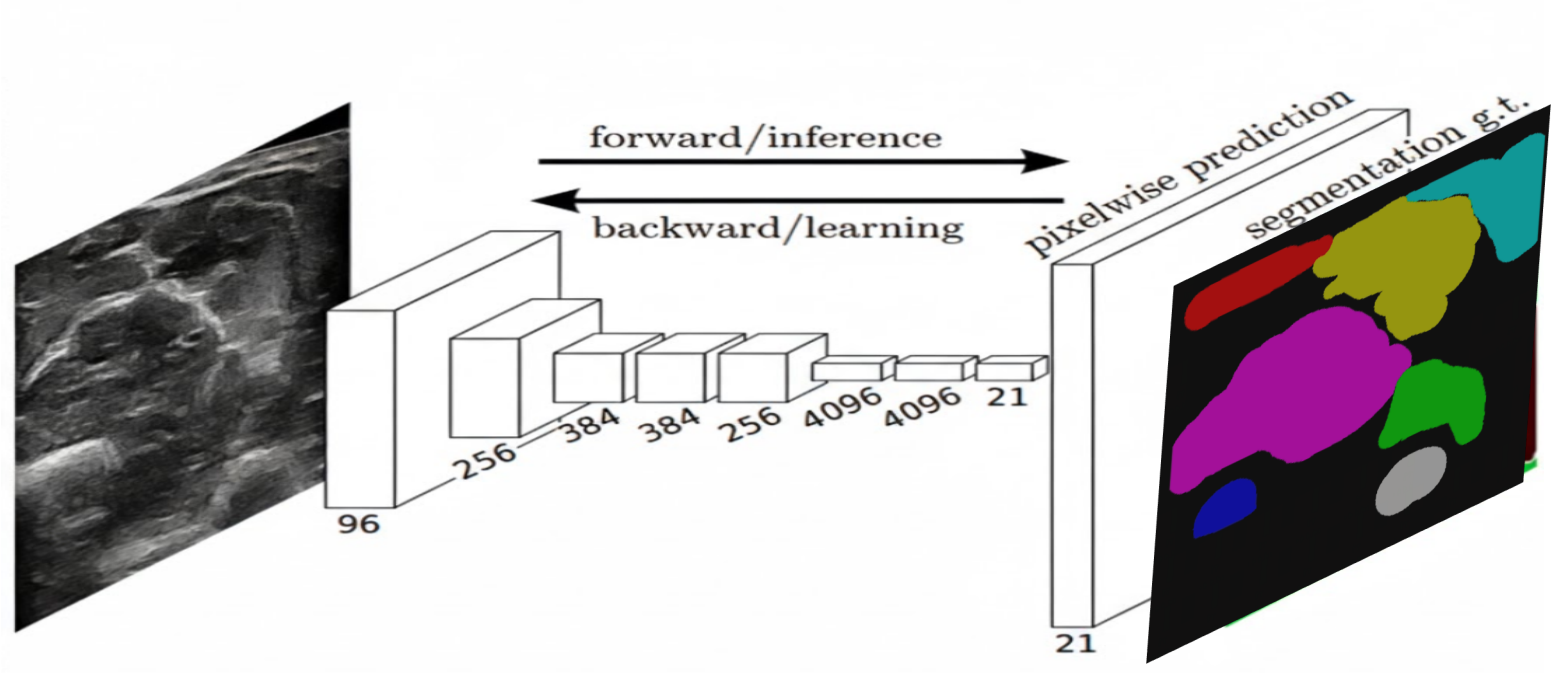
# *Hand motion prediction generalization via Deep Learning based muscle segmentation from Ultrasound imaging*

Authors:
- Eyal Amdur | 318849270 | eyal.amdur@campus.technion.ac.il | אייל אמדור
- Adir Bruchim | 318574654 | adir.bruchim@campus.technion.ac.il | אדיר ברוכים

Supervisors: Dean Zadok under VISTA lab



## Table of Contents

## Abstract

When building systems that let people control robotic or prosthetic hands using ultrasound, one of the biggest challenges is understanding the muscular structure in the forearm.
As different people have different muscle shapes and activation patterns, a system that works well for one person might not work as well for another.
To make these technologies reliable for everyone, we need better ways to read and interpret the muscle structure in ultrasound images.

One example of this kind of application can be seen in Dean Zadok's article "*Towards Predicting Fine Finger Motions from Ultrasound Images via Kinematic Representation*", where he analyzed ultrasound images to predict finger movements with a neural network.
While it worked as expected for the people it was trained on, it didn't generalize well to new users - showing how hard it is to rely only on raw ultrasound images.

In our project, we tried to improve this by adding a new step: automatically segmenting the ultrasound images into the different muscles in the forearm. By training a network to recognize and separate these muscles, we hope to create a clearer and more consistent representation that can help future models better predict movement across different users.

