Modular Behaviors Learned via Disentangled Feature Representations

Joshua Patterson[1] Garen Chouzadjian[2] and Julian Christensen[3].
https://github.com/tall-josh/fyp_diy_robo_car/tree/master/report

***Abstract*— In the following paper, we implement, evaluate and compare the ability of several machine learning models to infer the steering/throttle controls of a hypothetical RC vehicle using image data alone. We begin with a well-established architecture for control of autonomous vehicles, the end-to-end (E2E) model. We then investigate several generative models, namely; Auto-encoder, Variational Auto-encoder (VAE) and Beta Variational Auto-encoder (β-VAE), for learning efficient latent representations of the input data. After training each generative model, the encoder portion is frozen, and the embedding produced is used as the input to multiple independent fully connected networks (modules) that are then trained on steering and throttle behaviors separately. We analyze the generative model's ability to learn "disentangled representations", in the interest of improving a future autonomous vehicle's robustness to variation in input data, as well as the ability to learn new behaviors without impacting the performance of existing behaviors. Furthermore, we propose the design of a physical system, which we denote AVAE (Autonomous-VAE) that can be used to test the real-world performance of these architectures.**

I. ACKNOWLEDGEMENTS

Thank-you to Silverpond (http://silverpond.com.au/) for granting our team access to their development facilities and allowing us to pick your brains with all manner of machine learning questions. Thank-you to Andy Gelme and CCHS (http://www.hackmelbourne.org/) for providing hardware and software guidance and the data used to undertake this research. Thank-you to our supervisor Dr Chris McCarthy for guiding us through this project and keeping us focused.

II. INTRODUCTION

A. Feature Extraction and Entanglement

Designing an accurate predictive machine learning model involves determining a set of features that faithfully describes the data the model is presented with. Feature extraction is the method by which the dataset is reduced to it's most important and relevant attributes [1].

A major challenge within the machine learning community is in the extraction of features that generalize well to various domains [2]. Currently, implicit representations learned by many machine learning models appear to over-fit to the task being trained [7]. This phenomenon is commonly referred to as **feature entanglement** an example of which can be seen in figure 1. For a testing dataset that closely resembles the training dataset, feature entanglement is normally not worth considering, however this is not the case for many real-life scenarios. Scenarios in which we require a model to transfer knowledge so that it can learn numerous tasks fast, or reasoning about new data through combination of previously learned factors requires robust features that are disentangled from one another [8].

Fig. 1. Entangled dumbbell and arm features as illustrated by [4].

The aim of this research is to investigate whether multiple robotic behaviors can be independently trained using high level features extracted via an unsupervised learning model. The key motivations for conducting this research are to address the brittleness and poor scalability of Machine Learning (ML) models when tasked with incorporating additional behaviors after initial training has concluded.

We investigate the use of generative models to learn mappings from high dimensional (usually 2d video frames) input space to a low dimensional embedding space that captures the abstract structure of the original input data. Using this embedding we train multiple behaviors (steering and throttle controls) separately. We propose the advantages of this technique are 3 fold. 1) The abstract features learned by the generative models are of much lower dimensionality than the input data. This lower dimensionality allows for each individual behavior to be learned with a comparably smaller network hence reducing the amount of data needed for training. 2) The embeddings produced by the generative models may improve the properties necessary for knowledge transfer and generalization outside of the training data distribution 3) Additional behaviors may be learned independently from one another, and new behaviors do not effect the existing behaviors learned by the other modules.

III. RELATED WORK

A. Autonomous Driving with Deep Neural Networks

Present research into autonomous navigation techniques using deep neural networks (DNN's) offers a plethora of models trained on labeled datasets [15]. [16] describes these approaches as **behavior reflexive** as they map sensory input to driving commands. During training, a human drives the vehicle while the system records images paired with steering and/or throttle angles. [17] use this method to train a Convolutional Neural Net (CNN) to return steering angles that keep their vehicle close to the center lane of a road. We appropriate a similar implementation of this method [12] as a baseline for comparison against our modular approach.

