Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Master Thesis

Action Transcript Prediction from Multi-Modal Environments using Vision-Language Models

Author:
Yihao Wang

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Hilde Kühne

Supervisor:
Dr. Raphael Memmesheimer

Date: August 7, 2024

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Bonn 07.08.2024

Place, Date

Yihao Wang

Signature

Abstract

Domestic service robots aim to bring more convenience to human life by performing housework. Nowadays, open vocabulary brings an entirely new aspect: Different domestic tasks do not necessarily need to be implemented individually beforehand. In contrast, given a reasonable verbal task description, e.g., heat my lunch, bring out the trash. The domestic service robot should be able to comprehend these instructions along with the environment information it locates, e.g., RGB image and depth information, and perform the task. Considering the challenge of this task, more detailed step-by-step instructions are normally also given at the current stage. This challenge is referred to as the Vision-Language Navigation (VLN) and Vision-Language Manipulation (VLM) task.

We introduced LIAM. An end-to-end model that takes **L**anguage, **I**mage, **A**ction, and **M**ap information as input and predicts an action transcript. The language and image input are encoded with CLIP-backbone, in which we designed two different pre-training tasks to fine-tune the existing weights and pre-align the latent space. We froze the weight of both language and image encoders during the end-to-end training and trained only on autoencoder-like projection heads. For the action sequence and the semantic map, we used a learnable embedding layer and a map encoder to encode, respectively. All embeddings were concatenated and fed to a multi-modal Transformer encoder block, where modal-type encoding and causal attention were applied. These Transformer layers are expected to fuse the multi-modal inputs and learn a reasonable general representation, which can be used eventually to predict an action sequence.

We conducted experiments on the ALFRED challenge dataset, a simulator-generated benchmark for VLN and VLM tasks. Following previous approaches, we generated semantic maps for each trial in the preprocessing stage, which include obstacles, explored areas, class semantics, and pose as information.

We observed the CLIP models' limitations on simulator-generated data. We demonstrate the importance of pre-aligning embedding spaces from different modalities and find that incorporating semantic maps enhances the end-to-end model's performance. Despite these improvements, a significant gap exists between our approach and state-of-the-art methods. Proper training of an end-to-end model that integrates all information from different modalities remains challenging.

Contents

1 Introduction

With the rapid evolution of deep learning research, particularly in the natural language domain, we have witnessed the emergence of enormous Transformer-based models capable of generating high-quality texts and synthetic images (Brown et al. 2020; Yixin Liu et al. 2024; Ramesh et al. 2022). The interest in applying and evolving these methods in robotics, especially domestic service robots, is on the rise. The ultimate goal for domestic robotics research is to comprehend and perform complicated tasks successfully in every household from arbitrary verbal descriptions.

Artificial intelligence research has mimicked the structure and mechanisms of human brain nerves since the beginning of perceptron (Rosenblatt 1958). As human beings receive and process information in multiple modalities, for example, our brain often processes egocentric images and language instruction simultaneously to make a decision. In terms of deep learning, a model will receive input from different modalities, e.g., text, visual, and audio, and output prediction based on all this information. In 2021, OpenAI released a foundation model, CLIP, which innovatively uses a contrastive learning training paradigm for connecting text and images (Radford et al. 2021). The contrastive learning paradigm, which involves learning representations of data by contrasting similar and dissimilar pairs, is applied to textual and visual input. The model's strength in open-vocabulary zero-shot image classification draws much attention in multi-modality learning.

Many robotics applications soon utilized the CLIP-based model for the open-vocabulary approach in different down-streaming tasks, e.g., instance segmentation (Shafiullah et al. n.d.). However, the research in training a model that helps to predict the whole action transcript sequence based on the language instruction and visual information (e.g., RGB image, depth information) is still in its early phase. In terms of understanding arbitrary language instruction, which can be a combination of countless different action sequences, is a challenge in itself. In addition, realistic environments are very different from the setup, light conditions, and other factors. Objects to interact with also have different shapes and textures, which brings difficulties when it comes to interacting with those objects with the robot arms. Training a model that understands all information combined and has a strong generalisability requires a considerable amount of data.

Several simulator-generated datasets have been built eversince. Two major tasks for understanding a multi-modal environment are Visual-Language Navigation (VLN) and Visual-Language Manipulation (VLM). VLN requires the agent to navigate in an environment and to reach a goal specified by a natural language instruction. VLM requires the agent for more complicated manipulation of objects based on the natural language command. Most VLM datasets are built with robotic arms, which have a fixed overhead camera (Vuong et al. 2023).
A combination of VLN and VLM for domestic service robots is genuinely a more complicated task despite those datasets requiring less sophisticated interaction. In this case, a correctly predicted instance mask and the correct action are counted as a success case. In this thesis, we introduce an end-to-end multi-modal model LIAM, which receives inputs from different modalities (**L**anguage, **I**mage, **A**ction and **M**ap) and predicts an action sequence. A CLIP-like pre-training paradigm is introduced to pre-train the encoders for different modalities. We investigate our ideas with the ALFRED Challenge, a benchmark task that requires a robot to understand and execute complex natural language instructions in a simulated environment (described in chapter 4).

Contribution of our thesis:

1. In our thesis, we introduced two CLIP-like self-supervised training paradigms. These paradigms are designed to pre-align two or more different embedding spaces during the pre-training stage, enhancing the model's ability to understand and process multi-modal inputs. After the pre-training stage, we further trained the end-to-end model by freezing the heavy-headed encoder and training a light-designed projection adapter.

2. Semantic maps provide a high-level understanding of the environment and are typically used with deterministic policy. In our model, we introduced a novel approach that utilized semantic maps as an additional modality directly in addition to textual and visual inputs—the map modality is designed to serve as a knowledge base for the model. We explored the potential of a multi-modal model to fuse map information with visual and text inputs, thereby enhancing the robot's spatial understanding and decision-making capabilities.

3. Unlike most related works tackling the ALFRED challenge, which fine-tuned or used pre-trained object detection models, e.g., MaskRCNN (He, Gkioxari, et al. 2017). We followed FILM's language processing module (Min et al. n.d.), which predicts occurring objects from the language instruction for each trial. Leveraging this approach, we built an object vocabulary, replaced the detector with CLIPSeg (Lüddecke and Ecker 2022), and predicted the instance mask in an open vocabulary fashion.

This thesis is organized in the following manner. Section 2 covers all the fundamental knowledge essential for this thesis. This includes the attention mechanism of the transformer and the progression from fundamental contrastive learning in computer vision to modern clip-based multimodal models. Chapter 3 presents all related works to the ALFRED challenge, which our thesis followed and modified. Chapter 4 describes the ALFRED dataset and the pre-processing stage, i.e., generating semantics maps. In Chapter 5, we describe our model design and methodology in detail, including details of the pre-training and end-to-end training phases; we also report implementation details. Chapter 6 shows all the experiment results and our analysis of the different components. Chapter 7 concludes the thesis and lists limitations and potential future works.

2 Fundamentals

This chapter reviews all fundamental knowledge leading to the related work and our approach.

2.1 Transformers

Sequence-to-sequence transduction tasks are usually tackled with the recurrent neural network (RNN)-based architecture, e.g., Long Short-Term Memory (Hochreiter and Schmidhuber 1997). The biggest shortcoming of the RNN-based model is its inability to parallel computation, i.e., the hidden state h_t can only be computed once h_{t-1} has been first computed. (Vaswani et al. 2017) introduces Transformer, an encoder-decoder structure model. The core concept of the Transformer is its multi-head attention mechanism. Given an input X, query, key, and value (Q, K, V) are computed using three separate weight matrices (fully-connected layer). The output of the Transformer is the weighted sum of the values, where these weights are computed using the similarity score of the queries and keys. Mathematically, the attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad \text{(Vaswani et al. 2017)}$$

The similarity score function is defined using a scaled dot-product. In addition, Transformer introduced the multi-head attention mechanism. An n-head Transformer layer is implemented using n trainable linear projections of the queries, keys, and values. Thus, it allows the model to focus on different information scales (global or local). However, the attention mechanism does not consider temporal information since the dot-product-based similarity score won't change if the order of the sequence changes. Positional encoding is thus introduced to incorporate the temporal information into the input embedding. Transformer used the sinusoidal functions of different frequencies for its positional encoding (Vaswani et al. 2017). The success of the Transformer-based model has also drawn attention in the computer vision domain. The challenge in adapting a Transformer-based model in computer vision is the proper way to tokenize an image. If each pixel in the image

is treated as an individual, then a general image resolution of 224 × 224 with three channels would already be a vast computational cost. Vision Transformer (ViT) (Dosovitskiy et al. 2020) divides the image into a certain number of patches with size 16×16 as tokens, and these patch tokens will be further linearly embedded as input. Figure 2.1 demonstrates ViT's approach. ViT also applies positional en-

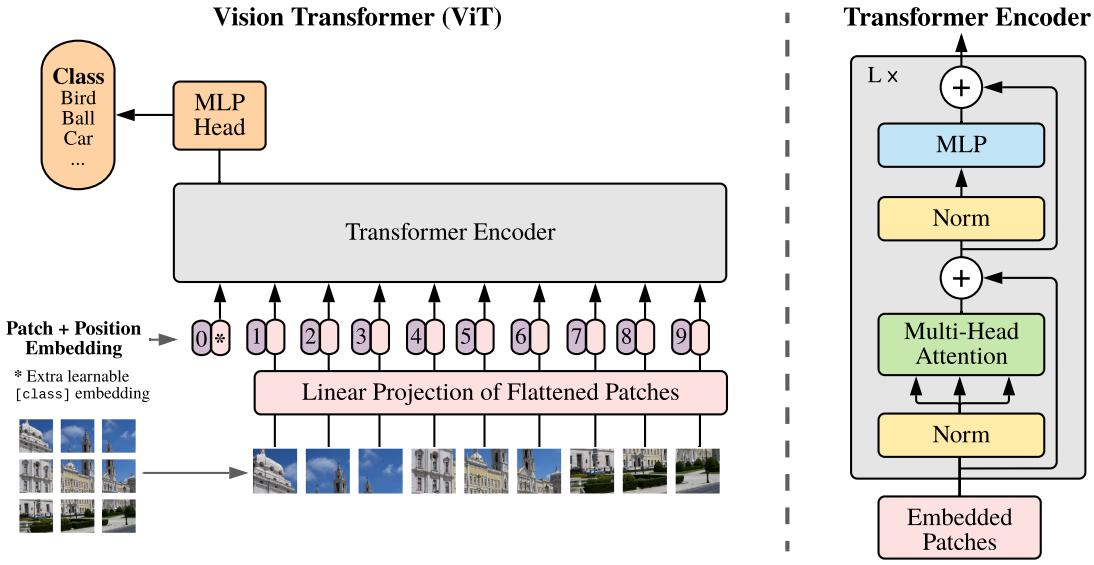

Figure 2.1: ViT architecture. An image is first split into 16 × 16 patches. The patch tokens are further embedded using linear projection. (Dosovitskiy et al. 2020)

coding to all patches and feeds them forward to Transformer encoder layers. The representation of the special "[CLS]" token serves as the global representation of the image and is utilized further for different vision down-streaming tasks.

2.2 Contrastive Learning

The research in contrastive learning aims to solve instance-level discrimination instead of discriminating among classes. One of the most basic contrastive training objectives is the following: Given a mini-batch of N images: $X_1^{org}, ..., X_N^{org}$. Images are first augmented to $X_1^{aug}, ..., X_N^{aug}$. For each image $X_i^{org}, \forall i \in N$, its encoded feature representation $f_{i,org}$ is called an anchor. The only positive pair is itself and the encoded feature of its augmented version $f_{i,aug}$, and all $X_{j\neq i}^{org}, X_{j\neq i}^{aug}, \forall j \in N$ are considered as negative. The training objective is to pull the encoded features of positive pairs as close as possible to each other in the embedding space, and negative pairs should be pushed away as their representations are dissimilar to the anchor.

(Wu et al. 2018) first proposed a feature memory bank-based approach. After computing the feature of the current query, a non-parametric softmax classifier is introduced. For given embedding v of image x, the probability that this image is classified as i is following:

$$P(i \mid v) = \frac{exp(v_i^T v/\tau)}{\sum_{j=1}^{n} exp(v_j^T v/\tau)}.$$

Instead of using the model output $w_i^T v$ as softmax classifier, they used exponential similarities $v_i^T v$. τ is a temperature scaling factor. This feature memory bank stores all features v_j, and it is periodically updated during training time. During the classification inference time, they compute the cosine similarity of the test image's embedding with all the features stored in the memory bank; the k-nearest classifier is used to output the final prediction.

