Vision Based Positioning: Supervised Contrastive Learning for Task-based Robotics

Emran Yasser Moustafa, Fin Kruseman Aretz, Niek Schattenberg
CS4245 - Seminar Computer Vision by Deep Learning
Group 4

I. Introduction

AUTONOMOUS mobile robotics (AMRs) play an important role in a variety of fields, from industrial automation to domestic assistance. For effective operation, these robots must navigate their environments using an array of sensors and actuators. In particular, vision-based robotics have seen success due to the rich semantic information captured by cameras. Typical robotics systems consist of a combination of hand crafted software modules that collaborate to enable perception, localization, path planning, and more. There is a strong incentive for robotics engineers to transition away from handcrafted systems towards learning end-to-end policies for navigation, with the primary driver for this being the increase in performance promised with deep learning [1].

State estimation is a key component of many robotics systems. It is the task of predicting the robot's state at a point in time. An example of this would be predicting a robot's position in space, progress in a task, role in an environment, etc. The paper "Contrastive Learning for Enhancing Robust Scene Transfer in Vision-based Agile Flight" [2] proposes a novel deep learning based alternative to the task of state estimation, based on the principles of contrastive learning. This is achieved by leveraging the feature extraction capabilities of a convolutional neural network (CNN) to learn a function which maps visual input to an embedded feature space that effectively represents the robot's state throughout a task. The resulting visual embeddings are contain rich task-specific information which allows better end-to-end control policies to be learned. The paper also demonstrates the scene transfer capabilities of this approach. They posit that task-specific contrastive pretraining leads to policies that not perform better, but also are more robust when brought to new environments.

Representation learning techniques, such as contrastive learning, appears to benefit deep learning systems in a number of ways, such as improved semi-supervised learning [3], improved supervised learning [4], faster learning in reinforcement learning (RL) environments [5], better performance in goal-conditioned RL [2], etc. Importantly in [2], their approach not only improved performance, but also imposed some meaning on the otherwise unexplainable latent space. These benefits are especially important in end-to-end learning for robotics, where explainablity is key and learning a reliable end-to-end policy is difficult.

For our project, we aimed to investigate the usefulness of contrastive learning in robotics. Namely, we explore two fundamental research questions with our project:

- Does contrastive learning lead to better performing, more explainable task-specific embeddings?
- Does contrastive pretraining lead to better end-to-end policies?

To explore these questions, we target Formula 1 as our environment of choice. The F1 is well-fitting environment for answering these questions as the vehicles perform a clear task and operate in well known environment. These vehicles are also fit with front-facing onboard cameras, which provides us with video footage showing drivers following a similar route but from multiple novel view points. This allows us to train the model to understand that specific route while also being robust to disturbances such as other drivers, rain, camera blur, etc.

For our project, we learn from the first-person view recordings how the Red Bull F1 driver Max Verstappen drives during his final five laps of the Sochi track in the Russian Grand Prix 2021 [6]. Specifically, we aim to predict the track progress, the fraction of track that is passed, for a given frame from the drivers point of view. We formulate this a regression task from a single image input and train using the contrastive pretraining to try and help learn end-to-end task by first learning implicit position on the track.

Potential use-cases for this system extend far beyond the racetrack, however. In industrial settings, robots could learn and adapt to dynamic environments, such as warehouse robots navigating around human workers and obstacles to move pallets efficiently. In domestic environments, robots could navigate from room to room using a learned representation of the home, adapting to changes in the environment such as moving furniture or people. In more specialized applications, UAVs could inspect wind turbines by following learned routes that account for environmental variations, or underwater robots could routinely inspect coral reefs, ensuring robust performance despite changes in underwater conditions or the seabed.

II. Methodology

In order to test whether contrastive learning has an increase in performance, we formulate an end-to-end vision based deep learning task: predicting track progress from only an image.

This formulation means our approach consists primarily of two distinct parts. First, we learn a representative vision embedding using contrastive learning. Then use this embedding to predict track progress.

A. Vision Embeddings

We learn an embedding function that takes images as input and output a vector in the embedding space. Our objective is to learn a set of parameters that minimizes the distance between the embeddings of images taken at similar track progress states, while maximizing the distance between the embeddings of images taken at dissimilar track progress states.