B. Feature Disentanglement

Research into the disentanglement of feature representations has been explored through a variety of applications, including facial expression classification [9], text-generation [10] and object detection [6]. Many approaches have produced promising results, however the majority of models are supervised and thereby provide limited application to real-world scenarios where a priori knowledge is unavailable [7]. Recent work by [2], [3] [5] [7], [8] and most recently [14] suggest Variational Auto-encoders (VAE) [3], specifically the βVAE variant [7] may provide an unsupervised solution to the entanglement problem.

A βVAE is an unsupervised learning model designed to extract only the most essential features within the input data. Taking high dimensional input data, in our case, [120x160x3] dimensional video frames. The βVAE learns a mapping to an efficient abstract representation known as a latent (or embedding) vector, z. The latent vector z has a much lower dimensionality than the original input data, while still encoding the semantic information needed to reconstruct the image. We aim to test whether z can be used to learn multiple behaviors used to control a scale RC-Car.

IV. Methodology

In order to assess the performance of our modular architecture to the traditional end-to-end approach, the Donkey-Car baseline [12] was trained using frames of video extracted from an existing dataset uploaded to YouTube [13]. The data was gathered by remotely driving an RC car around a marked track. The steering and throttle commands were encoded as 10-bit binary numbers and superimposed on each frame of the video. An example of the labeled data can be seen in fig 2. The architecture of the Donkey-Car CNN is shown in fig 3. The network was ported from [12]'s Keras implementation to TensorFlow. The baseline model was jointly trained to perform both steering and throttle behaviors. Steering data was quantized into 15 bins and trained as a classification task, whereas the throttle labels were normalized and trained as a regression task. As per [12] the losses for both steering and throttle were combined with weightings 0.9 and 0.01 respectively. Adam-optimization was used with a learning rate of 0.001.

The encoder and decoder for the Auto-encoder, VAE and $\beta = 5$-VAE are identical. Their design is inspired by the architecture proposed by [14], however to reduce complexity they do not include the recurrent layer at this point. Firstly the models are trained to reproduce sample images taken from the same dataset used to train the Donkey-Car model. A small amount of Gaussian noise with random rectangular blocks is added to the input data in an effort to train a more robust model. An example of inputs and reconstructions can be seen in fig 5 on the next page and fig 4. After the models are trained, the encoder portion of the network is frozen. The frozen encoder is then used as a feature extractor whereby the embedding vector is passed as the input to smaller fully connected networks used to learn steering and throttle behaviors interdependently. For comparison both behaviors

Fig. 2. Sample frame from training data. The encodings from left to right are 'version', 'throttle', 'steering' respectively. Once the encodings have been extracted they are then cropped out so the model does not merely learn to read the encodings.

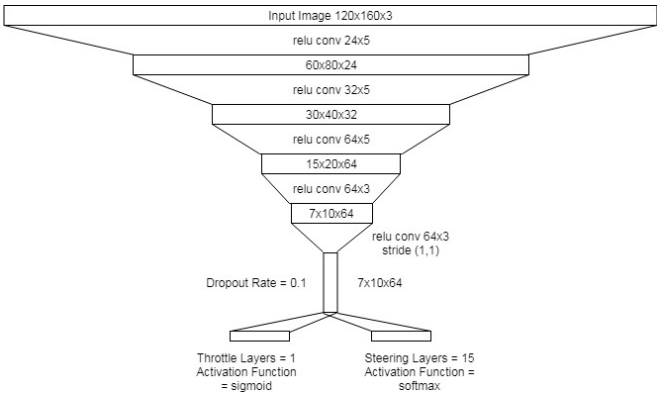

Fig. 3. Network architecture for the baseline Donkey-Car Network. All convolutions are square and have a stride of 2x2 unless indicated.

were trained using the encoders taken from the $\beta = 5$-VAE detailed in 6 on the next page as well as a vanilla VAE ($\beta = 1$) and a pure Auto-encoder.

Fig. 4. Top: sample input to encoder of Auto-encoder, Bottom: reconstruction produced by decoder.

