Learning to Orient Towards the Focus of Attention of a Group Conversation Using Variational Auto-encoders

Sally Ma
sally.ma@yale.edu
Yale University

Advisor: Marynel Vázquez
marynel.vazquez@yale.edu
Yale University

ABSTRACT

We investigate the effectiveness of using variational auto-encoders (VAEs) to compress high-dimensional observations into low-dimensional latent state representations, from which a robot could learn motion policies in orienting towards the speaker of a conversation. We study multiple procedures for training a VAE to learn a latent representation for reconstructing the input images and predicting action values for the robot. We then evaluate the learned policy in a simulated environment, and compare the VAEs' performance with that of convolutional neural networks (CNNs) that directly map high-dimensional observations to action values without learning a latent state representation. Our results suggest that the best training procedure of VAE for this task is training both the image reconstruction and action prediction together from scratch. The resulting VAE effectively learns a policy that allows the robot to orient towards the focus of attention of a group conversation by following the point of maximum social saliency (the point with maximum gaze rays' concurrence from group members). The latent space visualization also shows that the latent variables of the VAE model encode relevant information for action prediction. The best VAE model so far has comparable but not superior performance to the best CNN model. This suggests that additional constraints are needed to guide the model to learn better.

1 INTRODUCTION

Social robots often need to operate with and around groups of people, including partaking in group conversations. During social group conversations, a robot's body orientation is important because people frequently consider it communicative and meaningful [8]. For robots with a small number of degrees of freedom, such as Cobot [9] (which navigate university buildings and perform tasks in collaboration with nearby users) and Frog [3] (which operates as a museum tour guide leading groups of visitors from one place to another), body orientation controls the direction of important social features (e.g. faces) and many of the robot sensors. The robots' orientation can, therefore, substantially influence how people interpret their actions and sensing abilities. Research has also shown that body orientation could induce spatial reconfigurations during social interactions [4]. A robot turning towards the speaker in a group conversation is thus a subtle and effective strategy of conveying attentiveness to the conversation and redirecting the focus of a conversation.

Inspired by the potential of Variational Autoencoders (VAEs) in compressing high-dimensional data to a low-dimensional representation and disentangling complex social contexts [2], we at the Interactive Machines Group intend to explore using VAEs to allow the robot to learn motion policies in orienting towards the speaker of a conversation. Built on top of standard function approximators (neural networks) and able to be trained with stochastic gradient descent, VAEs emerge as an appealing approach to unsupervised learning of complicated distributions. In comparison with standard autoencoders (AEs), which are discriminative, VAEs are generative and probabilistic: whereas AEs force the encoder to produce a single discrete encoding for each input, VAEs force the encoder to produce a continuous probability distribution over the encodings, from which the decoder can sample to produce outputs. Given latent vectors not present in train data, AEs are incapable of generating reasonable outputs that look like they belong to the same distribution as the inputs of train data: AEs' latent space is purely optimized for image reconstruction loss, without penalization for generative purposes, so the latent space tends to have many clusters and empty space, and sampling from the latent space may not decode to a data point at all. On the other hand, VAEs are capable of doing so, learning a latent representation parameterized by Gaussian means and standard deviations, which correspond to a set of Gaussian distributions that can be sampled for generating output. VAEs are shown to be effective models for encoding sequential data, allowing them to perform tasks such as disentangling dynamic features from static features in video recordings [5]. VAEs thus demonstrate potential in disentangling complex social contexts for a robot, allowing the robot to understand and synthesize sequential context clues from its surroundings and better respond to changing social dynamics.

To study the effectiveness of using VAEs in compressing high-dimensional observations and learning a motion policy for orienting towards the speaker of the conversation, we first adapt a simulated environment built by Vázquez et al. [11] to generate a rich dataset for training neural network models. We preprocess the data to eliminate excess of data points in which the robot shows no motion. We then train multiple convolutional neural networks that directly map observation images to action values, and evaluate the learned policies' performance in simulation. Next, we train VAE models with different training procedures, and investigate the effect of varying different hyperparameters. We evaluate the VAEs' learned policies in simulation and compare the results with that of using a Convolutional Neural Network (CNN), a traditional method that does not learn a low-dimensional latent space for compression of the high-dimensional input data.

The rest of the report is organized as follows. Section 2 presents a list of prior work that helps contextualize the effort. Section 3 describes our main approach for studying the effectiveness of using VAEs in our task. Section 4 lays out the simulation experiments we performed to evaluate various models' performance in learning the motion policy, and analyzes the empirical results. Section 5

summarizes our key results and discusses potential future work, and section 6 presents acknowledgements.

Our key contributions in this work are: 1) presenting an evaluation of different procedures of training a VAE for the task of orienting towards the focus of attention in a group conversation, including the effects of different hyperparameters; 2) providing simulation experiments comparing the performance of various VAE models with those of convolutional neural networks that directly map from high-dimensional observations to action prediction.

