Contrastive Learning for Gesture and Skill Recognition

Nishan Shehadeh and Devin Jean

1 Introduction

The goal of this project was to implement a neural network for modeling the latent space from reconstructing robot kinematics from endoscope images obtained during robot-assisted procedures. The first part of this project was training Wu et al's [2] encoder-decoder network before building classifiers and plotting UMAPs from the resulting embedding space. Our plan, as outlined in our project proposal, was to use the addition of contrastive learning to the model to explore whether a new form of loss will improve the embedding space's separability and classification from XGBoost.

2 Background

2.1 Clinical Background

The problem of surgical gesture segmentation is to segment the intervals of time in which a particular gesture (e.g., suturing, knot-tying, etc.) is being performed during a robot-assisted surgery, ideally in a real-time manor. Surgical gesture segmentation can be used to judge whether or not the optimal sequence of gestures is used during a procedure or potentially how well a gesture was executed, both of which could be useful for training new surgeons and evaluating their performance. Typically, multimodal approaches are used, with RGB video and robot kinematics being the most common raw inputs prior to any pre-processing steps (e.g., optical flow). Additionally, learning embedding space can be applied to imitation learning techniques through using the learned representations in simulations of surgical scenes. This requires disentangled representations where objects and the environment are invariant to lighting, background, etc. (e.g., Motion2Vec [1]). Overall, this type of gesture recognition can improve surgical quality and patient outcome by creating a more general understanding of surgical processes.

2.2 Original Approach

Wu et al.'s original paper [2] focused on the multi- and cross-modal possibilities of having both kinematic and video data. Their model uses a cross-modal learning strategy for self-supervised learning, so the network learns the embedding space without being introduced to labels. In this instance, the input to the encoder is optical flow (25 frames), the output of the decoder is the predicted kinematic parameter sequence (25 tokens), and MSE is used as a training loss between the actual and predicted kinematics vectors. We used the same pre-processing to obtain optical flow for each frame, using all of the same parameters and setup as in [2]. For the actual training and classification of the network, we used all of the default hyper parameters (e.g., lr = 1e-4, 1000 epochs, weight decay = 1e-8).

2.3 Contrastive Learning

Contrastive learning is a machine learning paradigm that juxtaposes samples against each other in order to group similar samples together. It is particularly useful for tasks such as creating embedding spaces that are easily classified because it helps create more separable embedding spaces. Contrastive learning is also quite effective for self-supervised learning because it can train models to learn embedding spaces without any annotations or labels. By compressing data into latent space representations and using contrastive learning to group similar samples within the space, the embedding space can learn the high-level features of the input.

2.3.1 Contrastive Loss

Contrastive loss is was the first training objective used for contrastive learning. The loss function is formed by minimizing the euclidean distance between "similar" images. Given images I_i and I_j and their respective embeddings x_i and x_j, there is a label $Y = 0$ if similar and $Y = 1$ otherwise, the loss is given by:

$$L = (1 - Y) * ||x_i - x_j||^2 + Y * max(0, m - ||x_i - x_j||^2)$$

Although this loss does implement the contrastive learning paradigm by minimizing the distance between similar pairs, it simply forces intra-class distances to be smaller than the minimum inter-class distance by forcing all samples of the same class to be within a certain margin m.

2.3.2 Triplet Loss

Unlike standard contrastive loss, triplet loss uses three samples: an anchor, positive sample, and negative sample. In order to minimize the distance between similar samples and maximize the difference between different samples, triplet loss minimizes the euclidean distance between the anchor and positive sample and maximizes the euclidean distance between the anchor and the negative sample. The loss for anchor I, positive sample I^+, and negative sample I^- with corresponding embeddings x, x_i, x_j is given by:

$$L = max(0, ||x - x^+||^2 - ||x - x^-||^2 + m)$$

Triplet loss is the best-performing loss for contrastive learning. It does not use a threshold to distinguish between classes, so it can more easily accommodate outliers and variance between classes. Contrastive loss is more restrictive because is uses the same set distance for negative samples (the margin) and urges anchors and positive samples into the same point. Additionally, using the same set distance for negative samples can be restrictive as in the original contrastive loss. Lastly, it is less greedy than contrastive loss because it does not change the distances in a positive cluster if there is no interference from negative ones.