A. Comparisons between embedding vectors

Some additional experiments were also conduced to investigate whether the latent space discovered by the $\beta(5)$VAE was in fact 'disentangled'. We hypothesized the $\beta(5)$VAE would have less 'active' elements in the embedding vector and these active elements should correspond to individual features within the reconstructions. To test this we take the

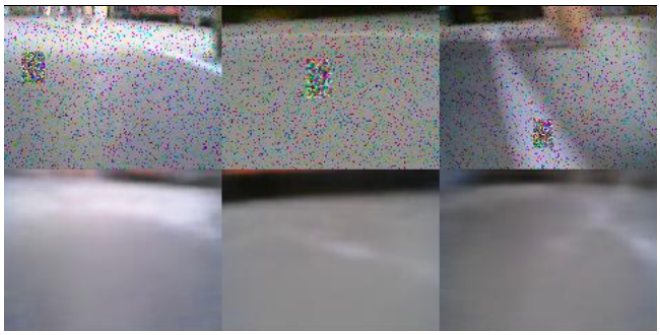

Fig. 5. Top: sample input to encoder of $\beta(5)$VAE, Bottom: reconstruction produced by decoder.

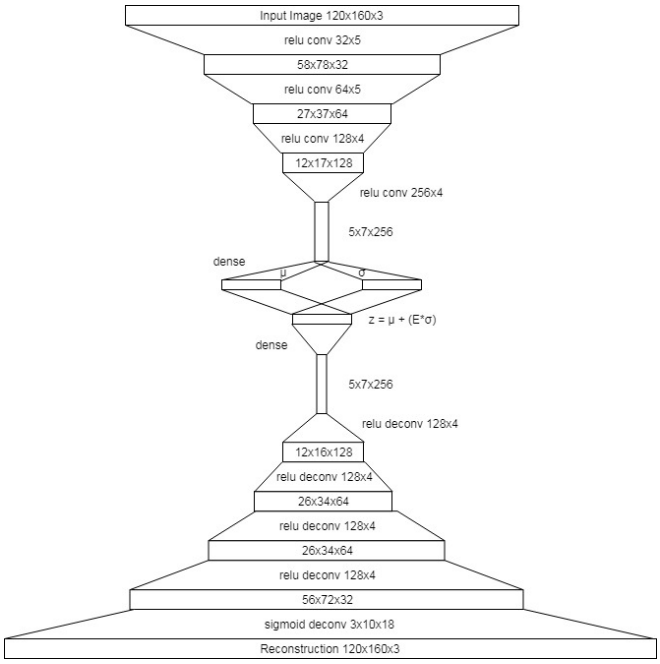

Fig. 6. Encoder - Decoder architecture for VAE and βVAE. The Auto-encoder also has an identical architecture, without the μ or σ sampling layers.

average embedding across the entire test set as a starting point. We then sweep each component of the embedding vector while holding the others constant and observe the resulting reconstructions.

The inverse of the above experiment can also shed light onto which elements are 'excited' by specific features. We took 1000 images from the steering data test set and passed them through the various encoder networks to compute their embedding vectors. We then plot the value of each element of the embedding against the steering command to observe whether any elements show correlation.

V. RESULTS

The end-to-end Donkey-Car baseline results for steering and throttle inference are shown in fig 7 and fig 8 respectively. Unfortunately, the range of throttle data was quite limited, with ground truth data only covering a range from 389 to 624 (approx 20% of the full 0 - 1023 range) making concrete comparisons difficult.

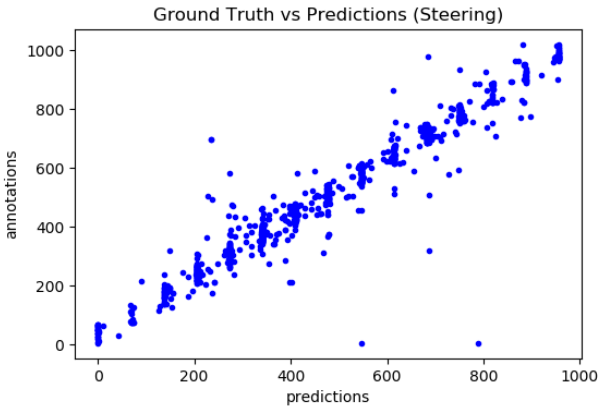

Fig. 7. Steering ground truth vs inference for Donkey-Car baseline. $\mu_{error} = 42.12 \approx 8\%$, $\sigma_{error} = 49.17$.

