JEPA Model For Two Room Navigation

Jack Chen
NYU Courant
zc2398@nyu.edu

Deniz Qian
NYU Courant
dq2024@nyu.edu

Matthew Dong
NYU Courant
matthew.dong@nyu.edu

I. Introduction

Our project focuses on developing and training a Joint Embedding Prediction Architecture (JEPA)-based model for an agent navigating a two-room environment. The model generates trajectories by processing sequences of observations and actions within a toy space consisting of two rooms separated by a wall with a gap in the middle. These representations are essential for capturing key elements of the environment, such as agent movement and the positions of walls and the hole in the wall.

The primary objective is to minimize the energy of predicted representations in latent space while effectively preventing representational collapse. Once trained, the model will be evaluated on its ability to accurately predict the agent's location across various scenarios. By undertaking this project, we aim to deepen the understanding of JEPA's applications in dynamic systems and explore solutions to real-world challenges such as efficiency and generalization.

II. Literature Review

Asides from the JEPA paper, we also reviewed the 3 papers that were provided in the project repository. The Variance-Invariance-Covariance Regularization (VicReg) method [1], introduces a self-supervised learning framework designed to learn meaningful representations without relying on negative samples or additional clustering steps. The variance term ensures that embeddings for a batch of samples retain sufficient variance along each dimension, thereby preventing representation collapse. Second, the invariance term promotes similarity between representations of augmented views of the same image. Lastly, the covariance term aims to reduce redundancy between embedding dimensions, fostering the extraction of diverse and informative features. VicReg's focus on decorrelation and feature diversity motivated us to explore this architecture. In our project, we used VicReg's principles to address challenges we encountered, like energy collapse and lack of diversity in embeddings. Specifically, the variance term helped our training stabilize by making sure the embeddings maintained sufficient variance. The invariance term aligned representations across different views of the two-room environment, reinforcing the model's ability to learn consistent spatial patterns.

The Bootstrap Your Own Latent (BYOL) framework, developed by Grill et al. [2], introduces an innovative self-supervised learning approach that removes negative samples, contrasting pairs, and clustering requirements. BYOL operates with two networks: an online network and a target network. The online network, comprising an encoder, projector, and predictor, is updated using gradient descent. In contrast, the target network, which consists of an encoder and projector, provides stable targets for the online network and is updated through an exponential moving average of the online network's weights. BYOL's paradigm focuses on predicting the representations of augmented views of the same image, minimizing the distance between predictions and target representations in its loss function. This approach eliminates the reliance on explicit negative pairs, reduces batch size requirements, and improves memory efficiency. BYOL's use of an online and target network structure could provide a robust way to stabilize learning without relying on negative pairs. This could help with our goal to simplify the training process. The framework is also not significantly affected by augmentation and batch size variations which could help with scaling our model and and using different regularization strategies in the future.

The work by Hadsell et al. [3] on dimensionality reduction explores learning invariant mappings that preserve the essential structure of input data while ensuring invariance to transformations. Central to this approach is the use of margin-based contrastive loss, which enforces a separation between similar and dissimilar data points by pulling similar points closer and pushing dissimilar points apart beyond a predefined margin. In our project, we introduced a contrastive margin error term into our loss function to enforce separation between embeddings of similar and dissimilar pairs within the two-room environment. This helped promote diversity among embeddings and ensured that the latent representations were discriminative. The use of contrastive loss also aligned with our goal of preventing overfitting.

As mentioned above, after analyzing these methodologies, we decided to implement the VicReg architecture as our initial approach. Its reliance on principled regularization to ensure the non-collapsing, invariant, and diverse nature of representations aligns with our research objectives. Moreover, VicReg's straightforward training paradigm provides a robust starting point compared to the more complex mechanisms employed by BYOL. The insights from Hadsell et al.'s work on margin-based contrastive loss will guide future modifications aimed at further enhancing our methodology. Of the three, we initially decided to begin by testing the VicReg architecture.