2 RELATED WORK

2.1 Motion Policy in Group Conversation

Past work on controlling robots' spatial behaviors in human conversations includes relying on tele-operations [12], [10] and generating autonomous robot motion through rule-based approaches [13] or generative models of proxemic behavior [6]. Rule-based approaches tend to generalize poorly to new situations, and generative models have not been tested with adapting a robot's position with respect to more than one person. More recently, Vázquez et al. evaluate using reinforcement learning (RL) to control robot orientation in a simulated environment that they built based on models from social psychology explaining spatial behavior during group conversations. They find that model-based RL agents are capable of learning good policies within a few minutes of interaction time [11]. The simulated environment they have constructed provides the basis for our work. We revise and adapt the simulation in order to generate images as our models' inputs; we also evaluate our models' performance by rolling out the model's learned motion policy in the simulation.

2.2 VAE for Learning Motion Policy through Simulation

The work by Bonatti et. al [1] most motivates our approach. They propose learning robust visuomotor policies for a drone's aerial navigation by training a cross-modal VAE with simulated data. The VAE takes in first-person-view (FPV) observation images generated from simulation and learns a rich low-dimensional state representation, which it then uses to learn a navigation policy of outputting velocity commands of the drone. Through multiple real-world navigation experiments, the authors show that their framework allows for simulation-to-real deployment of models learned purely in simulation, achieving over one kilometer of cumulative autonomous flight through obstacles.

We extend their work in 2 ways. Firstly, we investigate using VAE to learn the motion policy for a different task, focusing on the setting of orienting in group conversation, and explore other model architectures and hyperparameter settings. Secondly, we compare the effectiveness of different training procedures for VAEs. In addition to training both the image reconstruction branch and action prediction branch together from scratch, we investigate the effectiveness of 1) training only the image reconstruction branch, then freeze the encoder weights while training the action prediction branch to minimize action loss; 2) training only the image reconstruction branch, then allowing the encoder weights to change while training the action prediction branch.

3 METHOD

3.1 Data Collection and Preprocessing

To generate the dataset for training our models, we adapt the simulation environment proposed by Vázquez et al [11]. Figure 1 presents an example of the context of the simulation, where people maintain distinct spatial organizations known as F-formations (short for face-formations) while standing freely in open, public spaces. Face formations begin when the members of a group position themselves such that their transnational segments (segments that extend in front of each person) intersect, the intersection of which is known as the o-space of the F-formation. In the simulation, the robot and the people in its conversation form a circular F-formation.

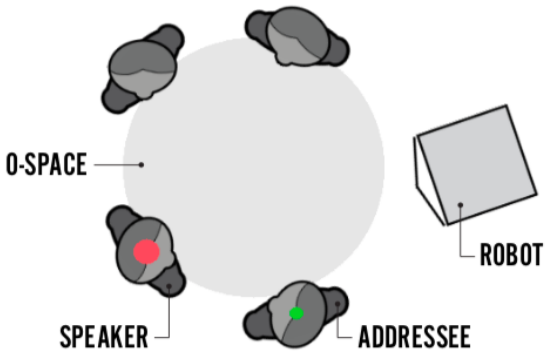

Figure 1: A simulated group conversation between a robot and four people. The red and green circles on top of the agents identify the speaker and addresses, respectively. The big gray circle represents the o-space of the group's face-formation. Figure borrowed with permission from [11].

Figure 2 demonstrates sample input images collected from running the simulation. The smaller dots in the first and third image, which can be either yellow or dark grey, represent the point of maximum social saliency induced by the people in the group. Social saliency models gaze concurrence, estimated using group members' gaze rays, and the point of maximum saliency is the one with greatest contribution from the group members' gaze rays [7]. The speaking wave is an additional indicator of who the speaker is. Both the speaking wave and the point of maximum social saliency are features occasionally present in the robot's observations.

We adapt the simulated environment such that at each time step in the simulation, a pair of observation image and its corresponding action value is collected. The action value represents the angular velocity of how much the robot should turn in radian such that it can change from its current orientation (as captured in the observation image) to pointing towards the speaker in one time step. Since we want to prevent oscillatory motions, when the action value is smaller than a threshold, we default the value to 0. We set the threshold to be 0.174533 radian (10 degrees) to be consistent with the calculation of bonus in the reward given to the robot during evaluation time.

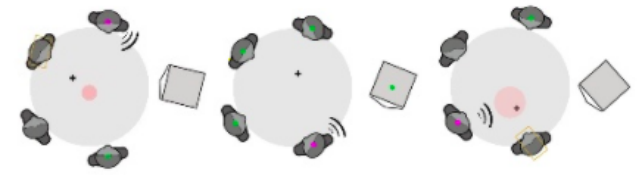

Figure 2: Sample input images generated from simulation, with the corresponding target action values of -0.486601, -0.557199, -0.490357 for how much to turn from current position in radian within one timestep of simulation (negative due to counterclockwise rotation)

Additionally, to prevent the model from having no training data that covers the scenarios where the errors it makes compound during evaluation time, leading the robot to point to odd angles (e.g. turning its back against the group) that an expert robot would never do and thus would not provide training data for how to recover from these scenarios, we randomize the robot orientation (to point to any direction in 360 degrees) during data collection at each time step with a certain probability. We collect two datasets, each having 6400 initial data points from 100 simulation runs of 64 time steps per run. The dataset has a randomization probability of 0.3 and 0.5 respectively, meaning that the robot would get a random orientation if the result of a random number generation between 0 and 1 is greater than 0.3 or 0.5. The randomization decision comes from observing empirically that the performance of the robot during evaluation drastically decreases after the mistakes it has made accumulates. The accumulated mistakes lead the robot to orientations not present in the train data without randomized orientation, thus it does not know what to do to recover from these unseen scenarios.