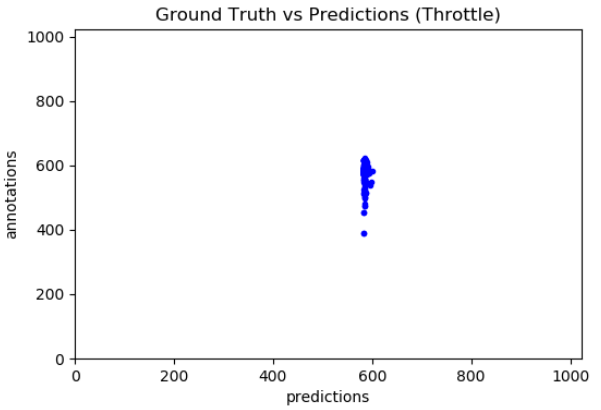

Fig. 8. Throttle ground truth vs inference for Donkey-Car baseline. $\mu_{error} = 8.31 \approx 1\%$, $\sigma_{error} = 15.93$.

To gain an understanding of the performance of the Donkey-Car network, we plot the inference of each frame sequentially, compared to ground truth data.

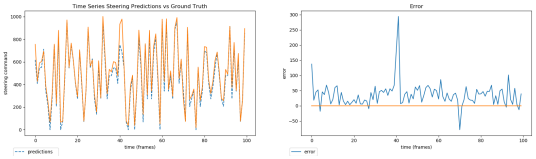

Fig. 9. LHS: Steering predictions wrt time (100 frames). RHS: Error wrt time

A. Behaviors

Using the trained encoder taken from the $\beta(5)$VAE as a feature extractor, a small fully connected classification network was connected as detailed below:

- `Encoder output (elements=50)`
- `Dense(neurons=40, activation=relu)`

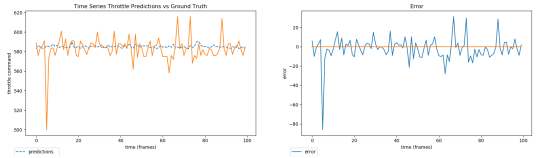

Fig. 10. LHS: Throttle predictions wrt time (100 frames). RHS: Error wrt time

- `Dense(neurons=30, activation=relu)`
- `Dense(neurons=20, activation=relu)`
- `Dense(neurons=15, activation=softmax)`

The steering behavior was then trained with the same dataset used for the end-to-end Donkey-Car network. For the throttle behavior the same fully connected architecture was used, however the final 15 neuron softmax layer was replaced with a single sigmoid node for regression. Again, for comparison, the VAE and Auto-encoder were also tested. Unfortunately, due to the low quality of the throttle data the results do not provide much insight so have been omitted. However, they can be found at our git repository.

Contrary to our initial presumption, we observed the correlation between the predicted and ground truth steering commands decreased from Auto-encoder to VAE and from VAE to $\beta(5)$VAE this can be seen in fig 11.

The time series prediction vs ground truth also exhibit the same pattern with average error increasing in the same fashion as the above mentioned correlations.

B. Embeddings

Finally in an attempt to observe the degree of disentanglement achieved by the $\beta(5)$VAE fig 13 on the next page shows the change in the reconstructed images as one element of the embedding vector is cycled from -6 to 6 with a step of 1. The figure attempts to visualize which elements of the embedding vector are responsible for specific features (if any). For comparison, we also observed the vanilla VAE and pure Auto-encoder. For conciseness only a selection of interesting embeddings are shown here, for the complete set of images see our git repository.

By inspection, we noticed, of the 50 elements in the embedding vector, the $\beta(5)$VAE utilizes approximately 12 in its reconstructions. With the VAE and Auto-encoder using 19 and 50 respectively. This is to be expected, as the addition of KL-Divergence loss in the VAE and the additional weighting applied within the $\beta(5)$VAE limits the networks ability to arbitrarily converge on features in place of a more structured embedding space.

Semantic features such as 'brightness' and 'leftness/rightness' seem to be captured by some of the elements in each style of encoder. What is interesting is that the $\beta(5)$VAE seems to have one element dedicated to 'brightness', and another to 'leftness/rightness' as opposed to the VAE and Auto-encoder whose embeddings appear more entangled.