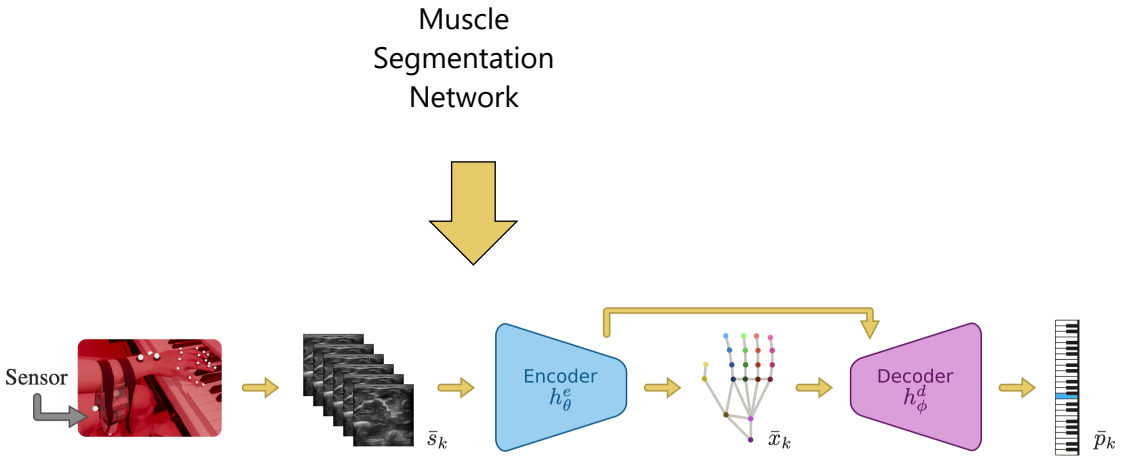



Figure 1: The original pipeline from the article, and where do we join

# Introduction

## Why Ultrasound?

Ultrasound is known as a powerful tool for understanding how muscles move and behave.
Instead of measuring electrical activity on the skin like EMG does, ultrasound shows the shape and movement of the muscles inside the arm.
This makes it easier to tell the difference between similar gestures or finger movements.

Ultrasound has other advantages too. It's safe, inexpensive, and already widely used in many fields.
For example, in sports medicine it helps track muscle fatigue, detect small injuries, and monitor recovery - all by looking at how muscles change shape during movement.
The same idea applies to human–machine interfaces: if we can "see" how muscles deform in real time, we can better understand what the person is trying to do.

Because every action creates a unique deformation pattern inside the forearm, ultrasound gives us rich information that often stays consistent across different people.
This makes it a promising sensor for controlling robotic or prosthetic hands.

Recently, data-driven methods like deep learning have been used to interpret these high-dimensional ultrasound signals. They've shown they can predict hand or finger motions, but they still fall short when it comes to more precise movements or generalizing to new users.

A recent study demonstrated that it's possible to predict hand motions from ultrasound images using a neural network, but the model didn't fully generalize - highlighting the need for better representations of the underlying muscle activity.

## Anatomical Overview

The forearm houses two main bones, the ulna and radius, that provide structure and anchor points for key muscles such as the FDS, FDP, FPL, FCR, and the FCU.

These muscles generate the forces responsible for finger flexion, wrist motion, and overall hand dexterity.
Their tendons extend into the hand, controlling the metacarpophalangeal (MP), proximal interphalangeal (PIP), and distal interphalangeal (DIP) joints.

During different finger movements, specific muscles contract and produce unique deformation patterns within the forearm.

As shown in the referenced study, these deformation patterns can be captured using ultrasound imaging and used to infer the intended motion of individual fingers, making the forearm an effective and non-invasive region for decoding fine motor activity.

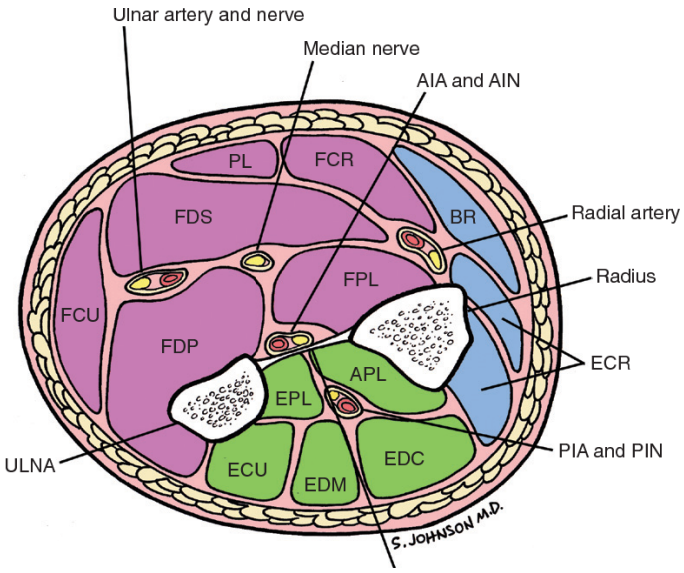



Figure 2: Cross section of the forearm muscles

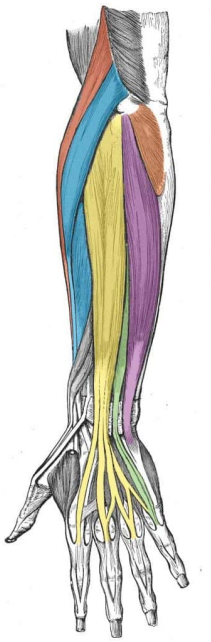


Figure 3: Muscles of the forearm flexing the fingers

## The solution

The goal of the project is to improve the generalization of motion prediction models by introducing a muscle segmentation step within the ultrasound processing pipeline.

Specifically, we aim to create a large dataset of segmented forearm muscles ultrasound images using a semi-supervised video object segmentation network to propagate initial segmentations.
The dataset will be enriched using different augmentations and used to train a neural network-based on the U-Net architecture - to predict segmentation of ultrasound images of the forearm into distinct muscle regions.
The full architecture and specifications will be described in detail in part "Training and Generalization Strategies", while the full pipeline we used is described below ("The system design")