If we frame this as a contrastive learning problem, at each instance we take an anchor image, I_{anchor}, and fetch a set of positive and negative images, I_{pos} and I_{neg}, respectively. A positive sample refers to an image taken from a state with similar track progress, while a negative sample refers to an image taken from a state with dissimilar track progress. Our objective then is to minimize the loss function Eq. 1. We decided to use a loss function similar to the one proposed in [4]. Instead however, we used cosine similarity to quantify the similarity between two embeddings as opposed to the dot product, with $\text{sim}(a, b)$ being the cosine similarity between vectors a and b. The variable tau, τ, is a hyperparameter typically referred to as the "temperature" in contrastive learning [7]. In short, this variable controls how much hard negative samples should be weighted; a low temperature punishing hard samples more. We set this value to 0.1.

$$\mathcal{L} = \sum_{I_{\text{anchor}} \in I} \frac{-1}{N_P} \sum_{p \in P} \log \frac{\exp(\text{sim}(I_{\text{anchor}}, I_{\text{pos}})/\tau)}{\sum_{i \in N \cup P} \exp(\text{sim}(I_{\text{anchor}}, I_{\text{i}})/\tau)} \tag{1}$$

B. Track Progress Prediction

In order to demonstrate the power of the learned contrastive embeddings, we target track progress prediction as a possible downstream robotics task that could be tackled with our approach. In this task, we aim to predict the fraction of the track that has been completed, given a frame from the vehicle's onboard camera. With this regression task the beginning of the track gets a value of 0 and the position on the track after the 20 seconds of video gets a value of 1. The five laps of the dataset are very similar in track progress over time, so the assumption is made that the progress in time represents the progress of the track.

A multi-layer perception (MLP) will be added on top of the vision encoder to go from the embedding space towards the track progress. The MLP needs to translate this embedding space into a number between 0.0 and 1.0.

The loss that is used is mean squared error (MSE). This loss is selected because it is a continuous function that can be used used in regression. Due to the assumption that time and track progress are correlated, we have a ground truth for the track progress namely the current frame number since the start of the lab, divided by the total amount of video frames in the 20 seconds.

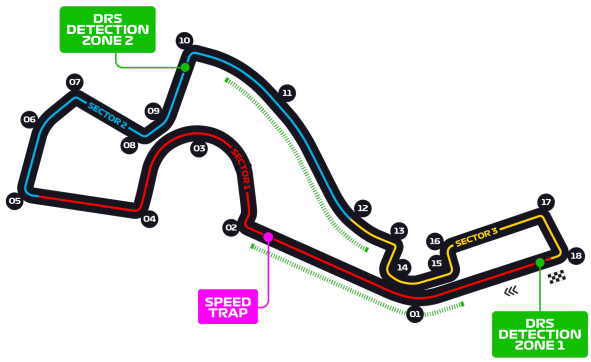

Fig. 1. Russian Grand Prix race track diagram [8]. Starting line indicated with a checkered flag. Turns and driving direction are also indicated. Source: [8]

III. Data

An ideal dataset for our project should contain multiple recordings of an agent performing the same task (i.e. a quadcopter flying through a set of hoops, a vehicle navigating the same route, robotic arm moving an object) with multiple viewpoints available for each stage in the task. This allows the learning algorithm to understand and adapt to changes in the environment, while still learning features important in distinguishing the state of the agent.

To this end, we have decided to use a recording of Max Verstappen completing the final five laps of the Russian Grand Prix [6]. The recording was taken from a front-facing camera onboard the vehicle. The recording is also available on YouTube. We believe this recording meets the criteria for a useful dataset for our application namely that there is sufficient variation between laps.

Our dataset contains a total of 14,025 frames, split into five laps of approximately 2800 frames each. Each lap is recorded at a rate of 25 frames per second. We manually split the video into five different laps using a video editing tool. Due to both time and compute constraints we decided to use only the first 500 frames of each lap. This represents the first 20 seconds of each lap or up until the third turn of the Sochi track, see Fig. 1. Furthermore, we down sample the video by half. The result is that each sequence contains 250 frames for use. We used three laps of our data for training, reserved one lap for validation and another for testing.