Thank you to the NYU HPC for providing great resources on how to use the system, and to the professors for a great class!

III. Methodology

In order to develop a working JEPA model, we explored various methodologies to address the challenges encountered during training. We initially utilized ResNet-18 and ResNet-50 as encoders in two separate versions of the model. These distinct configurations were aimed at evaluating the potential of different encoder architectures in addressing our objectives. However, both versions encountered critical challenges during training.

The version of the model using ResNet-18 consistently suffered from over-regularization. This version stagnated during training, unable to progress or generate informative representations. While regularization strategies were adopted based on established literature, their application in this context led to adverse outcomes. The version of the model using ResNet-50, on the other hand, consistently suffered from energy collapse. While this approach was theoretically sound based on the papers that we read, it proved ineffective in resolving the persistent issue of energy collapse, even after numerous adjustments to hyperparameters such as regularization strength, learning rate, dropout rates, and momentum. These setbacks prompted us to explore alternative methodologies before revisiting hyperparameter tuning.

As mentioned previously, we saw little benefit from the initial regularization; the variance of the model's embeddings was not decreasing as anticipated. To enhance the regularization mechanism, we introduced a contrastive margin error into the loss function inspired by the literature we read. This addition was designed to enforce a margin between positive and negative samples, thereby promoting greater separation and reducing variance in the embeddings. Contrastive margin error leverages the strengths of contrastive learning by explicitly encouraging the model to distinguish between similar and dissimilar pairs. This was expected to reinforce the regularization further and prevent variance stagnation.

We also noticed that our energy calculations were yielding excessively large values, which compromised the numerical stability of the training process and did not allow for effective optimization. To mitigate this, we recalibrated the energy computation by transitioning from the traditional L2 distance metric to Mean Squared Error (MSE). Switching to MSE was motivated by its ability to provide a stabler measure of energy, thereby preventing the explosive growth of loss values and ensuring reliable training progression. Despite these adjustments, the model continued to underperform, leading us to believe there were issues with the existing architecture.

As a result of our analysis of the BYOL paper, we decided to implement the BYOL architecture. We believed its predictor-based approach could improve the model's ability to align representations effectively between augmented views, potentially addressing lingering issues related to variance and feature alignment. BYOL's simplicity and demonstrated capacity to learn meaningful representations without relying on contrasting pairs aligned well with our goal of developing robust embeddings. However, by the time we decided to switch to BYOL, we were already deeply invested in the initial version of the model. The codebase had become increasingly complex due to the various adjustments and experimental additions we had implemented. Transitioning to a new model architecture required significant effort to adapt and integrate previous work into the new framework. Debugging and refining this implementation consumed a substantial amount of time, and despite our efforts, the BYOL-based model did not produce any meaningful results.

To explore temporal dependencies and sequential patterns in the data, we also experimented with incorporating Long Short-Term Memory (LSTM) units into the architecture. LSTMs are particularly well-suited for modeling sequential data, as they can retain and utilize information across extended time steps. We hoped that this modification would enhance the model's ability to capture temporal dynamics and improve overall performance. Unfortunately, this approach was also unsuccessful. Implementing and debugging the LSTM-based architecture required significant time and effort, further delaying progress without yielding any tangible improvements.

Following these challenges, we decided to return to using ResNet-50 as our encoder. Given the persistent issues with the earlier codebase, we opted to start fresh, rebuilding the model from the ground up to eliminate any hidden flaws or inefficiencies that might have hindered our progress. We implemented the BYOL framework with an encoder, expander, and predictor. Additionally, we incorporated contrastive margin-based regularization and kept the switch that we had done previously to mean squared error from L2 for energy calculations to ensure numerical stability. This reset allowed us to streamline our approach and focus on building a more robust and effective model. Through this process of exploring and refining methods, we worked to improve the JEPA model's ability to learn strong and useful representations. Each adjustment was based on practical observations and supported by theoretical understandings learned in class and from papers we read. In the end, the final architecture with ResNet-50 as our encoder gave us our best results yet. We then once again tuned our hyperparameters to try and get better results. These results are detailed in the next section of our report.