### Why adding segmentation?

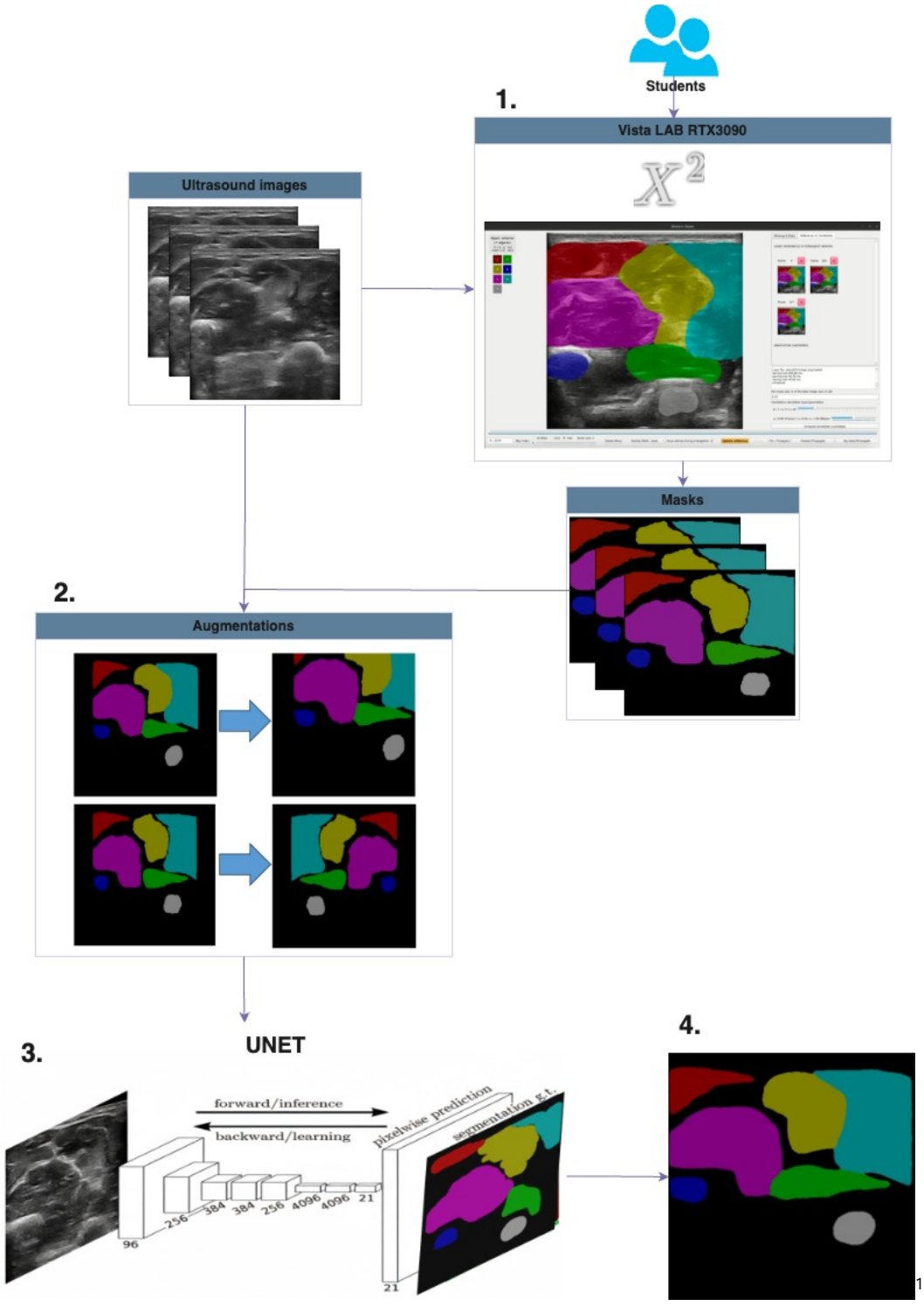
While the referenced study successfully inferred finger motions directly from raw ultrasound images, the resulting model exhibited limited generalization across subjects. This was primarily due to variations in muscle anatomy, probe placement, and the visual appearance of ultrasound signals, which differ significantly from one individual to another.

To address this limitation, we introduce a muscle segmentation preprocessing step that injects domain-specific knowledge into the learning pipeline. Instead of allowing the model to learn from raw pixel intensities - which can be noisy and subject-dependent - segmentation explicitly identifies and isolates the regions corresponding to key muscles involved in hand motion.

By doing so, we guide the model's attention toward physiologically meaningful structures, helping it focus on consistent anatomical cues rather than irrelevant background information.
This anatomical conditioning is expected to provide a more interpretable and robust representation of muscle activity, reduce sensitivity to variations in probe positioning, and ultimately improve the model's ability to generalize across subjects.

The segmentation serves as a bridge between raw imaging data and biomechanical understanding, transforming the input into a form that more directly reflects the underlying muscle dynamics driving finger movement.

## The system design




[1]

[1] https://www.geeksforgeeks.org/machine-learning/u-net-architecture-explained/

## Creating the dataset

### Why is this step needed?

The raw ultrasound videos used in this project originate from the experiments described in the referenced article.

While those recordings encode the deformation patterns that enabled the original researcher to infer finger motions, there was no publicly available, validated dataset of frame-wise muscle-label masks that would allow us to explicitly train anatomy-aware models.

Existing tools for ultrasound segmentation are limited: most available solutions were not trained on this specific region and require heavy manual supervision.

Because our goal was to add an intermediate, anatomy-driven step (muscle segmentation) to improve generalization, we created a dedicated dataset of ultrasound frames paired with per-pixel muscle labels and visualization overlays.

