Learning to Orient in Group Conversations via Social State Abstraction

William Hu
william.hu@yale.edu
Yale University

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

1 ABSTRACT

Social state abstraction is the process of compressing an agent's social context into a lower dimensional space. These compressed representations allow us to not only extract meaningful social cues, but also build policies on top. One social scenario that is rich with context is group conversations. In a group conversation, agents must not only be able to understand individual social cues, but also group social cues. We propose using convolutional neural networks (CNNs) and variational autoencoders (VAEs) to learn motion-orienting policies from social cues in group conversations. In particular, we are interested in teaching a robot how to orient itself as a human would. To experiment in a realistic environment, we created a simulation in Unity that uses motions from real, human conversations. These conversations are composed of three individuals: a haggler, who is trying to sell a product, and two listeners. We generated a dataset that replaces one of the participants with our Pepper robot and trained three separate CNN architectures on that dataset. We attempted to learn policies for both the regression and classification versions of this task, evaluating our models in a testing environment with real-time simulations. Our results show that while CNNs are unable to learn a proper motion-orienting policy, they are at least able to direct the robot towards the speaker in the testing environment. On the other hand, VAEs seem to perform better and are able to learn some form of compressed representation. The discrepancy in the performance of the two architectures suggests that the CNN model lacks sufficient structure, preventing it from properly disentangling the social context.

2 INTRODUCTION

As research in artificial intelligence continues to progress, robots are becoming increasingly sophisticated real-world agents. In particular, social robots, or robots that specialize in social interactions, are beginning to behave more and more like humans, even adapting to our advanced social protocols. For example, in a 2018 study, Yoon et al. developed a deep recurrent neural network model that allowed robots to generate gestures while speaking [12]. They successfully applied their model to the NAO robot, showing that robots can emulate and reproduce human gestures [12]. Another example is Kaspar, a humanoid robot designed to teach children with autism how to view the world from other people's perspectives [11]. Kaspar used object recognition and 3D orientation tracking algorithms, so that it could think about perspective as humans would [11].

Despite the advancements in human-robot interaction (HRI), robots have continued to struggle in larger social settings, such as group conversations. There have been many studies on improving robot behavior in one-on-one interactions, such as applying seq2seq models [3] and Bayesian networks [8] to learn gestures, but the problem is slightly different when the conversation is held among a group. In such a scenario, the robot must be able to process group social cues on top of individual social cues, making the problem more challenging.

In this report, we propose deep neural networks for compressing group social states into lower dimensional representations. In particular, we are interested in learning motion-orienting policies for group conversations. To accomplish this task, we experimented with two classes of models: convolutional neural networks (CNNs), which are discriminative, and variational autoencoders (VAEs), which are generative. CNNs are feedforward neural networks that specialize in image processing. They are able to compress images into their most meaningful features by extracting feature maps at each convolutional layer in the network [13]. VAEs, on the other hand, are a well-established class of models that specialize in the unsupervised learning of complex distributions [1]. While standard auto-encoders are discriminative and learn a compressed representation of the input, VAEs are generative and learn a latent representation of the input. Because of their architecture, VAEs are effective models for encoding sequential data [5]. In particular, a study by Yingzhen Li and Stephan Mandt showed that VAEs are capable of separating dynamic features from static features in video recordings [5].

The rest of this report lays out the context and results of our efforts. Before training our models, we built a simulation environment in Unity and generated a new dataset using joint information from the Haggling dataset [4]. We trained 3 different CNN architectures on our dataset, attempting both regression and classification. While the CNN was not able to learn a proper motion-orienting policy, we did acquire key insights into both the model architectures and the fundamental nature of the motion-orienting problem.

3 RELATED WORK

Surprisingly, there is not a significant amount of previous work in this area. This is likely because robust, motion-orienting policies are difficult to learn, and the problem is made even more challenging in multi-agent social contexts.

3.1 Multi-Modal Approaches to Learning in Group Conversations

Previous work on learning policies in group conversations has relied on multi-modal inputs [6], [7]. To derive their motion-orienting policies, Matsusaka et al. uses a rule-based approach with constraints [6], while Matsuyama et al. uses partially observable Markov decision processes (POMDP) [7]. Despite the successes of their approaches, we would like to depend on only one modality. In addition, rule-based systems are rarely generalizable, making them less effective than modern AI approaches.