The resulting dataset has a disproportionately large number of data points with action value equal to 0 (meaning robot not moving), even after setting the speaker's talking time average and minimum to be 0.1 time step. To achieve a more balanced dataset, preventing the model from learning to just not move, we keep only around 500 out of over 1000 or 1200 data points with action value equal to 0 in each dataset (see Figure 3).

3.2 Model Architectures and Training Procedures

We propose a VAE that takes an input image x_t representing the current state observation and uses a probabilistic encoder q_{img} to process the image into a multivariate Gaussian distribution $N(\mu, diag(\sigma^2))$, from which it samples a compressed vector z_t with dimensionality D. $N(\mu, diag(\sigma^2))$ models the conditional probability distribution $Q(z|x)$, defined to approximate the true underlying conditional distribution $P(z|x)$. μ and σ are both vectors with dimensionality D and are learned during training. The size of the latent dimension is a hyperparameter tuned during evaluation.

We use a common reparameterization trick for easier sampling of z_t from $N(\mu, diag(\sigma^2))$: we first sample from a standard Gaussian distribution $N(0, I)$, then use linear transformation to realize the latent vector $z_t = \mu + \epsilon * \sigma$, where $\epsilon \in N(0, I)$ is random noise.

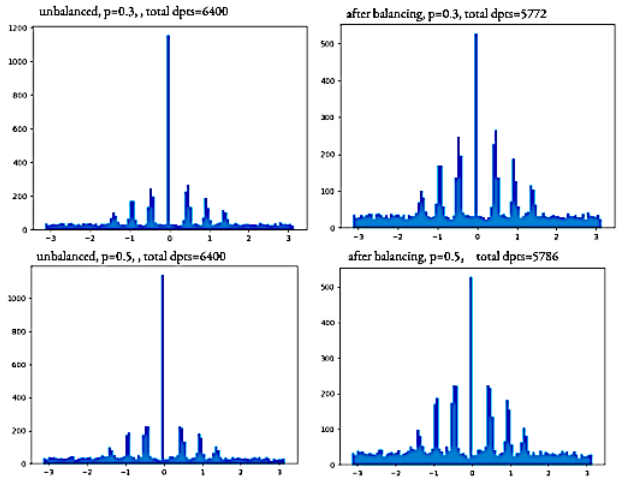

Figure 3: Summary of data distribution for the action values, with randomization probability set to 0.3 and 0.5 respectively; removing excess of data points in which action value = 0 cuts the size of dataset from 6400 to 5772 and 5786 respectively.

The VAE has an image reconstruction branch and an action prediction branch for decoding: the probabilistic image decoder p_{img} maps z_t to a reconstructed image $\hat{x_t}$, and the action decoder p_{action} maps z_t to an action value $\hat{\theta_t}$. The action corresponds to an angular velocity representing how much a robot should turn in a simulation time step in order to orient towards the speaker from its current orientation. The probabilistic decoder models the conditional probability distribution $P(x|z)$ for generating new samples given latent variables.

Figure 4 illustrates the full architecture of the VAE.

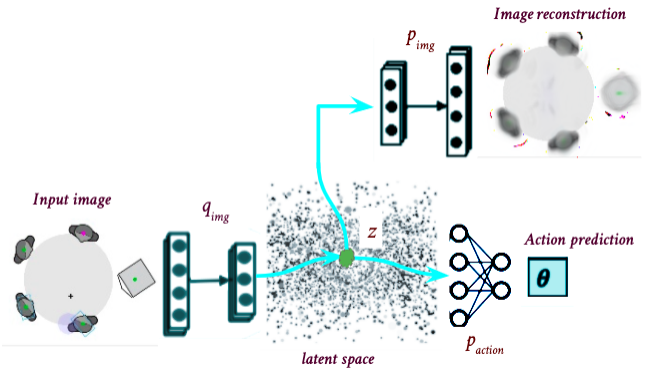

Figure 4: VAE architecture. Each data sample (input image) is encoded into a single latent space that can be decoded back into a reconstructed image, or transformed in an action value prediction for the robot's angular velocity command based on the input image

We evaluate the effectiveness of three different training procedures for VAEs:

(1) Train both the image reconstruction branch and the action prediction branch together from scratch
(2) Train the image reconstruction branch only, then load the trained weights and freeze them while training the action prediction branch only
(3) Train the image reconstruction branch only, then load the trained weights and allow the encoder weights to change while training the action prediction branch only

For the first training procedure, we consider three losses for each sample:

(1) Kullback–Leibler (KL) divergence loss between $Q(z|x)$ and $P(z)$, a regularization penalty for preventing the learned distribution $Q(z|x)$ from deviating too much from the prior distribution $P(z)$. Here we follow the convention to set $P(z)$ as standard Gaussian distribution $N(0, I)$, thus the KL divergence term can be expressed as:

$$D_{KL}[Q(z|x)||P(z)] = D_{KL}[N(\mu(x), diag(\sigma(x)^2))||N(0, I)]$$
$$= -0.5 * [1 + \sigma(x) - \mu(x)^2 - exp(\sigma(x))]$$

where $\sigma(x)$ and $\mu(x)$ are the latent vectors with values corresponding to an input image x, whose weights are learned during training
(2) Mean Squared Error (MSE) loss between actual and reconstructed images $(x_t, \hat{x_t})$
(3) MSE loss between target and predicted action values $(\theta_t, \hat{\theta}_t)$

For the second and third training procedure, we first consider only KL divergence and image reconstruction losses while training the image reconstruction branch, then only consider the action loss while training the action prediction branch. The weighting of the three losses in the weighted sum of losses is a hyperparameter tuned during experiments.

