Natural Language Processing Journal 7 (2024) 100072

Contents lists available at ScienceDirect

Natural Language Processing Journal

journal homepage: www.elsevier.com/locate/nlp

Enabling action crossmodality for a pretrained large language model

Anton Caesar [*], Ozan Özdemir, Cornelius Weber, Stefan Wermter

Knowledge Technology, University of Hamburg, Vogt-Koelln-Str. 30, Hamburg, 22527, Germany

ARTICLE INFO

Keywords:
Human–robot interaction
Machine translation
Action-vision-language model
Crossmodality
Robot learning

ABSTRACT

Natural language processing and vision tasks have recently seen large improvements through the rise of Transformer architectures. The high-performing large language models (LLMs) benefit from large textual datasets that are numerously available online. However, action and bidirectional action-language tasks are less developed, as these require more specific and labeled data. Therefore, we aim at enabling these robotic action capabilities for a pretrained LLM, while maintaining high efficiency with regards to the required training time and data size. To achieve this, we split up a Transformer-based LLM and insert a multimodal architecture into it. Specifically, we split a pretrained T5 LLM between its encoder and decoder parts, to insert a crossmodal Transformer component of a Paired Transformed Autoencoders (PTAE) bidirectional action-language model. The experiments are conducted on a new dataset, consisting of unimodal language translation and crossmodal bidirectional action-language translation. The natural language capabilities of the original T5 are re-established efficiently by training the crossmodal Transformer, which requires only one 5.7 millionth of the T5 model's original training data. Furthermore, the new model, called CrossT5, achieves high accuracy for the vision- and language-guided robotic action tasks. By design, the CrossT5 agent acts robustly when tested with language commands not included in the dataset. The results demonstrate that this novel approach is successful in combining the advanced linguistic capabilities of LLMs with the low-level robotic control skills of vision-action models. The code is available at this URL: https://github.com/samsoneko/CrossT5.

1. Introduction

Recently, large language models (LLMs) have gained popularity as versatile and powerful approaches to general purpose language processing. Being merely trained on large text corpora, they are capable of a wide range of tasks, including text generation, translation, classification and analysis as well as holding natural conversations, answering questions and generating action plans for robotic tasks. Popular implementations include PaLM 2 (Anil et al., 2023), LLaMA (Touvron et al., 2023) and GPT-4 (OpenAI, 2023), with the latter reaching a size of approximately 1 trillion parameters, as well as publicly available applications like ChatGPT (OpenAI, 2023), Bard (Anil et al., 2023) and Bing Chat (OpenAI, 2023) and open-source models such as BLOOM (Scao et al., 2023) and T5 (Raffel et al., 2020).

LLMs are based on the Transformer model (Vaswani et al., 2017), an encoder–decoder architecture incorporating self-attention (Lin et al., 2017). At its core, the Transformer converts one sequence of tokens to another, making it ideal for handling natural language. Bringing these capabilities of LLMs into the robotics domain to allow people communicate with a robot is not a trivial task. Through additional input processing, the architecture can also be adapted or extended for other modalities as well as crossmodal applications to translate between different modalities. Most of the popular multimodal implementations focus on adding visual capabilities such as image captioning, object recognition and visual information extraction to the Transformer architecture (Li et al., 2023; Zhu et al., 2023; Chen et al., 2023). Also, many implementations only deal with either image recognition/description (vision to language) or image generation (language to vision). Approaches that handle both directions rely on separate models trained for their respective tasks sharing their parameters (Zhang et al., 2021). Furthermore, they often require large-scale image–text datasets and only cover tasks on static images rather than image sequences or features extracted from them (Lee et al., 2023).

LLMs have a great potential for human–robot interaction (HRI) (Billing et al., 2023), while requiring more modalities than just vision and language to be combined. A robot perceives its physical environment via image sequences, proprioception or tactile input to carry out coordinated action. Through language, a robot can understand human commands and describe information on its own, hence, a robot should be able to translate between language and action in both directions, showing the same crossmodal behavior as expressed by humans. A common way to augment LLMs for robotic multimodality is via early

* Corresponding author.
E-mail address: anton.caesar@studium.uni-hamburg.de (A. Caesar).

https://doi.org/10.1016/j.nlp.2024.100072
Received 22 December 2023; Received in revised form 28 March 2024; Accepted 11 April 2024

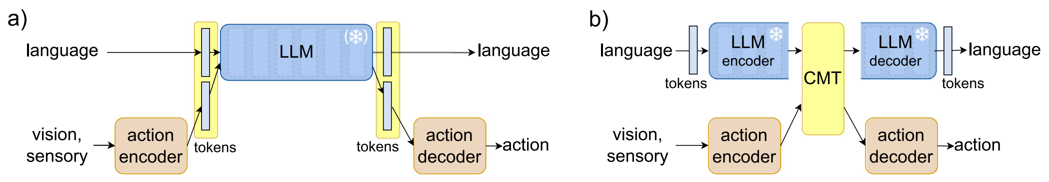

Fig. 1. Multimodal integration architectures combining LLMs with robotic behavior. (a) Integration in many conventional architectures is via the LLM's input and output tokens. (b) Our concept uses a crossmodal transformer (CMT) for intra-LLM integration.

fusion, where the robotic sensory input is converted into a textual format and is passed to the LLM as tokens of a uniform language prompt, as shown in Fig. 1a. Also the LLM's outputs are tokens, which may not be optimal to encode a robot's actions. Due to the novel tokens, fine-tuning of the LLM is highly advised (Brohan et al., 2023b). However, high-quality large datasets for multiple robotic modalities are scarce (Awais et al., 2023). Therefore, LLMs are being augmented with external tools (Mialon et al., 2023), and for human–robot collaboration it has been suggested to combine a nearly modality-independent physical task model with a dedicated dialogue model (Kleer et al., 2023).

Apart from the architecture, the most important factor influencing the performance of a model is the dataset. For language tasks, such as translation and summarization, there is an abundance of large-scale datasets.[1] For crossmodal vision and language processing on static images, datasets are also ready for use (Lin et al., 2015; Antol et al., 2015). However, for mixed datasets of robotic action (including vision and proprioception modalities) and language, data is harder to collect and requires more expense in labeling (Heinrich et al., 2018; Vuong et al., 2023).

To achieve multi- and crossmodal functionality on robotic action and natural language without the need for aligned large-scale datasets and expensive training, we propose a new approach, shown in Fig. 1b. We integrate a crossmodal architecture built for action-language tasks deeply into a pretrained LLM. This allows using the pretrained weights of the LLM while using the training procedure of the crossmodal architecture. The training of the new model, which is called CrossT5, uses a combination of two different datasets, one for the training of robotic action through the crossmodal architecture, the other for retraining the original LLM's features. This follows the assumption that the natural language capabilities of the LLM can deal with the addition of the new modality, and the resulting model can be used for both natural language processing as well as crossmodal action.

We use an HRI setup with the NICO robot (Kerzel et al., 2017) in the CoppeliaSim simulator (Rohmer et al., 2013), as visualized in Fig. 2. The robot perceives its environment through cameras in both eyes, and can interact with it by controlling its arms.

After conducting experiments with different dataset splits and loss calculations, we arrive at a final version of CrossT5 that satisfies our demands. The key advantages of our proposed approach include:

- The training does not need a large-scale language-action dataset to produce a working crossmodal model. The natural language capabilities are inherited from the LLM, and the action dataset can be simplistic.
- A tiny portion of a matching dataset is sufficient to reestablish the LLM's features during training. Afterwards, the new CrossT5 performs comparably on natural language tasks as the pretrained LLM before modification.
- The new architecture is flexible enough to allow for an easy exchange of the used language model, making it scalable for larger model variants or different LLMs.
- The training is very efficient. After only a short amount of training time, CrossT5 adapts to the LLM's language encodings and achieves high performance on both the natural language task and the robotic action-language tasks.
- In addition to the good results for both the linguistic T5 tasks and the multimodal Paired Transformed Autoencoders (PTAE) tasks (Özdemir et al., 2023), CrossT5 demonstrates high robustness for its language-to-action commands, having successfully adopted the linguistic capabilities of the T5.

2. Related work

While there are numerous approaches on vision-language and action-language crossmodality, many of them focus on specific modalities and can be divided into respective groups.