Building this dataset allowed us to:

1. Train and validate a U-Net style segmentation model tailored to forearm ultrasound
2. Quantify segmentation quality across subjects and motions

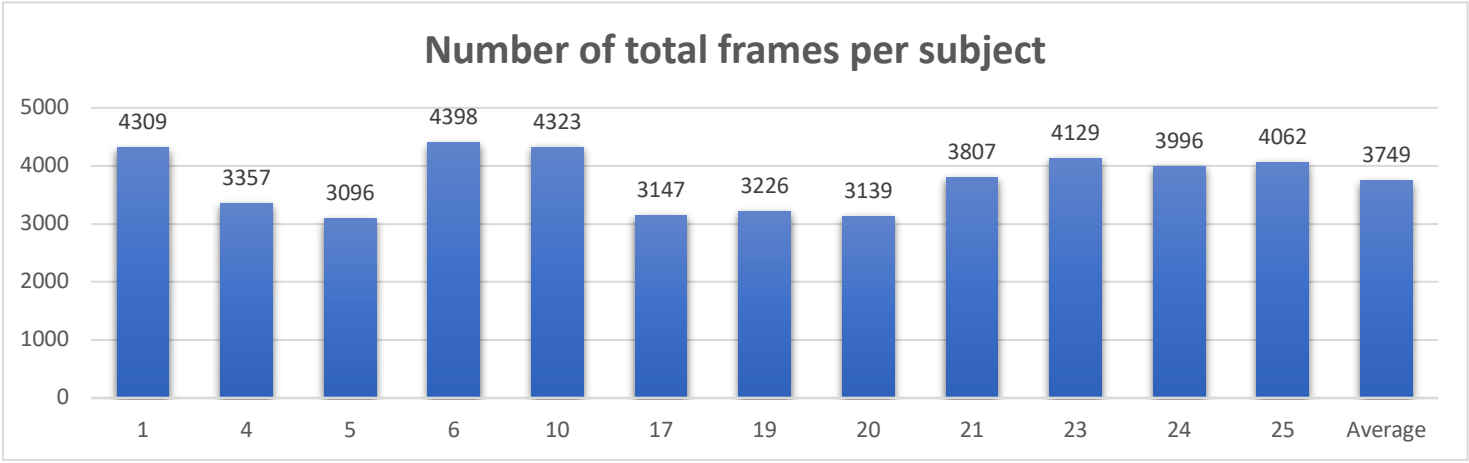
## The Original Data

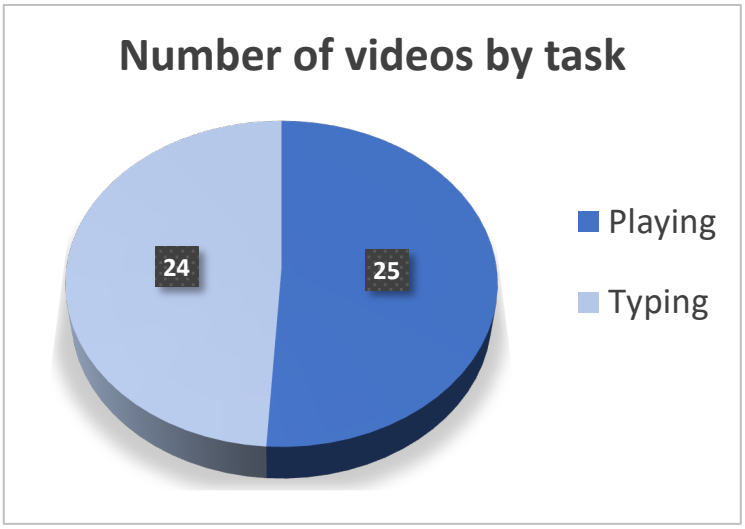
The dataset consisted of recordings from 12 subjects, each performing one of two tasks: playing the piano or typing on a keyboard.
During each task, the subjects' forearm muscles were captured using ultrasound, while a camera simultaneously recorded their hand movements. Each subject repeated the task two to three times, resulting in multiple video samples per person.

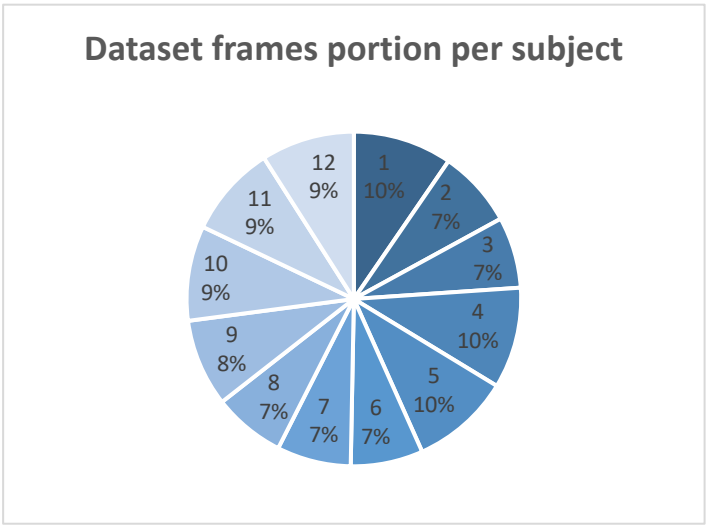
In this project, we focused specifically on the piano-playing task, as it produced more stable and informative results in the original experiment; however, the same methodology can be applied to the typing videos as well.

In total, we worked with 25 videos across 12 subjects, amounting to approximately 45,000 frames.

| Trial | Metric | 1 | 4 | 5 | 6 | 10 | 17 | 19 | 20 | 21 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Frames | 1464 | 1664 | 1592 | 1956 | 2202 | 1364 | 1599 | 1541 | 1617 | 2014 | 2074 | 2145 |
| | Time in seconds | 73 | 83 | 79 | 97 | 110 | 68 | 79 | 77 | 80 | 100 | 103 | 107 |
| 2 | Frames | 1323 | 1693 | 1504 | 2442 | 2121 | 1783 | 1627 | 1598 | 2190 | 2115 | 1922 | 1917 |
| | Time in seconds | 66 | 84 | 75 | 122 | 106 | 89 | 81 | 79 | 109 | 105 | 96 | 95 |
| 3 | Frames | 1522 | - | - | - | - | - | - | - | - | - | - | - |
| | Time in seconds | 76 | - | - | - | - | - | - | - | - | - | - | - |




Number of videos by task
24
25
Playing
Typing


Dataset frames portion per subject
1 10%
2 7%
3 7%
4 10%
5 10%
6 7%
7 7%
8 7%
9 8%
10 9%
11 9%
12 9%

## Tools and annotation pipeline

We evaluated several annotation strategies before settling on a pipeline that balanced annotation effort and temporal consistency.