To study the effectiveness of learning motion policy from a low-dimensional latent state representation, we compare its results with those of using a CNN. The CNN is a more traditional method that does not compress high-dimensional input data to a latent space for action prediction, but rather learn to extract features from input images in order to map each image directly to a corresponding action. The training of CNNs seeks to minimize MSE loss between target and predicted action values $(\theta_t, \hat{\theta}_t)$ for each data sample.

4 EXPERIMENTS

We evaluate various models and training procedures in three ways:

(1) Inspecting how train and validation losses change over epochs
(2) Rolling out the learned motion policy in the simulated environment and keeping track of the cumulative rewards received by the robot
(3) When appropriate, inspecting the reconstructed images of a VAE

For the second way, a reward r_{t+1} that the simulated environment provides to the robot as a result of taking an action a_t is

$$r_{t+1} = exp(-d^2) + b$$

where d is an angle in radian, between $[-\pi, \pi]$, representing the difference between the orientation of the robot and the angle of the direction towards the speaker from the robot's position; and the bonus b is given to reward 0 angular velocity commands when the difference d is smaller than 0.174533 radian (10 degrees). For each learned motion policy, we perform 5 simulation runs and select the intermediate (the 3rd best/worst) result to represent the model's performance.

We concentrate the evaluation on situations where the users are the active speakers and the focus of attention. These situations are more interesting to study than their counterparts because the robot does not have control of the interaction dynamics and must adapt to the flow of the conversation. Thus in our simulation, the robot will not be allowed to speak.

4.1 CNN Results

We first experiment with the case of having a convolutional neural network learn a mapping from high-dimensional data directly to action values. After experimenting with multiple network architectures, we find the best-performing network has a simple architecture, illustrated by Table 1. The model achieves 808 cumulative rewards, compared with 1655 achieved by the perfect policy (a perfect policy directly uses as the action value for each time step the difference between robot orientation and speaker direction computed by the environment). A video illustrating the robot using the best-performing CNN's learned policy in the simulation can be watched *here*.

layer	filter	stride	size	activation	regularizer
Conv2D	3x3	(2,2)	32	relu	L2(0.0005)
Conv2D	3x3	(2,2)	64	relu	L2(0.0005)
MaxPool2D	2x2	(2,2)			
Dropout(0.25)					
Dense			128	relu	
Dense			1	linear	

Table 1: CNN architecture: mapping an input image directly to an action value without learning a latent space

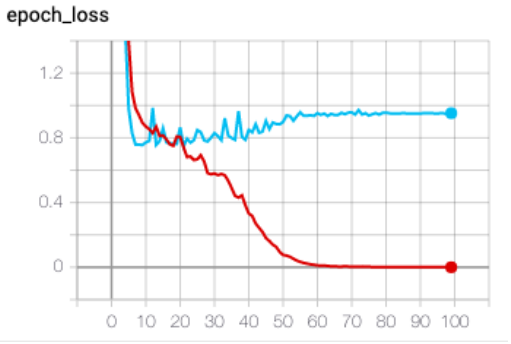

Figure 5: Although the VGG19 net has consistently decreasing train loss, ending with one very close to 0, its validation loss stops decreasing, suggesting that the network is overfitting

The second best-performing CNN is a VGG19 net. It has cumulative reward of 437, which is barely more than that of random prediction, which has 349 cumulative reward. Although the VGG19 net has consistently decreasing train loss, ending with one very close to 0, its validation loss stops decreasing, suggesting that the network is overfitting (see Figure 5). Additionally, VGG19 is a very deep neural network with later layers developed to capture features in realistic images not present in our simulated images, although early layers could be helpful in capturing low level features such as edge detection. These reasons could explain why VGG19 is the second best-performing CNN but doesn't perform significantly better random prediction. Other CNN models tried and tuned include: variants of the best-performing CNN where the relu activation functions are replaced by tanh at various places, some with an additional custom layer at the end that multiplies the output by 3.14, other CNNs with more layers, and YOLO net. However, none of these variants performed substantially better than random prediction.

