argument settings such as skipping the special tokens or not, cleaning up tokenization space or not. Additionally, the ¡pad¿ token in front of each response sequence is manually removed to get a clear English-translated sentence. This detokenized batch of descriptions is passed into the reward function along with their corresponding (a.) batch of image names (unsupervised approach) or (b.) batch of target language descriptions (supervised approach) to calculate a reward for each instance in the batch.
10) The batch of rewards received from the reward function in the previous step is passed to the step function along with a batch of queries and response tensors to fine-tune the model with respect to the learning rate set in Step 5. In Fig 1. Tuner is the same model we are training, for better visualization of data flow and training loop, it has been portrayed as a separate component.
11) Once the tuner receives the inputs required by PPO to tune the model in step 10, it changes the model parameters to better perform the task in an unsupervised (if 9. a is used) or supervised (if 9. b is used) manner to increase the reward that it receives with its predictions.
12) Once the fine-tuning step is performed on the model, the

model run-time behaviour, rewards it receives, entropy of the model sampling, KL divergence of the model along with many other details are recorded in instances created on weights & biases tied specifically to this particular training task.

The cycle from step 8 to step 12 repeats itself until all the batches are exhausted, The model has seen the entire training dataset while performing the number of steps mentioned in the model's configuration (step 5) and the training has run for the specified number of epochs. In this research, the number of epochs is set to be 1 in all experimental cases.

D. Experiments

The author put a lot of effort and time into getting this system up and running with significant results to share with the community. This section contains all the wrong turns that were made during the research and how certain settings resulted in bizarre and unethical results that should be avoided while fine-tuning with language models. In the author's opinion, this section provides a lot of guidance to the community to avoid the same mistakes as the author's and save themselves plenty of time and failures.

System setup:

In the initial stages of model training, several impediments hindered the learning process. The first mistake that prevented the model from learning anything during the training was by using two different opus models inside the training loop. The first model was the original model that needed to be trained and was used to generate response tokens, but due to a lack of experience with reinforcement learning, the author was also using hugging face's pipeline functionality to load another pretrained opus model just to retrieve proper English translations to pass to the reward function along with the images. This mistake prevented the model in training from inputting its own predictions to get the rewards. The author was unaware of using the decoder inside the training loop to get detokenized English translations from the response tokens to calculate the rewards. Later, correct changes were amended so that the model could keep a check on its performance based on the reward that it was getting on its own.

Secondly, it took some time and trials to figure out the maximum length of the tokenized queries and also if padding those queries will be beneficial for achieving the task or not. It turns out that having 'max_length' more than the average size of the descriptions with padding='max_length' introduced an unnecessary generation of words during the prediction, some did not make sense and some just repeated themselves. After multiple experiments with padding and truncation, the author realized that padding negatively affected the predictions with gibberish and repeated words. Hence, at last, padding is set to False (default), truncation is set to True as it is necessary to allow flexibility for the query size, and max length is set to 25.

Thirdly, it is crucial to avoid having any special or padding tokens before detokenizing the response tensors into a string. Missing this step will introduce noisy literals into the prediction string, which upon calculating the rewards will hinder their authenticity. To avoid this mistake and clean the predicted translations, the author set arguments skip_special_tokens=True and clean_up_tokenization_space=True inside the tokenizer.decode function. Other settings to take care of are generation kwargs which were set up and used in step 7. (see Fig. 1.). The author first had the minimum length parameter set to -1 and maximum new tokens to 99, which caused several abrupt failures in training because of 2 major errors that seem to be caused by them: Error 1: RuntimeError: The size of tensor a (177) must match the size of tensor b (40) at non-singleton dimension 1 and Error 2: Value error, the autorange detect [nan, nan], something like that on PPO.logs_stat line. The first error was solved by adding an argument (max_length=40) to the ViltProcessor object inside the reward function, which preprocesses the image and text into an encoding that further gets passed to ViLT's visual reasoning. The second error seemed to be the most reluctant of all because there were various reasons which affected the training to get the same error. One of the solutions was to set a minimum length parameter greater than 1, in our case, 2 and to improve the quality of the generation, and avoid unnecessary and repeated tokens in the output, the size of the max new tokens was set to be 13 instead of 99.