3 Models

In order to test contrastive learning in our problem domain, we tried a variety of models using different data modalities and pairing strategies. As mentioned earlier, contrastive learning is built for self-supervised models, so we were able to test just using optical flow to make pairs and using kinematic data. We overall maintained the same encoder-decoder structure as the original paper. The encoder is 4 CNN layers followed by 2 full connected layers, outputting a $d_r = 2048$ latent representation, and the decoder is a 4 layer connected network outputting 25 tokens of 76 dimensional kinematic data. One additional note for all of our models is that we added another loss term to prevent the network from simply pushing dissimilar points further and further away to trivially minimize the loss toward $-\infty$ due to the subtracted term. Specifically, given an embedded vector B, we add the term $||B||^2$ to force embedding vectors to be approximately spherical in distribution and with a soft bound on magnitude (balanced by the contrastive loss term). We note that it is important that the "containment" loss be on the same order as the contrastive loss (as otherwise one would overtake the other, either exploding out to infinity or collapsing everything to a single point), but could be fine-tuned with a constant weight. For all models that were trained, the weight used was simply 1.

3.1 Augmentation Contrastive Model

Shown in Figure 1, our first contrastive network used only optical flow data and triplet loss to perform contrastive learning. Positive pairs were created by adding Gaussian noise (with covariance matrix Σ) to each sample. Negative samples were defined as any other sample in the batch. We encoded the samples and corresponding noisy samples before calculating the triplet loss on the embeddings. We retained the MSE loss on the decoded kinematics.

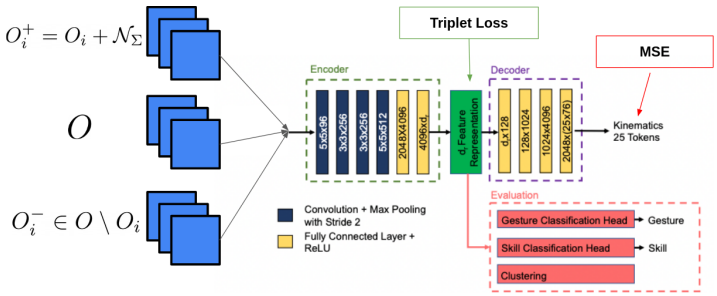

Figure 1: Contrastive Network 1 using Augmentation

3.2 Kinematics Contrastive Model

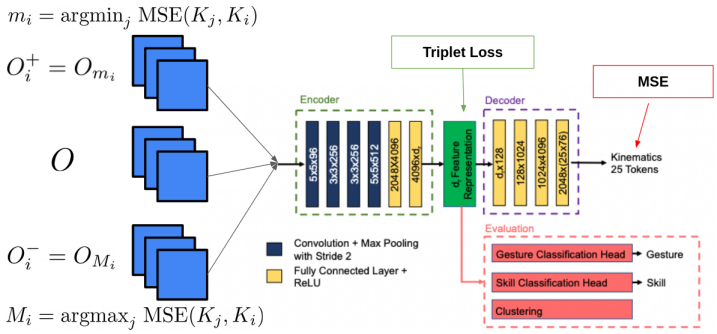

Figure 2: Contrastive Network 2 using Kinematics

Our second contrastive model, shown in Figure 2, used the actual kinematic data to create positive and negative samples. To capitalize on the multi-modal data, we used euclidean distance between the corresponding kinematic data for optical flow samples as a measure of the similarity between samples. For each anchor sample in a batch O, we found the corresponding most negative sample as the argmax of MSE between its kinematic data K and all other K_i for each sample O in the batch. Similarly, to find a positive sample, we found the argmin of MSE between K and all other K_i in the batch. We then used triplet loss to update the model based on these positive and negative samples.