When we sample from our reduced dataset, we make the assumption that images with the same index are recorded at the same point in the track. We also assume the track progress increases linearly with the lap time. Finally, we resize each image from 720x1280 to 244x244. This is again due to time and compute constraints. A sample of our data and the corresponding labels can be found in Fig. 2.

IV. Experiments

In this section we demonstrate the effect of contrastive learning in the embedding space. And we conduct two experiments where we aim to predict the track progress, with and without contrastive pretraining. One experiment tests the learned embedding with a fixed backbone, whilst the second used the contrastive learning as task-specific pretraining.

Fig. 2. Resized frames from reduced dataset with corresponding labels. Labels range from 0.0 to 1.0 indicating the amount of track progress made at each frame, 0.0 indicating the first frame of a sequence and 1.0 indicating the last frame of a sequence.

A. Model Training Setup

1) Vision Embeddings: Following [2], we decided to use ResNet50 [9] with a reprojection layer [10] for the backbone for our vision encoder. For the ResNet50 backbone, we started with a foundational model trained on ImageNet [11]. We restricted the reprojection layer such that the output was a 128-dimensional vector representing our embedding space. For training, we decided to use 6 positive samples and 12 negative samples for each anchor image. Again following [2], positive images are sampled from a 10% window around the anchor image, restricted by the beginning and end of the sequence. Negative images are sampled from a 10% window around a point in the sequence which is 50% of the sequence length away from the anchor image, restricted again by the beginning and end of the sequence. Notably, this formulation implies anchors at the beginning, middle and end of the sequence have a smaller population of positive and negative images to sample from. We also use a batch size of 250 images, the equivalent to one down sampled sequence in our reduced dataset. We used an Adam optimizer with a learning rate of $1\mathrm{e}{-4}$ and a weight decay equal to $1\mathrm{e}{-4}$.

2) Track Progress Prediction: To train for track progress prediction, we used the same backbone as described in IV-A1. We then trained two heads to form the full model: one with a single dense layer head and another with a three layer dense head. All layers in the head used a 128-dimensional hidden layer with the output being a single neuron. We used an Adam optimizer with a learning rate of $1\mathrm{e}{-5}$ and a weight decay equal to $1\mathrm{e}{-4}$. In order to quantitatively evaluate track progress prediction, we calculate the root mean square error between the predictions and the test lap. We define root mean square error as shown in Eq 2, where y_{pred} and y_{label} are the prediction and label, respectively, and n is the number of samples in the test set.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_{\mathrm{label}_i} - y_{\mathrm{pred}_i})^2} \tag{2}$$

B. Contrastive Pretraining

We trained the vision embedding model for 200 epochs, with the model converging at 150. In total the final model trained off approximately 112.5K data points.

In order to evaluate the quality of the embedding, we used both quantitative and qualitative indicators. Firstly, we applied a t-SNE dimensionality reduction [12] in order to visualise the embedding for the test lap, seen in Fig. 3. One should note, the t-SNE algorithm applies non linear scaling and transformations to the output in order to identify clusters and display them in an understandable manner. Therefore, one must not take the final visualisation as representing the actual points in the embedding space. The linear relation of the samples may thus not be linear in the actual embedding space. Alongside this indicator, we also looked at the measurable difference between the embeddings for different points in the track. To do so we selected an anchor image and a set of other images from random points in the sequence. We then obtained the embeddings for these images and compared their cosine similarity. Each comparison therefore would have a similarity score and a track progress difference, ranging from 0% to 100%. This visualisation can be seen in Fig. 4.

C. Embedding Experiment

In order to investigate the amount of task-specific information held in these embeddings, we performed an experiment where we trained three models; one with the contrastive pretrained backbone, one with a backbone pretrained on ImageNet [11], and another with a random initialised backbone (i.e. no pretraining). We trained these models for the regression task

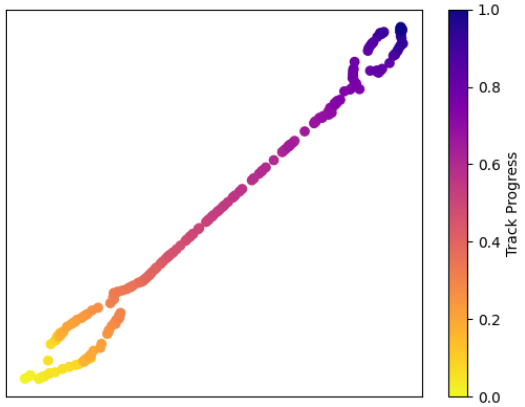

Fig. 3. Learned embedding of test data visualized using t-SNE. The colors correspond to the track progress ground truth of the samples. More yellow dots are from images with earlier track progress. More purple dots are from images with later track progress.