SimCLR (T. Chen et al. 2020) designed a straightforward training objective. The same image was augmented in two different ways, and the similarity of both representations is trained to maximize their agreement. A projection head (MLP+ReLU) was added after the encoding neural network, and this design was essential and brought over 10% performance improvement. The model is trained using a contrastive loss similar to the above.

Distillation with no labels (DINO) (Caron et al. 2021) is a more recent framework in contrastive learning. It operates by passing two different augmentations of one image sample to student and teacher networks separately. Both networks serve as feature extractors, using ViT architecture as the backbone. DINO introduces a centering operation in the teacher network to avoid model collapse, which is done by subtracting the mean value of all samples. The training objective for the student network is to predict the feature output from the teacher network.

2.3 CLIP: Contrastive Language-Image Pre-Training

Open-vocabulary approaches have garnered significant attention in the field of computer vision. In the past decade, most vision downstream tasks were trained via datasets with a fixed number of classes. The CLIP model, with its contrastive learning paradigm, brings a promising solution for inferencing open vocabulary vision tasks. The text encoder is a 12-layers Transformer encoder block with eight attention heads (Radford et al. 2021). The model offers two alternatives for image encoders: a ResNet-50-based feature encoder or a Vision Transformer (ViT)-based backbone (Dosovitskiy et al. 2020; He, X. Zhang, et al. 2016). After embedding the text and image using separate encoders, CLIP is trained by maximizing the

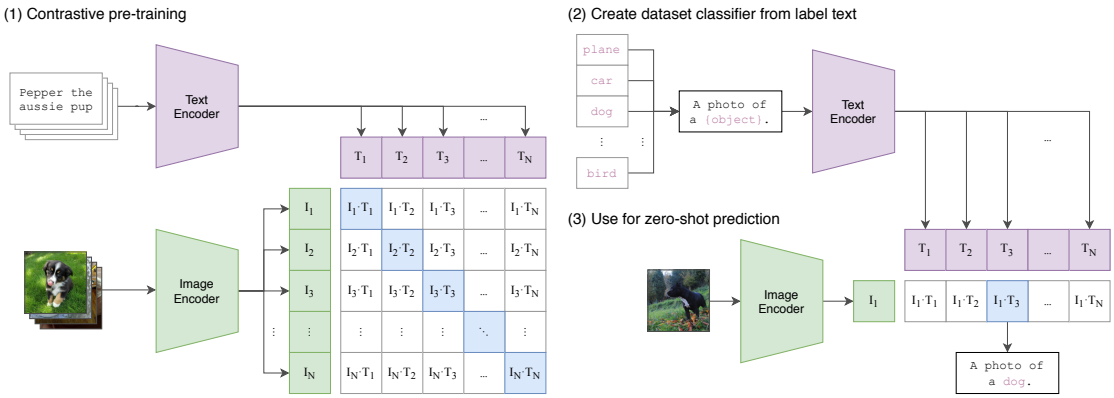

Figure 2.2: Diagram of CLIP main approach. (1) illustrates the CLIP pre-training stage, which matches the image and its corresponding label. (2) & (3) illustrates how image and text encoders are used in zero-shot inference. (Radford et al. 2021)

similarity score of the representation of corresponding image and text pairs. Both image and text feature representations I_f and T_f are first normalized to I_e and T_e using L2 normalization. The similarity score for both embeddings is chosen based on a scaled cosine similarity. Each entry (i, j) in the similarity matrix S is thus computed by:

$$S_{ij} = I_e^i \cdot T_e^j.$$

Similarity matrix S is then scaled to compute the final logits, $logits = S \cdot exp(\tau)$, where τ is the temperature hyperparameter. CLIP constraints that in each mini-batch, one image has only one corresponding label, and both sequences are put in corresponding order, namely, the i-th image and the i-th label are pairs. Thus, the ground truth label can be arranged as an identity matrix I_N, where N is the batch size (32768). CLIP computes both text-to-image and image-to-text losses using cross-entropy loss with the log-softmax, mathematically defined as:

$$\mathcal{L}_{\text{text-to-image}} = -\frac{1}{N}\sum_{i=1}^{N} log\left(\frac{exp(S_{ii}/\tau)}{\sum_{j=1}^{N} exp(S_{ij}/\tau)}\right),$$

$$\mathcal{L}_{\text{image-to-text}} = -\frac{1}{N}\sum_{j=1}^{N} log\left(\frac{exp(S_{jj}/\tau)}{\sum_{j=1}^{N} exp(S_{ij}/\tau)}\right),$$

The total contrastive loss is the average of both losses:

$$L_{\text{total}} = \frac{1}{2}(\mathcal{L}_{\text{text-to-image}} + \mathcal{L}_{\text{image-to-text}}).$$

The power of the CLIP models is the ability of zero-shot inference. CLIP creates the data classifier using a simple prompt: "A photo of {object}", where object is the name of the class label. All the text is then forwarded using the pre-trained text encoder. The new given unseen image is also forwarded using the pre-trained image encoder. A cosine similarity score is computed between all text embeddings and the single image embedding. The result of the classification is the text with the highest score.

A massive amount of work is built on CLIP models for different down-streaming tasks, e.g., Image classification (W. Kim, Son, and I. Kim 2021), Object Detection (Gu et al. n.d.; Zhou et al. 2022), semantic segmentation (Lüddecke and Ecker 2022; J. Xu et al. 2022), 3D recognition (R. Zhang et al. 2022), action prediction (H. Luo et al. 2022; Wang, Xing, and Yong Liu 2021). These approaches have designed similar methods adapted to their specific vision downstream tasks. For example, GroupViT first uses a group-based approach for semantic segmentation by introducing learnable group tokens to the ViT architecture (J. Xu et al. 2022). During inference time, each group is assigned the semantic label with the highest similarity score representing the text input.

Despite the outstanding performance of CLIP models, latency is always an important factor when people apply models in real life. A couple of works aimed to replace heavy Vision Transformer-based backbone with some mixture models. Mobile CLIP introduces Text-RepMixer, a hybrid text encoder of 1-d convolutions and self-attention layers, instead of the transformer architecture used in the CLIP model along with MCi. This hybrid image encoder is an improved version of FastViT (Vasu, Gabriel, et al. 2023). In addition, MobileCLIP utilizes a well-designed reinforced dataset for the pretraining. These design choices and architecture modifications not only accelerated CLIP (VIT-B/16) with 2.3 $\times$ speed but also a better average performance on 38 different datasets (Vasu, Pouransari, et al. 2024).

The CLIP contrastive training paradigm can also be adapted (Figure 2.3) when faced with tasks of sequential information, such as action recognition task. In this case, the model should learn to encode the corresponding video sequence and its annotated label to have a high cosine similarity score. The only challenge is to compute a reasonable sequence representation of the video sequence from the video encoder. Action CLIP proposes three approaches Wang, Xing, and Yong Liu 2021: a) non-parametric mean pooling operation, which computes the mean value of all separate frame representations straightforwardly. This method brings efficiency to the model training but does not have the ability to learn the temporal information in the video sequence. The other two more advanced approaches add a learnable 1D convolution layer/LSTM layer, or transformer blocks and take

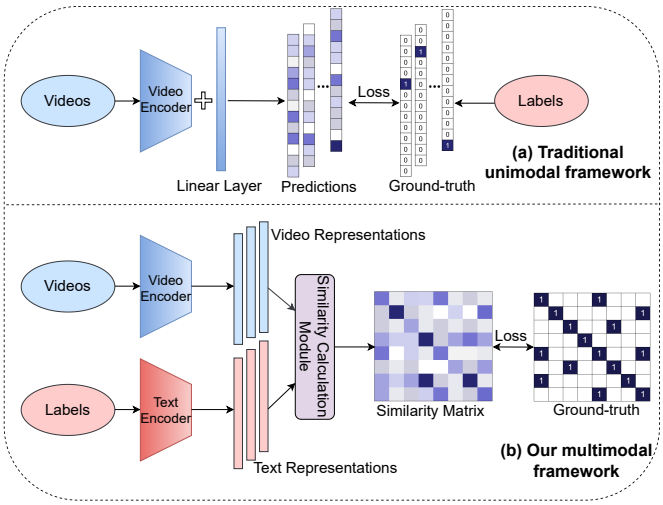

Figure 2.3: Unlike classical action recognition training, ActionCLIP proposes adapting the CLIP training paradigm by learning the similarity mapping from text and video modality (Wang, Xing, and Yong Liu 2021).

the temporal information also into consideration. A subsequent mean pooling is also applied to compute the final action representation. Another modification in ActionCLIP is the training loss. Since the amount of the video is larger than the number of labels, a CLIP-like one-hot ground truth matrix can not be formed, and thus, Kullback-Leibler (KL) divergence loss is used as an alternative to the cross entropy loss. For two given discrete probability distributions, KL divergence loss is mathematically defined as:

$$\mathcal{D}_{KL}(P \parallel Q) = \sum_i P \cdot log\left(\frac{P(i)}{Q(i)}\right).$$

For continuous probability distributions:

$$\mathcal{D}_{KL}(P \parallel Q) = \int_{-\infty}^{\infty} P(x) \cdot log\frac{P(x)}{Q(x)} dx.$$

In the case of ActionCLIP, P is the softmax-normalized similarity matrix and Q is the ground truth matrix (Wang, Xing, and Yong Liu 2021). The softmax-nomarlized similarity score is thus defined as

$$P_{\text{video-to-text}} = \frac{exp(s(x, y_i)/\tau)}{\sum_{j=1}^{N} exp(s(x, y_j)/\tau)},$$

where $s(\cdot, \cdot)$ denotes the consine similarity of video x and its corresponding label

y_i. Similarly,

$$P_{\text{text-to-video}} = \frac{exp(s(y, x_i)/\tau)}{\sum_{j=1}^{N} exp(s(y, x_j)/\tau)}.$$

The total loss is defined as:

$$\mathcal{L} = \frac{1}{2}\mathbb{E}_{(x,y)\backsim\mathcal{D}}\left[KL(P_{\text{video-to-text}}, Q_{\text{video-to-text}} + KL(P_{\text{text-to-video}}, Q_{\text{text-to-video}})\right],$$

where $\mathcal{D}$ denotes the dataset.

3 Related Works

This chapter introduces related works for Vision-Language Navigation (VLN) and Vision-Language Manipulation (VLM) task that have been benchmarked on the ALFRED challenge.

3.1 End-To-End Models

ALFRED's baseline model is a Convolutional Neural Networks-Long Short-Term Memory (CNN-LSTM) architecture (Shridhar et al. 2020). ALFRED uses a frozen ResNet-18 CNN architecture to extract the image feature and a bidirectional LSTM encoder to extract the text feature. In addition, action-based soft-attention is performed on the language features. Mathematically, given the last hidden state h_{t-1} and the language embedding x, the soft-attention score is computed with

$$\alpha_t = Softmax((W_x h_{t-1})^T x),$$

where a fully-connected layer is applied to the hidden state. The weighted attention distribution is applied to the language again, $\hat{x}_t = \alpha_t^T x$. At the end of each time step, given the image and text features along with the previous action, ALFRED concatenates all the input $[v_t; \hat{x}_t; a_{t-1}]$. An LSTM decoder further outputs a new hidden state for the current step. Two separate networks are trained to predict the action and its corresponding binary mask for the object if an interaction action is predicted.
Episodic Transformer (E.T.) is an end-to-end attention-based transformer model that consists of three modality-specific encoders and one multi-modality fusion encoder, thus having no hidden states (Pashevich, Schmid, and C. Sun 2021).

Figure 3.1 demonstrates the architecture of the E.T. model. E.T. introduces four different encoders. During the training stage of the model, all tokenized language instructions $x_{1:L}$, ego-centric images $v_{1:T}$, and all actions $a_{1:T}$ are first encoded using separate modal encoders. Then, all embeddings are concatenated and forwarded through a two layer multi-modal transformer encoder. The output embedding is forwarded to a simple fc layer and predicts action. In addition, E.T. design a sequence-to-sequence translation task to pre-train the language encoder.