3.3 Fourier Transform Kinematics Model

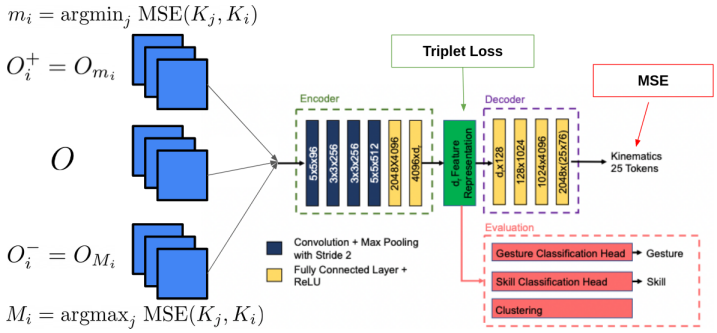

Figure 3: Contrastive Network 2 using Kinematics

One potential issue that was speculated with our second model was that we were using the MSE between kinematics sequences (which are a time series of 25 kinematics snapshots) to determine positive and negative pairs. However, shifting the components of a vector (e.g., by sliding time in the 25-frame kinematics sequence) yields an entirely different overall vector, despite representing (almost) the same moment in time and gesture being performed. Because of this, our third model uses an approach which

is invariant of time shifts in the 25-frame input/output sequence. Specifically, we employ the Fourier transform,

$$\widehat{f(t)}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-2\pi it\omega}dt,$$

which transforms the original problem in terms of time into a distribution in the frequency domain. Importantly, $\widehat{f(t)} = \widehat{f(t+c)}$ for any $c \in \mathbb{R}$ because $\hat{f}$ integrates out t over the entire domain $\mathbb{R}$. Thus, the Fourier transform of our kinematics sequence (applied along the time axis only) yields a time-invariant frequency spectrum that we can then compare by our existing MSE approach. Our model for this approach is shown in Figure 3. The only notable difference from the second model (aside from the Fourier transform pre-processing) is that the size of the last layer of the decoder is doubled. This is because $\hat{f}$ is a complex-valued function. Specifically, for a kinematics sequence K, the model learns to predict $\text{Re}(\widehat{K}) \mid\mid \text{Im}(\widehat{K})$, i.e., the concatenation of the real and imaginary terms of $\widehat{K}$.

4 Results

User (out)	Original	Augmented Contrastive Loss	Kinematic Triplet Loss	Kinematic FFT Triplet Loss
1	0.7361	0.6461	0.6127	0.5857
2	0.0137	0.0033	0.0031	0.0006
3	0.3423	0.3421	0.1846	0.2166
4	0.3126	0.3396	0.3596	0.3000
5	0.0324	0.0000	0.0036	0.0004
6	0.8831	0.9705	0.9574	0.9179
7	0.7486	0.7644	0.7133	0.7305
8	0.8579	0.8596	0.8734	0.9179
Avg (std)	0.4908 (0.3599)	0.4907 (0.3758)	0.4635 (0.3796)	0.4587 (0.3809)

Figure 4: Gesture Classification Results

The results from using the trained models we have described for gesture classification (using leave-one-user-out validation) are tabulated in Figure 4. Included are the "Original" results of the unmodified model from Wu et al. [2] with default arguments and hyper parameters. From this, the first thing that we notice is that the average classification accuracy of all the models is comparable, which unfortunately means that we have not demonstrated the improved classification accuracy that w hypothesized contrasting learning would provide. However, the standard deviations on this average value is extremely high due to the low sample size (8 users) and high range of values depending on which user was left out.

UMAP projections were generated for all models, and colored by various categories such as gesture, skill, and user; these projections are shown in Figure 5. From the raw results, we saw that there was poor performance for users 2 and 5 across the board (for all models). In the user column of UMAP projections, this corresponds to the orange and purple points. By cross-examining these points with the skill column of projections (same projection just different coloring), we see that these points correspond almost exactly to the points belonging to the orange "Intermediate" skill category.

5 Future Work

The most obvious avenue for future work overall is to modify the actual encoding and decoding networks for our models. More advanced embedding models can better account for long range dependencies and temporal information, which could prove extremely valuable considering we are passing in 25 frames of optical flow data.

