Introduction

Github Link: https://github.com/rituraj2847/ViLT-ALFRED.

Our End-to-End approach is based on Vision-and-Language Transformer, ViLT [1]. The encoder relies on attention-based multi-layer transformer encoders that take the goal instruction, the step instruction and the last 8 (hyperparameter) actions as language section and front, left and right egocentric images as image section. The transformer predicts action and two objects, the target object and the receptacle object. This predicted action is used again in a recurrent manner as part of the action history. Below we have described the model architecture and the algorithm we implemented.

0.1 Model Architecture

As shown in Fig. 1 our model is takes in 2 sets of inputs, the language tokens and the image tokens. The language tokens take goal instruction, step instruction, and the previous 8 actions as action history. The Image tokens take front, left and right views. ViLT model accepts 512 tokens as input, where the first 40 tokens are reserved for the Language tokens.

For output, we have taken concatenation of the pooler output, and the last hidden state at 40^{th} index, each has a dimension of 768. This concatenated vector is passed through an FC layer to make the vector modal redundant and transforms it into a subspace of 2048 dimensions. Further, three FC layers are used to predict the following action, the target object and the receptacle object.

In Fig. 1, the goal is from the "Pick and Place" task, which will have around 4 instructions to perform (low level instructions). Each step instruction is provided recurrently after the same goal's previous instruction is completed. The Action history provided in the input tells us the previous actions taken by the agent only for the current step instruction. For example, in the figure, the current step instruction asks the agent to pick up the pillow, and for this task, the

previous actions were *LookDown, MoveAhead* and a few more actions.