Hyperparameters and Rewards:

The biggest leap in improving the quality of the training and stabilizing the performance of both unsupervised and supervised models; is related to setting the appropriate learning rate and batch size in the model configuration settings (Step 5.). Fig 2. shows a performance comparison of how learning rate and batch size affected the training curve in the form of avoiding the explosion of KL divergence beyond a reasonable limit (10).

Among all the combinations between various learning rates and batch sizes, the author believes that for specifically Multi30K dataset, the best values out of all the tried options are 1e-9 and 580 respectively. For more details, check out the saved experimental runs of the model on author's weight & biases profile.

Unsupervised Rewards

The most important and ideal component for experimentation in any kind of RL system is the reward function, and the author did not let go of any chance to do so. The experiments with reward function include training with raw logit scores from the ViLT model as rewards, binning these continuous scores into discrete rewards while trying multiple bin ranges as well as reward values for each bin, and changing the number of bins from 4 to 3 to 2. Eventually, discrete rewards did not work as expected, so they were dropped, and the author moved on to normalized rewards. Scores were turned into normalized rewards by restricting their maximum and minimum value with +10 and -10 respectively and performing min-max scaling on each reward value. Furthermore, normalized rewards with range (0, 1) started causing the same error during training that we encountered before: Value error, the autorange detect

Fig. 2. KL Divergence curve during training with different learning rates and batch sizes. The chart above shows that there is a steady and small increase in KL divergence, which is ideal for the system, whereas the chart below shows an explosion in KL divergence value by crossing the threshold value of 10 and damaging the learning of the model.

[nan, nan]. To avoid this, the rewards after normalization were scaled to different ranges: 0 to 10, 0 to 100, -50 to 50, -30 to 70 and many more. The ranges with negative minimums were introduced with the thought of maintaining the behaviour shown by ViLT's visual reasoning score (negative value for unreasonable/poor relation between the image and text). Other than global normalization, the author also experimented with batch normalization and then scaled up the ranges using different multipliers. Last but not least, in order to emphasize good predictions and severely punish bad ones, the author tried changing the nature of rewards from linear to quadratic and even cubic, but these approaches did not work as the polynomial nature of the reward function will be a poor choice for ensuring the model's smooth and stable gradient optimization. Conclusively, the reward function is set to produce globally normalized scores via min-max scaling with -8 and +8 as min and max respectively and then further scaling up the range to 0-100 as the unsupervised rewards.

Supervised Rewards

While experimenting with supervised rewards, the easiest approach of setting BLEU values between the target and prediction text as the policy reward did not work quite well. After that, although the approach mentioned in section 3.D is considered best practice, during the training, the author observed a directly proportional relationship between the run-time rewards graph and the loss graph. In the author's understanding, they both should be inversely proportional in nature (if the reward increases, the loss should decrease and vice versa), so just for experimentation, the supervised reward function was slightly modified to give a negative of what it was producing before (cumulative BLEU score between the target and predicted text descriptions) to enforce direct relationship. Surprisingly, this approach backfired in an unethical, unexpected and disturbing way, Although the model loss and rewards showed a similar relationship as what the author expected to see, the increase in KL divergence resulted in the presence of pornographic words in the prediction descriptions. Conclusively, this approach was dropped, and the model was again fine-tuned with the best practice reward function (section 3.D) but with a lower learning rate than before (1e-8 to 1e-9).

E. Evalutaion

A standard set of MT evaluations metrics are used: BLEU (Papineni et al., 2001) [35] and METEOR (Denkowski & Lavie, 2014) [36] to evaluate and compare the performance of pretrained, unsupervised and supervised models. Additionally, significance testing is performed on BLEU scores generated for in-domain **2016** testset via all the models mentioned above to compare the differences between pretrained and supervised, and pretrained and unsupervised.