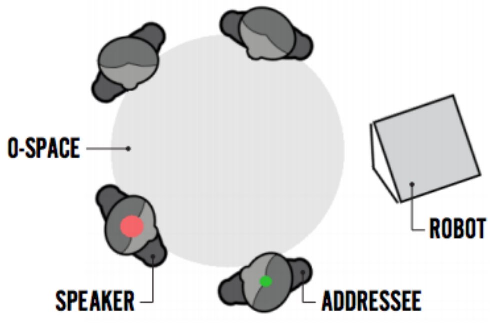

Figure 1: The simulation used in Vázquez et al. The robot has access to a camera and a microphone array for identifying the speaker [10].

In 2016, Vázquez et al. published a study that investigated the effectiveness of reinforcement learning in deriving motion-orienting policies [10]. As with the previously mentioned research, they trained their robot on multi-modal inputs. Their work showed that robots are able to learn robust policies after 3 to 7 minutes of training [10]. While their results are promising, their simulation only considers the group conversation in a 2D plane. We would like to extend their work to a more realistic setting with moving human speakers and a humanoid conversation partner.

3.2 Learning Motion-Orienting Policies via Deep Neural Networks

Historically, CNNs have shown great potential in learning motion-orienting policies. In a paper by Smolyanskiy et al., a low-flying drone was able to learn a motion policy via deep convolutional neural networks [9]. The model passes images captured from the drone to an 18-layer ResNet, which then extracts the relevant features for the lower dimensional representation. At the end of the model, there are fully connected layers that predict a 3D orientation and a 3D offset, which the drone then uses to reposition itself [9].

The success of [9] demonstrates that convolutional neural networks can be effective at learning motion-orienting policies from a single modality. We hope to build on their work by applying CNNs to a different task, namely orienting in group conversations. We would like to explore various CNN architectures, including those that contain ResNet layers. While our task is fundamentally different, there are many similarities in how we frame the problems and how we approach them.

4 METHOD

The project was divided into two main objectives: to first generate a motion-orienting policy using deep neural networks and then to apply that generated policy to a group conversation.

4.1 Joint Transfer

The conversation groups are composed of two human avatars and a Pepper robot. The human avatars were imported from Microsoft's

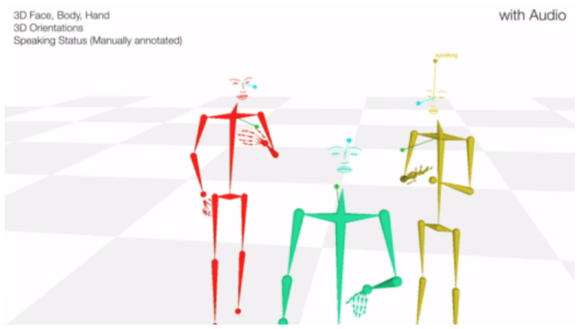

Figure 2: A visual representation of the Haggling dataset.

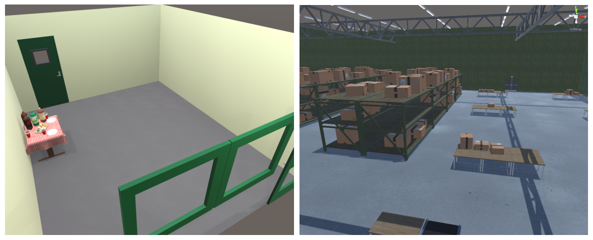

Figure 3: The two Unity scenes used in our experiments. The left image is the cocktail party scene, while the right image is the warehouse scene. We chose more than one scene so that the models would learn to ignore the environments.

Rocketbox library, while the model for the Pepper robot was provided to us from previous experiments in the Interactive Machines Group. To generate the poses of the Rocketbox avatars, we used joint information from the Haggling dataset, which was created by the CMU Panoptic Studio [4]. As shown in Figure 3, the Haggling dataset consists of conversations between three human participants, with joint information for each individual.

The Haggling dataset uses the OpenGL simulation in Python, while our setup uses Unity and the Robot Operating System (ROS), so we had to transfer the joint information from OpenGL to ROS. We chose a Unity environment for our experiments because the simulations were more realistic than the skeleton poses originally provided by the Haggling dataset. For joint transfer, we first mapped the joints of the Panoptic avatars to our avatars (including the robot) and then recomputed the positions and angles of the joints. To make the problem of learning a motion-orienting policy more tractable, we decided to limit the robot's action space, so that it only needed to orient its head and its body. For the robot, we converted the orientations of the Panoptic avatar's face and body into quaternions and then passed that information to our Unity simulation. The angles are relative to the previous time step, so the robot only has to predict how much to rotate instead of an exact orientation angle.