2.1. Vision-to-language with LLMs

Many approaches on multimodal crossmodality focus on adding image processing and understanding capabilities for LLMs (Mialon et al., 2023), which are either frozen or fine-tuned as part of the new architecture. While these approaches are crossmodal, they are not capable of bidirectional tasks.

For instance, BLIP-2 (Li et al., 2023) leverages the performance of pretrained language models and image encoders to enable vision-to-language generation without expensive training. For this, the image encoder and the language model are connected through a Querying Transformer (Q-Former), through which the vision and language Transformers share some of their parameters. For the experiments, an OPT (Zhang et al., 2022) model is used for the decoder-only LLM variant, while FLAN-T5 (Chung et al., 2022) is used for the encoder–decoder variant. Despite having fewer trainable parameters than its competitors, BLIP-2 achieves state-of-the-art performance, even outperforming much larger models in zero-shot VQA.

Similarly, the Flamingo model (Alayrac et al., 2022) incorporates pretrained and frozen LLMs and vision encoders. It accepts visual and text data as an input, and can perform tasks such as captioning, VQA and visual dialogue. Similar to our model, it combines the two modalities using gated cross-attention layers, where the keys and values are obtained from the visual features and the queries from the text language input. In few-shot learning, Flamingo sets a new state of the art on all of the 16 considered benchmarks, and is often on par with models specifically fine-tuned for a respective task.

MiniGPT-4 (Zhu et al., 2023) explores the advanced multimodal vision-to-language capabilities of GPT-4. It incorporates a frozen LLM, the Vicuna model (Chiang et al., 2023) built upon LLaMA, that is connected to a frozen vision encoder through a linear projection layer. This projection layer is trained on aligning the visual features with the LLM, which are passed to the language model in a combined text prompt. MiniGPT-4 expresses a variety of capabilities similar to those of GPT-4, processing visual information through a correct alignment of features.

PaLM-E (Driess et al., 2023) utilizes a similar method, but expands the concept to further input modalities, each with their respective encoder. After being embedded by the encoders, the input is fed to an LLM, which is a decoder-only PaLM (Chowdhery et al., 2023). PaLM-E

[1] https://commoncrawl.org/

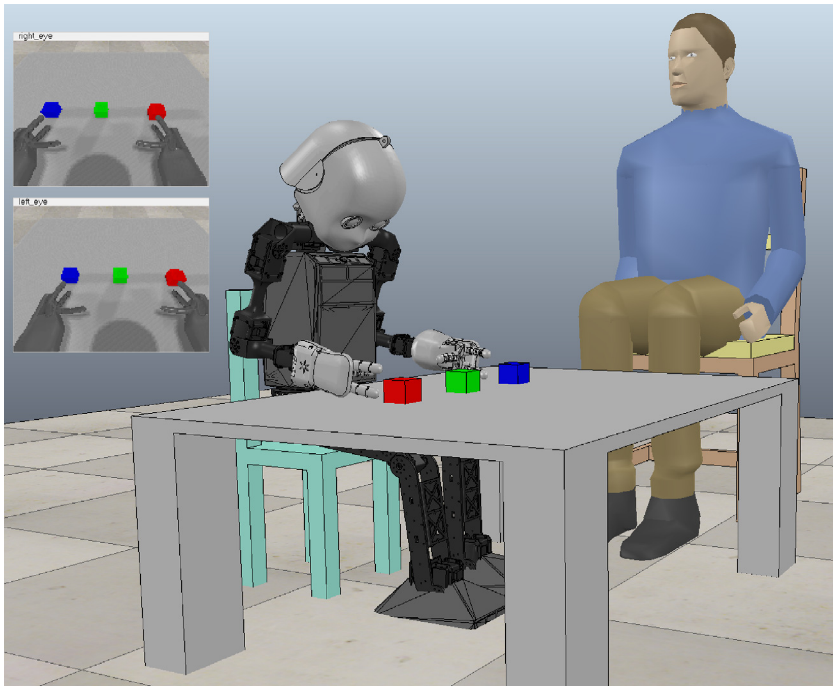

Fig. 2. NICO robot setup. NICO is supposed to converse with the human, execute commands, and comment on the scene, which is captured by its left and right eye cameras.

performs well on VQA tasks as well as robotic manipulation planning. However, the performance on NLG tasks drops significantly when using a smaller PaLM variant as an LLM.

VisionLLM (Wang et al., 2023) treats images as foreign languages, enabling the LLM to comprehend and execute vision-centric tasks. Other projects leaning more towards action execution show that, although still limited to language output, LLMs can generate action plans (Huang et al., 2022) and interact with multiple sensory modalities (Zhao et al., 2023) in a variety of different tasks in a virtual environment, even without being trained on them.

2.2. Action execution through multimodal input

Another category of approaches is not built for language production, but instead deals with the execution of robotic action based on multimodal input (Shridhar et al., 2023), in most cases language and vision/proprioception. Many of these approaches also focus on generalization (Guhur et al., 2023; Jang et al., 2022).

The VIMA model (Jiang et al., 2023) uses an encoder–decoder architecture to translate from language and vision to robotic output. The model accepts combined prompts consisting of language commands and descriptions as well as visual information, using the pretrained T5 (Text-to-Text Transfer Transformer) (Raffel et al., 2020) as a language encoder. VIMA outperforms state-of-the-art approaches on different robotic tasks, including zero-shot generalization.

Improving on the problem of lacking adaption in action execution, the Language-Informed Latent Actions with Corrections (LILAC) framework (Cui et al., 2023) corrects robotic action through language commands in real time. The ATLA model (Ren et al., 2023) demonstrates that language descriptions can help in the adaption to unseen tools in robotic policies. ATLA uses LLMs to generate these descriptions as well as obtain the respective feature representations. InstructRL (Liu et al., 2023) uses one unified multimodal encoder to encode both language and vision for robotic tasks in a virtual environment.

2.3. Action-centric crossmodality

Action-centric crossmodal approaches are able to reason and describe actions and visual information, but are not built for independent language-only tasks.

The RT-2 model (Brohan et al., 2023b) has recently demonstrated robotic action-language crossmodality for a large-scale pretrained LLM. Its architecture treats the robotic action as an extension of language, tokenizing it on the input level and de-tokenizing it for a robotic output. While having a high computation cost, RT-2 performs much better than earlier models like RT-1 (Brohan et al., 2023a), and generalizes to unseen objects and backgrounds while maintaining a high accuracy. Trained on the Open X-Embodiment dataset, a consolidated dataset with 22 robotic embodiments, the RT-X models show a further improvement in success rate (Vuong et al., 2023).

Another approach, the Mani-GPT model (Zhang et al., 2023), incorporates an LLM and a network that generates output for grasping objects. Its input consists of object labels from a visual module, the past dialogue history and the human instructions. Based on the nature of the input, the model classifies it into one of several response categories and generates a corresponding output. A similar approach, SayCan (Ichter et al., 2023), leverages the semantic knowledge of LLMs for action execution of low-level skills. The model uses the PaLM LLM and is connected to a robotic system that lets it navigate through and manipulate its environment.

While these approaches demonstrate bidirectional crossmodality to a certain extent, being able to generate both language and action, their language output is limited to the context of the visual or robotic scene the model is confronted with as they do not involve pretrained LLMs for general-purpose dialogues.