V. Results

Countless experiments and tweaking gave away the final settings for both supervised and unsupervised models that work well to achieve the aim of this research. The results mentioned in this section clearly show that the PPO policy is working properly, hence, the results are authentic.

Before moving forward, the function to calculate the BLEU score for the entire dataset is imported from a package: sacrebleu while the function to do the same for METEOR is self-made. The METEOR score for each instance in the dataset is calculated using the package: nltk and the mean of all the scores calculated from the dataset is the resultant METEOR score for that dataset. BLEU (Table 2) and METEOR (Table 3) scores are calculated to evaluate the models (pretrained (baseline), supervised and unsupervised) on a total of 5 datasets (validation, 2016, 2017, 2018, 2017 COCO) as mentioned in section 4.A. The Opus model, when imported pretrained, uses MarianMTModel and MarianTokonizer as the base class for both model and tokenizer respectively hence, during testing, they are directly imported and used for evaluation after getting loaded with the trained parameters. The validation dataset is included in the results because the model was finalized without ever running on the validation dataset and hence, it works as another in-domain testset. Firstly, some comments on the performance of the supervised model in comparison to the baseline (pretrained); to justify the proper functioning of the PPO pipeline and provide a solid ground to comment on the performance of the unsupervised approach.

TABLE II
PERFORMANCE EVALUATION WITH BLEU SCORE ON VARIOUS TESTSETS

Evaluation	**BLEU**				
Models	Validation	Test 2016 Flickr	Test 2017 Flickr	Test 2018 Flickr	Test 2017 mscoco
Pretrained	35.39	34.98	**35.87**	**33.10**	28.26
Supervised	**35.40**	**35.41**	35.84	33.05	**28.38**
Unsupervised	34.81	34.84	35.17	32.32	27.89

TABLE III
PERFORMANCE EVALUATION WITH METEOR SCORE ON VARIOUS TESTSETS

Evaluation	**METEOR**				
Models	Validation	Test 2016 Flickr	Test 2017 Flickr	Test 2018 Flickr	Test 2017 mscoco
Pretrained	**0.5847**	0.5671	**0.5823**	0.5578	0.5326
Supervised	0.5838	**0.5680**	0.5811	**0.5593**	**0.5336**
Unsupervised	0.5805	0.5672	0.5778	0.5545	0.5317

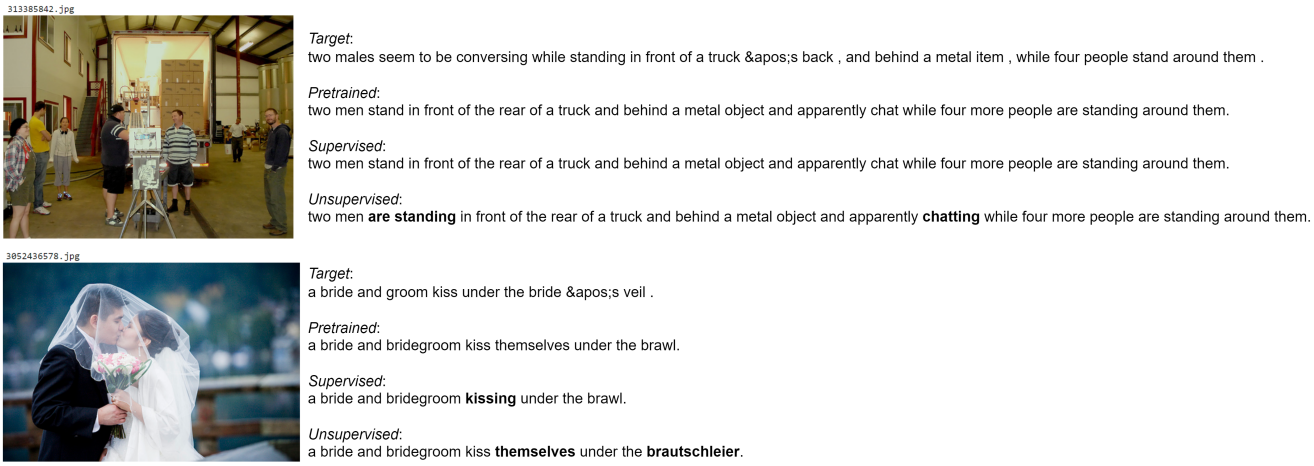

Fig. 3. Two examples showcasing target description, their corresponding images and predicted descriptions from pretrained (baseline), supervised and unsupervised models in order to perform qualitative analysis of the proposed approach (unsupervised)