4.2 Data Collection

Once we had the simulation set up, we created a dataset using the robot's built-in camera. The Haggling dataset was recorded at 29.97

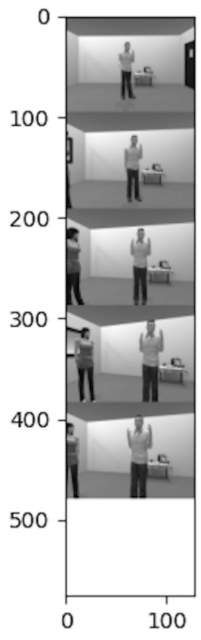

Figure 4: An example input from our generated dataset.

FPS, which is around the recording rate of the camera in Unity, but due to processing lag in our simulation, we decided to stream the dataset at 10 FPS. This meant that there were now multiple images per Haggling dataset frame. To create a 1-to-1 mapping, we simply chose the first image for each frame as the representative.

Once we synchronized the Unity camera images with the Haggling dataset, we began collecting data. We wrote scripts that used ROS to stream the Haggling dataset to our avatars in Unity and saved the images collected from the robot's camera as a rosbag. To increase the size of our dataset, we had the robot take turns as each of the three individuals in the conversation. We recorded a "medium-sized dataset" with 10 sequences in the cocktail party scene and a complete dataset with all of the sequences in both scenes.

4.3 Dataset Generation

After recording the data, we had to prepare the dataset for our models. To make our task simpler, we only considered the parts of a sequence where the robot was not speaking. The parts where the robot was speaking were spliced out. In addition, we converted the camera images to grayscale because we did not believe that color was important to our task.

In order to give the robot sufficient context, we augmented each image with the previous four images and added a "haggling" bit as a 6th "image." The haggling bit indicates whether the robot is the haggler and could provide additional context for how the robot should orient itself, as the haggler behaves differently from the other individuals. In addition, we decided to make the haggling bit as large as an image, so that the network would not just ignore it. An example is shown in Figure 4

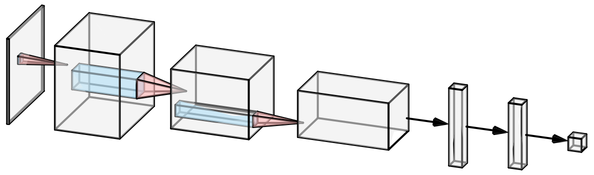

Figure 5: A visual representation of the CNN approach. For our 3 architectures, the number of convolutional and max pooling layers depends on the model, but the structure of the fully connected layers is the same. Made with http://alexlenail.me/NN-SVG/LeNet.html.

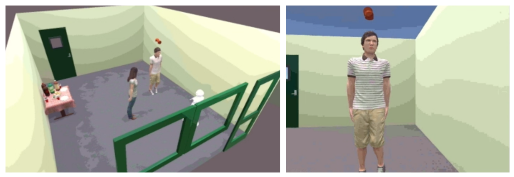

Figure 6: Images of our test environment. The left is the overall view of the conversation, while the right is the output of the robot's camera. The red marker indicates the speaker.

4.4 Building the Models

We explored two different model architectures: deep convolutional neural networks (CNNs) and variational autoencoders (VAEs). I built the CNN models, while Sally Ma, another student in the Interactive Machines Group, built the VAE models.

We explored 3 main architectures for the CNN model:

(1) A simple, 4-layer CNN
(2) A 50-layer ResNet [2]
(3) An unraveled, 10-layer ResNet

We started with the medium-sized dataset to make sure that our models were learning, before trying out the larger, complete dataset. We primarily focused on regression, training the CNN models to predict continuous action values from the input images. Towards the end, we reframed the problem as a classification task, training the CNN models to predict bins of action values. A diagram of the CNN model architecture can be found in Figure 5.

For regression, we used mean squared error (MSE) as the loss function, while for classification, we used categorical cross-entropy (CCE).