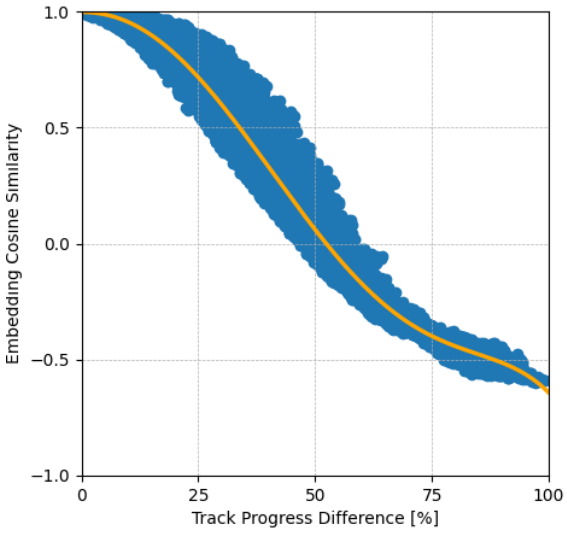

Fig. 4. Cosine similarity between the embedding of an anchor image and a comparison image. Track difference between the anchor image and comparison image is displayed on the horizontal axis. Cosine similarity between the two latent vectors is displayed on the vertical axis. Each comparison is denoted with a blue dot. A degree five polynomial function is fit to the data, displayed in orange.

of predicting the amount of progress made on the track so far. The labels ranged from 0.0 to 1.0, corresponding to 0% and 100% track progress.

As mentioned in IV-A2, we trained each model twice: first with a one layer head and then with a three layer head. In these experiments, we froze the weights of the backbone during training so as to keep the embeddings for the three networks the same. We performed no activation on the output of the reprojection layer before feeding it to the head. This has effect of the one layer head only being capable of performing a linear transformation on the output of the convolutional backbone. In comparison to this, the three layer head can perform a non-linear transformation on the output of the backbone. In order to evaluate performance of each model, we calculated the root mean square error between the predictions and the labels for each frame. The performance of each model can be found in Tab. I and Fig. 5.

TABLE I
EMBEDDINGS EXPERIMENT - ROOT MEAN SQUARE ERROR ON TEST SET, THE BEST RESULTS FOR EACH ARCHITECTURE ARE HIGHLIGHTED IN BOLD

	One Layer	Three Layer
No Pretraining	0.117	0.089
Standard Pretraining	**0.068**	0.033
Contrastive Pretraining	0.121	**0.031**

D. Pretraining Comparison

In order to investigate the value of learning a contrastive representation of the task as pretraining for learning a set of parameters for the end-to-end regression task, we decided to conduct the same experiment as in IV-C however this time not freezing the weights on the backbone during training. The contrastive learning done would then serve purely as pretraining for the task rather than as a pretrained learned embedding. We again used the root mean square error between the predictions and the labels for each frame as our performance indicator. The performance of each model can be found in Tab. II and Figs. 6.

TABLE II
PRETRAINING EXPERIMENT - ROOT MEAN SQUARE ERROR ON TEST SET, THE BEST RESULTS FOR EACH ARCHITECTURE ARE HIGHLIGHTED IN BOLD

	One Layer	Three Layer
No Pretraining	0.022	0.016
Standard Pretraining	**0.015**	**0.015**
Contrastive Pretraining	0.018	0.020

V. DISCUSSION

In this section, we aim to explain the possible reasons for the observed performance of our model, highlight weak points and assumptions in our approach, as well as identify area of possible future work.

Dataset assumptions: Our generation and use of the dataset makes a lot of assumptions that we were unable to validate. As mentioned earlier, in generating the dataset we manually divided the input video into five laps using a video editing tool. If our division of the laps is different by a handful of frames, this error propagates throughout the entire dataset. This is very possible as the original video was recorded at 25 FPS.