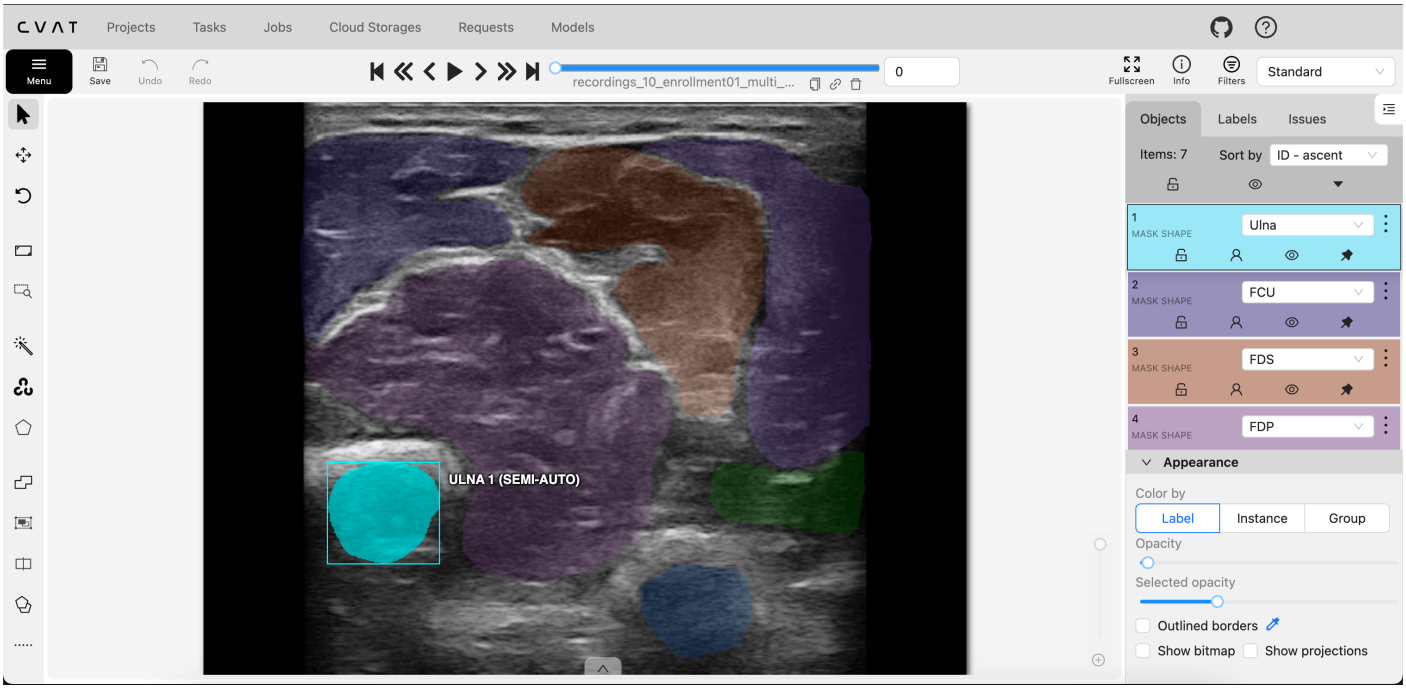
### Initial attempts: DEXTR and CVAT + SAM2

Our first experiments included DEXTR[2], a CNN-based interactive segmentation tool, and CVAT combined with SAM2 for semi-automatic frame-by-frame labeling.

DEXTR required setting the boundaries for each muscle on every frame, which made the process very slow and inconsistent due to continuous muscle deformation and ultrasound noise.

CVAT[3]+ SAM2[4] improved per-frame segmentation and provided an AI-based annotation, but still offered limited reliable temporal propagation.
Each frame remained essentially independent, and interpolation failed to preserve the changes in muscle shape.

These methods were thus useful only for single-frame segmentation and could not be used for producing large temporally coherent mask sequences of ultrasound videos.




[2] https://cvlsegmentation.github.io/dextr/
[3] https://app.cvat.ai/
[4] https://ai.meta.com/sam2/