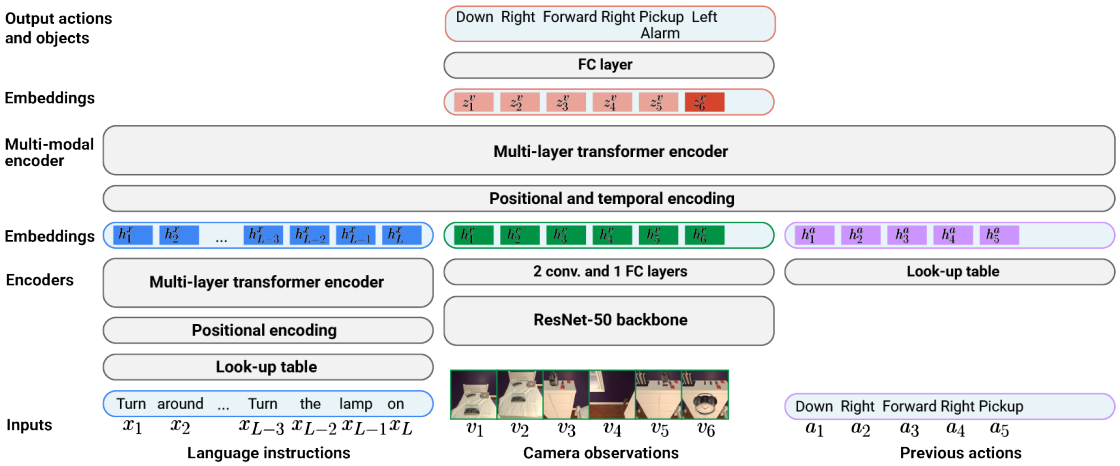

Figure 3.1: Architecture of the Episodic Transformer (Pashevich, Schmid, and C. Sun 2021)

In addition, E.T. also predicts an object list, which has the same length as the action sequence. If the prediction of the current action is navigation, the object prediction head is expected to predict "NoObject." Both action sequence and object sequence are trained using cross-entropy loss.
During inference time, the E.T. agent first takes the entire language instruction, one frame at the beginning as the input. All frames and actions are stored and accumulated after each time step and fed to the model. The last element of the predicted action sequence is used as the action for the next step. If the action belongs to one of the seven interaction actions, E.T. takes the predicted object class at current time step and used a pre-trained Mask R-CNN to predict the object mask.

3.2 Modular Methods

The Hierarchical Language-Conditioned Spatial Model (HLSM) presents a hierarchical approach to address the challenges of the ALFRED benchmark (Blukis et al. 2022). HLSM introduces an observation model to predict a semantic map from the ego-centric RGB image. As shown in Figure 3.2, given an ego-centric RGB view I_t at the time step t, the observation model first predicts the semantic segmentation I_t^S using Mask R-CNN and depth I_t^D using U-Net-based neural networks, which are pre-trained using the data generated from Alfred environment. A pinhole camera model projects both predictions to a semantic point cloud. In the end, this semantic point cloud is binned into a voxel map. This cumulative voxel map serves as the state representation and is updated over time as the agent

explores the environment.

A high-level controller π^H takes this state representation, along with the language

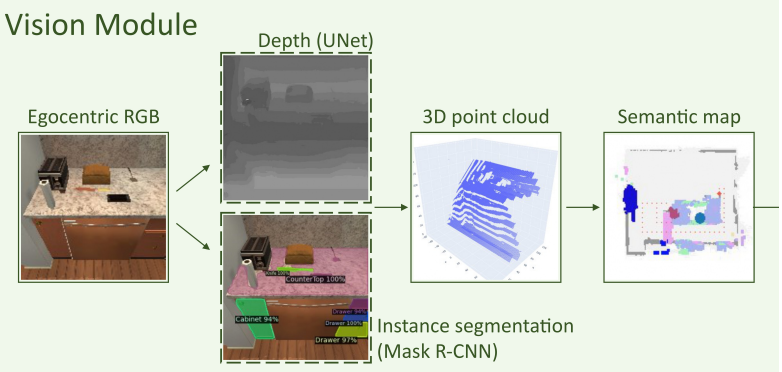

Figure 3.2: Predicting semantic map from ego-centric RGB. (Illustraion from FILM++ (Inoue and Ohashi 2022))

instructions and the subgoal history as the input, to predict the next subgoal (action type, action argument, 3D mask). A low-level controller π^L, predicts the particular action sequence, encompassing navigation and exploration for the agent to achieve the subgoal predicted by the high-level controller.

FILM (abbreviation for Following Instructions in Language with Modular Meth-

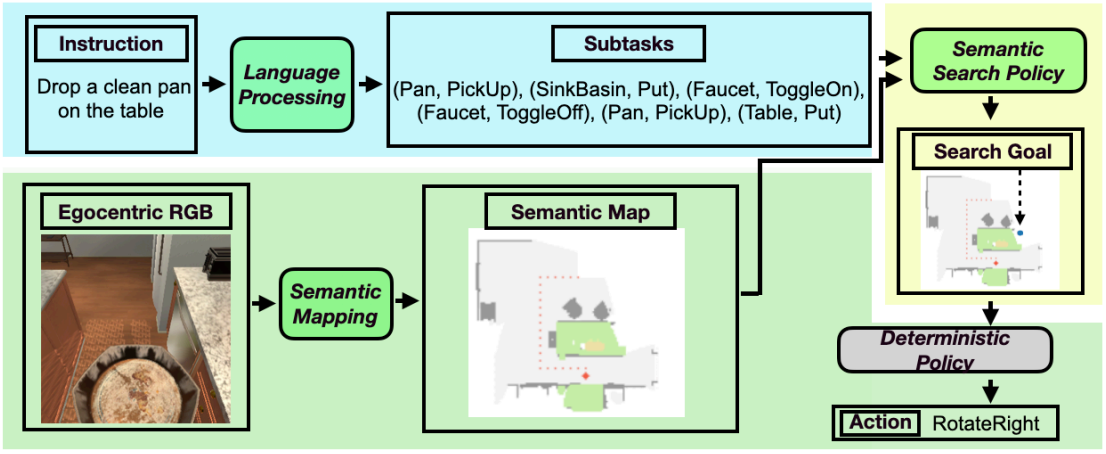

Figure 3.3: Pipeline of FILM approach. (Min et al. n.d.)

ods) proposes a pipeline of three learned modules (Min et al. n.d.). Figure 3.3 illustrates FILM's approach for tackling the Alfred challenge. A language processing module first converts the task description into an action transcript using pre-defined templates, e.g., "Place a black plate with a spoon in it on top of the kitchen table" is converted into [('Spoon', 'PickupObject'), ('Plate', 'PutO-

bject'),('Plate', 'PickupObject'), ('DiningTable', 'PutObject')]. A semantic mapping module processes ego-centric RGB images into a 2d semantic map, following the HLSM approach described above. Furthermore, a semantic search policy predicts the possible distribution of objects' locations. A deterministic policy is designed to predict the action decision based on the predicted action transcript and the predicted semantic map. While looping through the generated action transcript, if the object are found in the semantic map, then the agent will navigate itself to that object via the Fast Marching Method. Otherwise, the agent will use the semantic policy prediction and navigates to the location where the object is most likely located. The pointer of tracking the process of the action transcript will be updated if the interaction action is successfully performed.
FILM++ (Inoue and Ohashi 2022) followed FILM's approach. FILM's semantic search policy is time-consuming in terms of required extra training, especially in preparing the synthetic training data. FILM++ replaces it with prompting to established language models, i.e., BERT. A prompt, "Something you find at [Y] is [X]." is given, and the possibilities of the location are predicted. This modular block is fine-tuned using a less time-consuming method, and the whole model doubles its performance compared to FILM.

4 Dataset

This chapter introduces the ALFRED dataset and our pre-processing stage of generating semantic maps from it.

4.1 Introduction

All experiments are conducted using ALFRED (Action Learning From Realistic Environments and Directives), a simulator, i.e., AI2-THOR (Kolve et al. 2017) generated, the recent benchmark for learning a mapping task, from vision (ego-centric RGB image) and natural language input to an action transcript for domestic service robot (Shridhar et al. 2020). The dataset consists of 7 different household tasks, directly related to real-world scenarios, i.e., Pick & Place, Stack & Place, Pick two & Place, Clean & Place, Heat & Place, Cool & Place, and Examine in light.
84 object classes parameterize all tasks in 120 scenes; all object classes have numbers of individual instances with different colors, textures, etc. The agent needs to execute one action from 12 different actions, which includes five navigation actions (MoveAhead, RotateRight, RotateLeft, LookUp, and LookDown) and seven manipulation actions (Pickup, Put, Open, Close, ToggleOn, ToggleOff, and Slice). This covers the most fundamental actions, and a considerable number of household tasks can be accomplished by combining these 12 actions. All navigation actions are bounded with a discretized value respectively, i.e., 90 degrees for rotating and 25cm for moving ahead.
ALFRED's challenge provides the following annotations: an initial state of the simulated room, language instructions, and an expert demonstration trajectory. The simulated room's initial state consists of the scene's index, along with the environment layouts, i.e., poses and states (e.g., sliced or not) of all moveable objects in this scene. This data is used to set up the environment in the simulator and is not intended to be used as training data. The agent only has access to the RGB observations and does not have information on the environment state during training or the inference. Otherwise, the agent knows the relative position of the objects instead of detecting them using vision models.

Language instructions contain one sentence of global goal instruction, e.g., ”Pick up the alarm clock and turn on the lamp. «goal»”. Moreover, a list of sentences for subgoal instructions: ”Turn left and face the dresser.” -> ”Pick up the alarm clock from the dresser.” -> ”Turn left, look and then face the lamp.” -> ”Turn the lamp on.” -> ”«stop».” The expert demonstration trajectory is the annotated data, which is meant to be fed to the model. The annotation consists of a sequence of discrete actions, the object's mask, whenever the interaction with objects is involved. Figure 4.1 illustrates the frame data and the expert demonstration trajectory of abovementioned task.

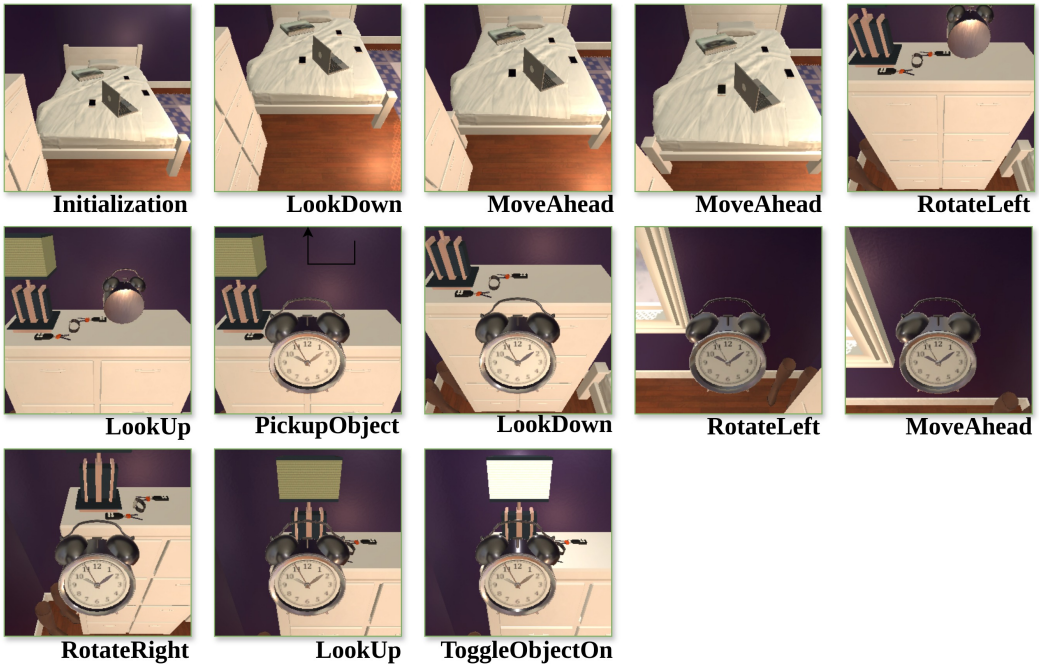

Figure 4.1: Example task: ”Pick up the alarm clock and turn on the lamp.”. ALFRED includes frame and action class at each time step. The object's mask is also given for interaction actions, i.e., PickupObject and ToggleObjectOn.