4.2 VAE Results

For the VAE, after extensive architecture search and hyperparameter tuning, we find that the best performing model is given by training procedure (1), where we train both the image reconstruction branch and the action prediction branch together from scratch. We use a batch size of 32, a learning rate of 1e-4, Adams optimizer, 150 train epochs, and (KL divergence, image reconstruction, action prediction) losses having weights of (8, 1, 1). Table 2 illustrates architecture of its encoder, and Table 3 illustrates that of its decoder for image reconstruction. For action prediction, the best performing model simply outputs 1 action value directly from the 10-dimensional latent vector (see Table 4).

layer	filter	stride	size	activation	regularizer
Conv2D	3x3	(2,2)	32	relu	L2(0.0005)
Conv2D	3x3	(2,2)	64	relu	L2(0.0005)
MaxPool2D	2x2	(2,2)			
Dropout(0.25)					
Dense			64	relu	
Dense			32	relu	
Dense			10	linear	

Table 2: Encoder architecture: encodes input image into a latent space with dimension = 10

Figure 6 and 7 illustrate how the train and validation losses vary while training the VAE. We could observe that the KL divergence and image reconstruction loss remain low throughout the training. On the other hand, though the train action loss decreases consistently throughout the training procedure except for initial oscillation, the validation action loss suggests overfitting. Thus we save the weights for when the validation action loss is at minimum.

Figure 9 illustrates the reconstructed images from the best-performing VAE. By contrast, Figure 8 illustrates the reconstructed images from using the same architecture as the best-performing VAE, but first training the encoder and the decoder only, without the action prediction branch. It's notable how the robot is only a round blob in the reconstructed images from a VAE that does not include a trained action prediction branch, whereas the robot is more square and has a triangular head representing its orientation in the reconstructed images from a VAE that includes an action prediction branch. This is likely because a round blob is an average of input images, which minimizes image reconstruction loss; on the other hand, the presence of triangular head representing robot orientation suggests that the VAE with action policy branch recognizes the importance of the robot's current orientation, in order to more accurately predict how much it should turn to orient towards the speaker in the group.

layer	filter	stride	size	activation
Dense			128x128x64	relu
Reshape(128,128,64)				
Dropout(0.25)				
UpSampling2D	2x2	(2,2)		
Conv2DTranspose	3x3	(2,2)	64	relu
Conv2DTranspose	3x3	(2,2)	32	relu
Conv2DTranspose	3x3	(2,2)	3	relu

Table 3: Image decoder architecture: decodes latent space into reconstructed image; a regularizer of L2(0.0005) is also applied to each Conv2DTranspose layer

layer	filter	stride	size	activation
Dense			1	linear

Table 4: Action decoder architecture: decodes latent space into an action value

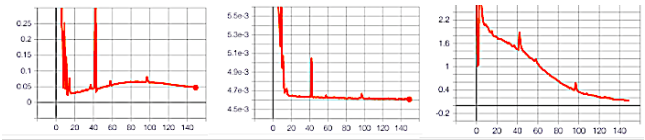

Figure 6: Train loss for KL divergence, image reconstruction, and action prediction in order, for the best-performing VAE: KL divergence drops to 0.025 at the lowest at ends with 0.05; image construction loss drops below 4.7e-3 after 10 epochs and ends with 4.6e-3; action loss decreases consistently, except for the initial oscillation, to 0.1 after 150 epochs

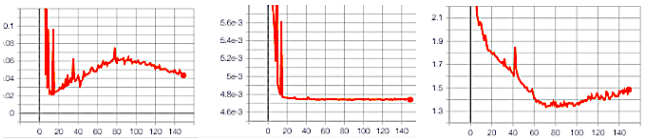

Figure 7: Validation loss for KL divergence, image reconstruction, and action prediction in order, for the best performing VAE: KL divergence drops to 0.02 at the lowest at ends with 0.04; image construction loss drops below 4.8e-3 after 10 epochs and ends with 4.75e-3; action loss drops to 1.32 at the minimum at epoch 81, then starts increasing

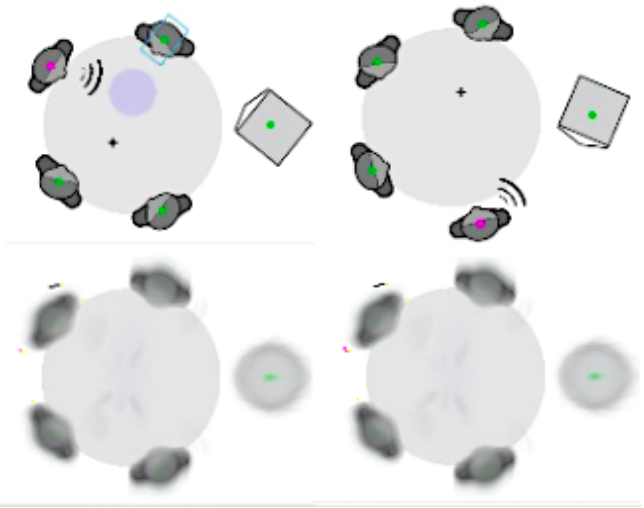

Figure 8: Sampled original images (above) and its corresponding reconstructed images (below) by the best-performing encoder+decoder trained without the action branch