Moreover, this source of error affects how we use the dataset in training the track progress prediction model. The generation and use of the labels is done under the assumption that equal times in each sequence correspond to the same point on the track. This assumption is supported by the fact that we only used the first 20 seconds of driving in our reduced dataset. However, any variation in Max's driving within the first three turns of the track would cause our assumption to break. This is especially possible due to the presence of cars at the end of the last two sequences.

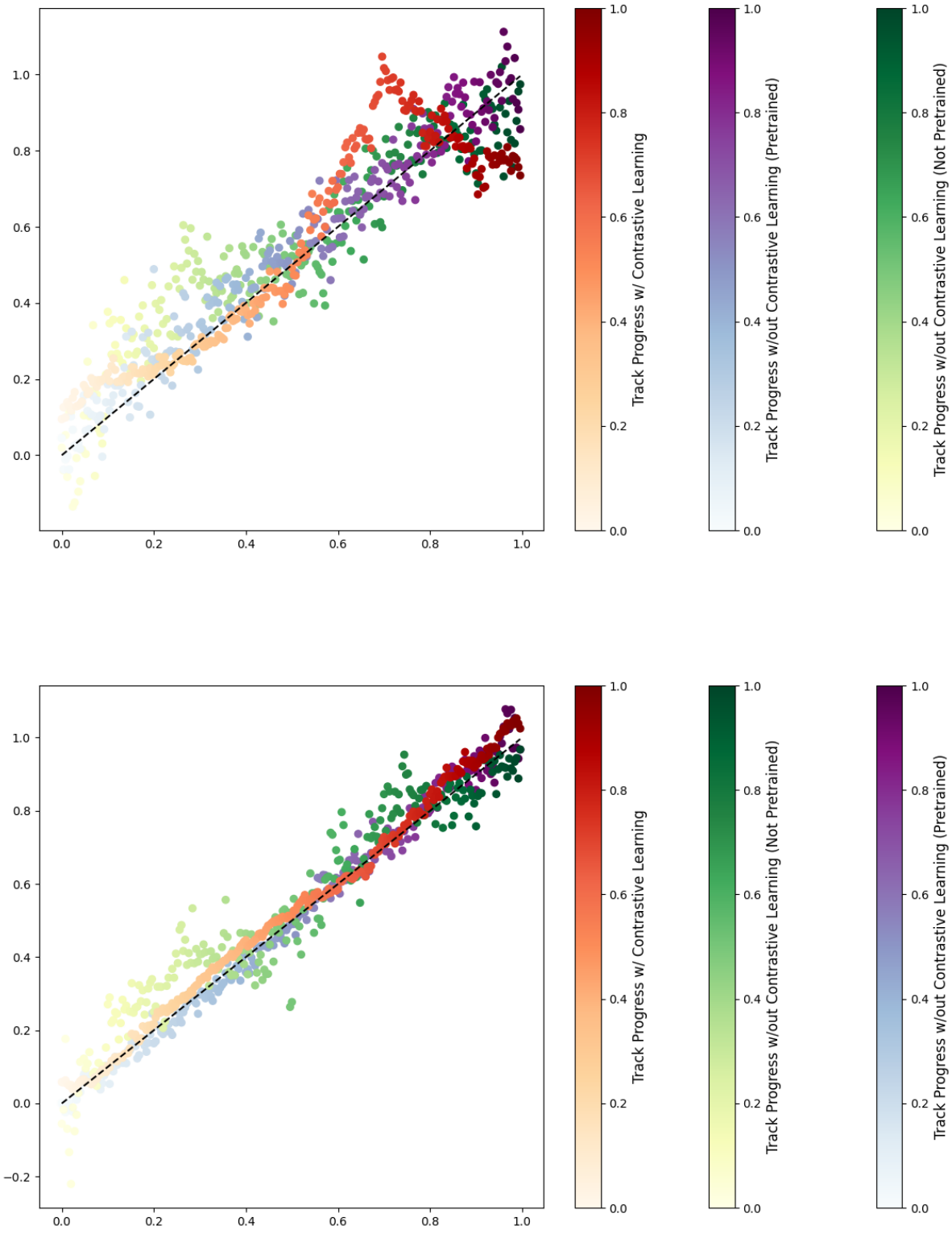

Fig. 5. Track Progress Prediction for embeddings experiment. One layer head models displayed on top. Three layer head models displayed on the bottom. Ground truth track progress displayed on the horizontal axis. Predicted ground truth on the vertical axis. Lighter colours denote earlier track progress, i.e. left on the horizontal axis. Darker colours denote later track progress, i.e. right on the horizontal axis. Model trained with contrastive learning coloured in red. Model trained with ImageNet pretraining coloured in blue-purple. Model with no pretraining coloured in green.