4.5 Evaluation

Before generating our dataset, we set aside 20% of the sequences in the Haggling dataset for testing. To evaluate our models, we looked at how the losses changed over training epochs and how the models performed in real-time simulations. For the latter, we built a testing environment that allows us to plug in our best models and visualize the robot's actions. An example is shown in Figure 6. The marker indicates which avatar is talking.

5 EXPERIMENTS

We trained various CNN models on the generated datasets and compared their performance to the VAE models.

5.1 Regression on the Medium-Sized Dataset

We first trained the CNN model on the medium-sized dataset. As shown in Figure 7, the CNNs do not really seem to be learning. The top lines are the validation loss, while the bottom lines are the training loss. The 4-layer CNN, which is the topmost graph, does show a downward trend in training loss towards the end, but further training did not decrease the validation loss. The 50-layer ResNet overfits on the data by epoch 28, likely because the model has too many parameters, while the 10-layer ResNet shows a steady decrease in training loss but no change in validation loss. In most of the models, the CNNs do not really learn to predict meaningful action values from the images. They just learn to predict an average angle that minimizes the loss.

The overall trends in the graphs suggest that the models are all overfitting on the medium-sized dataset. As a result, we experimented with dropout layers and L2 regularization to discourage overfitting. The extra regularization did not end up affecting performance significantly, and both the training and validation losses became flat lines.

The VAE model, on the other hand, seemed to perform better on the medium-sized dataset. As shown in Figure 8, the losses for both image reconstruction and the action predictions are decreasing over time. One potential explanation is that the lack of structure in the CNN model may be preventing it from learning effectively. In particular, the loss function of the VAE includes a regularization term based on the KL-divergence of the learned distribution Q and the true distribution P. This term adds additional structure to the model by keeping Q relatively close to P.

The best encoder for the VAE was the unraveled, 10-layer CNN, so for future experiments, we primarily looked at that CNN architecture for training and evaluation.

5.2 Regression on the Complete Dataset

Because the CNN could not learn a proper policy from the medium-sized dataset, we hypothesized that more data could improve its performance. We trained the CNN on the complete dataset, but the results did not improve significantly. As shown in Figure 9, the training and validation losses for the unraveled CNN are relatively flat. We ended up taking a deeper look at the complete dataset and noticed that there were problems with the vertical axis positions of the avatars in the conversation. It seems that Unity starts to bug out after running for a long time. As a result, we decided to use the medium-sized dataset for future experiments.

5.3 Analyzing the Action Space

Another hypothesis for why the CNN was not learning properly was that the action space is too skewed. The distributions of face rotation angles in the medium-sized dataset and the full dataset are shown in the top two images of Figure 10. As seen in the plots, there are significantly more data points around 0, which encourages the model to predict values that are always close to 0. In addition, we considered the possibility that the last 5 images are too close temporally to be helpful. The original dataset is recorded at 29.97 FPS, so the last 5 frames constitute about 166 milliseconds, which is a little too short for humans to move significantly. As a result, we decided to space out the images such that 10 frames of the dataset would pass between each pair of images. This created slightly better distributions, as shown in the bottom two images of Figure 10, but they are still skewed. We trained on this dataset as well, but the results were not any different.