2.4. Full crossmodality

A bidirectional approach to both language as well as visual/robotic action tasks is the PTAE (Paired Transformed Autoencoder) model (Özdemir et al., 2023), which connects two separate input and output channels through a Crossmodal Transformer (CMT) (Irshad et al., 2021) to enable action-to-language, language-to-action and unimodal language and action skills. The action dataset that the model is trained on focuses on moving three colored cubes positioned on a table in front of the robot. For this, the model receives a textual description of the action as well as visual features and the robotic joint sequence. The PTAE model achieves high performance on the specified tasks, even

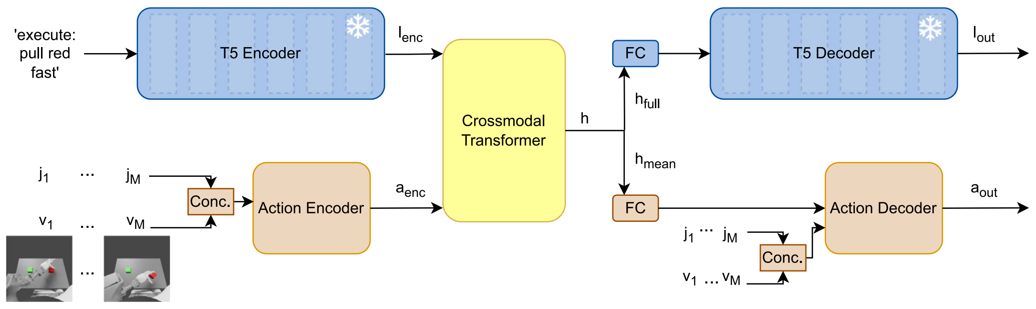

Fig. 3. CrossT5 architecture. The T5 encoder and decoder are integrated with the PTAE architecture via a crossmodal transformer (CMT). j vectors represent joint angle values of NICO's arms, while v vectors are visual features extracted from images recorded by NICO's eye cameras. Conc. denotes concatenation and FC is a fully connected layer.

with a small number of crossmodal training samples (vision/action with language labels), given sufficient unimodal experience (vision/action samples independent of language samples). However, the model is not built for language tasks outside of its multimodal training data, such as natural dialogue.

The multimodal Gato agent (Reed et al., 2022) is able to perform many different tasks in various modalities, such as image captioning, acting as a chatbot and playing Atari games, all on a single pretrained model. It is trained on a wide variety of data involving visual, language and action modalities. All of its modalities are fed to the model in a batched, tokenized input. While Gato's capabilities are numerous, it requires extensive training on a large number of datasets to achieve its crossmodal proficiency. In contrast, our approach uses a pretrained LLM to obtain language proficiency without requiring LLM fine-tuning.

3. Proposed approach

Our proposed model integrates a pretrained LLM with the crossmodal PTAE architecture. PTAE consists of two input encoders, one for language and one for action, and two respective decoders. The two input and output modalities are connected via the CMT in the middle. For action encoding and decoding, PTAE uses LSTMs. Since PTAE only features simple language processing, we extend it with an LLM to enhance its linguistic capabilities.

To decide on the LLM used, a few factors were taken into consideration. The model has to be pretrained, open source, and compact enough to be run locally and be fine-tuned without major expense. Furthermore, its architecture needs to be compatible with the PTAE model to allow for a seamless combination of the two models. A model that meets all these requirements is the Transformer architecture T5 (Raffel et al., 2020). It has high performance in its largest variant, but for our concept study, we utilize the T5-small variant with only 60 million parameters. A fine-tuned variant of T5, FLAN-T5, would also be suitable, but evaluations by Google concluded that this fine-tuning tends to result in worse performance for smaller model variants (Wei et al., 2022). Another model compact enough would be the smallest variant of BLOOM (Scao et al., 2023), featuring 560 million parameters. BLOOM has no encoders but a stack of 70 decoder blocks, hence there is no distinguished single point to integrate the PTAE. In contrast, T5 matches the encoder–decoder architecture of the PTAE model, making the gap between encoder and decoder an obvious point of integration. This yields the new CrossT5 model.

As presented before, most other vision-language approaches instead make use of early fusion, combining the data at an initial part of the model. These models also implement only a single multimodal direction but are either not built for language-to-action capabilities (BLIP-2, MiniGPT-4) or not for action-to-language capabilities (RT-2, VIMA).

3.1. Model architecture

CrossT5 follows the same schematic as the PTAE model. It retains the CMT and the two encoders and decoders, of which the action encoder and decoder remain unchanged from the PTAE, while the language encoder and decoder are from T5. The CMT takes the language encoding as *query*, while the action encoding is taken as *key* and *value*. After applying scaled dot product attention, it outputs the crossmodal hidden representation vector h, which is passed onto the two decoders. T5-small consists of 6 encoder and 6 decoder layers, which are split up in the middle to connect it with the CMT (Fig. 3).

The hidden dimension of the CMT and therefore of the output vector h is set to 512 to match the encoding of the T5-small (as opposed to 256 in PTAE). In the PTAE model, the mean over the temporal dimension of the vector h was used as the input for the language and action decoders. Since the tokens of the T5 language encoding are presented as one sequence, this step is unnecessary for the T5 language decoder and only done for the action decoder.

As part of the approach to the research question, we freeze the T5 weights. In contrast, we train the CMT and the action encoder and decoder from scratch. The T5 decoder is still used for backpropagating the language error during loss calculation, but its weights are not modified.

3.2. CLANT dataset

Our dataset *CLANT* (**C**rossmodal **L**anguage-**A**ction and **N**atural **T**ranslation) combines two separate datasets: an *action dataset* is used for adding the bidirectional action capabilities to the model, while a *language dataset* is needed to reestablish the T5's capabilities, since its encoder and decoder are separated by the CMT.

Action dataset. We train the crossmodal tasks of the architecture using the *NICO Coppeliasim Dataset*. A variant of this dataset was used for training the PTAE model (Özdemir et al., 2023). The dataset consists of 1440 samples in total, out of which 360 samples (25%) are used for testing, while 1080 samples (75%) are used for training.

Each sample consists of a sequence of images and joint values, and a description of an action carried out by the NICO robot (Kerzel et al., 2017). NICO sits in front of a table, on which three colored cubes are placed in three pre-configured positions. The entire scene is generated on Coppeliasim (Rohmer et al., 2013) using inverse kinematics. NICO has a camera in each eye, overseeing the table, its hands, arms and their shadows.

In each sample, NICO moves one cube using either its right or left arm. During this action, the camera records, depending on the sample, a sequence of $T = 40$, 60 or 85 images. From each image, a 30-dimensional feature vector is extracted using the channel-separated convolutional autoencoder (Özdemir et al., 2021), resulting in a matrix of $30 \times T$ features for each action. The joint angle values of both arms are also recorded for each of the T time steps, with 5 joint values for each arm.

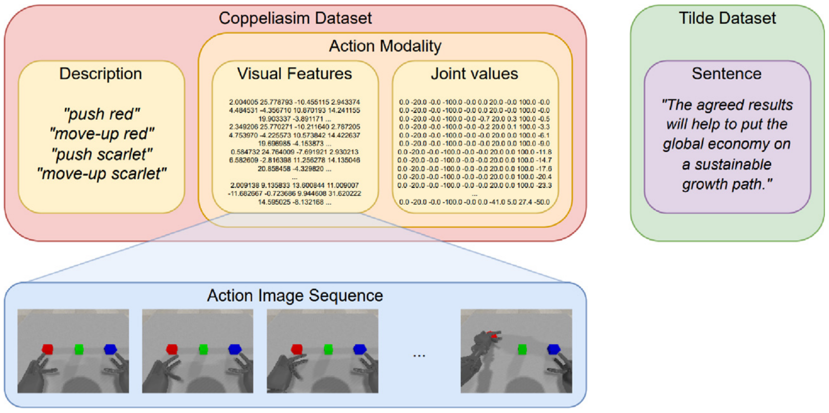

Fig. 4. Setup of a CLANT dataset sample. The NICO Coppeliasim dataset provides a textual action description, joint values and the visual features extracted from the action image sequence before training, while the Tilde RAPID 2019 dataset provides a natural language translation sample. These two datasets are combined into a single dataset, from which, depending on the performed task, different parts are used or ignored on the input level.

Each textual description consist of two words, one describing the desired action and the other describing the color of the target cube. There are 6 different cube colors and 4 distinct actions, each of which has an alternative name. In total, the action vocabulary consists of:

- **12 color words** (6 original/6 alternative): "red/scarlet", "green/harlequin", "blue/azure", "yellow/blonde", "cyan/greenish-blue", "violet/purple"
- **8 action words** (4 original/4 alternative): "push/move-up", "pull/move-down", "slide-to-left/move-sideways-to-left", "slide-to-right/move-sideways-to-right"

This results in $12 \times 8 = 96$ distinct textual commands.

The dataset includes one sample for each distinct *visual* action. This means that the alternative language descriptions do not increase the number of possible situations. Also, as visible in the vocabulary above, a language description does not specify the position of the cube, but its color.