Another test we conducted was an attempt to visualize any correlation between steering command and embedding

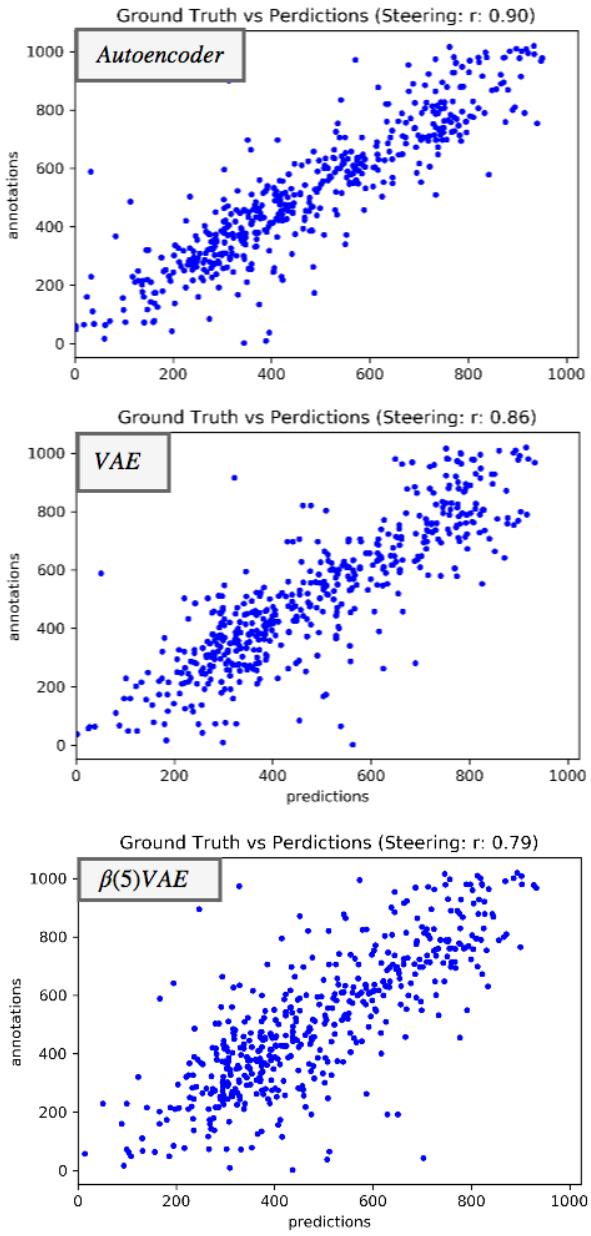

Fig. 11. Scatter plots of steering predictions from identical steering behavior networks with differing encoders used as feature extractors. X: Predictions, Y: Ground Truth

elements. However, upon inspection there seemed to be no observable relationship. fig 14 on the following page shows the correlation between element 49 of the $\beta(5)$VAE (the element observed in fig 13 on the next page to seemingly be related to direction) and steering annotations taken from the Donkey-Car test set.

VI. CONCLUSION

Our investigation into the use of disentangled embedding vectors indicates βVAEs do tend to learn fewer and seemingly more independent features. However, this disentanglement does come at the cost of quality reconstructions. This phenomenon was indicated by the literature [7], [8]. These