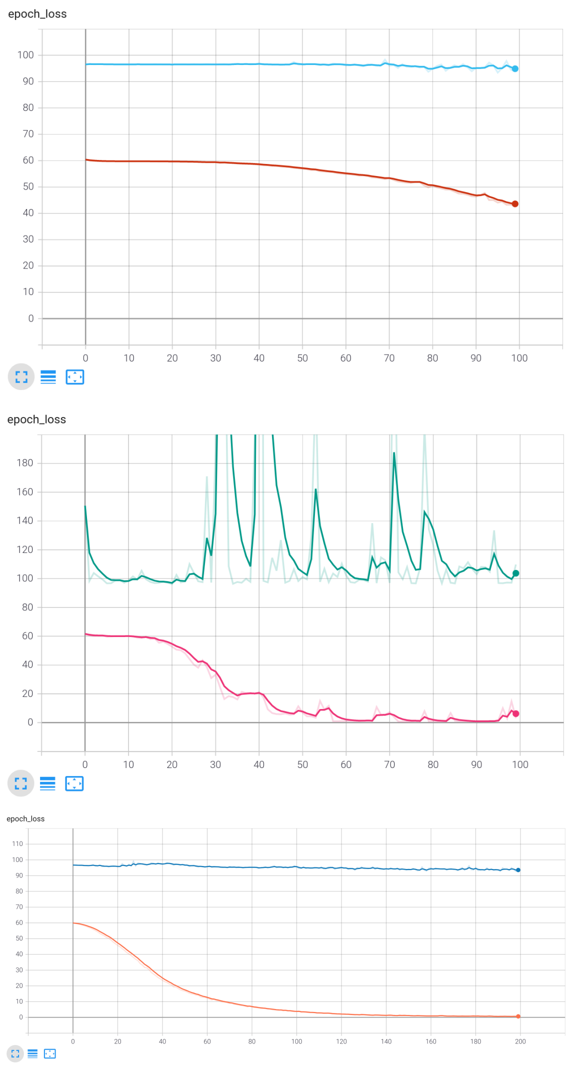

Figure 7: Loss graphs for CNN models trained on the medium-sized dataset. The upper lines are the validation loss, while the lower lines are the training loss. The topmost graph is the simple, 4-layer CNN, the second graph is the 50-layer ResNet, and the bottom graph is the unraveled ResNet. Despite their different structures, all of the models seem to be overfitting.

5.4 Classification on the Medium-Sized Dataset

With limited results on the regression task, we switched over to the classification task. We grouped the actions into 7 bins of width 10, with centers from -30 to 30. We ignore rotations greater than 30

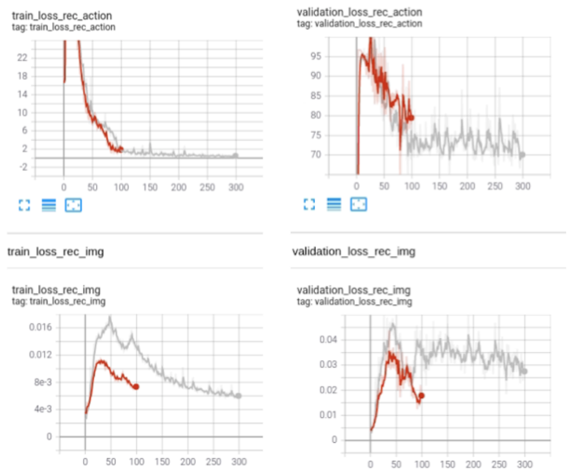

Figure 8: Loss graphs for the VAE model on the medium-sized dataset. The best encoder architecture was the unraveled, 10-layer ResNet. The red plots correspond to a model that was trained on 100 epochs, while the gray plots correspond to a model that was trained on 300 epochs. The VAE seems to be learning, given the downward trend of the losses. Credit to Sally Ma for the graphs.

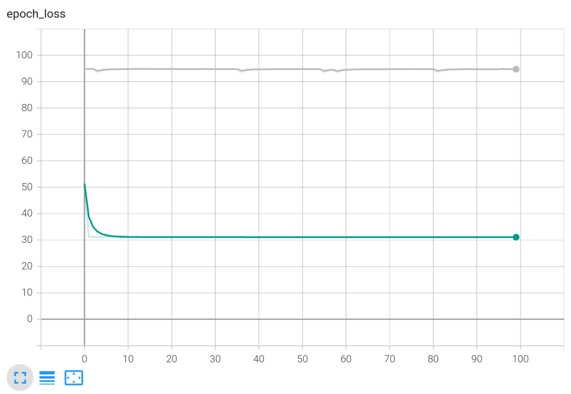

Figure 9: The loss graph for the unraveled, 10-layer CNN on the medium-sized dataset. The upper line is the validation loss, while the lower line is the training loss. Both lines are flat, which indicates that the model is not really learning.

degrees because each data point now spans 50 frames, and realistically, the robot should not have to turn more than 30 degrees in less than 2 seconds. To help with the skewed distribution of action values, we added weights to each of the bins, so that the classes would have equal weight when training.