Since there are always exactly three cubes on the table, with four possible actions, the total number of possible action sequences for a scene is $3 \times 4 = 12$. The number of distinct cube configurations is the number of subsets of 3 cubes from a set of 6 possible cube colors, which is 120. This makes a total of $12 \times 120 = 1440$ samples contained in the dataset.

Language dataset. The language dataset used for training the natural language capabilities of CrossT5 is the *Tilde RAPID 2019 German to English* dataset (WMT, 2019). It consists of more than a million sentence pairs in German and English taken from the press release database of the European Commission. As the original T5 model is already trained on English to German translation, this dataset is well suited for training the new model. We focus on language translation because it can be evaluated more easily than text summary or information extraction, and works well on shorter text samples.

For the new CLANT dataset, the first 1440 samples from RAPID 2019 are taken and added to the NICO Coppeliasim dataset as an additional category. The contents of one CLANT dataset sample are visualized in Fig. 4. As the content of RAPID 2019 is not ordered by any specification, taking the samples from the beginning does not cause the data to be restricted in vocabulary or versatility. 1440 translation samples would be an insufficient amount of data to train from scratch for a complex task like translation, but this is beyond the scope of this work. The CMT in the center of CrossT5 only has to learn how to identically reproduce the encoding from the T5 encoder to the T5 decoder for the language translation to still work.

3.3. Training setup

The new architecture should allow for the user to specify whether one is performing actions or having a conversation. To implement this kind of control, we introduce different mode signals, so the model can differentiate between the different operation modes. In the original PTAE model, mode signals were also used during training and inference. Following the same idea, the new signals now not only distinguish between the unimodal and crossmodal tasks from the NICO Coppeliasim dataset, but also the natural language translation task of the integrated T5 model. The signals are:

- **Translate**: Unimodal natural language translation. This mode trains the new model on natural language translation samples taken from the Tilde RAPID dataset to keep T5's capabilities intact.
- **Describe**: Crossmodal action-to-language translation. This mode trains the model to describe the perceived NICO action.
- **Execute**: Crossmodal language-to-action translation. This mode trains the model to generate the correct sequence of joint values corresponding to the NICO action description.
- **Repeat Action**: Unimodal action-to-action translation. This mode trains the model to repeat the sequences of joint values from the received NICO action.
- **Repeat Language**: Unimodal language-to-language translation. This mode trains the model to repeat the language description from the received NICO sample.

We refer to the last four signals as "PTAE signals". During training, one of the signals is chosen at random to determine the mode according to a varying probability distribution. In **Translate** mode, the model receives an English sentence sample from the Tilde RAPID 2019 dataset as its language input, with the added prefix "*Translate English to German:* ". This prefix is needed for the pretrained T5 model to differentiate the translation task from the rest of its skill set. In **Translate** mode, the action input is set as the repeated initial joint configuration of the robot concatenated with zeros for the visual features. The language target is set as the equivalent output that the original T5 model produces, while the action target is the initial joint for all time steps. This trains the robot to not move during an unimodal language task by design.

For the PTAE signals, the action is given to the model in the form of the joint value sequence of NICO's arms concatenated to the corresponding matrix of visual features. If the mode signal is **Execute** or **Repeat Language**, the action input is instead provided as the repetition of the joint values and visual features at the first time step. In the same way, the language input is the two-word description of the action, except for the **Describe** or **Repeat Action** mode, where the description is left out. In all cases, the desired mode signal term is appended as a prefix to the language input.

In **Execute** and **Repeat Action** modes, the action target is the correct sequence of joint values for the action, while the language target is an empty string. In the **Describe** mode, the action target is the repeated final time step joint configuration, while in the **Repeat Language** mode, the action target is the repeated initial joint configuration. In both modes, the language target is the corresponding two-word description of the action.

The **Translate** mode can be seen as an indicator that the new model treats the input in the same way the pretrained T5 would do. The *Translate* task was chosen because compared to its other tasks, T5-small displayed the best performance in English-to-German translation, and it can be easily measured quantitatively.

Technically, T5 supports a maximum encoding length of up to 512 tokens with positional encoding. By default, this number is capped at 20 tokens. Since the translation dataset includes sentences longer than 20 tokens, we set the maximum sequence length to 60 tokens.

During training, the T5 decoder uses *teacher forcing*. This means that instead of generating one token per time step, using the previously generated tokens as an input, the decoder uses the ground-truth token. Then, the error is calculated between all generated tokens and the ground-truth tokens individually. Because this method does not rely on the previous t tokens for a token $t+1$ to be predicted, the entire encoding can be processed in parallel with only one decoder run. As the T5 language encodings can be up to 60 tokens in length and would normally require the same number of decoder iterations, this makes the training much more efficient. The LSTM action decoder inherited from the PTAE model uses teacher forcing as well but does not run in parallel.

3.4. Loss calculation

For the action loss of CrossT5, we keep the calculation implementation of the PTAE model, as the action decoder and the general training process remain unchanged. This implementation takes the mean squared error (MSE) of the produced joint sequence and the target joint sequence. For the calculation of the language loss, there are two straightforward options, both at different points in the model architecture. We train CrossT5 with these two loss calculations to evaluate their usability for the modified architecture.

h-vector loss. In this calculation mode, the loss is defined as the MSE between the hidden vector h of the CMT and the target hidden vector $\hat{h}$:

$$L_{lang} = \frac{1}{N}\sum_{t=1}^{N}(h_t - \hat{h}_t)^2, \tag{1}$$

where t indexes the numerical values within an encoding of length N. This removes the need to run the T5 decoder and saves on computational expense. For the T5 translation, the target $\hat{h}$ is the encoding generated by the T5 encoder. The idea behind this is that for translation, the T5 encoder already produces an encoding that, when passed to the decoder without change, generates a good result. The CMT only needs to learn not to modify the encoding. While this proves to be true and works well for the T5 translation tasks, the targets for the crossmodal tasks of the NICO Coppeliasim dataset have to be given additional attention. Since not using the T5 decoder during training means only comparing numerical encodings against each other, we need T5 encodings that evidently produce correct sentences such as "*push blue*" or "*pull red*" in the decoder. The encodings of prompts like "*Translate English to English: NICO command*" reliably decode to "*NICO command*" in about 90% of cases and enable us to evaluate this training method. This means that during training, if, e.g. in **Describe** mode, the model receives an encoded action "*push blue*", the generated output vector h is compared to the T5 encoding of "*Translate English to English: push blue*". Since these two encodings differ in their token count, varying between 2 and 10 tokens, which causes problems in the loss calculation, we pad all language encodings of the PTAE signals to 20 tokens. Unfortunately, this method shows to be insufficient, which can be seen in the bar plot in Fig. 5. The language performance for **Describe** and **Repeat Language** is 0% and 1.36% respectively. Even for **Execute**, where the model is trained to give an empty output, the score is just 0.56%. We believe that this outcome is due to the use of padding tokens. The only PTAE signal that CrossT5 learns to some extent is **Repeat Action**. Here, the target is also an empty output, but unlike for **Execute**, the language input for **Repeat Action** is just *"repeat action:"*, and the action input is a joint sequence. This results in the T5 decoder not receiving a usable encoding, and giving an empty output in 53.89% of the cases.

T5 decoder loss. This method uses the loss calculation implemented in T5. All language output is decoded through T5 with teacher forcing, the loss calculated as the cross-entropy loss between target x and prediction y:

$$L_{lang} = \frac{1}{M}\sum_{t=1}^{M}\left(-\sum_{i=1}^{V} x_t^{[i]} \log y_t^{[i]}\right), \tag{2}$$

where t indexes the tokens within a sequence of length M, and V is the vocabulary size. The error is then backpropagated through the decoder into the CMT. This increases the accuracy for the PTAE signals dramatically, because calculating the loss at the actual output layer teaches the model to generate encodings directly corresponding to, for example, "*push blue*" and not "*Translate English to English: push blue*". Unfortunately, as displayed in Fig. 5 in the middle, this training method does not work well for the T5 Translation. When training only with the **T5 decoder** loss calculation, the Translation performance drops to 5.86% on the test data (Fig. 5). Though the dataset used for training the **Translate** mode aligns with what the pretrained T5 model can already do, for the same input prompt, the pretrained model often generates an output completely different from the target. Although most dataset targets and the respective T5 predictions are identical in terms of information and grammatical correctness, the flexible syntax of the German language means that they can be completely different in their word order and sentence structure. For this reason, the cross-entropy loss calculates a large error for an acceptable translation. This interferes with the CMT, which then tries to learn to change the translation encodings in accordance with the dataset targets instead of learning to identically map them to the decoder.