IV. Results

We also made a copy of the data and moved it to the scratch directories of the main Greene cluster in HPC, in order to be able to perform more tests. However, the training time per epoch rises from around 15-20 minutes per epoch on the burst cluster, to around 2 hours per epoch on Greene. This is a 8 times decrease in training efficiency, and therefore makes it unfeasible to train and test any of the models that we had created on Greene. This limited our ability to test as many different architectures and differing hyperparameters as we wanted. Eventually, one of our members ran into issues getting GPU allocation, even as the other members were able to obtain GPUs on the burst cluster. At a crucial time in the project, this also heavily limited our ability to test different model variations.

Fig. 1. Initial Resnet50 based model.

As displayed in Figure 1, our initial Resnet50 model did not perform very well, and barely learned any new methods of generalization. At base, the MockModel tends to have around 260 normal loss and 200 wall loss, so this model only improved by around 10 loss points. Initially, we thought that a decrease of 10 loss points could potentially be improved with hyperparameter tuning, so we tried many runs with a variety of different configurations. Our logic was that because the training loss was consistently collapsing below the validation loss within one single epoch, the model was very likely overfitting to the training data. We also tried different version of learning rate schedulers such as CyclicLR, StepLR, and CosineAnnealingLR, along with warm-up epochs. From previous experience, Adam as an optimizer tends to perform better than SGD, but for the sake of ensuring that we are confident which optimizer would be the best for our specific use case, we tested both.

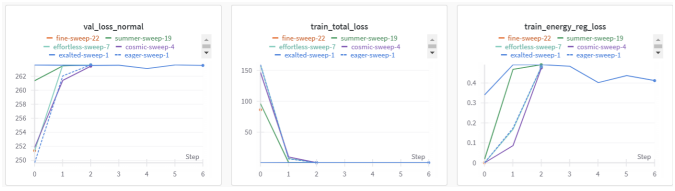

Fig. 2. Some results from our first tests. Every iteration of the model collapses.

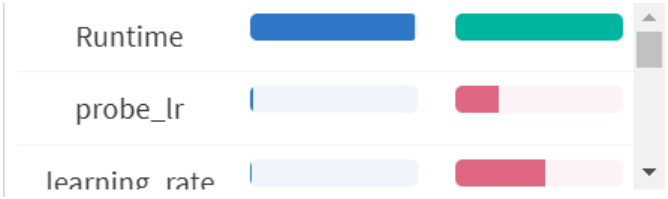

Fig. 3. Metrics from the first tests, suggesting that we should decrease our learning rate. We also decreased the learning rate because the model loss was collapsing so fast that it was definitely overfitting.

After referring to our metric outputs in Figure 3, we decided to decrease the learning rate. Despite the decrease in learning rate from $2e-4$ (the number referenced in the VicReg paper) to $1e-5$, the model still indicated signs of overfitting to the training data. As displayed in Figure 5, our loss values were also very unstable, exploding at the end of each epoch. This occurs for both energy loss and MSE loss. To mitigate this issue, we added gradient clipping, as well as further adjusting the learning rate with the learning rate schedulers that were mentioned previously. We were also running into issues where our energy loss was on a scale vastly higher than our variance loss, leading to issues in optimization. As a result, these methods did not produce good results.

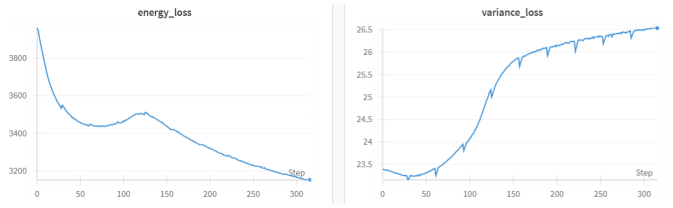

Fig. 4. Loss comparison. As you can see, the energy loss is on a scale vastly higher than the variance loss.