4.2 Data Distribution

In this section, we reported the data distribution, including the data split, the number of different task types, and the number of low-level actions among all the annotations.
The data splits of ALFRED in the following way: 21,023 annotations for training, 820 annotations for ”valid-seen”, 821 annotations for ”valid-unseen”. ALFRED

dataset is annotated using the Amazon Mechanical Turk (MTurk) service (Shridhar et al. 2020).

Figure 4.2 shows the distribution of different task types. ALFRED maintains

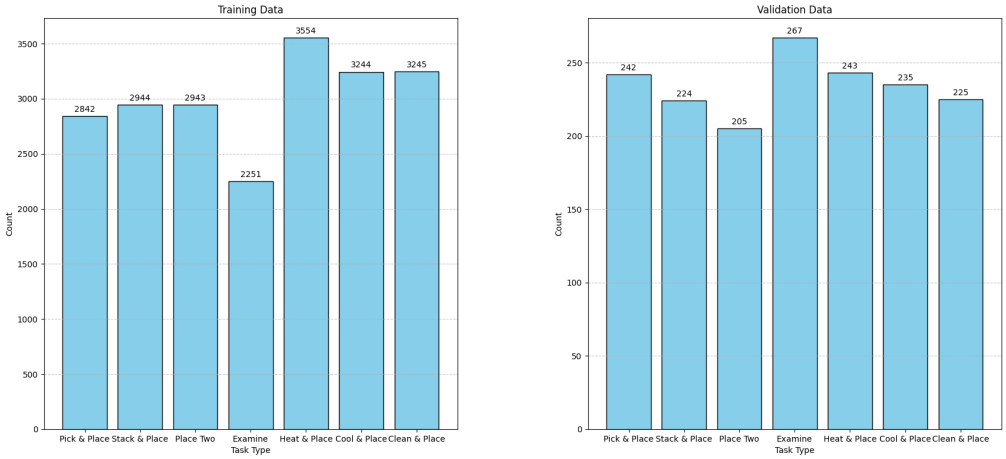

Figure 4.2: Data distribution of the task type in the ALFRED challenge. First histogram shows the distribution of the training data, second shows the distribution of the validation data.

relatively evenly distributed task types among the dataset.

Figure 4.3 displays the action class distribution within the training and validation data. The class index is defined as follows: ”0: LookDown, 1: LookUp, 2: RotateLeft, 3: RotateRight, 4: MoveAhead, 5: PickupObject, 6: PutObject, 7: SliceObject, 8: OpenObject, 9: CloseObject, 10: ToggleObjectOn, 11: ToggleObjectOff.” The histograms clearly show that the actions MoveAhead and LookDown are overrepresented, occupying the majority of the dataset. In contrast, interaction actions are significantly underrepresented, as they typically occur only once or twice within each trial. This underrepresentation poses a significant challenge to achieving effective sequence matching.

4.3 Semantic Map Generation

We followed HLSM and FILM (Blukis et al. 2022; Min et al. n.d.)’s approach for generating semantic maps based on the ground truth data. Given the egocentric RGB image, we use pre-trained U-Net to predict depth and pre-trained Mask-RCNN to perform instance segmentation for each time step. Both pieces of information are integrated into a 3D point cloud. Our semantic map has the

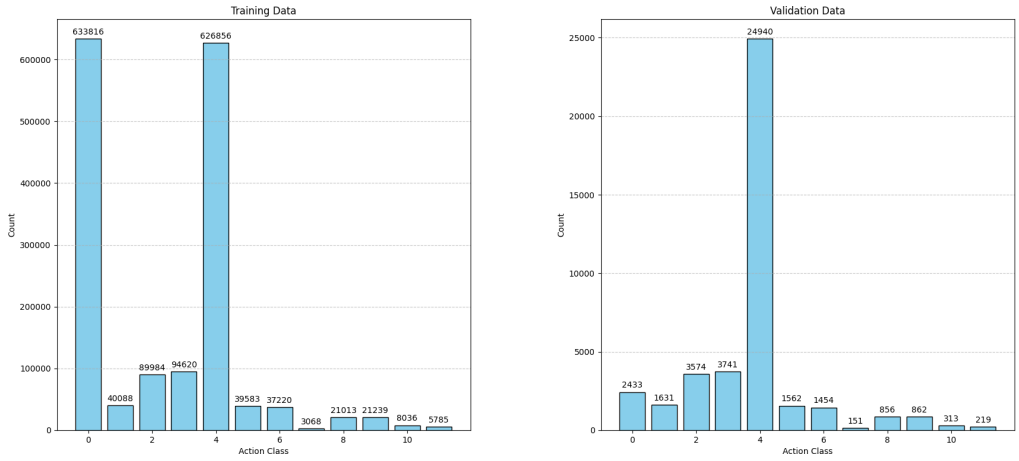

Figure 4.3: Data distribution of action classes in the annotation. First histogram shows the distribution of the training data, second shows the distribution of the validation data.

shape (C + 3) × M × M. Following previous work (Min et al. n.d.), we set M with the value 240, equivalent to a 12m × 12m room in the real world. C represents the semantic channels that occur in the current trial. The additional three channels involved the obstacle map, explored map, and pose of the agent. Unlike previous works that utilize the map information inside their modular approaches, our approach considers the map as a modality, the same as the visual and text input. Under these circumstances, the current location and orientation give extra information to this map. We represent the pose with a circle with a diameter of 5 pixels. The orientation is represented with a line starting from the circle's center with 4 pixels. This pose representation is coarse but sufficient as the orientation is constrained to only four directions in the simulator. We show a trial with the ego-centric RGB sequence and the corresponding semantic map sequence in Figure 4.4. This allocentric map is initialized with zero (white space) and accumulated after each time step. The dark gray color represents the obstacle, and the light gray color represents the explored places (All seen places, which are non-object nor obstacle, is considered as explored). We use random different colors to represent different object classes.

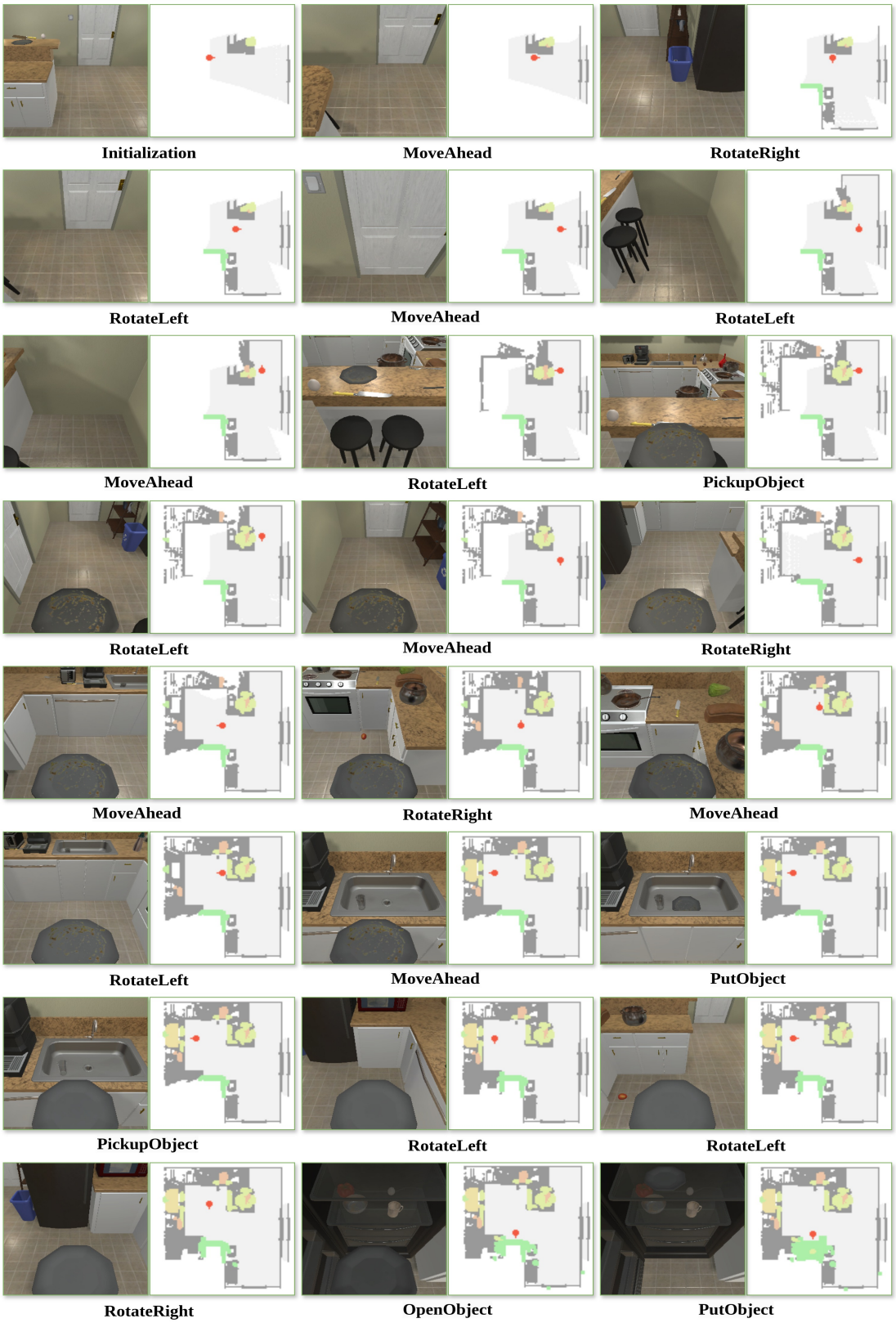

Figure 4.4: An example of generating semantic map from ego-centric RGB input.

5 Main Approach

This chapter discusses the details of our main approach. We follow and modifiy the model architecture of the Episodic Transformer and the primary approach from FILM/FILM++ to construct a semantic map for each trial.

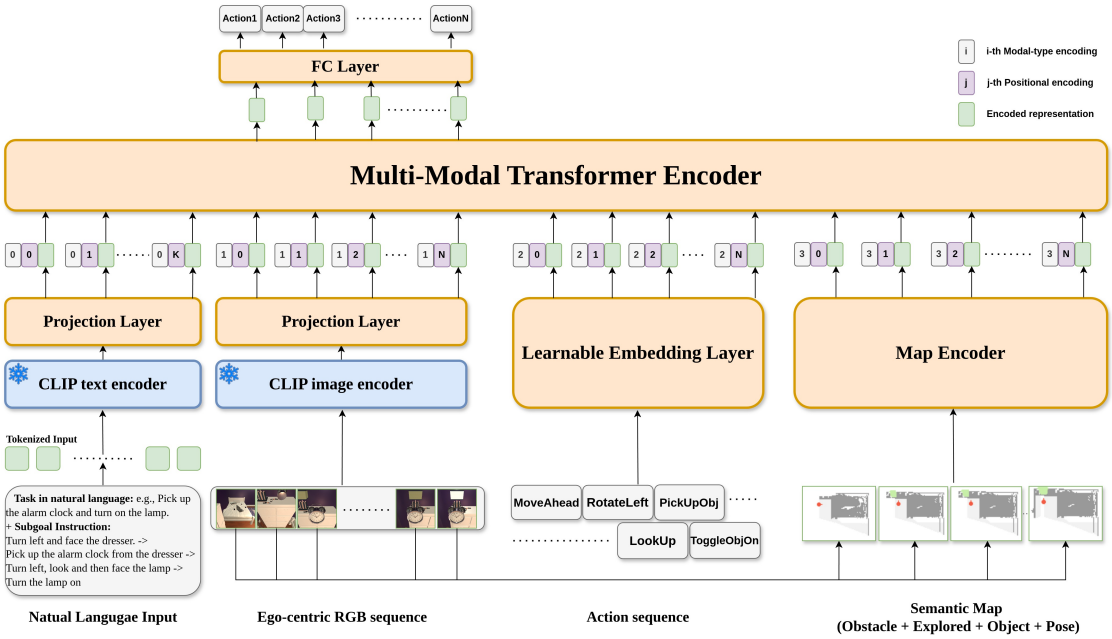

Figure 5.1: Model architecture of LIAM. The blue blocks are all layers that were frozen during the end-to-end model training; the orange blocks are the parts that were trained. After adding the positional encoding and a learnable modal-type encoding, a multi-modal fusion layer fuses representations from different latent spaces. A fully-connected layer is employed to predict the matching action sequence based on the input.