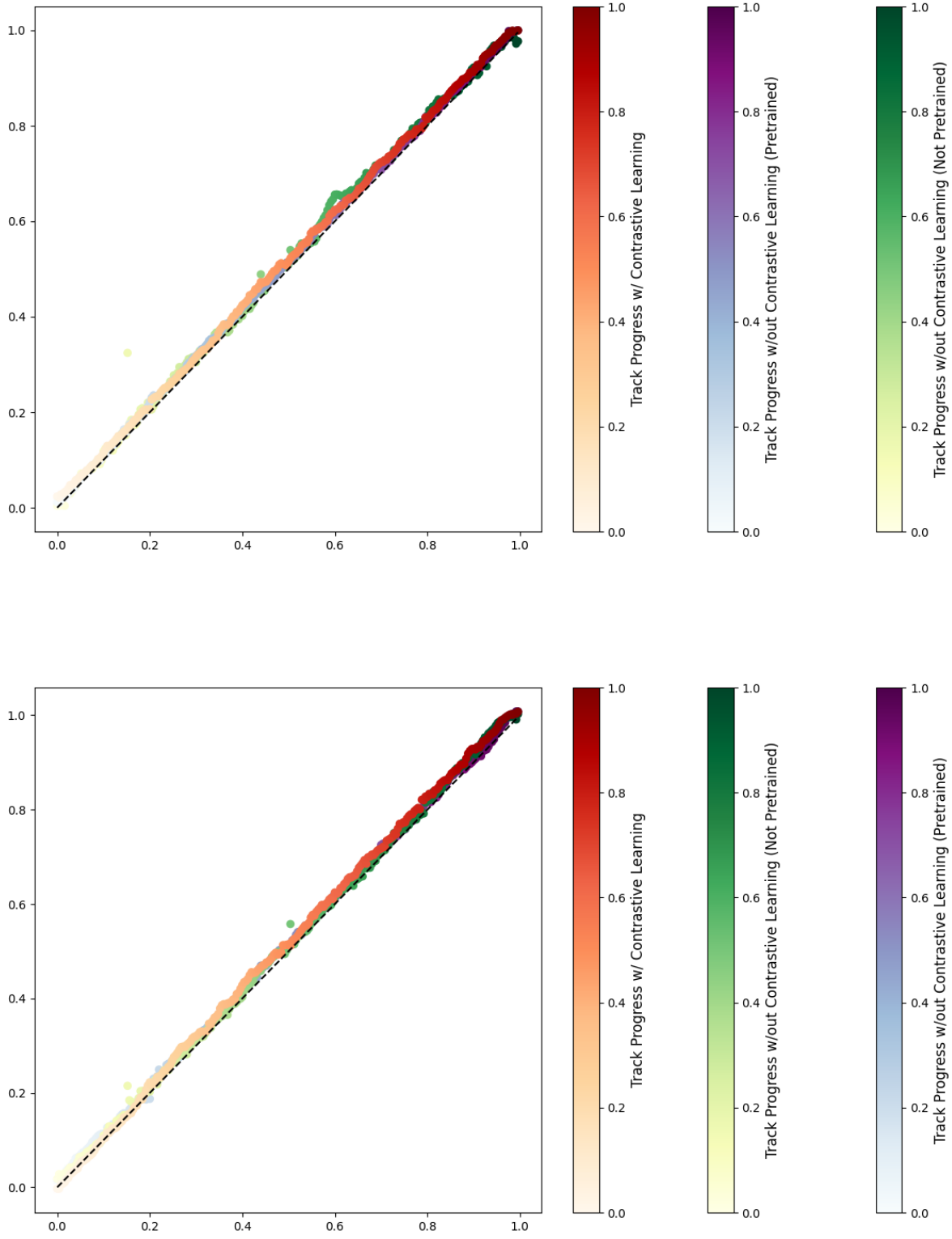

Fig. 6. Track Progress Prediction for pretraining experiment. One layer head models displayed on top. Three layer head models displayed on the bottom. Ground truth track progress displayed on the horizontal axis. Predicted ground truth on the vertical axis. Lighter colours denote earlier track progress, i.e. left on the horizontal axis. Darker colours denote later track progress, i.e. right on the horizontal axis. Model trained with contrastive learning coloured in red. Model trained with ImageNet pretraining coloured in blue-purple. Model with no pretraining coloured in green.