Comparison with the baseline

Supervised Model

As seen in Table 2, supervised is out-performing the others on in-domain validation and 2016 testset (+0.01 BLEU and +0.43 BLEU respectively), while on out-of-domain 2017, 2018 and 2017 COCO testsets (-0.03 BLEU, -0.05 BLEU and +0.12 BLEU respectively), its performance is still evaluated better but with METEOR (Table 3)(-0.012 METEOR, +0.015 METEOR and +0.010 METEOR respectively) than BLEU. In overall, supervised Opus on Multi30K beats pretrained Opus, which is what to be expected with supervised fine-tuning. Additionally, after performing significance testing on the BLEU scores obtained from pretrained and supervised models, the value from the Permutation test: 0.0021 is less than the p-value (0.01) hence, it can be said that the difference between this pair of models is statistically significant. This result validates the optimization performed by the PPO in the author's implementation, hence, making the evaluation scores on unsupervised approach credible.

Unsupervised Model

In Table 2, the proposed approach (unsupervised model fine-tuned using visual signals) gives a tight competition to the pretrained baseline. On in-domain validation and 2016 testset, it is keeping up with the baseline with just a slight decrease of -0.58 BLEU and -0.14 BLEU respectively while, with out-of-domain 2017, 2018, 2017 COCO testset the difference is -0.70 BLEU, -0.78 BLEU and -0.37 BLEU respectively. On the other hand, with the METEOR metric (Table 3), the unsupervised model surpassed the baseline on the in-domain 2016 testset while having a tiny difference of -0.0042, -0.0045, -0.0033 and -0.0009 METEOR score on validation, 2017, 2018, and 2017 COCO testset respectively. Additionally, after performing significance testing on the BLEU scores obtained from pretrained and unsupervised models, the value from the Permutation test: 0.9332 is more than the p-value (0.01) hence, it can not be said that the difference between this pair of models is statistically significant. These results show that this approach can be further improved to surpass the baseline in no time, It points the research community towards improving the multimodal rewards by incorporating a better multimodal

model than ViLT, a more refined reward system or both and so much more. The qualitative analysis shown below proves the last point.

Qualitative Analysis

This section provides a manual evaluation of the performance of the proposed model with the help of two example predictions shown in Fig 3. Unlike unsupervised, pretrained and supervised models do not have any access to the image corresponding to the input. These two examples are strategically selected to show that an unsupervised model can handle both long and short descriptions. The first example on the top in Fig 3., shows that both the pretrained and supervised models produced exactly the same translation whereas the unsupervised model improved the grammar of the sentence by translating it in present continuous (are standing, chatting) form. The author suggests that this is possible because the model had access to an additional source of context (visual signals) available during training through the proposed novel multimodal unsupervised rewards. Another example shown in Fig 3., has a smaller target translation this time and both the pretrained and supervised model mistranslated bride 's (bride's) into a brawl which means something totally different from 'veil', but the unsupervised model left that word in input language rather than destroying the meaning of the translation. This again was made possible because the model has learned through visual signals that a word like brawl isn't semantically similar to brawl and would produce a poor multimodal unsupervised reward.

Conclusively, through all the results, it is clear that there is still a lot of scope for research in figuring out the best model or mechanism or both to incorporate visual signals into the reinforcement learning of the pretrained LLMs. However, this novel research being the first to target the approach of introducing multimodal unsupervised reward for reinforcement learning on large language models is successful in showcasing the potential of this approach.