Figure 5.1 illustrates the model architecture. We have adopted E.T.'s idea of encoding different modalities separately, replacing the image and text encoders with CLIP encoders. To manage the high computational cost, we freeze the weights and biases of the CLIP backbone during model end-to-end training, ensuring efficient resource utilization. A pre-training stage is introduced to pre-align the embedding space of different modalities. We also design a simple projection head to project

the embedding. This learnable autoencoder-wise head is learned to make further adjustments to the embedding. We use a single embedding layer to map the action sequence to the hidden dimension. The last modality is the semantic map. We follow FILM's approach (Min et al. n.d.) and build a semantic map with chosen information: the obstacle map, explored area, semantic map for objects, and the pose (agent's position and orientation). A map encoder is introduced to encode the semantic maps from each time step. After all modalities are embedded in their own latent space using the independent encoder, we first apply positional encoding and modal-type encoding to each representation, then concatenate all the representations, and feed them further to a multi-modality fusing layer to learn a general representation. A causal attention mask is also applied in this stage. This layer plays a crucial role in learning a general representation by fusing the representations from different latent spaces. Ultimately, we employ a fully connected layer to predict an output sequence based on all the tokens of visual representations, which have the same number of mappings of actions, so that we can train the model with the ground truth action sequence. Details of model design and training are explained below.

5.1 Pre-training Stage

5.1.1 Contrastive Alignment

CLIP was initially trained using data gathered from the Internet, resulting in bias and significant performance differences across different datasets in zero-shot inference. For instance, while CLIP excels on datasets related to cars and food, it struggles on simpler datasets like MNIST compared to the supervision approach (Deng 2012; Radford et al. 2021). This discrepancy can be attributed to the rarity of handwritten digits on the Internet. Given that the ALFRED dataset is simulator-generated, it deviates slightly from reality. Hence, a pre-training stage to align the latent space of image and action becomes imperative.

We first pre-align the CLIP image encoder and action embedding via contrastive learning. The learning objective of the model is to match the correct pair of action embedding and image embedding. Unlike CLIP, where one image has one corresponding label, in our case, two consecutive images have one corresponding label, which denotes an action. Figure 5.2 shows an example of the ground truth of one mini-batch. Unlike the original CLIP, which forms an $N \times N$ affinity matrix (N denotes the batch size), our ground truth matrix cannot maintain symmetry based on its data nature, i.e., the action column has a small fixed size. It requires

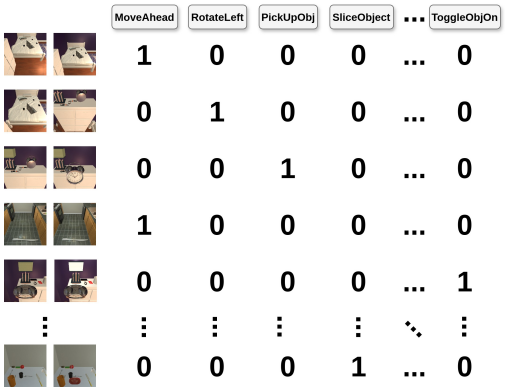

Figure 5.2: An example of the ground truth from one mini-batch for alignment of visual and action embedding space. One action corresponds to two consecutive frames.

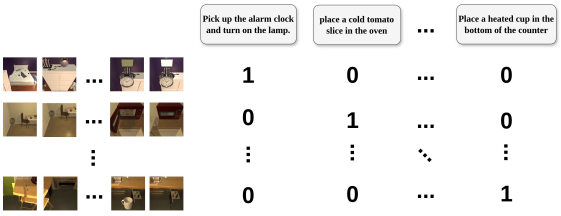

Figure 5.3: Ground truth for alignment of visual and language embedding space. One frame sequence corresponds to one language instruction.

an extra data processing stage to choose exactly 12 different actions for each mini-batch to form the ground truth similar to CLIP. Our approach counts the number of unique actions that appear in each mini-batch instead and forms all occuring actions as the column of the matrix. Notice that the number of columns differs from each mini-batch, and is no longer necessarily the same as the number of the action classes. Each row denotes the representation of two consecutive frames. The matrix entry (i,j) has the value one if the i-th frames representation corresponds to the action class j.

A common approach for computing the representation of a video sequence can be as straightforward as using only a parameter-free approach, namely computing the mean value of the representations of all frames. Another possibility is using a learnable network to learn the fusion of multiple representations. In our approach, we use f$(\cdot,\cdot)$, a 1D convolution layer along with global average pooling to compute a fused representation of two consecutive images. The action embedding follows E.T.'s look-up table approach, using an embedding layer that embeds all 14 classes (12 classes + 2 special tokens) into 768-dimensional dense vectors. With the help of this embedding layer, all discrete action classes are encoded into continuous vector space representations. After having the normalized representations of both frames I_e and action classes A_e, we compute a cosine similarity score of both vectors and fine-tuned the CLIP backbone following CLIP's approach. With $a_i \in A_e, I_t \in I_e$:

$$P_{I2A}(I) = \frac{exp(f(I_t, I_{t+1}), a_i)/\tau)}{\sum_{j=1}^{N-1} exp(f(I_t, I_{t+1}), a_j)/\tau)}$$

$$P_{A2I}(A) = \frac{exp(a, f(I_{t_i}, I_{t_i+1}))/\tau)}{\sum_{j=1}^{N-1} exp(a_j, f(I_{t_i}, I_{t_i+1}))/\tau)}$$

We notate text, action, and image with alphabet T, A, and I. I2A denotes "image-to-action," and all later mentioned abbreviations (X2X) follow this pattern. Function f$(\cdot, \cdot)$ denotes the fusion function of two consecutive images at time step t and t + 1 as described above. τ is the temperature, a learnable parameter scales the similarity score. Following CLIP, we initialize the temperature with a value of 0.07 and use 100 as an upper threshold to clip the temperature value during the training stage, preventing the scaling factor from becoming too large (Wang, Xing, and Yong Liu 2021) and 0.01 as a lower threshold bound to prevent dividing by zero error.

Following ActionCLIP (Wang, Xing, and Yong Liu 2021), we use the KL divergence loss to fine-tune the CLIP backbone. The total loss of the contrastive alignment pre-training is the average of both image-to-action and action-to-image loss.

$$\mathcal{L}_{\text{Image-Action-Total}} = \frac{1}{2}\mathbb{E}_{(x,y)\backsim\mathcal{D}}\left[KL(P_{\text{I2A}}(I), Q_{\text{I2A}}(I)) + KL(P_{\text{A2I}}(A), Q_{\text{A2I}}(A))\right]$$

, $Q(\cdot)$ denotes the ground truth similarity matrix.

5.1.2 Triple Contrastive Pre-training

Our contrastive learning pre-aligns the latent space of the vision and action embedding. However, the alignment of the language embedding space with other latent spaces is thus neglected. To address this, we introduce a triple contrastive pre-training stage. This method not only aligns the vision and action embedding spaces but also aligns the language embedding space with them. This mutual training ensures that the loss can be back-propagated to all encoders. In addition to Figure 5.2, Figure 5.3 shows the example of the ground truth of aligning the CLIP text encoder and image encoder. The training objective is straightforward: the representation of the language instruction and its corresponding frame sequence should be similar to each other. Figure 5.4 illustrates the basic idea of the triple contrastive alignment. We first compute the representation of every

single frame (N frames) using the CLIP image encoder and further process this sequence of representations in two ways.
Considering the computing resource for the image sequence representation, we choose the parameter-free approach to compute the mean value of all CLIP embeddings of all frames to get the sequence representation. For every two consecutive frames in this list, we use the 1D convolution layer to get the representation of both frames as the same methodology described above 5.1.1. Thus the visual embedding space is aligned simultaneously with both text and action embedding space. As shown below, we define cross-entropy loss $\mathcal{L}_{\text{Text-Image}}$ for learning the pairing of visual and text sequences following original CLIP. For aligning action and image, we continue using KL divergence loss $\mathcal{L}_{\text{Image-Action}}$.

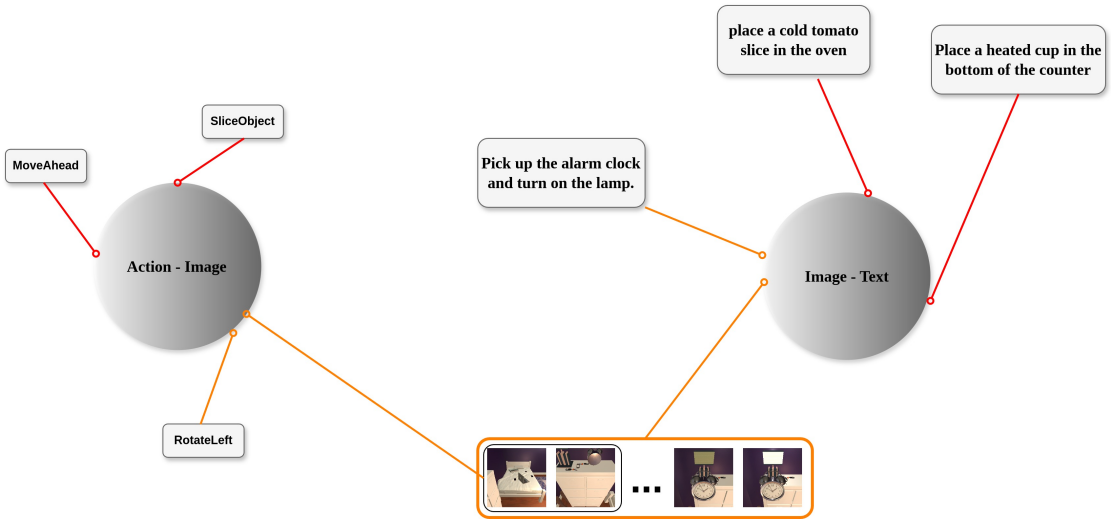

Figure 5.4: An illustration of the triple contrastive alignment. The representation of the whole frame sequence should be pulled together with the representation of its corresponding global language instruction. At the same time, the representation of every two consecutive frames should be pulled as close as possible to the representation of its corresponding action. The representation of frames is outputted by the same visual encoder, which serves as a **"bridge"** in between.

$$\mathcal{L}_{\text{Text-Image}} = -\frac{1}{2N}\left(\sum_{i=1}^{N} log\left(\frac{exp(S_{ii}/\tau)}{\sum_{j=1}^{N} exp(S_{ij}/\tau)}\right) + \sum_{j=1}^{N} log\left(\frac{exp(S_{jj}/\tau)}{\sum_{j=1}^{N} exp(S_{ij}/\tau)}\right)\right)$$

$$\mathcal{L}_{\text{Image-Action}} = \frac{1}{2}\mathbb{E}_{(I,A)\backsim\mathcal{D}}\left[KL(P_{\text{I2A}}(I), Q_{\text{I2A}}(I)) + KL(P_{\text{A2I}}(A), Q_{\text{A2I}}(A))\right]$$

The total loss of triple contrastive alignment is defined as:

$$\mathcal{L}_{\text{total}} = (1 - \alpha) \times \mathcal{L}_{\text{Text-Image}} + \alpha \times \mathcal{L}_{\text{Image-Action}}$$

We introduce α as a hyperparameter for weighing both losses. This hyperparameter gives us more flexibility. Furthermore, each sample in a batch is a whole sequence. Three or four sequences-constructed mini-batch consists of averages of over 200 actions. With the computational resource constraints, we can only set a tiny batch size for triple contrastive alignment. However, the objective of pairing the correct image and text sequence is now more leisurely among each mini-batch. Therefore we assign a small weighted $\mathcal{L}_{\text{Text-Image}}$ in the total loss ($\alpha = 0.8$).

5.2 Projection Layer

We introduce an autoencoder-like design for our projection layer. CLIP-encoded features have a hidden dimension of 512, as shown in figure 5.5 (left). We then downsample the input feature to a smaller hidden dimension (384 in our case) and apply the Gaussian Error Linear Units (GELU) as the nonlinear function (Hendrycks and Gimpel 2016). GELU has recently become a more favorable choice for nonlinear functions, especially in transformer-based models. The hidden representation is then passed to another fully connected layer, which maps the feature dimension to 768. To ensure stability, we employ layer normalization and include a residual connection in our design.

We have two separate projection heads with the same architecture for text and image encoders. The map encoder (see Figure 5.5 Right) has different structures. In consideration of a good tradeoff of the computation resource, we use the familiar process of global adaptive average pooling to downsample the map tensor. A fully connected layer with the non-linear function GELU project downsamples the map to the same feature dimension as other modalities' embeddings (768).