To solve this problem, we propose a new way of calculating the loss. Since training the translation using the h-**vector** loss already proved to be sufficient, we keep it only for the **Translate** mode. In each training iteration, the current mode signal is checked in the loss function. If it is **Translate**, the function calculates the MSE loss between the h-vector and the T5 encoder output. If it is any of the PTAE signals, the function instead runs the T5 decoder and calculates the cross-entropy loss of its output compared to the target of the NICO sample. This is possible because of the way the T5 model is integrated in the new architecture, making it optional during training. We name this dynamic way of switching the loss calculation **mixed** loss. The total loss combines the corresponding language loss L_{lang} and the MSE-defined action loss L_{act} (see Özdemir et al., 2023):

$$L_{tot} = \alpha L_{lang} + \beta L_{act}, \tag{3}$$

where, for our experiments, $\alpha = \beta = 1$. The results shown in Fig. 5 on the right confirm the success of this method, as the **mixed** loss calculation results in a good accuracy for all mode signals.

4. Experiment results

We train the CrossT5 model for up to 10,000 epochs as a trade-off between good performance and training time. To give both the translation training and the NICO training the necessary attention, we

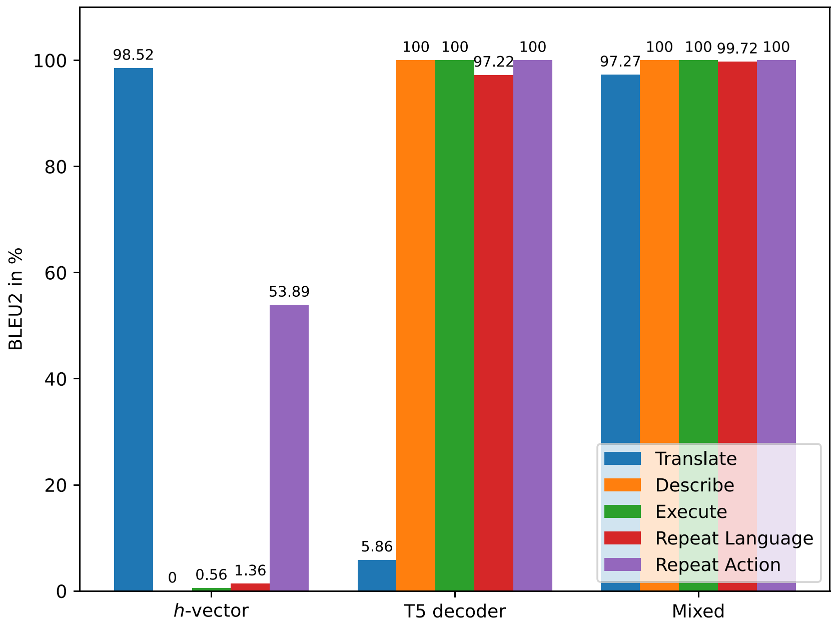

Fig. 5. Language test performance in the three loss modes "*h*-vector", "T5 decoder" and "Mixed", trained for 10,000 epochs.

Table 1
Language test performance with different shares of the **Translate** signal, trained for 5,000 epochs.

Signal	10%	25%	50%	75%	90%
Translate	57.48	71.33	91.48	92.67	92.04
Describe	96.94	99.17	99.17	93.33	94.44
Execute	100	100	100	100	100
Repeat Language	100	99.44	99.17	99.37	99.27
Repeat Action	100	100	100	100	100

set a split of 3/4 for **Translate** and 1/4 for the PTAE signals for training. The PTAE signals are further split into 1/3 for **Describe**, 1/3 for **Execute** and 1/6 for **Repeat Action** and **Repeat Language** each. It is difficult to determine whether this is an optimal split because the model performs well with different splits as well. Both 1/2 for **Translate** and 1/2 for the PTAE signals and also 9/10 for **Translate** and 1/10 for the PTAE signals only affect the accuracy by 1%–2%, shown in the results in Table 1. Also apparent is the consistently good performance for all NICO signals, where only the performance for **Describe** drops slightly at a 75% or higher share of the **Translate** signal.

Training CrossT5 on a system equipped with an RTX 3080 Ti, a run with 1000 epochs requires around 35 min. The entire 10,000 epochs therefore take just under 6 h to complete.

Evaluating the translation performance is difficult. As explained in the introduction of the **mixed** loss calculation, T5 and therefore also the new CrossT5 often give a grammatically and factually correct output that only differs from the dataset target in terms of syntax structure or choice of words. In addition to that, because we used T5-small, the T5 variant with only 60 million parameters, for the training, not all translations are accurate even for the pure T5 model. When evaluating the translation performance based on the dataset targets, the calculated BLEU2 accuracy only reaches about 10% at 10,000 epochs. This is a poor score for the translation performance, but even more a score not representative of the spirit of our proposed architecture. Because we train CrossT5 on identically reproducing the language encodings in the CMT, its translation performance can never exceed and most likely not fully reach the level of T5. With that in mind, the fairest way of estimating the translation performance of CrossT5 is not to compare its output with the dataset target, but instead with the output of the pure T5 model. This gives us an estimate of how much of the original T5 performance is retained in the new model.

For evaluating the NICO performance, the output is compared with the actual dataset targets, not the PTAE output, because CrossT5 is trained from scratch on these tasks. The language accuracy for the PTAE signals **Describe** and **Repeat Language** is calculated as the BLEU2 score compared to the target sentences, while the action accuracy for **Execute** and **Repeat Action** is the deviation from the correct joint sequence, calculated as the normalized root-mean-squared error (NRMSE).

Figs. 6 and 7 show the language and action results of CrossT5 over 10,000 epochs, evaluated on the test data. At 10,000 epochs, the model achieves a BLEU2 score of 97% for the **Translate** signal and 99%–100% for the PTAE signals. This demonstrates that almost all of the initial language translation capabilities of T5 stay intact, while the added crossmodal skills also achieve high performance. This is confirmed by the fact that most of the PTAE signals have already reached over 90% accuracy after only 3000 epochs. Furthermore, as the CMT quickly learns that the language output for **Execute** and **Repeat Action** should always be empty, these signals even reach 100% accuracy after only 1000 epochs.

Also in terms of action accuracy, CrossT5 performs well, with an NRMSE of 0.85% at maximum and an average of 0.4% at 10,000 epochs. It is noticeable that for the PTAE signals, the crossmodal signals **Describe** and **Execute** always have a higher error than the unimodal signals **Repeat Language** and **Repeat Action**. In addition to that, the action performance of the **Translate** signal is much better than all PTAE signals, having an NRMSE of only 0.025%.

As the weights of the CMT are initialized randomly, the language performance at 0 epochs displayed in Fig. 6 is at chance level. For the action performance in Fig. 7, the NRMSE score of around 18% for the action output is worse than the average distance of any dataset joint sequence from the initial joint values (around 6%) and worse than the average distance of all joint sequences to each other (around 11%) in the dataset. For the language accuracy in Fig. 6, the BLEU2 score of the output is 0% for all signals, except for the **Execute** signal, where it is about 90%. This comes from the fact that by chance, some random initialization seeds cause the T5 decoder to always generate an empty output for a certain input, which is the target for **Execute** and **Repeat Action**. For instance, another run with a different seed instead resulted

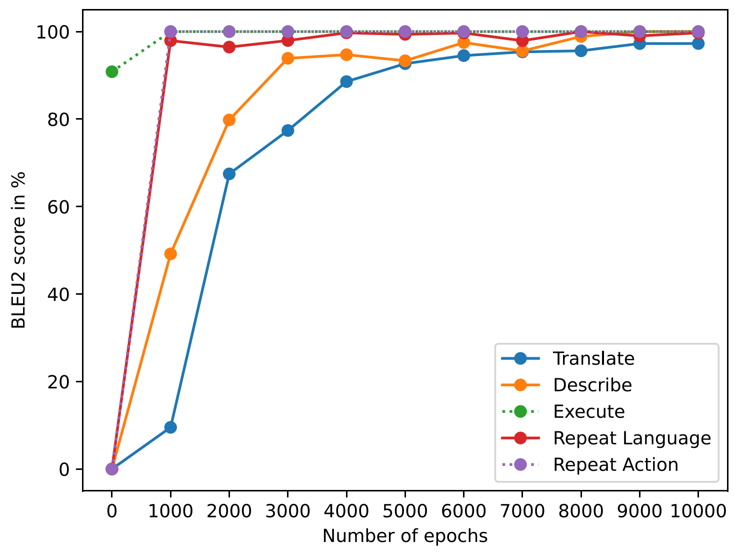

Fig. 6. CrossT5 language test performance over training, calculated as the BLEU2 score of the language output compared to the target.