## XMem++ (video object segmentation / propagation)

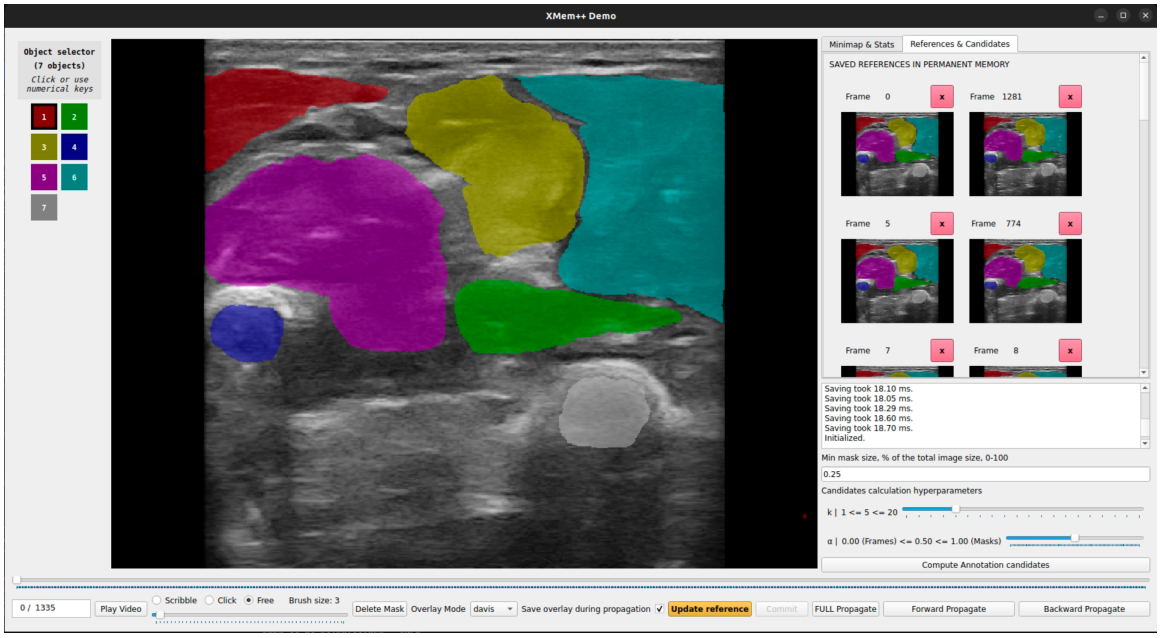
To overcome the lack of temporal consistency, we adopted XMem++[5], a video object segmentation framework.

XMem++ builds upon the original XMem architecture, which employs a memory-based attention mechanism to propagate object masks across frames by storing and retrieving features from previously labeled frames.
This allows it to maintain strong temporal coherence even in challenging conditions such as motion blur or hard-to-psot muscle boundaries.

In our pipeline, we manually annotated a small number of keyframes per video representing distinct muscles, manually segmenting every ~200 frames per video or so. XMem++ then used these annotated frames as references to propagate segmentation masks throughout the entire video sequence.

Unlike the previous tools, XMem++ effectively tracked the deforming muscles across frames, producing temporally stable masks that required only limited manual correction. It also handled small probe motions and moderate appearance changes well, which are common in ultrasound recordings.

The main challenge with XMem++ was computational: propagation is GPU-intensive. While the model performed well when propagating just a few muscle groups, using all seven classes in our project often caused CUDA out-of-memory errors.
Dean helped us overcome this issue by increasing the GPU capacity of the server we were using, after which the program ran seamlessly.




[5] https://github.com/mbzuai-metaverse/XMem2

## Processing the videos

### Preprocess

Although XMem++'s propagation worked great, we had to run some preprocessing before passing the videos to the model.

### Resizing the videos

The original ultrasound videos had a resolution of 640×480, which introduced two main issues:

1. Although the frame size was 640×480, the actual ultrasound content occupied only 480×480 pixels, leaving black borders around the image that were irrelevant to the segmentation task.
   To address this, we cropped the videos before feeding them into the pipeline.

2. Using 480×480 inputs and outputs led to many model parameters, which caused overfitting early in training (low training loss vs. high validation loss).
   To reduce model complexity and improve generalization, we resized the inputs and outputs to 240×240.

### Quantizing the pixels

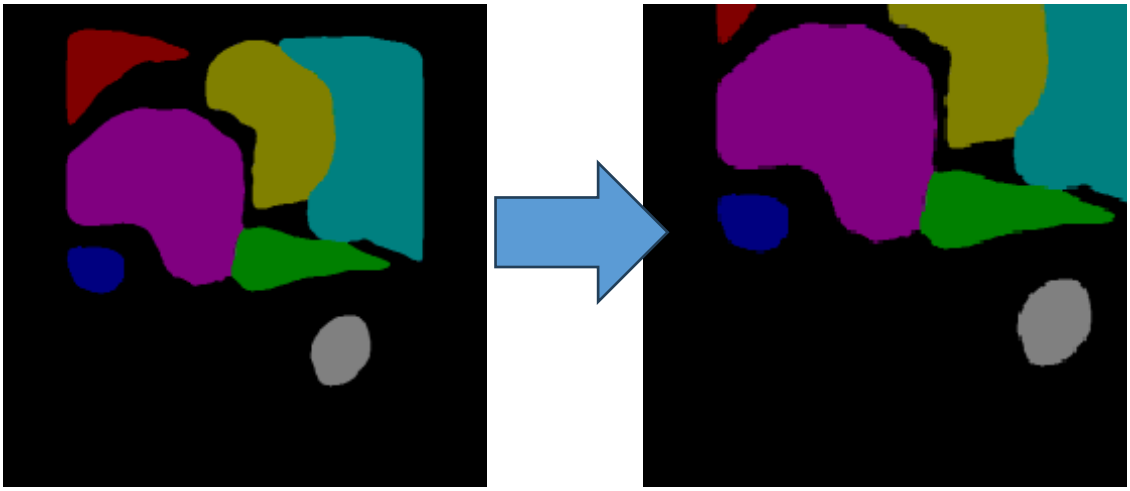
While XMem++ allowed us to segment the muscles in one out of seven colors (and another class for the background), the output contained multiple pixels with color similar to the ones we used, off by a couple of RGBs.
For example, while most of the black pixels were considered as (0, 0, 0) RGB - some were (1, 0, 0) or (1, 1, 0).
Instead of 8 different pixel colors, a frame contained on average a couple of thousand different pixel colors.
This caused a conflict when training on the images and predicting the desired output classes.
To solve this – we wrote a function to go over the frames and replace them with ones where every pixel is replaced with the value of the closest color form the palette we used.