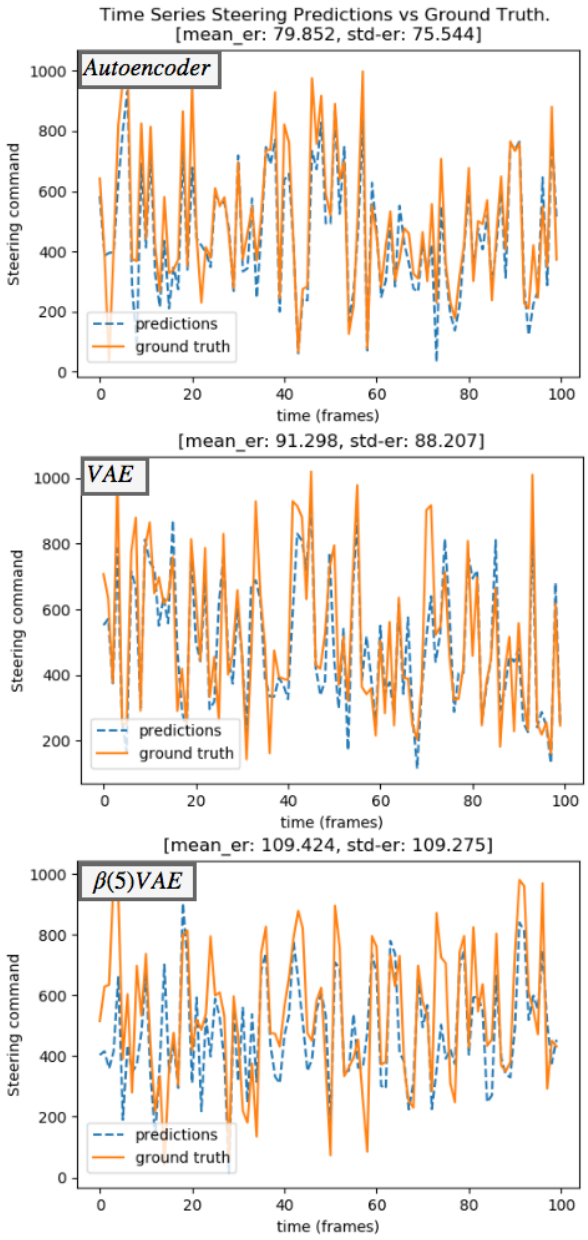

Fig. 12. 100 sequential frames of steering inferences produced by identical steering behavior networks using different encoders as feature extractors.

disentangled features however, did not seem to aid in the performance of steering or throttle behaviors. Our observations imply the quantity of less-optimal features is preferential to a smaller number of higher quality features. However, we demonstrate that the embedding vector produced by the encoder of an Auto-encoder, VAE or βVAE provides sufficient information to allow a regression or classification network to perform some level of inference. Whether the error is sufficiently low enough to control an autonomous RC car remains to be seen. A drop in performance between the fully connected Donkey-Car network and modular networks was expected. The drop can be attributed to the lower number of trainable parameters at the modular network's disposal.

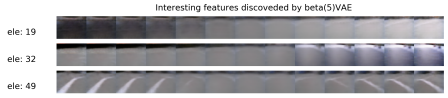

Fig. 13. Visualization of interesting elements extracted by the $\beta(5)$VAE. Each row begins at the center cell, then the element of the embedding vector associated with that feature is swept +- 6 with a step of 1 while holding all other elements stationary. The center images is the reconstruction of the mean vector over the entire test set.

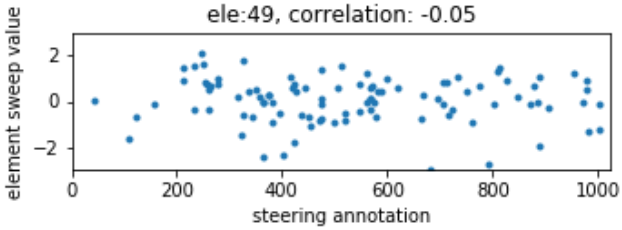

Fig. 14. Visualization of correlation between element 49 of $\beta(5)$VAE's embedding vector and steering annotations taken from the Donkey-Car test set. Correlation is taken as the Pearson correlation coefficient.

The Donkey-Car's steering component has 209,278 trainable parameters in comparison to the modular steering network with only 4,100. On the other hand the steering and throttle modules do benefit from a feature extractor that is more powerful than the Donkey-Car network. Recent literature [2], [14] does indicate the use of embedding vectors obtained via βVAEs can contain sufficient information for a network to perform complex tasks. Further investigation is required to produce a sufficiently effective encoder.

A. *Future Work*

Further investigation is required to assess the use of embedding vectors for use as feature extractors across multiple tasks and domains. Some future avenues for exploration are:

- Train the βVAE with additional noise and data from several domains ie: simulation, multiple tracks, camera perspectives, image augmentation. This may allow the encoder to learn more robust and useful features.
- Add a temporal component to the embedding. A Recurrent-NN can be used to learn temporal features within data. The use of RNNs and disentangled embeddings was recently demonstrated in [14]. The addition of temporal information may allow behaviors such as throttle control to perform more accurately.
- Insert additional sensor information into the behavior networks. Each behavior may also benefit from sensor data that contains information not easily conveyable in an image sequence, such as accelerometer or encoder data. These features may be added via an additional node feeding into the behavior modules as needed.
- Real world testing, until now only real world data gathered from a similar RC-Car has been used for training and testing. The natural next step is to implement the networks on AVAE, our purpose built RC platform. This platform is described in the next section.