VI. Conclusion

This paper proposes a novel approach to formulate rewards for reinforcement learning, which allows unsupervised machine translation using multiple modes of data. In this research, linguistic and visual data are combined to provide different sources of context before performing the translation. This approach in unsupervised machine translation is one of its kind and opens new doors for researchers to explore more. This paper includes an additional supervised approach with the same system but with only linguistic data to validate the strong ground of this setup, hence, making sure of the credibility of results produced by the unsupervised model and allowing trustworthy observations and their analysis. The paper also presents extensive experimentations to leave as little room for doubt as possible with respect to figuring out the optimal settings for the system. Also, the difficulty of the task should be noted, as the author had to spend a lot of time researching and failing before succeeding in this uncharted territory. Even though our unsupervised model's scores (using BLEU and METEOR) were close to the baseline model, the quality of its translations is noteworthy. For future work, the results show strong promise to extend this area of research and the author expects motivated researchers to further work towards finding better alternatives to multimodal transformer for visual reasoning other than ViLT, overall reward formulation and try tweaking other similar components that played a crucial part in the proposed system.

References

[1] Ive, J. et al. (2021) Exploring supervised and unsupervised rewards in machine translation, arXiv.org. Available at: https://arxiv.org/abs/2102.11403v1.

[2] Liu, Y. et al. (2023) Summary of CHATGPT/GPT-4 research and perspective towards the future of large language models, arXiv.org. Available at: https://doi.org/10.48550/arXiv.2304.01852.

[3] Yu, L. et al. (2017) Seqgan: Sequence generative adversarial nets with policy gradient, Proceedings of the AAAI Conference on Artificial Intelligence. Available at: https://doi.org/10.1609/aaai.v31i1.10804.

[4] Ranzato, M. et al. (2016) Sequence level training with recurrent neural networks, arXiv.org. Available at: https://arxiv.org/abs/1511.06732v7.

[5] Bahdanau, D. et al. (2017) An actor-critic algorithm for Sequence Prediction, arXiv.org. Available at: https://arxiv.org/abs/1607.07086.

[6] Ronald J Williams. 1992. Simple statistical gradientfollowing algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256.

[7] Kreutzer, J., Uyheng, J. and Riezler, S. (2018) 'Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning', Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) [Preprint]. doi:10.18653/v1/p18-1165.

[8] Konda, V.R. and Tsitsiklis, J.N. (1999) Actor-Critic Algorithms. NIPS Proceedings, 13, 1008-1014.

[9] He, D. et al. (2017) Decoding with value networks for Neural Machine Translation, Advances in Neural Information Processing Systems. Available at: https://dl.acm.org/doi/abs/10.5555/3294771.3294788.

[10] Choshen, L. et al. (2020) On the weaknesses of reinforcement learning for neural machine translation, arXiv.org. Available at: https://arxiv.org/abs/1907.01752v4.

[11] Li, J. et al. (2016) Deep Reinforcement Learning for Dialogue Generation, arXiv.org. Available at: https://arxiv.org/abs/1606.01541.

[12] Nguyen, K., Daumé III, H. and Boyd-Graber, J. (2017) 'Reinforcement learning for bandit neural machine translation with simulated human feedback', Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing [Preprint]. doi:10.18653/v1/d17-1153.

[13] Donato, D. et al. (2022) Mad for robust reinforcement learning in machine translation, arXiv.org. Available at: https://arxiv.org/abs/2207.08583.

[14] Schulman, J. et al. (2017) Proximal policy optimization algorithms, arXiv.org. Available at: https://arxiv.org/abs/1707.06347.

[15] Baltrušaitis, T., Ahuja, C. and Morency, L.-P. (2017) Multimodal Machine Learning: A Survey and taxonomy, arXiv.org. Available at: https://arxiv.org/abs/1705.09406v2.

[16] Atrey, P.K. et al. (2010) Multimodal Fusion for Multimedia Analysis: A survey - multimedia systems, SpringerLink. Available at: https://link.springer.com/article/10.1007/s00530-010-0182-0.