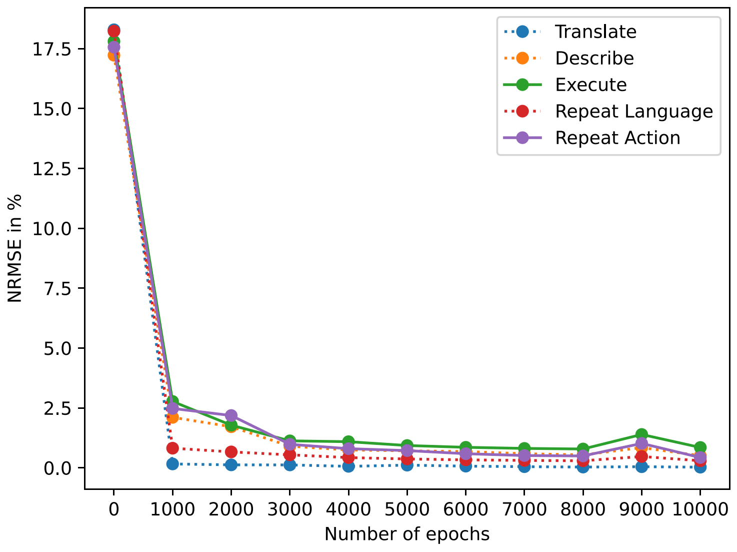

Fig. 7. CrossT5 action test performance over training, calculated as the NRMSE between the action output and the target joint sequence.

in an 80% accuracy for the **Repeat Action** signal, and 0% for all others including **Execute**.

4.1. Practical action evaluation

To better assess the actual performance of the action execution, the synthetic accuracy calculated as the NRMSE is not sufficient. Therefore, we conduct an additional practical evaluation, in which the joint values generated by CrossT5 for the **Execute** signal are executed in a simulated 3D environment, the CoppeliaSim Edu V4.3.0 (Rohmer et al., 2013). For this, the same setup as in the dataset is used, consisting of the NICO robot in front of a table with three cubes on it. For each generated action, the respective cube configuration is loaded in the simulator.

To evaluate whether an action is actually successful, three factors are taken into consideration:

- whether the correct cube is being pushed in the right direction, and farther than a minimum threshold
- whether the correct cube is being pushed in the right direction, and farther or equal than the dataset threshold
- whether no other cubes are being touched or accidentally moved

Based on these factors, an automatic score is calculated for each generated joint sequence for the **Execute** prompts in the test dataset.

To determine how the threshold for successful actions should look like, the joint sequences as contained in the dataset are also executed and used as a benchmark. To reach this threshold, the execution has to be of at least the same quality as in the corresponding dataset sample, meaning a movement by roughly the same magnitude and no other

Table 2
Quality of the action execution for the checkpoints ranging from 1,000 to 10,000 epochs, given as the share of "successful" and "perfect" executions of all the evaluated test dataset samples.

Quality	Epochs									
	1,000	2,000	3,000	4,000	5,000	6,000	7,000	8,000	9,000	10,000
Successful	53.17	95.00	96.39	98.06	98.61	98.61	97.78	98.33	93.61	95.83
Perfect	34.17	75.56	75.00	79.17	77.22	81.67	83.33	82.50	59.44	81.11

cubes touched. Even if an executed action does not reach the same quality as the dataset sample, it could still be considered a success if the magnitude of the cube movement is above a minimum threshold. This distinction is necessary as many of the generated movements are just below the magnitude of the dataset example, but are de facto still successful (no other cubes are touched, and the correct cube is very clearly moved in the right direction with a coordinated movement).

To appropriately represent this fact, we distinguish between a "perfect" action, indicating an execution quality equal to the dataset, and a "successful" action, indicating an execution quality that would still be seen as a success, i.e. the correct cube is moved in the correct direction but not reaching the threshold set for perfect actions. The checkpoints from 1000 to 10,000 epochs are evaluated on the test dataset, and the results can be seen in Table 2.

As the results show, CrossT5 achieves high quality in the practical evaluation of the generated joint sequences. In the best case, the model manages to execute 98.61% of the actions successfully, while 83.33% are of almost identical quality to the corresponding dataset samples. Interestingly, these scores are valid for the checkpoints at 6000 and 7000 epochs, while the 10,000 epoch checkpoint reaches 95.83% and 81.11%. Also, the checkpoint at 9000 iterations is an outlier again, scoring worse than even the one at 2000 in both metrics.

4.2. Language robustness

As explained in Section 1, another intention of the project is to explore how the combination of a crossmodal architecture better suited for simple language input with an LLM could enable additional language robustness even without a more complex language-action dataset. To determine whether this assumption would prove true, we conduct an additional robustness evaluation on the final CrossT5 model.

In the NICO Coppeliasim Dataset used for training CrossT5, each sample contains a language description of the respective action that can both serve as an input command in the language-to-action **Execute** signal (e.g. "execute: push red") or a target output in the action-to-language **Describe** signal. These language descriptions are built from a limited vocabulary, only containing words for the color of the intended cube and the direction it should be pushed in. Although the dataset already includes alternative descriptions for both color and direction (e.g. "scarlet" instead of "red" and "move-up" instead of "push"), these variations are also hardcoded and do not teach the model any actual semantic understanding. In the original PTAE architecture, the descriptions were one-hot encoded, further limiting the actual language processed by the model.

In the new CrossT5 architecture, the language descriptions are encoded and processed through T5, but still, only the simple language commands like "push red" are used for the training. To evaluate whether the model can handle more complex language, 9 alternative command patterns are used, testing different aspects of more natural language such as varying vocabulary, word order, and additional phrasings such as politeness and adverbs. By applying these 9 command patterns, more complex language input is constructed from the simple descriptions used during training, as visualized in Table 3.

We reevaluate the 10,000 epoch checkpoint on these 9 robustness modes and afterwards test its practical action performance in the same way as in Section 4.1. The results can be seen in Table 4.

The results show a small to medium performance drop in some of the robustness modes, which is expected for the more complex commands as in mode 2 and 6. However, the overall performance, especially for the "successful" actions, remains almost unchanged in many cases, even exceeding the accuracy of the simple dataset commands in modes 7 and 8. Even in mode 6, which produces the worst results, the accuracy of 84.72% is still respectable. Not reflected in Table 4 but still important to mention is the fact that even for the failed action executions, in none of the cases did the model confuse colors or commands and move the wrong cube.

Overall, CrossT5 demonstrates a good performance on more natural input, in many cases reaching or even exceeding the performance on the original dataset samples. It should be mentioned that the robustness applies to the language-to-action **Execute** signal only; as the language output has been specifically trained to align to the dataset descriptions, the action-to-language **Describe** or language-to-language **Repeat Language** signals still produce simple language outputs. For example, a command like "repeat language: would you please push red" will still output "push red" in the majority of cases.

5. Discussion

With the performance of CrossT5 being close to 100% in both the translation tasks from the pretrained T5 as well as the crossmodal language-action tasks from the NICO dataset, CrossT5 shows promising initial results. The experiments are only conducted with one dataset and one language model. Extending the model with larger T5 variants would be straightforward, but more adjustments would be necessary for different LLM architectures. While the bottleneck between the encoder and decoder of T5 is an obvious place to insert the PTAE via CMT, many LLMs use decoder-only architectures. Finding an optimal pair of layers for CMT insertion in such models will require experimental exploration. Alternative architectures, e.g. those that add small trainable matrices between a frozen network's layers, as done for fine-tuning for multi-task learning (Hu et al., 2022; Rusu et al., 2022), could also be investigated for a multimodal enhancement of LLMs.