5.3 Multi-modal Transformer

The multi-modal transformer is a comprehensive model that aims to integrate all different modalities into one global representation, namely language, visual frames, action sequence, and semantic map in our case. It utilizes a two-layer transformer encoder with 12 attention heads. We use causal attention following E.T. (Pashevich, Schmid, and C. Sun 2021). The attention mask follows these rules: Language tokens should attend to the language itself. Visual frames are

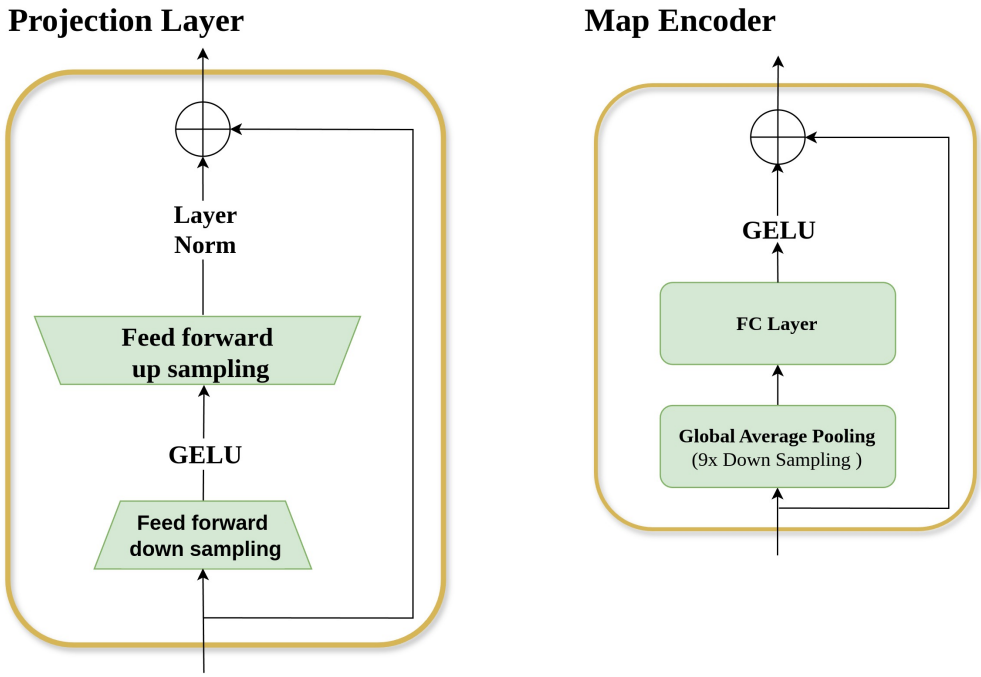

Figure 5.5: Left: Projection head appends after both CLIP encoders. Right: Map encoder encodes the current map and outputs a representation.

allowed to attend to all language tokens, but they can only attend to all the frames, actions, and semantic maps before current time step t. The same rules are applied to action tokens and semantic maps. Sinusoidal positional encoding is also applied.

In addition, we add a learnable modal-type encoding, first introduced by VILT (W. Kim, Son, and I. Kim 2021). This encoding brings extra information and clarifies different modality types to the Transformer model; it also avoids the need for explicit separation tokens like "[SEP]." The modal-type encoding is trainable and expect to give the fusion model the ability to distinguish different modalities.

5.4 Zero-shot Semantic Segmentation

The ALFRED challenge also requires us to predict the object's mask during inference. Following FILM/FILM++ (Inoue and Ohashi 2022; Min et al. n.d.), we use a zero-shot approach instead of training an extra model for prediction. For a given task, we first predict all the objects occurring in this trial, including 1. The target object: objects with which the agent will interact. 2. The receptacle object, e.g., table, fridge, etc. 3. The parent objects are all moveable receptacle objects,

e.g., Plates. The parent object class does not have interaction with the receptacle object class. 4. ToggleObject, e.g., Light switch. 5. Knife, which depends on whether the slice action is predicted in the sequence. All object categories are predicted by their own pre-trained BERT model (Kenton and Toutanova 2019). All 5 BERT models have the same architecture with a different classification head. An additional BERT is used to predict the task type out of all seven tasks in ALFRED. Given all the prediction objects, an action transcript is generated based on the task type. For example, a high-level action transcript for the task "Examine a remote control by the light of a floor lamp." is ('RemoteControl', 'PickupObject'), ('FloorLamp', 'ToggleObjectOn'). We start from the beginning and use a pointer to track progress.
We utilize CLIPSeg (Lüddecke and Ecker 2022) for zero-shot inference to predict the object mask. In the example mentioned above, the open vocabulary input for the first step into the CLIPSeg model is the "RemoteControl" solely. We update the transcript tracker whenever the simulator tells us this action has been predicted and performed successfully. This part is also the only deterministic policy we used during the inference time. A potential challenge is that the CLIPSeg model accepts input size with shape (352, 352), and the ego-centric RGB image in the AI2Thor has shape (300, 300). We use bilinear interpolation to convert the image size.

5.5 LIAM Pipeline Walkthrough

In this section, we define our input and output mathematically to explain our approach more deeply. Given input:

$$\begin{aligned} \text{Tokenized language instruction} &: [L_1, \ldots L_m] \\ \text{EGO-centric RGB input} &: [I_1, ..., I_n] \\ \text{Action sequence} &: [A_1, ..., A_n] \\ \text{Semantic map sequence} &: [M_1, ..., M_n] \end{aligned}$$

Notice that language tokens have length m, and all other modalities have length n. For image sequence $I_1, \ldots, I_n$, only n - 1 actions are involved. We append "«stop»" as the last action. Four different encoders encode all input to their latent space

and output F_l, F_v, F_a &F_m as the feature representation.

$$\begin{aligned}
&\overline{F}_{mod} = F_{mod} + mod^{Type} + mod^{Pos}, \text{ for mod in L, I, A, M.} \\
&F^0_{total} = [\overline{F_L}; \overline{F_I}; \overline{F_A}; \overline{F_M}] \\
&\overline{F}^l_{total} = LN(MA(\overline{F}^{l-1}_{total})) + \overline{F}^{l-1}_{total}, \qquad l = 1, ..., \mathbb{L} \\
&\overline{F}^l_I = \overline{F}^d_{total}[m+1 : m+1+n] \\
&O = \overline{F}^d_I D
\end{aligned}$$

After applying the modal-type and positional encoding to each feature representation of different modalities, we concatenate all the representations to F^0_{total}. The multi-modal Transformer encoder consists of $\mathbb{L}$ layers, and each layer of encoder blocks consists of Multi-headed Attention (MA), Layer Normalization (LN), and residual connection with the output of the previous layer. Finally, we slice the total representation and take the representation of the visual part, namely from index m + 1 to m + 1 + n, for the final action sequence prediction. The predicted action sequence is computed by multiplying the sliced matrix with a trained linear layer D.

The end-to-end model is trained using cross-entropy loss between the generated action sequence and the ground truth action sequence. For predicted action sequence P = $[\hat{a_1} \dots \hat{a_n}]$ and ground truth Q = $[a_1 \dots a_n]$, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{action}} = -\sum_{i=1}^{n}\sum_{c=1}^{13} a_{i,c} log(\hat{a_{i,c}}),$$

where c represents the action class out of 14 classes (12 classes + «stop» + «pad»). However, the «pad» token is masked out during the computation of the loss value. Following E.T. (Pashevich, Schmid, and C. Sun 2021), we also use auxiliary losses in the training. First, we predict the object class for each frame ("NoObject" is also a class). The predicted object class is a list with the same length of the action. Similary to the action, for predicted object sequence $P_o = [\hat{o_1} \dots \hat{o_n}]$ and ground truth Q = $[o_1 \dots o_n]$, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{object}} = -\sum_{i=1}^{n}\sum_{c=1}^{85} o_{i,c} log(\hat{o_{i,c}}),$$

where c represents the object class out of 85 classes (84 object classes + NoObject). We also predict the goal progress with the initial thoughts for giving the model a better understanding of the current progress of the current trial. The ground

truth Q^{gp} of the goal progress is computed by .

$$Q^{gp} = \left\{ \frac{i+1}{n} \mid i \in \{0, 1, 2, \ldots, n-1\} \right\},$$

where n is the length of the ground truth action for current trial. For predicted goal progress P^{gp}, we use Mean Square Error (MSE) loss, defined as:

$$\mathcal{L}_{gp} = \frac{1}{n} \sum_{i=1}^{n} (Q_i^{gp} - P_i^{gp})^2$$

The total loss is then defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{action} + \alpha \mathcal{L}_{object} + \beta \mathcal{L}_{gp}.$$

where α, β are hyperparameters for weighing the auxiliary loss. We set all weights to 0.1 in our experiments.

5.6 Implementation Details

The models are implemented using the Pytorch package (Paszke et al. 2019) and trained on NVIDIA RTX A6000 (50.3 GB memory). Table 5.1 shows the detailed hyperparameters we used for the training. These hyperparameters are optimized for our training using the optuna package (Akiba et al. 2019). We use a two-layer transformer encoder with eight heads for the model architecture. We use a trivial 768 as our hidden dimension. For training our end-to-end model, we use batch sizes 32 and 2 as the accumulation step. The accumulation step is the gradient accumulation technique, namely we backpropagate the gradient for 64 samples (2 mini-batches). We use the learning rate 1e-4 and AdamW (Loshchilov and Hutter n.d.) as our optimizer. We adapt this learning rate using a cosine annealing schedule (Loshchilov and Hutter 2022). We train each model with a maximum of 20 epochs. Some models start to overfit after 16 epochs, so we apply the early stop technique and pick the best result for each model.

Hyperparameters	**Value**
Multi-modal Transformer Encoder	
Num_heads	8
Num_layers	2
Hidden_dim	768
Dropout ratio	0.1
Pre-Training	
Batch size (I2A)	64
Batch size (I2A + T2A)	3
Length Sequence (I2A + T2A)	21
Learning rate	5e-5
Scheduler	CosineAnnealingLR
Optimizer	AdamW
Epochs	11
End-to-End Training	
Batch size	32
Accumulation step	2
Learning rate	1e-4
Optimizer	AdamW
Scheduler	CosineAnnealingLR
Epochs	20

Table 5.1: Implementation details

6 Evaluation

In this section, we introduce the metrics that we used to evaluate our model and analyse the results. All experiments are conducted using ALFRED.

6.1 Evaluation Metrics

For the pre-training stage, we measure the percentage of correct matching of the action and the visual representations (I2A) and the percentage of correct matching of the text and visual sequence representations.
For the ALFRED challenge, we first evaluate the generated action sequence. We report the accuracy score, the element-wise correct prediction percentage of the generated action sequence, and the ground truth sequence. The precision score computes the percentage of the correctly predicted class (True Positives TP) among all predicted classes as positives (True Positives TP + False Positives FP),

$$\text{Precision} = \frac{TP}{TP + FP}.$$

The recall score computes the percentage of the correctly predicted TP among the sum of true positives and false negatives, namely the sum of this class in the ground truth,

$$\text{Recall} = \frac{TP}{TP + FN}.$$

The F1 score is the harmonic mean of the precision and the recall score, defined as

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

We compute the average precision score among all different classes.
We also report all the metrics that ALFRED has used for the leaderboard:

1. Success Rate (**SR**) measures the number of successfully performed tasks among the seen and unseen data.

2. The post-conditioned success rate (**GC**) measures the proportion of successfully performed subtasks for each episode.

3. Because of the vast difference in the length of each task among the dataset (e.g., Heat & Place is a more challenging task compared to Pick & Place, which requires longer steps to complete), there is a path-length weighted version for SR and GC, respectively (**PLWSR, PLWGC**). We first evaluate the model performance on the validation data.

We use "valid_seen" as our validation dataset, which contains tasks that the model has seen during training, and "valid_unseen" as our test dataset, which contains tasks that the model has never encountered during training. The environment in unseen data has never appeared in the training set.

6.2 Pre-training Stage

We pick three different CLIP backbones: MobileCLIP-S0 (54.7 million parameters), OpenCLIP with Resnet-50 backbone (114.1 million parameters), and OpenCLIP with the vision transformer backbone (152.7 million parameters).