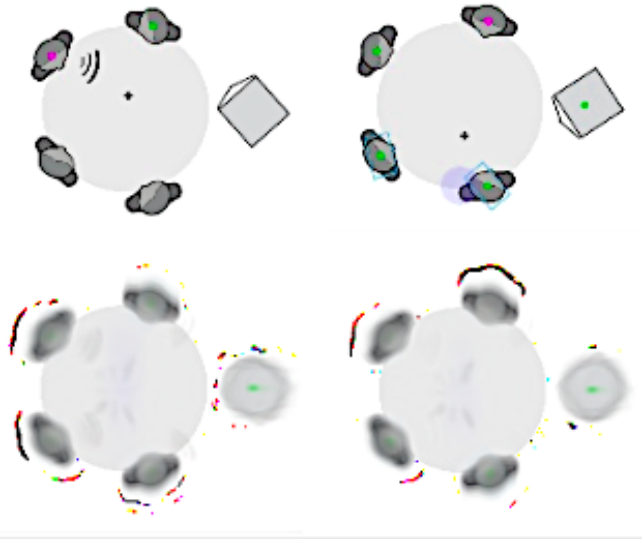

Figure 9: Sampled original images (above) and its corresponding reconstructed images (below) by the best performing VAE

The second best-performing VAE model uses the same architecture as the best-performing VAE, and is given by first training the image reconstruction branch only, then allowing the encoder weights to change while training action prediction branch. The videos illustrating the robot using the best and second-best learned policy in the simulation can be watched *here*. We observe that the robot has learned to follow the point of maximum social saliency to obtain a close approximation to the speaker during the conversation.

The training procedure where we freeze the encoder and decoder weights while training the action prediction branch does not work well for the best-performing VAE architecture. The train and validation action losses only oscillate rather than decrease during training, which makes sense, since the action decoder directly predicts a value from the latent representation–without a changing encoder that adapts the latent representation for better action prediction. The action decoder alone is too simple to have the capacity for learning the motion policy.

Figure 10 displays in juxtaposition the performance of the 6 most relevant cases: the perfect policy, the best performing CNN, the second best performing CNN (VGG19), the best-performing and second-best VAE models, and an untrained agent using random prediction. The gap between the perfect policy and the best VAE model demonstrates that there is still room for improvement in finding the most optimal VAE model, with more architectural search and hyperparameter tuning.

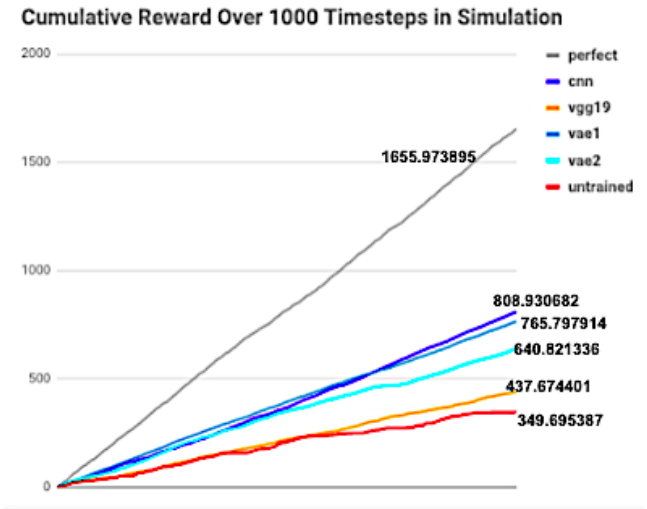

Figure 10: Comparison of cumulative rewards obtained by the robot over 1000 time steps in simulation using 6 modes (in order of max to min): 1) the perfect angular velocity given by the environment; 2) the best-performing CNN; 3) the best performing VAE, training both action branch and image reconstruction branch together from scratch; 4) the second best performing VAE, first training the image reconstruction branch only and tuning the encoder further while training the action branch; 5) the second best-performing CNN using VGG19 net; 6) random prediction

Other architectural and hyperparameter settings we have explored for VAEs include: using the best-performing CNN's architecture as the encoder with various the size of latent dimensions and the number of dense layers between the convolutional layers and the latent representation, switching the UpSampling2D layers to Conv2DTranspose layers and vice versa in the image decoder,

the number and size of dense layers in action decoder, and varying number of train epochs, learning rate, batch size, and weights assigned to the three losses. The setting that yields the most significant improvement, with much better action loss decrease during training, better action prediction during evaluation, and image reconstruction that encodes robot orientation, is having two dense layers of size 64 and 32 after the convolutional layers to get to the 10-dimensional latent layer in the encoder.

4.3 Latent Space Visualization

We visualize the latent space of the best performing VAE model to understand its performance better (Figure 11).

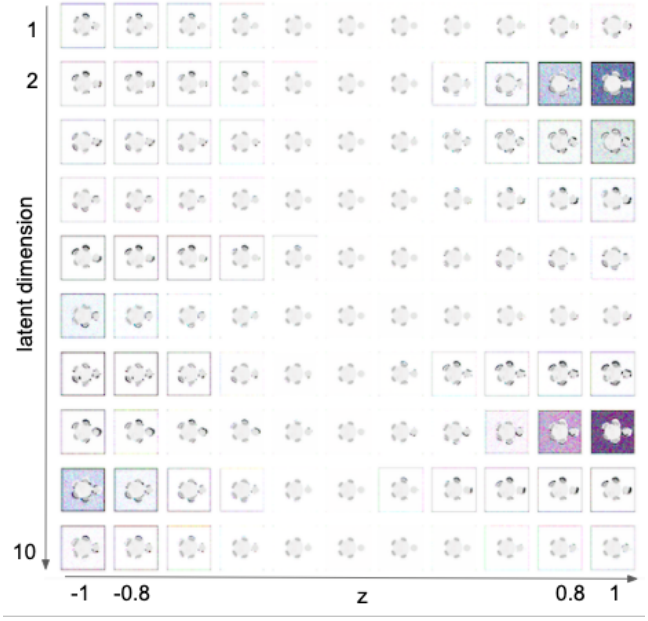

Figure 11: Visualization of the VAE model's latent space. Each row corresponds to variations in z values in one of the 10 latent dimensions from -1 to 1 with a 0.2 increment at each step, while all the z values in the other dimensions are set to 0