An advantage of integrating a pretrained language model and only training the crossmodal architecture to keep its capabilities is that it saves a lot of computational resources during training. Given the frozen language model, the dataset can be much smaller than the datasets used for training state-of-the-art LLMs. With only 1080 training and 360 test samples, CrossT5 successfully re-established the T5's original translation abilities. Moreover, the model calculates the loss for the LLM tasks on the hidden representation vector h, which is outputted by the CMT and does not have to use the language model decoder output. For this reason, the English-to-German training does not require targets. Even for the integration of LLMs with different skills, a small collection of suitable inference prompts suffices to reliably retain their capabilities during the training of the new CrossT5 model.

The number of parameters in the CMT is 2,629,120 (the action model has an additional 3,021,322 parameters) as opposed to the 60 million frozen parameters of T5-small. This is in the order of 22.8 times fewer parameters. The language dataset used to retrain T5's original abilities has a size of 131 KB, as opposed to the 745 GB of the C4 dataset the original T5 was trained on. This is about 5.7 million times less data. The training time of CrossT5 was 6 h on an RTX 3080 Ti GPU. While there is no official information about the training time of T5, similar projects using the T5 architecture (Sarti and Nissim, 2022; Ulčar and

Table 3
The 9 different robustness patterns. "direction" and "color" refer to the corresponding word from the dataset sample.

Nr.	Robustness pattern	Example for "push red"
1	Please + *direction* + *color*	Please push red
2	Would you please + *direction* + *color*	Would you please push red
3	*direction* + the + *color* + cube	Push the red cube
4	*direction* + the + *color* + cube now	Push the red cube now
5	Please + *direction* + the + *color* + cube	Please push the red cube
6	Would you please + *direction* + the + *color* + cube	Would you please push the red cube
7	*color* + *direction*	Red push
8	The + *color* + cube + *direction*	The red cube push
9	**Alternative direction word** + *color*	Shove red

Table 4
Quality of the action execution for the 10,000 epoch checkpoint, evaluated on the 9 different robustness modes.

Quality	Robustness mode								
	1	2	3	4	5	6	7	8	9
Successful	95.28	88.61	95.28	93.89	94.17	84.72	96.67	96.39	95.56
Perfect	79.17	68.06	76.67	75.83	74.44	64.17	80.00	78.89	78.61

Robnik-Šikonja, 2023) give estimates of multiple days to over a week even for the smallest checkpoint, while being trained on four 40 GB A100 GPUs. With only these little resources spent, CrossT5 achieves over 97% of the performance of the original T5 after retraining.

In addition to the surprisingly good results for both the crossmodal PTAE tasks as well as the translation from T5, CrossT5 also displays impressive performance when tested with more complex language commands on the **Execute** signal. Even though the model has only been trained on the simple descriptions from the NICO Coppeliasim dataset, such as "push red", it is able to execute complex commands like "Would you please push the red cube" in more than 84% of cases. Other command variations, like switching the word order ("red push") or using alternative descriptions ("shove red") do not deteriorate the practical performance. This shows that CrossT5 not only retains the features of T5, but can furthermore leverage its advanced linguistic capabilities for the crossmodal PTAE tasks as well.

Currently, the CrossT5 model has not been tested on its ability to generalize to new scenes and actions, as the CLANT dataset comes with limited actions and pre-configured object positions. As we wanted to prove the feasibility of our approach with limited resources available, we did not extend our testing to larger datasets with more variety in their setups and/or continuous object positioning. Based on the promising test results, we are confident that the model will also be able to perform well on more complex datasets. As the architecture is easily scalable, it can be used with different and even larger variants of T5 and other LLMs by merely adjusting the hidden dimension size of the CMT. As a first experiment in that direction, we tested the model with FLAN-T5, which features the same size variants and architecture as T5, but an extended skill set, and the model achieved good performance. Also, we are currently working on integrating reinforcement learning into the model to train it to reach continuous object positions without requiring more teacher trajectories.

6. Conclusion

Towards bringing together an LLM's conversation capabilities with a robot's sense and action capabilities, we have proposed a model that strongly combines a robotic model with an LLM, i.e. where both share internal representations within a CMT with mutual reading and writing access. The CMT, inserted between the T5 encoder and decoder, adapts well to T5's internal representation, while requiring only little training data and effort. This leaves the performance of T5 intact.

In the evaluation, the new CrossT5 model achieves close to 100% accuracy in the added crossmodal action tasks and retains the performance of a representative task of its T5 LLM. Furthermore, it shows high performance in both the practical action evaluation and the robustness for more complex language-to-action input. This demonstrates that the approach of splitting up a frozen LLM and combining it with a crossmodal architecture delivers on its premise. The resulting model displays good language-action performance while staying flexible enough not to damage the skill set of the language model. By its design, it also requires little computational cost in training and is robust to more complex language-to-action commands even without dedicated training. Hence, the procedure is efficient to enable crossmodal capabilities for an LLM.

CRediT authorship contribution statement

Anton Caesar: Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Conceptualization. **Ozan Özdemir:** Writing – review & editing, Writing – original draft. **Cornelius Weber:** Writing – review & editing, Writing – original draft. **Stefan Wermter:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Yuan Yao and Zheni Zeng for useful feedback. This research was partially funded by the German Research Foundation (DFG) in the project Crossmodal Learning (TRR-169).

References

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al., 2022. Flamingo: A visual language model for few-shot learning. In: Advances in Neural Information Processing Systems, vol. 35, pp. 23716–23736.

Anil, R., Dai, A.M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al., 2023. Palm 2 Technical Report. arXiv:2305.10403.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D., 2015. VQA: Visual question answering. In: 2015 IEEE International Conference on Computer Vision. ICCV, Santiago, Chile, IEEE Computer Society, pp. 2425–2433. http://dx.doi.org/10.1109/ICCV.2015.279.

Awais, M., Naseer, M., Khan, S., Anwer, R.M., Cholakkal, H., Shah, M., Yang, M.-H., Khan, F.S., 2023. Foundational models defining a new era in vision: A survey and outlook. arXiv preprint arXiv:2307.13721, arXiv:2307.13721.

Billing, E., Rosén, J., Lamb, M., 2023. Language models for human-robot interaction. In: Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction. HRI '23, Association for Computing Machinery, New York, NY, USA, pp. 905–906. http://dx.doi.org/10.1145/3568294.3580040.

Brohan, A., Brown, N., Carbajal, J., et al., 2023a. RT-1: Robotics transformer for real-world control at scale. arXiv preprint arXiv:2212.06817, arXiv:2212.06817.

Brohan, A., Brown, N., Carbajal, J., et al., 2023b. RT-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818, arXiv:2307.15818.

Chen, G., Zheng, Y.-D., Wang, J., Xu, J., Huang, Y., Pan, J., Wang, Y., Wang, Y., Qiao, Y., Lu, T., Wang, L., 2023. VideoLLM: Modeling video sequence with large language models. arXiv preprint arXiv:2305.13292, arXiv:2305.13292.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., Stoica, I., Xing, E.P., 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al., 2023. Palm: Scaling language modeling with pathways. J. Mach. Learn. Res. 24 (240), 1–113.

Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al., 2022. Scaling instruction-finetuned language models. arXiv:2210.11416.

Cui, Y., Karamcheti, S., Palleti, R., Shivakumar, N., Liang, P., Sadigh, D., 2023. No, to the right: Online language corrections for robotic manipulation via shared autonomy. In: Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction. HRI '23, ACM, http://dx.doi.org/10.1145/3568162.3578623.

Driess, D., Xia, F., Sajjadi, M.S.M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., Florence, P., 2023. PaLM-E: An embodied multimodal language model. In: Proceedings of the 40th International Conference on Machine Learning. ICML '23, JMLR.org, pp. 8469–8488.