In order to be systematic with our hyperparameter tuning, we used Weights and Biases (wandb) to tune our model. This meant setting up the configuration for each model and testing each version of the model with the relevant versions of the hyperparameters filled in. Even after multiple runs testing these different variations and using grid search to go through all of the hyperparameters, none of the different variations would result in proper training. From this point onwards, we knew that we needed to rework our architecture.

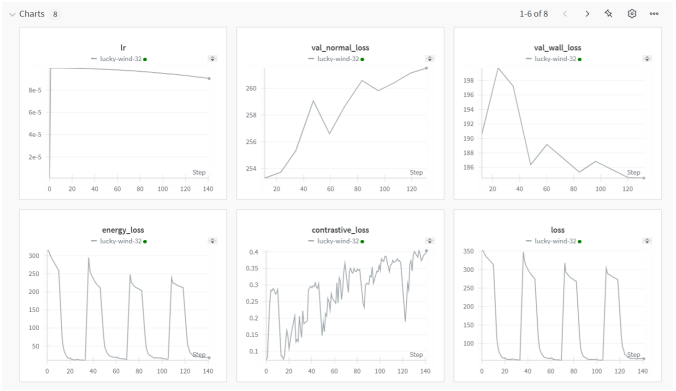

Fig. 5. Training statistics from one of our JEPA based models.

We did some preliminary tests in trying to implement the BYOL architecture. As mentioned in the methodology section, this meant replacing the entire model that we had set up previously, while still ensuring that the data dimensions were maintained. The BYOL model as shown in the paper needed extensive reworks in order to work with the dataset that we had been given for this project. However, even after these edits, the model was still not training properly, and loss was collapsing. The same was occurring for the LSTM based version of the model.

To expedite training and allow for more extensive experimentation, we aimed to utilize Distributed Data Parallel (DDP) from PyTorch. Successfully implementing DDP would have significantly reduced training time, enabling us to test a broader range of model configurations more efficiently. This could have helped us identify optimal hyperparameters faster and potentially spot architectural issues earlier in the process. The plan was to leverage resources such as 4 GPUs from Greene or 2 GPUs from Burst. Based on previous projects where we had successfully implemented DDP, we anticipated that this integration would be straightforward.

However, despite referencing code from our earlier projects and conducting numerous tests, we were unable to get DDP working with the current codebase. The issue seemed to stem from structural differences between the provided code and the examples we were referencing. This setback consumed valuable time that could have been spent on other aspects of the project, and unfortunately, the effort did not yield any tangible benefits.

In our final iteration of the model before the submission deadline, we were able to get some noticeable improvement in the model. Instead of collapsing immediately, the regularization techniques that we added to the model were able to keep the energy function from collapsing as soon as the model started training, so the validation loss was able to go down.

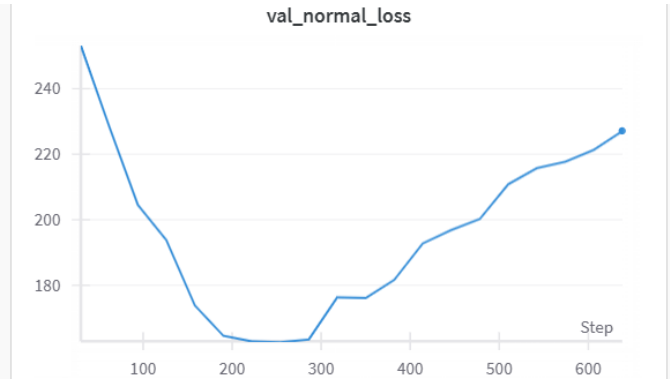

Fig. 6. The graph of the normal loss for our best model. The graph for our wall loss was very similar in shape, with different numbers.