We observe that when z is set to 0 or close to 0 (the middle columns), the images are blurrier and the robot in the images is a round blob, suggesting that these are likely the more averaged images. All the corresponding action values for each latent dimension demonstrate a consistent decreasing or increasing order along that dimension with reasonable numbers, with a maximum value of 1.00724566 and minimum value of -1.05971515. For example, the 11 action values for the fifth latent dimension as we shift z from -1 to 1 are: [-0.82769436 -0.66740245 -0.50711054 -0.34681857 -0.18652664 -0.02623472 0.13405721 0.29434913 0.45464107 0.61493295 0.77522486]. The fifth latent dimension (Figure 12) also shows a clear change in the robot orientation when we vary z from -1 to 1, with sensible corresponding action values. We observe that the most prominent grey sound wave moves from the second speaker (counting from the bottom up clockwise) to the third speaker, while the robot orientation changes from pointing towards the lower left corner to the upper left corner. The action values corresponding to the image reconstruction also make sense. For the left image, the robot needs to turn clockwise to point towards the speaker with the most prominent grey sound wave, hence a positive action value. For the right image, the robot needs to turn counter-clockwise to point towards the speaker with the most prominent grey sound wave, hence a negative action value.

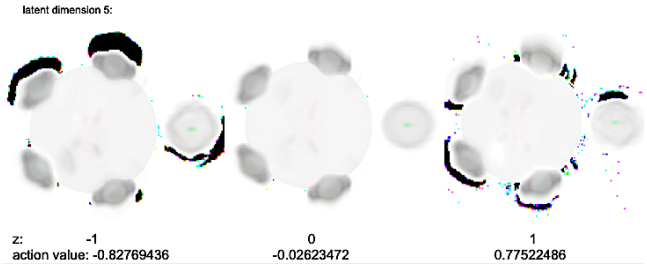

Figure 12: The left end, middle, and right end of visualization at latent dimension 5 (image borders cropped for bigger display). A positive action value means counter-clockwise rotation and a negative action value means clockwise rotation.

The eighth and ninth latent dimensions display clear changes in the locations of the most prominent sound waves as we vary z. For the eighth latent dimension, we see the most prominent sound wave shifts from the second speaker to the third speaker (counting from bottom clockwise), while the opposite happens for the ninth latent dimension. The opposite change in the location of the most prominent sound wave corresponds to the opposite trend of the action values for these two latent dimensions: the eighth latent dimension has action value going from positive to negative, and vice versa for the ninth latent dimension.

Since the robot behavior in simulation shows that the feature the model has learned to follow is the point of maximum social saliency, it's curious that none of the reconstruction images display this feature prominently. One likelihood is that the prominent grey sound wave is actually the model's decoding of the point of maximum social saliency (which is also often grey). Regardless, the latent space visualization shows that the latent variables can encode relevant information about the action prediction, including the robot orientation and indications of the current speaker.

Additional visualization and the corresponding action values are provided in the submitted code and not presented here due to the large size of the images. See the README of the code for more details.

5 CONCLUSION AND FUTURE WORK

We study the effectiveness of using VAEs in compressing high-dimensional observations into low-dimensional latent representation, from which the robot learns a motion policy for orienting towards the speaker of a conversation. We adapt a simulated environment, from which we generate a rich, robust dataset to train our models and evaluate the models' performance in action. We perform architectural search and hyperparameter tuning, including varying the training procedures, and subsequently enable VAE models to

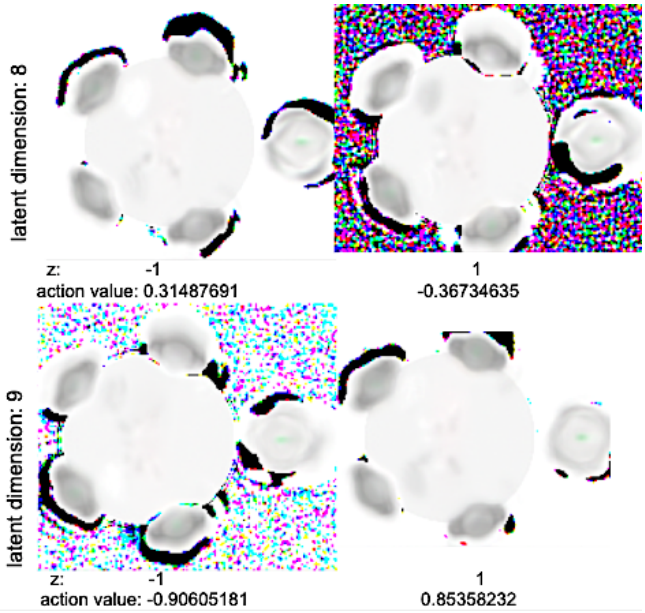

Figure 13: The left and right end of visualization at latent dimension 8 and 9 (image borders cropped for bigger display)

work for our task. We also compare the top VAEs' performance with best-performing traditional methods that map high-dimensional observations directly to action prediction without learning a latent space for context compression.