## Augmentations

Because the dataset we created was small to begin with (12 subjects with 2 videos each) we decided to enrich it using a couple of methods.
This allowed us to triple the number of videos per subject.

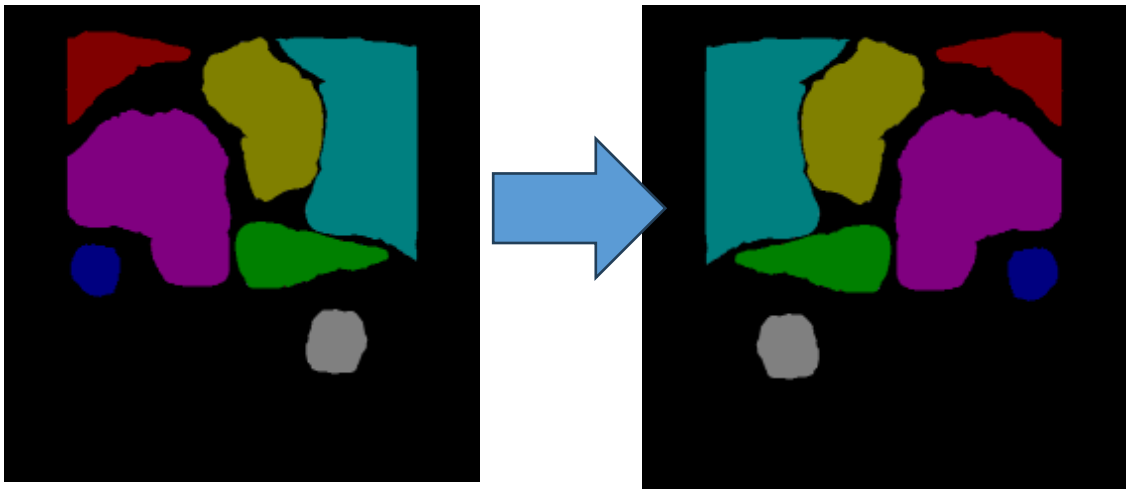
### Random Resize & Crop

We employed Torch's **RandomResizedCrop** function to extract enlarged sub-regions of the original frames, exposing only partial areas of the image.
This approach helps simulate variations caused by device misplacement or differences in muscle proportions across subjects.



### Flip

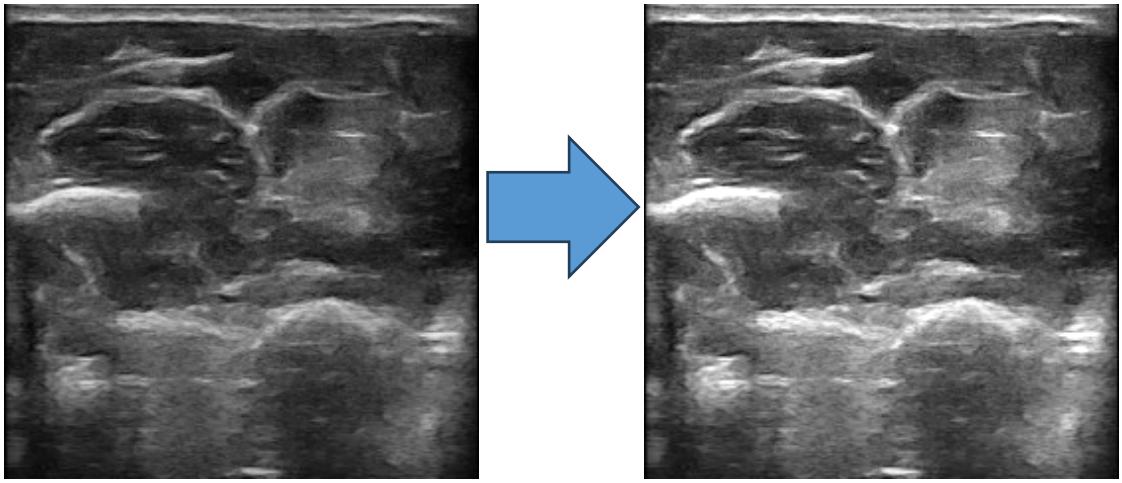
Similarly, we used OpenCV's flip function to horizontally mirror the original frames.
This augmentation simulates both hands (left and right), as the original dataset included recordings of only one hand.



## Random Gamma

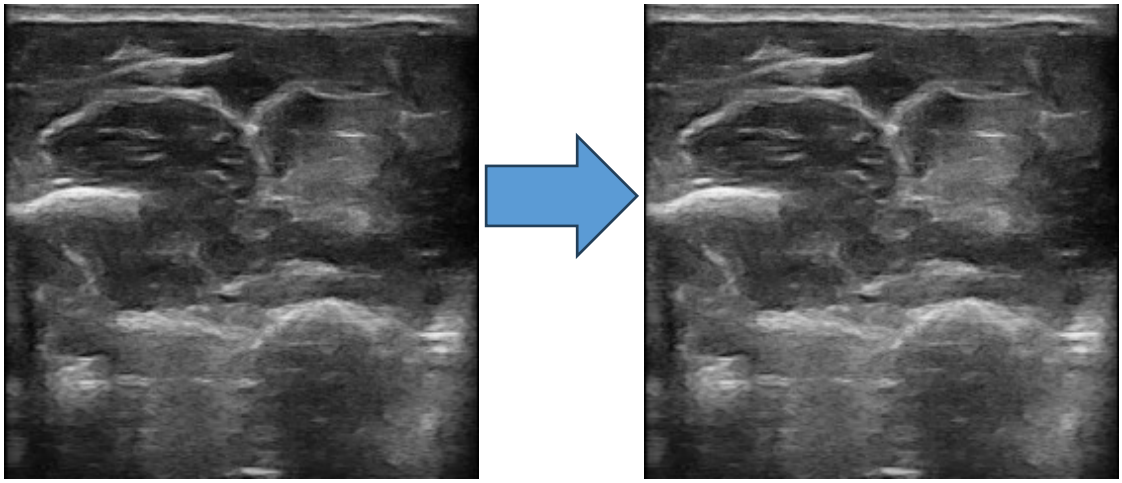
We first convert each raw frame to an 8-bit image and then draw a gamma value uniformly from the configured range (default $0.7 - 1.5$).
We then use this value to remap each pixel accordingly, effectively brightening or dimming the entire image while preserving relative contrast.
Frames then return to their original dtype to keep the rest of the pipeline unchanged.