Guhur, P.-L., Chen, S., Pinel, R.G., Tapaswi, M., Laptev, I., Schmid, C., 2023. Instruction-driven history-aware policies for robotic manipulations. In: Liu, K., Kulic, D., Ichnowski, J. (Eds.), Proceedings of the 6th Conference on Robot Learning. In: Proceedings of Machine Learning Research, vol. 205, PMLR, pp. 175–187.

Heinrich, S., Kerzel, M., Strahl, E., Wermter, S., 2018. Embodied multi-modal interaction in language learning: The EMIL data collection. In: Proceedings of the ICDL-EpiRob Workshop on Active Vision, Attention, and Learning. ICDL-Epirob 2018 AVAL, p. 2p.

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., 2022. LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations. ICLR, URL: https://openreview.net/forum?id=nZeVKeeFYf9.

Huang, W., Abbeel, P., Pathak, D., Mordatch, I., 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (Eds.), Proceedings of the 39th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 162, PMLR, pp. 9118–9147.

Ichter, B., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A.T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Joshi, N.J., Jeffrey, K., Ruano, R.J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., Fu, C.K., 2023. Do as I can, not as I say: Grounding language in robotic affordances. In: Liu, K., Kulic, D., Ichnowski, J. (Eds.), Proceedings of the 6th Conference on Robot Learning. In: Proceedings of Machine Learning Research, vol. 205, PMLR, pp. 287–318.

Irshad, M.Z., Ma, C.-Y., Kira, Z., 2021. Hierarchical cross-modal agent for robotics vision-and-language navigation. In: 2021 IEEE International Conference on Robotics and Automation. ICRA, pp. 13238–13246. http://dx.doi.org/10.1109/ICRA48506.2021.9561806.

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., Finn, C., 2022. BC-Z: Zero-shot task generalization with robotic imitation learning. arXiv preprint arXiv:2202.02005, arXiv:2202.02005.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., Fan, L., 2023. VIMA: General robot manipulation with multimodal prompts. arXiv preprint arXiv:2210.03094, arXiv:2210.03094.

Kerzel, M., Strahl, E., Magg, S., Navarro-Guerrero, N., Heinrich, S., Wermter, S., 2017. NICO — Neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction. In: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication. RO-MAN, pp. 113–120. http://dx.doi.org/10.1109/ROMAN.2017.8172289.

Kleer, N., Rekrut, M., Wolter, J., Schwartz, T., Feld, M., 2023. A multimodal teach-in approach to the pick-and-place problem in human-robot collaboration. In: Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction. HRI '23, Association for Computing Machinery, New York, NY, USA, pp. 81–85. http://dx.doi.org/10.1145/3568294.3580047.

Lee, S., Kim, W.J., Ye, J.C., 2023. LLM itself can read and generate CXR images. arXiv preprint arXiv:2305.11490, arXiv:2305.11490.

Li, J., Li, D., Savarese, S., Hoi, S., 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597, arXiv:2301.12597.

Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y., 2017. A structured self-attentive sentence embedding. In: 5th International Conference on Learning Representations (ICLR), Toulon, France, April 24-26, 2017, Conference Track Proceedings.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P., 2015. Microsoft COCO: Common Objects in Context. arXiv preprint arXiv:1405.0312, arXiv:1405.0312.

Liu, H., Lee, L., Lee, K., Abbeel, P., 2023. Instruction-following agents with multimodal transformer. arXiv preprint arXiv:2210.13431, arXiv:2210.13431.

Mialon, G., Dessi, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Roziere, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., Scialom, T., 2023. Augmented language models: A survey. Trans. Mach. Learn. Res..

OpenAI, 2023. GPT-4 technical report. arXiv preprint arXiv:2303.08774, arXiv:2303.08774.

Özdemir, O., Kerzel, M., Weber, C., Lee, J.H., Hafez, M.B., Bruns, P., Wermter, S., 2023. Learning bidirectional action-language translation with limited supervision and testing with incongruent input. Appl. Artif. Intell. 37 (1), http://dx.doi.org/10.1080/08839514.2023.2179167.

Özdemir, O., Kerzel, M., Wermter, S., 2021. Embodied language learning with paired variational autoencoders. In: 2021 IEEE International Conference on Development and Learning. ICDL, IEEE, pp. 1–6. http://dx.doi.org/10.1109/ICDL49984.2021.9515668.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. 21 (1), 1–67.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S.G., Novikov, A., Barth-maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J.T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., de Freitas, N., 2022. A generalist agent. Trans. Mach. Learn. Res. URL: https://openreview.net/forum?id=1ikK0kHjvj.

Ren, A.Z., Govil, B., Yang, T.-Y., Narasimhan, K.R., Majumdar, A., 2023. Leveraging language for accelerated learning of tool manipulation. In: Liu, K., Kulic, D., Ichnowski, J. (Eds.), Proceedings of the 6th Conference on Robot Learning. In: Proceedings of Machine Learning Research, vol. 205, PMLR, pp. 1531–1541.

Rohmer, E., Singh, S.P.N., Freese, M., 2013. CoppeliaSim (formerly V-REP): a versatile and scalable robot simulation framework. In: Proc. of the International Conference on Intelligent Robots and Systems. IROS, https://www.coppeliarobotics.com.

Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R., 2022. Progressive neural networks. arXiv:1606.04671.

Sarti, G., Nissim, M., 2022. IT5: Large-scale text-to-text pretraining for Italian language understanding and generation. ArXiv preprint 2203.03759, URL: https://arxiv.org/abs/2203.03759.

Scao, B.W.T.L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A.S., Yvon, F., Gallé, M., et al., 2023. BLOOM: A 176B-Parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100, arXiv:2211.05100.

Shridhar, M., Manuelli, L., Fox, D., 2023. Perceiver-actor: A multi-task transformer for robotic manipulation. In: Liu, K., Kulic, D., Ichnowski, J. (Eds.), Proceedings of the 6th Conference on Robot Learning. In: Proceedings of Machine Learning Research, vol. 205, PMLR, pp. 785–799.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G., 2023. LLaMA: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, arXiv:2302.13971.

Ulčar, M., Robnik-Šikonja, M., 2023. Sequence-to-sequence pretraining for a less-resourced Slovenian language. Front. Artif. Intell. 6, http://dx.doi.org/10.3389/frai.2023.932519.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), In: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc..

Vuong, Q., Levine, S., Walke, H.R., et al., 2023. Open X-Embodiment: Robotic learning datasets and RT-X models. In: Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023.

Wang, W., Chen, Z., Chen, X., Wu, J., Zhu, X., Zeng, G., Luo, P., Lu, T., Zhou, J., Qiao, Y., Dai, J., 2023. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. In: Oh, A., Neumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (Eds.), In: Advances in Neural Information Processing Systems, vol. 36, Curran Associates, Inc., pp. 61501–61513.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V., 2022. Finetuned language models are zero-shot learners. In: International Conference on Learning Representations. ICLR, Virtual Event, April 25-29, 2022, URL: https://openreview.net/forum?id=gEZrGCozdqR.

WMT, 2019. ACL 2019 fourth conference on machine translation (WMT19), shared task: Machine translation of news. URL: http://www.statmt.org/wmt19/translation-task.html.

Zhang, Z., Chai, W., Wang, J., 2023. Mani-GPT: A generative model for interactive robotic manipulation. Procedia Comput. Sci. 226, 149–156. http://dx.doi.org/10.1016/j.procs.2023.10.649, Proceedings of International Conference on Biomimetic Intelligence and Robotics.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X.V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P.S., Sridhar, A., Wang, T., Zettlemoyer, L., 2022. OPT: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068, arXiv:2205.01068.

Zhang, H., Yin, W., Fang, Y., Li, L., Duan, B., Wu, Z., Sun, Y., Tian, H., Wu, H., Wang, H., 2021. ERNIE-ViLG: Unified generative pre-training for bidirectional vision-language generation. arXiv preprint arXiv:2112.15283, arXiv:2112.15283.

Zhao, X., Li, M., Weber, C., Hafez, B., Wermter, S., 2023. Chat with the environment: Interactive multimodal perception using large language models. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS.

Zhu, D., Chen, J., Shen, X., Li, X., Elhoseiny, M., 2023. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592, arXiv:2304.10592.