From our results, VAE demonstrates potential in helping the robot learn a robust motion policy of orienting towards the speaker of the conversation. It reaches comparable performance to the best-performing CNN that does not learn a low-dimensional latent representation for context compression. Rolling out the learned policy in the simulation environment demonstrates that the current best model allows the robot to learn a policy that approximates the focus of attention in a group conversation with the point of maximum social saliency. The reconstructed images from the best-performing VAE, given by training both action and image reconstruction branches from scratch together, show that it recognizes the importance of the robot's current orientation in predicting actions. Visualizing the latent space of the best performing VAE model shows that the latent variables can encode relevant information about the action prediction, such as the robot's current orientation and indication of the current speaker.

We see that with a relatively generic VAE model as used in this project, the model may learn features (such as the point of maximum social saliency) that we did not expect it to learn, rather than learning the most informative feature–the red dot indicating the speaker. We think that adding more constraints to the model would help guide it to learn what it should learn better. For instance, we could add an additional loss to the latent space learned by the encoder, such that the latent distribution does not change too much as it takes image frame by frame, given that the input images represent sequential observations. Since the real world normally does not have sudden drastic changes, this constraint will help guide the model to learn not to change the latent distribution drastically based on a single new image, resulting in a smoother latent distribution. We could also crop out each speaker to form separate images, and feed the full image as well as the cropped images to the model as input, to guide the model to focus on each agent's behavior better. Future work could explore the different types of constraints imposed on the model and see if, how, and why additional guidance helps the model learn desired features better.

6 ACKNOWLEDGEMENTS

I wish to thank Professor Marynel Vázquez for advising me throughout the project. I would also like to thank fellow lab member Nathan Tsoi for his support. This project is a continuation of their efforts, as it is founded upon the simulation environment that Professor Vázquez built, and Nathan set up the development environment and performed initial experiments, both of which laid key foundation for this project. I am grateful to them for all their contributions.

REFERENCES

[1] Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. 2019. Learning Visuomotor Policies for Aerial Navigation Using Cross-Modal Representations. arXiv:cs.CV/1909.06993

[2] Carl Doersch. 2016. Tutorial on Variational Autoencoders. *ArXiv* abs/1606.05908 (2016).

[3] Vanessa Evers, Nuno Menezes, Luis Merino, Dariu Gavrila, Fernando Nabais, Maja Pantic, Paulo Alvito, and Daphne Karreman. 2014. The Development and Real-World Deployment of FROG, the Fun Robotic Outdoor Guide. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction (HRI '14)*. Association for Computing Machinery, New York, NY, USA, 100. https://doi.org/10.1145/2559636.2559649

[4] Hideaki Kuzuoka, Yuya Suzuki, Jun Yamashita, and Keiichi Yamazaki. 2010. Reconfiguring spatial formation arrangement by robot body orientation. 285–292. https://doi.org/10.1109/HRI.2010.5453182

[5] Yingzhen Li and Stephan Mandt. 2018. Disentangled Sequential Autoencoder. In *ICML*.

[6] Ross Mead and Maja Mataric. 2016. *Perceptual Models of Human-Robot Proxemics*. Vol. 109. 261–276. https://doi.org/10.1007/978-3-319-23778-7_18

[7] Hyun Soo Park, Eakta Jain, and Yaser Sheikh. 2012. 3D Social Saliency from Head-mounted Cameras. In *NIPS*.

[8] Martin Saerbeck and Christoph Bartneck. 2010. Perception of affect elicited by robot motion. 53–60. https://doi.org/10.1109/HRI.2010.5453269

[9] Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. 2015. CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 4423–4429.

[10] J. Vroon, M. Joosse, M. Lohse, J. Kolkmeier, J. Kim, K. Truong, G. Englebienne, D. Heylen, and V. Evers. 2015. Dynamics of social positioning patterns in group-robot interactions. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 394–399. https://doi.org/10.1109/ROMAN.2015.7333633

[11] Marynel Vázquez, Aaron Steinfeld, and Scott Hudson. 2016. Maintaining awareness of the focus of attention of a conversation: A robot-centric reinforcement learning approach. 36–43. https://doi.org/10.1109/ROMAN.2016.7745088

[12] Marynel Vázquez, Aaron Steinfeld, Scott E. Hudson, and Jodi Forlizzi. 2014. Spatial and Other Social Engagement Cues in a Child-Robot Interaction: Effects of a Sidekick. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction (HRI '14)*. Association for Computing Machinery, New York, NY, USA, 391–398. https://doi.org/10.1145/2559636.2559684

[13] Mohammad Abu Yousuf, Yoshinori Kobayashi, Yoshinori Kuno, Akiko Yamazaki, and Keiichi Yamazaki. 2012. Development of a Mobile Museum Guide Robot That Can Configure Spatial Formation with Visitors. In *Intelligent Computing Technology*, De-Shuang Huang, Changjun Jiang, Vitoantonio Bevilacqua, and Juan Carlos Figueroa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 423–432.