Also, Motion2Vec presents an interesting paradigm involving using the labelled data in a dataset to create pseudo-labels for unlabelled samples. They use an LSTM to create these pseudo labels for unlabelled data and further refine the embedding space. Given the promising results of Motion2Vec,

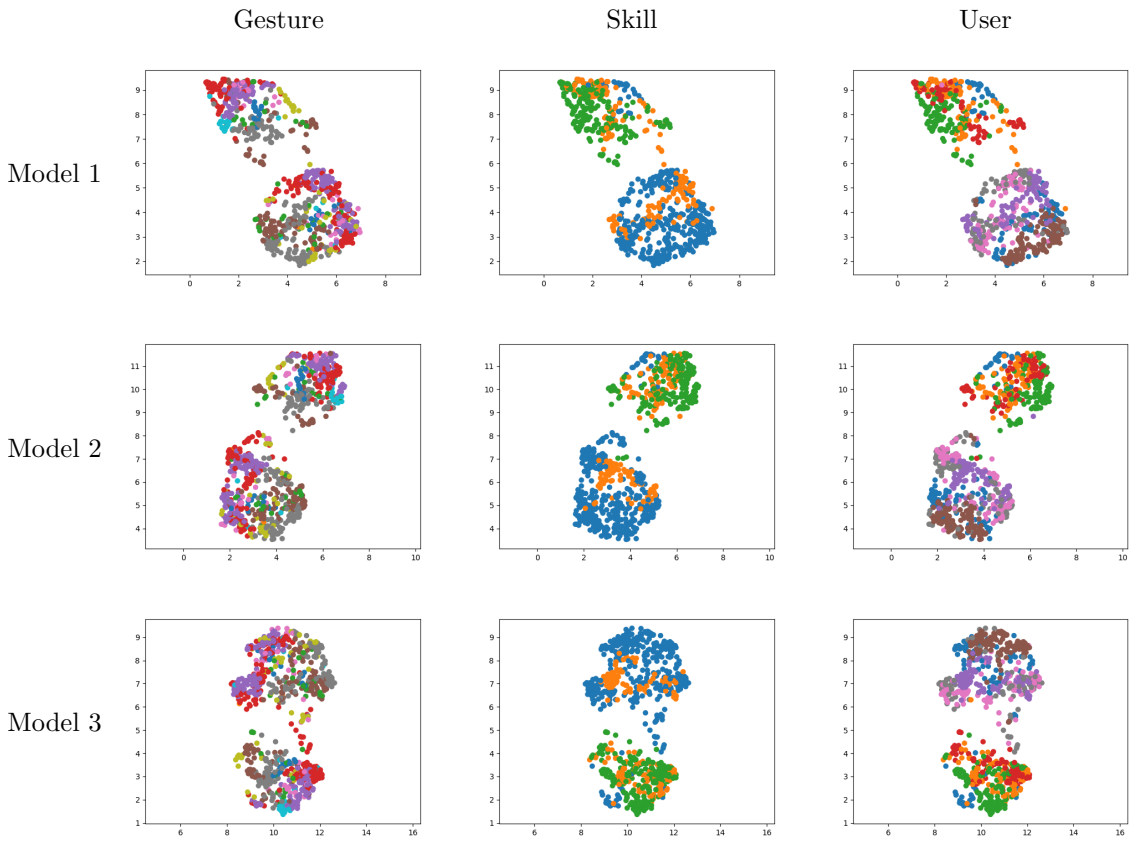

Figure 5: UMAP projections of resulting embedding space

using sequence learning of some type to aid in the creation of positive and negative pairs in the embedding space could prove beneficial for creating a well-defined representation space.

6 Conclusion

We have now introduced three different approaches to applying contrastive learning to train self-supervised embedding vectors that can be used for other purposes such as surgical gesture classification/segmentation. Unfortunately, from these results we do not see the improved gesture classification accuracy that we hypothesized contrastive learning would provide. However, we believe that this can be explained by the small data size. Specifically, we saw that every model performed poorly for users 2 and 5, which were the (only) intermediate skill level users. The reason for this could be that different skill levels perform gestures differently, even to the point that the union of novice and expert data is insufficient to properly generalize to intermediate users, despite the seeming overlap in the 2D UMAP projection. A larger dataset with more surgeons of varying levels could perhaps be used to overcome this dataset issue, at which point these contrastive models might be more effective.

References

[1] Tanwani, A. K., Sermanet, P., Yan, A., Anand, R., Phielipp, M., and Goldberg, K. Motion2vec: Semi-supervised representation learning from surgical videos. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 2174–2181.

[2] Wu, J. Y., Tamhane, A., Kazanzides, P., and Unberath, M. Cross-modal self-supervised representation learning for gesture and skill recognition in robotic surgery. *International Journal of Computer Assisted Radiology and Surgery 16*, 5 (2021), 779–787.