Models	**Validation**		**Test**	
	ACC (I2A)	ACC (T2I)	ACC (I2A)	ACC (T2I)
Contrastive alignment of image and action				
MobileCLIP S0 (Zero-shot)	7.51	-	7.32	-
MobileCLIP S0	38.34	-	43.21	-
OpenCLIP-RN50 (Zero-shot)	5.63	-	5.51	-
OpenCLIP-RN50	61.41	-	58.78	-
OpenCLIP-ViT-B-32 (Zero-shot)	5.84	-	8.26	-
OpenCLIP-ViT-B-32	**65.45**	-	**61.58**	-
Triple contrastive alignment				
MobileCLIP S0 (Zero-shot)	3.21	35.04	3.34	33.03
MobileCLIP S0	80.28	60.22	**68.97**	35.95
OpenCLIP-RN50 (Zero-shot)	3.76	31.51	4.78	34.61
OpenCLIP-RN50	70.56	32.67	65.54	**51.82**
OpenCLIP-ViT-B-32 (Zero-shot)	3.89	45.62	5.67	37.65
OpenCLIP-ViT-B-32	**88.22**	**75.43**	65.98	40.69

Table 6.1: The matching accuracy of correct image to text (I2A) pair and correct text to image (T2I) pair.

Table 6.1 presents the accuracy score of correct matching pairs among the mini-batches. Firstly, we can clearly see that no CLIP model has the ability to correctly

match two consecutive visual images with their corresponding actions in a zero-shot manner. As discussed before, CLIP is a model that is sensitive to the dataset. However, both image-action contrastive and triple contrastive alignment can be adequately trained and show significant accuracy gains in matching corresponding pairs of two consecutive images and action or pairs of the image sequence and text sequence. We use batch size 64 for image-action contrastive alignment training, batch size three, and additionally, constraint each sequence to length 21 (during validation) for triple-contrastive alignment to keep a fair comparison between both accuracy scores. We observed that using zero-shot inference, the accuracy for text sequence and image sequence matching is around 33%, and the accuracy for two image frames and action matching lies between 3% and 8%. Both scores denote the random guessing from the model. After fine-tuning, the accuracy of predicting the correct image and text sequence pair is increased using MobileCLIP backbone and CLIP VIT-B-32 backbone to 60.22% and 45.62%, however still not entirely correct out of the mini-batch.
Furthermore, we observe that the accuracy score for unseen data is still low compared to the random guess, indicating an overfitting problem in the sequence matching task. This result suggests that the model is not generalizing well to unseen data. The potential reason for this could be the suboptimal mean operation for computing the sequence representation. Surprisingly, CLIP with ResNet-50 backbone has a relatively higher score in the sequence matching task but a lower score in the action and visual matching. However, we observe that triple-contrastive alignment indeed benefits accuracy in matching the correct action label of the consecutive images using all three backbones. Overall, based on its performance in the action-image matching task, we decide to use pre-aligned CLIP VIT-B-32 as the backbone to extract the features of our input.

6.3 ALFRED Challenge: Quantitative Results

Before running the inference code on the ALFRED challenge, we evaluate the generated action sequence using the abovementioned metrics compared to the ground truth action sequence and find the optimal model architecture and design choice with the best performance. Table 6.2 demonstrates the result using three different CLIP backbones, i.e., the pre-trained model from the official repository, which does zero-shot on our dataset, contrastive pre-aligned in the image and action space, and triple contrastive pre-aligned backbone. For each frozen backbone, we evaluate the prediction using 1. the E.T.-like baseline model 2. Adding auxiliary loss as discussed in ref and 3. Adding semantic maps as an additional modality.

4. Adding both 2 and 3.

Models	Valid Seen				Valid Unseen			
	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
Zero-shot CLIP encoder	73.00	68.69	56.29	58.64	71.87	58.13	48.82	49.31
weighted loss	64.51	-	-	-	55.78			
+ Aux. Loss	76.92	80.77	66.66	70.51	75.67	70.49	59.92	62.20
+ Semantic Map	78.08	78.35	**68.81**	71.69	76.49	65.17	58.35	59.54
+ Semantic Map & Aux. Loss	**78.43**	**82.05**	68.07	**72.46**	**76.59**	**71.65**	**60.16**	**62.74**
Contrastive-aligned CLIP	76.97	**71.30**	**63.60**	**64.17**	79.56	49.95	42.94	44.02
+ Aux. Loss	76.80	51.41	43.89	44.62	79.71	**51.41**	**43.89**	**44.62**
+ Semantic Map	**79.30**	61.73	55.81	56.94	**81.51**	43.49	39.32	39.52
+ Semantic Map & Aux. Loss	78.03	61.72	53.78	56.10	80.10	43.60	37.40	38.66
Triple-contrastive-aligned CLIP	77.52	**84.33**	66.73	72.15	76.89	73.12	62.06	64.57
+ Aux. Loss	76.68	82.03	65.23	69.96	75.64	66.06	55.63	57.54
+ Semantic Map	**77.98**	77.04	**70.90**	**72.63**	75.79	**69.35**	**67.54**	**67.02**
+ Semantic Map & Aux. Loss	77.97	82.43	66.35	71.39	**76.97**	68.69	57.87	60.02

Table 6.2: We report the accuracy, precision, recall, and F1 score for our end-to-end model predicted sequence and the ground truth.

From all the experiments, first, we observe that the fine-tuning of the CLIP backbone brought benefits in increasing the accuracy, precision, recall, and F1 score. (baseline model without using auxiliary loss and map as modality). Although the improvement in the accuracy is not significant, e.g., 73.00 %, 76.97%, 77.52% in the seen data, 71.87%, 79.56%, 76.89% in the unseen data, respectively. The improvement in F1 score is relatively huge, 8% in the seen data, and around 15% improvement in unseen data, when the triple contrastive pre-alignment is performed. Second, we observe that the auxiliary loss only brings an advantage when we use the non-fine-tuned CLIP backbone. The auxiliary damages the model's training and worsens performance for both pre-aligned CLIP backbones. Third, the semantic map improved the model's performance for the zero-shot backbone and triple-aligned backbone, showing the potential of using the map as an extra modality to enhance the model's spatial understanding rather than solely given the ego-centric RGB images. The backbone fine-tuned with image-to-action alignment has an acceptable accuracy score but a relatively low F1 score; this indicates that the model still suffers from the imbalanced data problem. As the dataset is skewed towards the action classes, we also explore the effect of using weighted cross-entropy loss. We assign a weight to each class based on its share of the total data. However, this method intensifies the overfitting problem, resulting in a model that is highly challenging to train (strongly fluctuated loss curve). Based on these results, we select **Triple-contrastive-aligned backbone + Semantic Map modality** as our model for further experiments on the ALFRED challenge.

Action Class	Valid Seen			Valid Unseen		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Accurate predictions						
MoveAhead	75.60	98.85	85.66	75.90	98.68	85.79
CloseObject	99.06	93.21	95.85	97.93	97.95	**97.86**
ToggleObjectOn	99.11	85.39	91.02	96.81	92.35	94.15
ToggleObjectOff	100.00	97.19	**98.41**	81.25	80.62	80.92
«stop»	100.00	92.13	95.77	99.63	94.51	96.85
Moderate Predictions						
LookDown	90.22	52.43	65.99	91.16	59.99	71.87
OpenObject	80.19	72.93	75.79	60.59	74.38	65.51
LookUp	89.55	66.88	76.20	85.38	54.39	65.53
PickupObject	91.76	69.24	78.50	91.33	65.99	76.18
PutObject	89.30	70.98	78.65	84.13	69.02	75.09
Poor predictions						
RotateLeft	63.46	15.29	24.44	41.81	12.39	18.93
RotateRight	55.54	17.51	26.46	48.48	14.40	21.70
SliceObject	51.98	40.00	41.85	30.73	27.08	27.96

Table 6.3: We evaluate the class-wise score from the best model above, clustered in 3 performance classes. Accurate predictions: These classes have a relatively high F1 Score (over 85%). Moderate Predictions: Classes with high precision and moderate recall scores, indicates a significant number of false negative predictions. Poor predictions: Classes with moderate precision and very low recall scores indicate that the model struggles to predict these classes.

We investigate each action class's precision, recall, and F1 score individually (Table 6.3). We cluster all class predictions in 3 performance classes. "MoveAhead", "ToggleObjectOff", "ToggleObjectOn", " «stop»", and "CloseObject" are five classes that the model is good at predicting. Surprisingly, as the navigation actions dominate the action classes, especially MoveAhead, the model still has the ability to correctly predict some action classes and not simply predict those dominant classes to gain a high accuracy score. The classes in which the model has moderate performance have typically a high precision but moderate recall score. This result suggests the model predicts few false positives; however, a significant number of false negatives. In other words, when the model predicts these classes, they are usually reliable, but they are still missing many predictions among whole dataset. "RotateLeft", "RotateRight" and "SliceObject" are three classes with

moderate precision score and meager recall score. The reason for the lousy prediction on "SliceObject" comes down to the lack of examples. However, the rotation action is worse predicted and needs further investigation. Our possible explanation would be that ALFRED data hard parametrized the rotation action with 90 degrees, which means that in many cases (when the furniture is not big enough to spot after turning 90 degrees). We only have two non-correlated frames next to each other, which might seem completely random to the model. In addition, the evaluation of both seen and unseen data is similar, denoting the generalizability of our model.

Table 6.4 reports the result of the ALFRED challenge, comparing our approach to the related works. Our approach can solve specific tasks as the GC scores reported; however, it did not have a successful case for the whole task. From our observation of runs in the simulator, our model wanders around in the environment most of the time but does not make rational decisions for the whole task. It can understand the preceding part of the language instructions by doing the first couple instructions but still lack of ability to comprehend the whole instructions. Our assumption for this behavior is that we are still not incorporating the map information correctly into the LIAM model, or the multi-modal Transformer encoder did not learn an excellent way to fuse all modality information.

Approaches	**SR**		**PLWSR**		**GC**		**PLWGC**	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
Ours	0.00	0.00	0.00	0.00	0.060	0.067	0.069	0.064
E.T. (Pashevich, Schmid, and C. Sun 2021)	**28.77**	5.04	-	-	**36.47**	15.01	-	-
HLSM (Blukis et al. 2022)	25.11	16.29	6.69	4.34	35.79	27.24	11.53	8.45
FILM (Min et al. n.d.)	25.77	**24.46**	10.39	**9.67**	36.15	**34.75**	**14.17**	**13.13**

Table 6.4: Evaluation on ALFRED Challenge

6.4 Qualitative Results

In this section, we demonstrate some qualitative examples of the agent's inference inside the simulator.

Figure 6.2 shows the decision of our agent for the task "Pick up the basketball and turn on the desk lamp in the bedroom." The agent can comprehend most of the sub-goal instructions and correctly find and pick up the basketball at the foot of the bed. Furthermore, the agent rotates left twice and goes to the desk. However, this task eventually failed because of the model's wrong prediction of the lamp's location (which should be on the other side of the desk).

We show examples of using the CLIPSeg (Lüddecke and Ecker 2022) model in

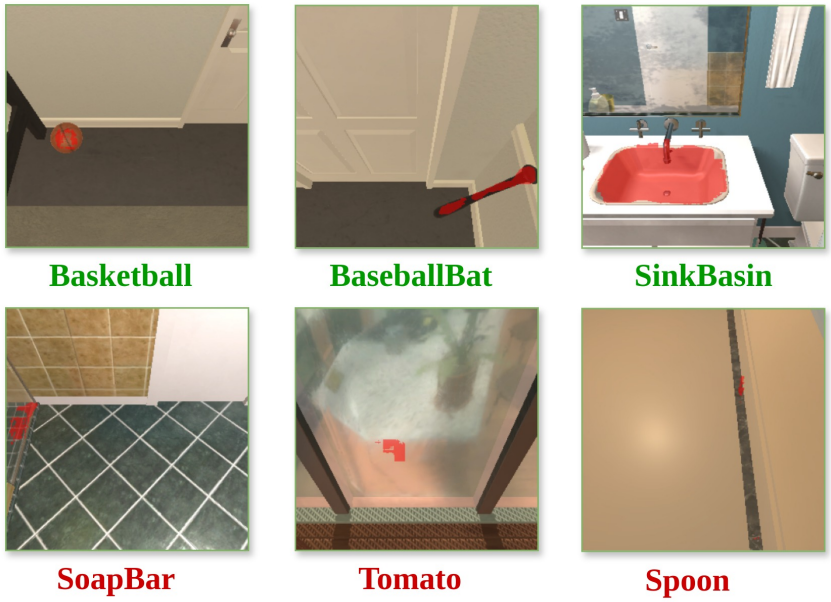

Figure 6.1: Qualitative result of masks, predicted using zero-shot CLIPSeg model. The first row shows the success cases and the second row are the wrong predicted masks.

zero-shot fashion in the simulator in Figure 6.1. The first row shows three correctly masked objects: basketball, baseball bat, and sink. From our observation, the correctly segmented instances are usually large, and their colors greatly contrast the surroundings. The second row shows the failure case. For example, the tiles are wrongly recognized as soap bars because of its square shape. Tomatoes and spoons are also wrongly segmented from windows or floors. In the third image, CLIPSeg predicted a spoon-like shape from the background, showing its limitations in the simulator.