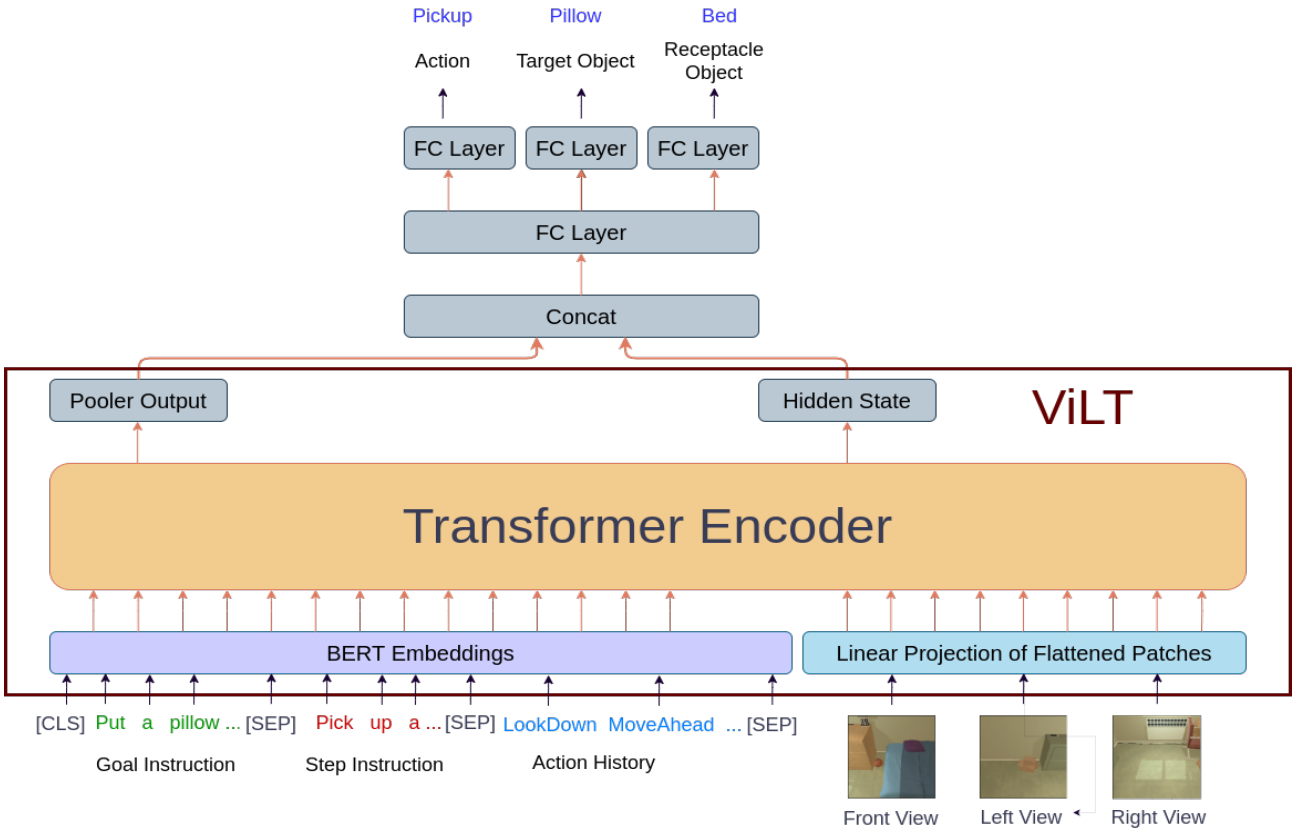

Figure 1: Model Architecture

0.1.1 Language Processing

The language input of our architecture is allotted the first 40 tokens out of 512 input tokens. We provide goal instruction, step instruction and action history, all separated by a $[SEP]$ token. Step Instruction corresponding to the same goal increments when our model predicts a *STOP* action. If the particular step instruction is the last step, we start a new annotation data with a new goal.

Action history consists of the last 8 (hyperparameter) actions taken by the model in each iteration corresponding to each step instruction. With each time step recurrently, we append the action history with the latest predicted action. When the step instruction changes, the action history is set as None. We have set the action history size hyperparameter as 8.

The input language tokens are sent to Bert Tokenizer to generate tokens, which are then forwarded to the model.

0.1.2 Image Processing

ViLT model [1] forwards linear projection of flattened patches of the input image to the transformer encoder. For the image input, instead of just providing the current egocentric image, we provide 3 images, the front view, left view and the right view to provide a panoramic view of the scene. We obtain the left and right view of the scene by rotating the agent by 90^o in the left and right directions. Since we can only rotate the agent by 90^o, creating a panoramic view takes 4 steps. In ViLT, for the available pretrained checkpoint, 482 tokens are fixed for image tokens.

0.1.3 Our Algorithm

Our model (ViLT-Alfred) takes the input and predicts actions and objects. When our model predicts a *Stop* action, we change the step instruction or both goal and step instruction, depending on if the step instruction corresponds to the last step for that trajectory (or goal). In our implementation, we tried to deal with the following 2 bottlenecks.

Collision Detection: In our implementation, there are instances of collisions. Here our agent collides with the environment. An agent can take a maximum of up to 10 collisions before the trajectory is marked as a failure. Such collisions occur only when the action predicted is *MoveAhead*. To deal with such failures, if a collision is predicted, and the subsequent action predicted is *MoveAhead*, we replace it with either *RotateLeft* or *RotateRight*, whichever has a higher probability, and append it in the action history. This replaced action allows the agent more exploration instead of trajectory failure caused by collision.

Exploration: Here, we aim to handle cases where a navigation subgoal fails and try to ensure more exploration.

When our model predicts *Stop* action and our fine-tuned MaskRCNN cannot get the target object's bounding box in the current frame, all the subsequent manipulation actions fail. We need to restart the navigation instruction for more explorations to deal with such failures. Our xxploration function randomly replaces *Stop* prediction in such scenarios from among the top two most probable navigation actions (*MoveAhead, RotateLeft, RotateRight, LookUp, LookDown*) This is repeated until our navigation is a success or our agent exceeds 1000 steps or causes more than 10 failed actions.

0.1.4 Results

Results in Tables 1, 2 and 3 are taken on validation data.

Table 1: Evaluation Metrics

	Validation				**Test**	
	Seen		**Unseen**		**Seen**	**Unseen**
Models	Task SR	GC	Task SR	GC		
ViLT-Alfred (Ours)	5.6	15.1	2.4	11.7		

Table 2: SubGoal Evaluation - Validation

	Models	**Goto**	**Pick**	**Put**	**Cool**	**Heat**	**Clean**	**Slice**	**Toggle**	**Avg.**
Seen	**ViLT-Alfred (Ours)**	43	61	68	71	90	22	76	77	73
Unseen	**ViLT-Alfred (Ours)**	30	52	53	75	89	86	60	84	61

Table 3: Task Ablations - Validation

	Models	**Pick & Place**	**Stack & Place**	**Pick two & Place**	**Clean & Place**	**Heat & Place**	**Cool & Place**	**Examine in Light**
Seen	**ViLT-Alfred (Ours)**	10.56	0.12	8.13	0.0	7.84	5.93	4.25
Unseen	**ViLT-Alfred (Ours)**	1.0	0.91	0.0	2.65	0.0	0.0	8.67

Bibliography

[1] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021. i, iii