Poor dataset diversity: Another issue to raise concerns about is the diversity of the dataset. Due to the sparse amount of data, reliable and robust validation is difficult. Only a single sequence was used for testing, meaning each label could only be validated with one frame. This reasonably calls into question any intuitions concluded from these results.

Distance metric selection: In this project, we followed [2] and used cosine similarity as our distance metric during contrastive learning. This may not have been the best metric for our application, since our ideal learned representation of the task would be a line in the embedded space. A purely orientation based distance metric may not cause issues that can be seen using the evaluation methods we've used Section IV-B. However, issues may arise when using the learned embedding for prediction. We suspect that a distance metric such as dot product or weighted dot product may be a better fit for our application, as used in [4][5].

Embeddings experiment analysis: As seen in the results from the embeddings experiments, see Tab. I and Fig. 5, the contrastive pretraining we used in our approach had a negative effect when only a one layer head was used, but a beneficial effect when a three layer. Our intuition is that this is caused by the use of a linear (i.e. one-layer) head in combination with the use of cosine similarity as the distance metric in contrastive learning.

To clarify again, in the one layer head model we take the output of the dense reprojection head and feed it to another single dense layer, with no activation in between. This means the two dense layers behave the same as one dense layer. This restricts the model to only performing a linear transformation on the output of the backbone. As mentioned in V-*Distance metric selection*, the use of cosine similarity in contrastive learning causes positive and negative samples to have different orientation about the origin. This metric does not separate the points in the embedding in terms of absolute distance, making a linear regression to be fit to the data.

In support of this, when we look at the results of the three layer experiment we see the contrastive training has a lower root mean square error compared to both the single layer model and the other three layer model. We suspect this is due to the model being capable of learning a non-linear transformation to the embeddings in order to separate early track states from late track states.

If we were to pursue this project further, we would be interested to see the change in the one layer heads performance when a dot product is used as compared to cosine similarity. We would expect the disparity between the results from a one layer head and a three layer head to reduce.

Poor pretraining performance: According to the model performance on the test set, it appears that using the contrastive learning approach we propose has a negative effect on the models ability to determine track progress. We've drawn two intuitions from these results.

First, we believe the drop in performance when using a three layer head may be more prone to over fitting when compared to the the one layer alternative. The reason for this could be that a good representation is already learned in pretraining and therefore a smaller model is needed. This is supported if we compare the training loss between the two models, see III.

TABLE III
TRAINING LOSS BETWEEN ONE LAYER HEAD (TOP ROW) AND THREE LAYER HEAD (BOTTOM ROW) MODELS WITH CONTRASTIVE PRETRAINING. NUMBER OF TRAINING EPOCHS DISPLAYED IN TOP ROW. BEST PERFORMANCE AT EACH EPOCH INDICATED IN BOLD.

	100 epochs	200 epochs	300 epochs	400 epochs
One Layer	1.99e-4	8.21e-5	**3.37e-5**	**2.83e-5**
Three Layer	**6.83e-5**	**3.33e-5**	4.65e-5	4.14e-5

In our pretraining experiment, the single layer head model and the three layer head model converged after 217 and 205 epochs, respectively. Intuitively you would expect the larger model to require more training, so the three layer model converging quicker and at the point in which training loss is lowest could suggest a overfitting has occurred. More experiments would need to be done to confirm this, however.

Secondly, we believe that the model trained without contrastive learning perform well in comparison to those trained with contrastive learning due to them not learning task-specific information, but rather learning visual cues in the image that indicate the track progress. To illustrate this, we visualised the embeddings for the test set generated by the ResNet50 model with only pretraining on ImageNet (i.e. no task specific pretraining), see 7. Here we see that similar images are mapped together, not due task specific information. This is likely a failure in selecting a task that would display this kind of behaviour. If we were to continue our investigation into this problem, we would instead select a task that contains images that are semantically similar but have significantly different meanings in the context of the task. For example, data recorded from a vehicles taking different routes at an intersection may have similar viewpoints but very different tasks.