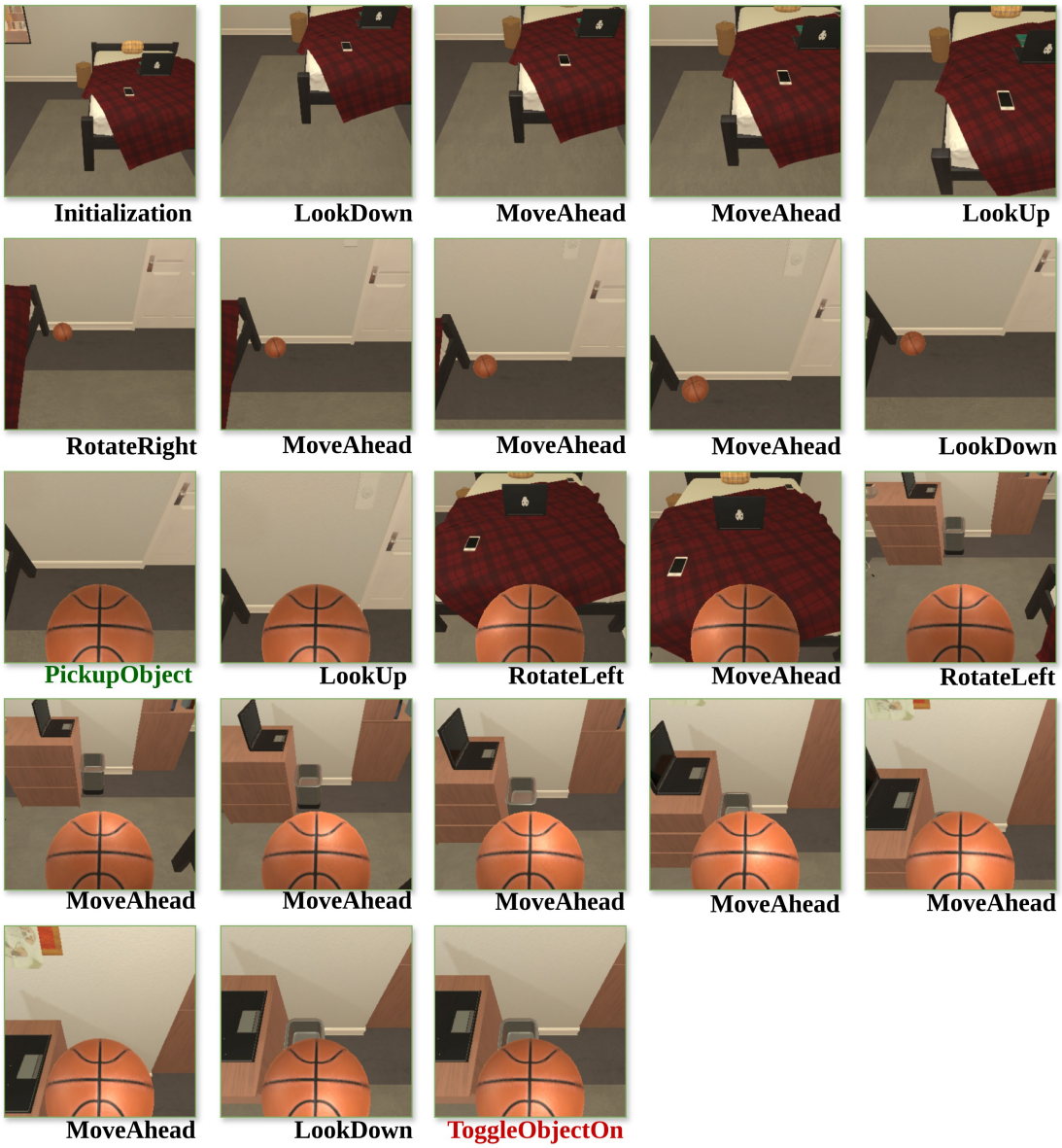

Figure 6.2: Example task: "Look at the basketball in the light from the lamp". The given step-by-step instructions are as follows: 'Walk to the foot of the bed.' -> 'Pick up the basketball from the floor.' -> 'Go to the desk to your left.' -> 'Turn on the lamp.'

7 Conclusion

In this thesis, we introduce LIAM, a multi-modal model incorporating natural language input, ego-centric RGB image input, action history, and an accumulated semantic map. The semantic map is generated using the ego-centric RGB image based on semantic segmentation prediction and depth estimation. Because all the encoder encodes corresponding input in its own latent space, we design two pre-training tasks to pre-align the embedding space. The baseline contrastive pre-alignment of the image and the action space is inspired by the idea of ActionCLIP (Wang, Xing, and Yong Liu 2021). This pre-training objective aims to maximize the agreement of the action representation and the global representation of two consecutive images. We use a light-head 1D convolution to compute this global representation. However, the pre-training task does not consider the alignment of the text encoding backbone. We introduce a triple contrastive alignment to tackle this weakness. Given several trials, we first compute the representation of each individual frame from the annotation data using CLIP backbone. We use a parameter-free operation, i.e., mean pooling, to compute the global visual sequence representation. We follow our contrastive alignment approach for each of the two consecutive images to compute their global representation. The agreement between the image sequence representation and the text sequence representation, the two-frames representation and the action representation are expected to be maximized. Both optimizations are carried out at the same time so that the image encoder serves as a bridge and aligns the three latent embedding spaces simultaneously. After concatenating all embeddings, we fuse them using a 2-layer Transformer encoder. A learnable modal-type encoding is applied at this stage so that the model can separate different encoding types.

Our model outperforms our baseline, using the OpenAI-released CLIP model and no map as an additional modality. We show the importance of pre-aligning the embedding spaces from different modalities. In addition, using semantic maps as a modality to the end-to-end model brought benefits as well.

Despite the acceptable prediction of the action sequence, the gap between our approach and the state-of-the-art approaches in the ALFRED challenge is enormous. Our model has trouble during inference time on the challenge. Our assumption is that CLIP models are limited to extracting the proper representation of simulator-

based models. Besides, training an end-to-end model that can make complicated decision sequence from different modalities input, like human brains, is still challenging, especially when combining the instruction understanding with holistic spatial information understanding.
Future work holds immense potential for further enhancing our model. One avenue could be integrating the segmentation model inside our training stage and the semantic map generation stage. Furthermore, the methodology for encoding the map spatial information is another area that needs to be further investigated. We are particularly intrigued by the potential of considering maps as a knowledge base modality to the Visual-Language models.

List of Figures

List of Tables

Bibliography

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama (2019). "Optuna: a next-generation hyperparameter optimization framework." In: *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*, pp. 2623–2631.

Blukis, Valts, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi (2022). "A persistent spatial semantic representation for high-level natural language instruction execution." In: *Conference on robot learning*. PMLR, pp. 706–717.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). "Language models are few-shot learners." In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin (2021). "Emerging properties in self-supervised vision transformers." In: *Proceedings of the ieee/cvf international conference on computer vision*, pp. 9650–9660.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020). "A simple framework for contrastive learning of visual representations." In: *International conference on machine learning*. PMLR, pp. 1597–1607.

Deng, Li (2012). "The mnist database of handwritten digit images for machine learning research [best of the web]." In: *Ieee signal processing magazine* 29.6, pp. 141–142.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. (2020). "An image is worth 16x16 words: transformers for image recognition at scale." In: *International conference on learning representations*.

Gu, Xiuye, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui (n.d.). "Open-vocabulary object detection via vision and language knowledge distillation." In: *International conference on learning representations*.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). "Mask r-cnn." In: *Proceedings of the ieee international conference on computer vision*, pp. 2961–2969.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition." In: *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 770–778.

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian error linear units (gelus)." In: *Arxiv preprint arxiv:1606.08415*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780.

Inoue, Yuki and Hiroki Ohashi (2022). "Prompter: utilizing large language model prompting for a data efficient embodied instruction following." In: *Arxiv preprint arxiv:2211.03267*.

Kenton, Jacob Devlin Ming-Wei Chang and Lee Kristina Toutanova (2019). "Bert: pre-training of deep bidirectional transformers for language understanding." In: *Proceedings of naacl-hlt*, pp. 4171–4186.

Kim, Wonjae, Bokyung Son, and Ildoo Kim (2021). "Vilt: vision-and-language transformer without convolution or region supervision." In: *International conference on machine learning*. PMLR, pp. 5583–5594.

Kolve, Eric, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. (2017). "Ai2-thor: an interactive 3d environment for visual ai." In: *Arxiv preprint arxiv:1712.05474*.

Liu, Yixin, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. (2024). "Sora: a review on background, technology, limitations, and opportunities of large vision models." In: *Arxiv preprint arxiv:2402.17177*.

Loshchilov, Ilya and Frank Hutter (2022). "Sgdr: stochastic gradient descent with warm restarts." In: *International conference on learning representations*.

– (n.d.). "Decoupled weight decay regularization." In: *International conference on learning representations*.

Lüddecke, Timo and Alexander Ecker (2022). "Image segmentation using text and image prompts." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 7086–7096.

Luo, Huaishao, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li (2022). "Clip4clip: an empirical study of clip for end to end video clip retrieval and captioning." In: *Neurocomputing* 508, pp. 293–304.

Min, So Yeon, Devendra Singh Chaplot, Pradeep Kumar Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov (n.d.). "Film: following instructions in language with modular methods." In: *International conference on learning representations*.

Pashevich, Alexander, Cordelia Schmid, and Chen Sun (2021). "Episodic transformer for vision-and-language navigation." In: *Proceedings of the ieee/cvf international conference on computer vision*, pp. 15942–15952.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. (2019). "Pytorch: an imperative style, high-performance deep learning library." In: *Advances in neural information processing systems* 32.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark,

et al. (2021). "Learning transferable visual models from natural language supervision." In: *International conference on machine learning*. PMLR, pp. 8748–8763.

Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen (2022). "Hierarchical text-conditional image generation with clip latents." In: *Arxiv preprint arxiv:2204.06125* 1.2, p. 3.

Rosenblatt, F. (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6, pp. 386–408.

Shafiullah, Nur Muhammad Mahi, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam (n.d.). "Clip-fields: weakly supervised semantic fields for robotic memory." In: *Icra2023 workshop on pretraining for robotics (pt4r)*.

Shridhar, Mohit, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox (2020). "Alfred: a benchmark for interpreting grounded instructions for everyday tasks." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 10740–10749.

Vasu, Pavan Kumar Anasosalu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan (2023). "Fastvit: a fast hybrid vision transformer using structural reparameterization." In: *Proceedings of the ieee/cvf international conference on computer vision*, pp. 5785–5795.

Vasu, Pavan Kumar Anasosalu, Hadi Pouransari, Fartash Faghri, Raviteja Vemulapalli, and Oncel Tuzel (2024). "Mobileclip: fast image-text models through multi-modal reinforced training." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 15963–15974.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In: *Advances in neural information processing systems* 30.

Vuong, Quan, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, 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, Mengmeng, Jiazheng Xing, and Yong Liu (2021). "Actionclip: a new paradigm for video action recognition." In: *Arxiv preprint arxiv:2109.08472*.

Wu, Zhirong, Yuanjun Xiong, Stella X Yu, and Dahua Lin (2018). "Unsupervised feature learning via non-parametric instance discrimination." In: *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 3733–3742.

Xu, Jiarui, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang (2022). "Groupvit: semantic segmentation emerges from text supervision." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 18134–18144.

Zhang, Renrui, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li (2022). "Pointclip: point cloud understand-

ing by clip." In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 8552–8562.

Zhou, Xingyi, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra (2022). "Detecting twenty-thousand classes using image-level supervision." In: *European conference on computer vision*. Springer, pp. 350–368.