B. Physical System - AVAE

The proposed physical system is shown in 15. A 1:16th scale car (TURNIGY 4x4 Mini Trooper) is to be equipped with a robotics platform board (BeagleBone Blue) and quad-core camera (JeVois Smart Machine Vision Camera). An embedded computer (Raspberry Pi 3) is to be used for data acquisition and communication over an MQTT server hosted on-board. A laptop (Intel Core i7 2.4GHz CPU, 8GB ram) running the Tensorflow API is to be used for implementation of the machine learning models and a GPU (Nvidia GeForce GTX 1080 Ti) is to be used for the training of said models.

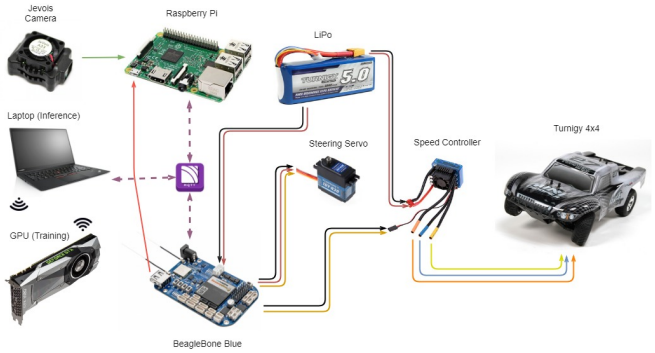

Fig. 15. System Setup. Dark Red/Black indicate +- power wires, light red indicates power over USB, yellow indicates PWM control signals, purple indicates data link over MQTT, blue/orange/light yellow indicate motor control signals, and green indicates data over USB

REFERENCES

[1] E. Alpaydin, "Introduction to Machine Learning", The MIT Press, p.110, ISBN 978-0-262-01243-0, 2010.*

[2] I. Higgins, A. Pal, A .A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, A. Lerchner. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. ICML, 2017.*

[3] D. P. Kingma, and M. Welling, Auto-encoding variational bayes. ICLR, 2014.*

[4] A. Mordvintsev, C. Olah, M. Tyka, Inceptionism: Going Deeper into Neural Networks, Google Research Blog, 2015. (research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html) sourced: Aug 15-17*

[5] C. Doersch, Tutorial on Variational Auto-encoders. Carnegie Mellon / UC Berkeley, 2016*

[6] G. E. Hinton et al, Transforming Auto-encoders, International Conference on Artificial Neural Networks, 2011, sourced: May 12-18*

[7] C. Burgess, I. Higgins, et. al, Understanding disentangling in -VAE, DeepMind London Apr 10-18, sourced: May 12-18*

[8] I. Higgins, et. al, -VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, ICLR 2017, sourced: May 12-18*

[9] Y. Bengio et al, "Disentangling factors of variation for facial expression recognition", Universite de Montreal, Department of Computer Science and Operations Research, sourced: May 10-18.

[10] M. Larsson, A. Nilsson, M. Kgebck, Disentangled Representations for Manipulation of Sentiment in Text, NIPS 2017 Workshop, sourced: May 12-18*

[11] T. Chen et al, "Isolating Sources of Disentanglement in Variational Auto-encoders", arXiv preprint arXiv:1802.04942, 2018, sourced: May 12-18*

[12] W. Roscoe, donkeycar: a python self driving library, https://github.com/wroscoe/donkey, sourced: Mar 09-18, commit: 04900dc*

[13] A. Gelme, "Autonomous Vehicles: Training video #1 2017-10-08", https://www.youtube.com/watch?v=akavVLkza-o, sourced: Dec 03-17*

[14] D. Ha, J. Schmidhuber, "World Models", (https://arxiv.org/pdf/1803.10122.pdf) sourced May 10-18*

[15] B Paden. et al, "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles", IEEE Transactions on Intelligent Vehicles, 1(1), 2016. sourced May 10-18*

[16] C. Chen et al, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving", In ICCV, 2015. sourced May 10-18*

[17] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars.", arXiv:1604.07316, 2016. sourced May 10-18*