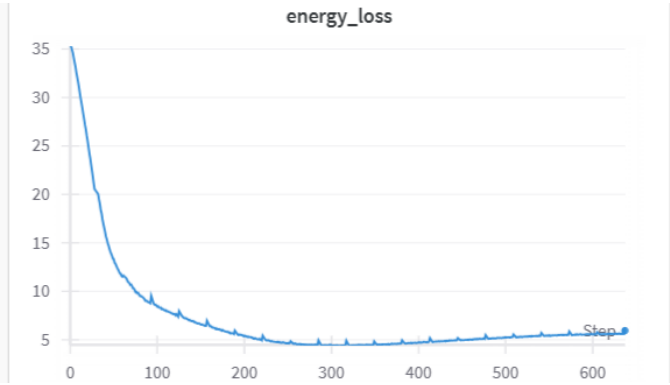

Fig. 7. Graph of the energy loss metric for our best model.

As training progresses, eventually the validation loss for both normal loss and wall loss stop decreasing and start going up again. There could be a number of reasons for this. It is possible that our regularization is not strong enough, and that the model still eventually collapses.

V. Future Work

With additional time to work on the project, we would have been able to further refine and improve the model. Although we made some progress and achieved improvements toward the end of the project, these advancements came too late to make a significant impact. Unfortunately, we were unable to implement or test further enhancements before the project deadline.

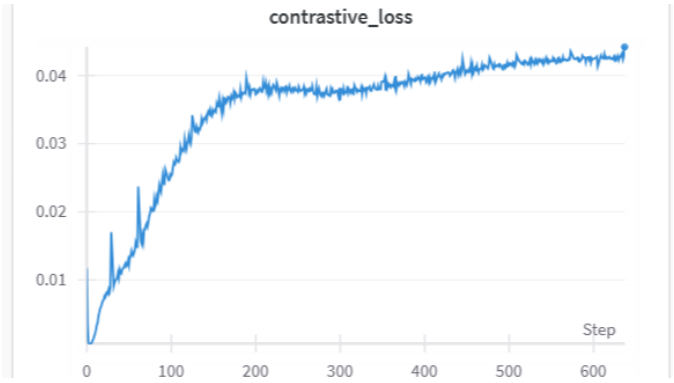

Fig. 8. Graph of contrastive loss for best model.

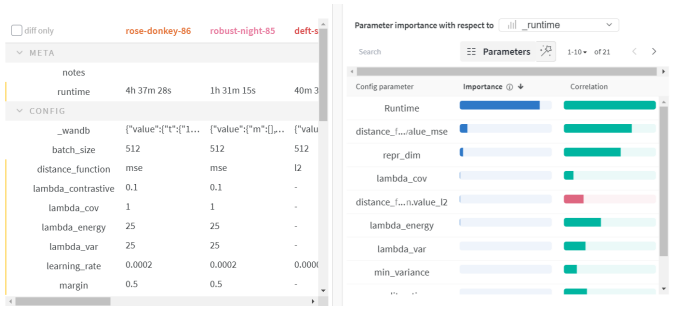

Fig. 9. Information on each of the metrics that we used in our best model and how they affect validation loss.

We would aim to utilize the information that we know about each of the metrics to adjust our hyperparameters accordingly. For example, it seems like increasing the dimensions of the model results in better validation loss by a pretty significant amount. Therefore, we would run the model again and test larger values of 'repr_dim'. We would do this for each of the hyperparameters seen, look at how the model performs on the validation datasets, and then continue to adjust accordingly until we get the best results that we can get with our current architecture. Once that process is complete, if the validation loss of the model is not low enough for our purposes, we were planning on going back and testing some of the architectures that not worked previously for us, now that we have a working version of the model.

References

[1] A. Bardes, J. Ponce, and Y. LeCun, "Vicreg: Variance-invariance-covariance regularization for self-supervised learning," 2022. [Online]. Available: https://arxiv.org/abs/2105.04906

[2] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," 2020. [Online]. Available: https://arxiv.org/abs/2006.07733

[3] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:8281592