Even with the modifications, the CNN model still struggled to learn a proper policy. As shown in Figure 11, the loss was decreasing, but the accuracy fluctuated wildly. More importantly, precision and recall both declined as the model trained, indicating that the model could not really differentiate between the classes. One potential explanation is that the dataset has many similar images with

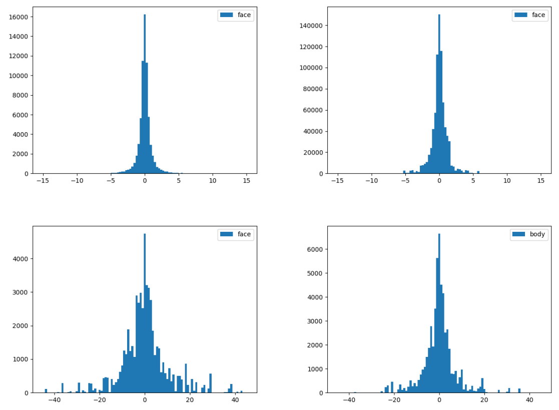

Figure 10: Graphs of the action space for the medium-sized and full datasets. The top left image shows the distribution of face rotation angles for the medium-sized dataset, while the top right image shows the distribution of face rotation angles for the full dataset. The bottom two images show the distribution of the face and body rotation angles for the full dataset after adding 10 frames between each pair of images.

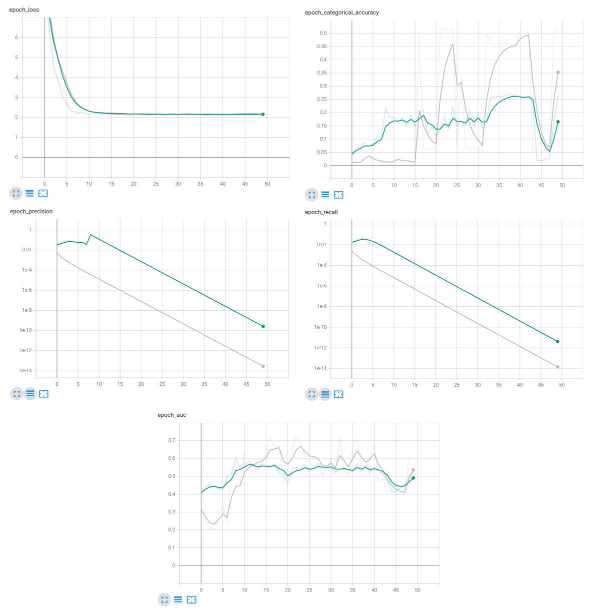

Figure 11: Metrics for the unraveled, 10-layer CNN trained to do classification. The green line is the training metric, and the gray line is the validation metric. While the loss does seem to decrease initially, the accuracy jumps around, and the precision and recall do not improve.

drastically different action values. While this is likely the case, there must still be overall trends in the data, as the VAE was able to learn from the medium-sized dataset.

5.5 Video Demonstration

Here is a video demonstration of our CNN model in action. This particular model predicts continuous actions and was trained on the complete dataset. Overall, the CNN does a reasonable job of orienting the robot towards the haggler, but the robot moves very slowly. As mentioned before, this is because the action space is dominated by values that are close 0. For reference, the video is playing at 8x speed.

6 CONCLUSION

Through our experiments, we investigated the effectiveness of CNN and VAE architectures in learning motion-orienting policies for group conversations. The CNN, despite its previous success in [9], was not able to learn a proper policy for this task. When trained on the medium-sized dataset, the CNN architectures all overfit. Even with aggressive regularization and dropout, the CNN models were not able to improve their performance significantly.

In an attempt to further simplify the problem, we switched to the classification version of the task, in which the angles that the robot could rotate were placed into 7 discrete bins. We redistributed the weights of each class as well, so that minority classes would have more weight during training. The accuracy of the model managed to reach around 50%, but the precision and loss graphs indicate that the model was not really learning.

The VAE model, on the other hand, showed more promise. Its loss functions were decreasing when trained on the medium-sized dataset, and the image reconstructions looked reasonable. One potential explanation for the discrepancy is that the CNN model lacks sufficient structure to learn properly. The VAE model is trained under more constraints, including a KL-divergence penalty that encourages its learned distribution to stay close to the true distribution.

Despite the flaws of the CNN, however, the robot does perform acceptably in the real-time testing environment. While slow, it orients itself towards the speaker in most sequences. The small predictions are likely a result of the skewed distribution of our training data, which has a much greater representation of angles close to 0.

Lastly, we have made our dataset publicly available. The dataset captures group conversations among two humans and a robot in a semi-realistic, simulation environment. The code for our project can be found here on GitLab. We have included the original code for collecting the data, along with the code for training our models and running the test environment.

7 FUTURE WORK

There are many potential directions for future work, from improving our experiments to adding additional constraints. One interesting idea is to augment the input so that it contains auditory input in addition to visual input. At the moment, the only input to the models are the images captured by the robot, but a second modality, like audio, could be provide extra information that improves general performance.