[17] Bhatt, C.A. and Kankanhalli, M.S. (2010) 'Multimedia Data Mining: State of the art and Challenges', Multimedia Tools and Applications, 51(1), pp. 35–76. doi:10.1007/s11042-010-0645-5.

[18] Jiang, S. et al. (2022) Deep Learning for Technical Document Classification, arXiv.org. Available at: https://arxiv.org/abs/2106.14269.

[19] Tiulpin, A. et al. (2019) Multimodal Machine Learning-based knee osteoarthritis progression prediction from plain radiographs and Clinical Data, Nature News. Available at: https://www.nature.com/articles/s41598-019-56527-3.

[20] Cho, S. et al. (2023) A framework for understanding unstructured financial documents using RPA and Multimodal Approach, MDPI. Available at: https://doi.org/10.3390/electronics12040939.

[21] Artetxe, M. et al. (2018) Unsupervised neural machine translation, arXiv.org. Available at: https://arxiv.org/abs/1710.11041.

[22] Lample, G. et al. (2018) Unsupervised machine translation using monolingual corpora only, arXiv.org. Available at: https://arxiv.org/abs/1711.00043.
[23] Anuchitanukul, A. and Ive, J. (2022) 'Surf: Semantic-level unsupervised reward function for machine translation', Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies [Preprint]. doi:10.18653/v1/2022.naacl-main.334.
[24] Lu, J. et al. (2019) Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, arXiv.org. Available at: https://arxiv.org/abs/1908.02265.
[25] Kim, W., Son, B. and Kim, I. (2021) Vilt: Vision-and-language transformer without convolution or region supervision, arXiv.org. Available at: https://arxiv.org/abs/2102.03334.
[26] Li, L.H. et al. (2019) VisualBERT: A simple and performant baseline for vision and language, arXiv.org. Available at: https://arxiv.org/abs/1908.03557.
[27] Jabbar, M.S., Shin, J. and Cho, J.-D. (2022) Ai Ekphrasis: Multi-modal learning with foundation models for fine-grained poetry retrieval, MDPI. Available at: https://www.mdpi.com/2079-9292/11/8/1275.
[28] Radford, A. et al. (2021) Learning transferable visual models from Natural Language Supervision, arXiv.org. Available at: https://arxiv.org/abs/2103.00020.
[29] Salvador, A. et al. (2017) 'Learning cross-modal embeddings for cooking recipes and food images', 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [Preprint]. doi:10.1109/cvpr.2017.327.
[30] Raffel, C. et al. (2020) Exploring the limits of transfer learning with a unified text-to-text transformer, arXiv.org. Available at: https://arxiv.org/abs/1910.10683.
[31] Tiedemann, J. and Thottingal, S. (2020) Opus-mt – building open translation services for the world, ACL Anthology. In Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, pages 479–480, Lisboa, Portugal. European Association for Machine Translation. Available at: https://aclanthology.org/2020.eamt-1.61.
[32] Sutskever, I., Vinyals, O. and Le, Q.V. (2014) Sequence to sequence learning with Neural Networks, arXiv.org. Available at: https://arxiv.org/abs/1409.3215.
[33] Dai, Z., Xie, Q. and Hovy, E. (2018) From credit assignment to entropy regularization: Two new algorithms for neural sequence prediction, arXiv.org. Available at: https://arxiv.org/abs/1804.10974.
[34] Elliott, D. et al. (2016) Multi30k: Multilingual English-German image descriptions, arXiv.org. Available at: https://arxiv.org/abs/1605.00459.
[35] Papineni, K. et al. (2001) 'Bleu', Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02 [Preprint]. doi:10.3115/1073083.1073135.
[36] Denkowski, M. and Lavie, A. (2014) 'Meteor Universal: Language specific translation evaluation for any target language', Proceedings of the Ninth Workshop on Statistical Machine Translation [Preprint]. doi:10.3115/v1/w14-3348.