Notably, we also found that in [3] they found that fine tuning on a three layer reprojection head as opposed to a single layer reprojection head leads to lower performance, on the task of image classification.

We would also like to mention that the results obtained in this experiment could largely be influenced by faults in the dataset. As seen in the results in Tab. II, the models near perfectly predict the track progress labels. However in Fig. 6, it appears the models predictions appear slightly greater than the ground truth. If the recording for the test lap was started slightly later than those in the training set, we would expect to see results similar to what we observe.

VI. CONCLUSION

In this study, we have explored the application of supervised contrastive learning to enhance task-specific embeddings for vision-based robotics. Through our experiments, we found that while contrastive learning can offer some benefits in terms of representation learning, its effectiveness is highly dependent on the specific architecture and task design. Our results indicated a mixed performance between a three-layer head model and a one-layer head model in predicting track progress; while the three-layer model showed superior performance in

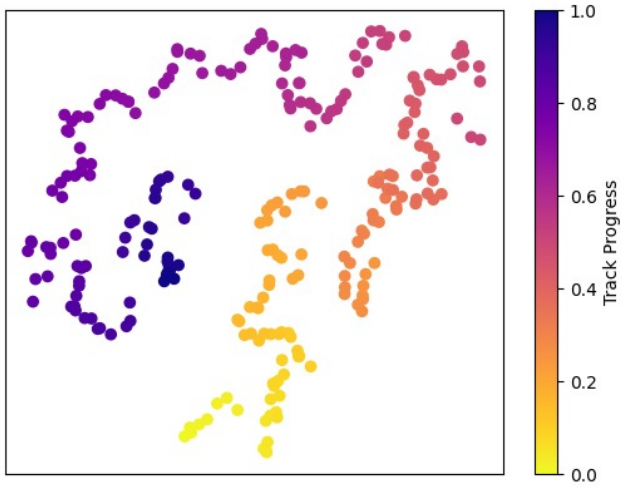

Fig. 7. Embedding of test data visualized using t-SNE of ResNet50 network pretrained on ImageNet. The colors correspond to the track progress ground truth of the samples. More yellow dots are from images with earlier track progress. More purple dots are from images with later track progress.

the embedding experiment (frozen backbone weights), the one-layer model outperformed it in the pretraining experiment (no frozen weights). This variability suggests that the optimal model depth may vary based on specific experimental conditions. We believe these results are indicative of the embedding space representation learned during contrastive training.

The analysis of embedding space experiment revealed that contrastive learning helps in creating more distinct and task-relevant representations. In the t-SNE graphs, the contrastive pretraining embedding shows a clear line where each point in the track lies close to the previous in the embedding whereas without shows large jumps from one position of the track to the next. However, it also highlighted the importance of selecting appropriate distance metrics and task-specific datasets to avoid potential pitfalls such as overfitting or inadequate generalization. Specifically, we speculate that the use of cosine similarity as a distance metric led to worse performance and perhaps an alternative distance metric, such as dot product, may yield better results. With this metric, the models without contrastive pretraining performed better in most cases.

Moreover, the study underscores the importance of carefully selecting datasets to effectively demonstrate the benefits of contrastive learning. The inherent errors in our dataset and the suboptimal task design limited our ability to confidently assess the advantages of contrastive learning. Future research should address these limitations by using more reliable dataset with more illustrative objectives.

References

[1] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[2] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," *arXiv preprint arXiv:2309.09865*, 2023.

[3] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[4] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[5] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 5639–5650.

[6] F. 1, "Max verstappen's final five laps in sochi — 2021 russian grand prix," 2022. [Online]. Available: https://www.youtube.com/watch?v=m0eASI6N6vE

[7] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2495–2504.

[8] Formula 1, "FORMULA 1 VTB RUSSIAN GRAND PRIX 2021," 2021, Accessed: 2024-06-16. [Online]. Available: https://www.formula1.com/en/racing/2021/russia/circuit

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[12] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.