Another direction for future work is to experiment with alternative model architectures. Two potential options, in particular, are convolutional recurrent neural networks and reinforcement learning models. Convolutional RNNs, like ordinary RNNs, specialize in learning sequential information. The nature of a conversation is sequential, so there are likely many temporal signals that could be learned. Reinforcement learning, on the other hand, has seen previous success in [10] and is one of the go-to approaches for robotics tasks.

Finally, we could reattempt the regression task but upsample images with higher rotation angles. One problem we saw in our test environment was that the robot turns at small angles, due to the skewed distribution of our action space. This is one approach to redistributing the training data so that it is more even.

8 ACKNOWLEDGEMENTS

I would like to thank Professor Marynel Vázquez for advising me throughout this project. Her feedback and insights were instrumental to our success. I would also like to thank Nathan Tsoi and Sally Ma, who are other members of the Interactive Machines Group. Sally and I worked on this project together, but we focused on different tasks. The VAE experiments mentioned in this report were all conducted by her.

REFERENCES

[1] Carl Doersch. 2021. Tutorial on Variational Autoencoders. arXiv:stat.ML/1606.05908

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[3] Minjie Hua, Fuyuan Shi, Yibing Nan, Kai Wang, Hao Chen, and Shiguo Lian. 2019. Towards More Realistic Human-Robot Conversation: A Seq2Seq-based Body Gesture Interaction System. *CoRR* abs/1905.01641 (2019). arXiv:1905.01641 http://arxiv.org/abs/1905.01641

[4] Hanbyul Joo, Tomas Simon, Mina Cikara, and Yaser Sheikh. 2019. Towards Social Artificial Intelligence: Nonverbal Social Signal Prediction in A Triadic Interaction. In *CVPR*.

[5] Yingzhen Li and Stephan Mandt. 2018. A Deep Generative Model for Disentangled Representations of Sequential Data. *CoRR* abs/1803.02991 (2018). arXiv:1803.02991 http://arxiv.org/abs/1803.02991

[6] Y. Matsusaka, S. Fujie, and T. Kobayashi. 2001. Modeling of conversational strategy for the robot participating in the group conversation. In *INTERSPEECH*.

[7] Yoichi Matsuyama, Iwao Akiba, Shinya Fujie, and Tetsunori Kobayashi. 2015. Four-participant group conversation: A facilitation robot controlling engagement density as the fourth participant. *Computer Speech Language* 33, 1 (2015), 1–24. https://doi.org/10.1016/j.csl.2014.12.001

[8] Ross Mead and Maja J. Mataric. 2014. Perceptual Models of Human-Robot Proxemics. In *Experimental Robotics - The 14th International Symposium on Experimental Robotics, ISER 2014, June 15-18, 2014, Marrakech and Essaouira, Morocco (Springer Tracts in Advanced Robotics)*, M. Ani Hsieh, Oussama Khatib, and Vijay Kumar (Eds.), Vol. 109. Springer, 261–276. https://doi.org/10.1007/978-3-319-23778-7_18

[9] Nikolai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, and Stan Birchfield. 2017. Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 4241–4247. https://doi.org/10.1109/IROS.2017.8206285

[10] Marynel Vázquez, Aaron Steinfeld, and Scott E. Hudson. 2016. Maintaining awareness of the focus of attention of a conversation: A robot-centric reinforcement learning approach. In *25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016, New York, NY, USA, August 26-31, 2016*. IEEE, 36–43. https://doi.org/10.1109/ROMAN.2016.7745088

[11] Luke Jai Wood, Abolfazl Zaraki, Ben Robins, and Kerstin Dautenhahn. 2019. Developing Kaspar: A humanoid robot for children with autism. *International Journal of Social Robotics* (2019).

[12] Youngwoo Yoon, Woo-Ri Ko, Minsu Jang, Jaeyeon Lee, Jaehong Kim, and Geehyuk Lee. 2018. Robots Learn Social Skills: End-to-End Learning of Co-Speech Gesture Generation for Humanoid Robots. arXiv:cs.RO/1810.12541

[13] Matthew D. Zeiler and Rob Fergus. 2013. Visualizing and Understanding Convolutional Networks. *CoRR* abs/1311.2901 (2013). arXiv:1311.2901 http://arxiv.org/abs/1311.2901