By presenting the model with this spectrum of gain settings, it gets used to brighter or dimmer scans and still pays attention to the muscle shapes.



## Speckle Noise

For speckle, we treat the ultrasound image as floating-point, sample a Gaussian noise tensor with mean 0 and configurable standard deviation, and scale it by the original pixel values to create multiplicative noise - simulating the grainy texture produced by coherent imaging.
After adding the noise, values are clipped back into the valid range.
Seeing this kind of realistic speckle during training helps the model tell actual anatomy from scanner noise, so it's steadier on live scans.



## Training and Generalization Strategies

The core of the segmentation system is a U-Net architecture, a CNN widely used for biomedical image segmentation tasks.
This specific implementation utilizes the **segmentation_models_pytorch** library and is designed for multi-class semantic segmentation, classifying each pixel into one of 8 classes (7 distinct muscle groups and 1 background class).

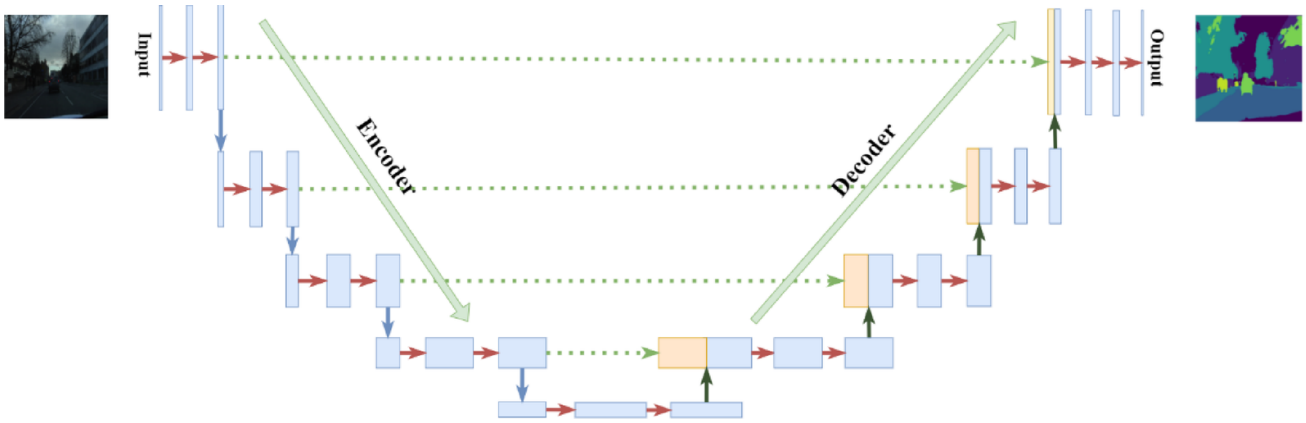
### U-Net Components

#### Encoder (Feature Extraction)

The model's encoder is based on a ResNet34 backbone.
ResNet layers are crucial for deep feature extraction, and the 34-layer depth provides hierarchical feature representations.
Critically, the encoder is pre-trained on ImageNet weights.
This transfer learning approach leverages knowledge from a large natural image dataset, significantly improving convergence and feature robustness, which is beneficial when training with limited medical imaging data.
This allows us to use the relatively small dataset we created to only fine tune the model, instead of training it from scratch.
The encoder is adapted to accept single-channel (grayscale) ultrasound images as input.
Input shape: $(batch\_size, 1, H, W)$.

#### Decoder (Reconstruction)

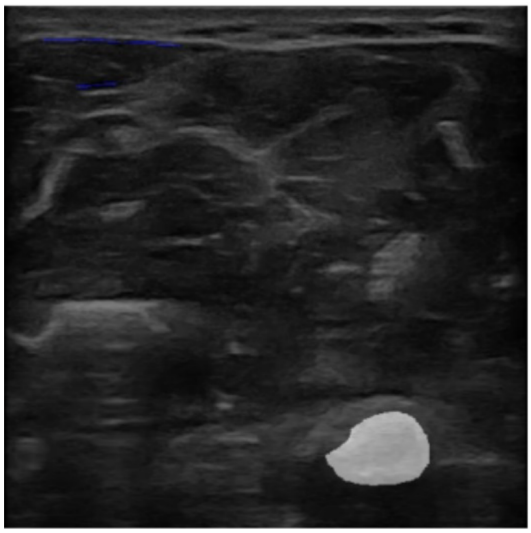
The decoder progressively upsamples the compressed features from the encoder, reconstructing the image to the original input resolution.
The output of the decoder is an 8-channel feature map (raw logits), corresponding to the log-probabilities for each of the 8 classes at every pixel location.
Output shape: $(batch\_size, 8, H, W)$, where the 8 channels represent the score for each class.



## Loss Function and Class Weighting

The model is trained using **Cross-Entropy Loss** ($\text{nn.CrossEntropyLoss}$). This loss function is suitable for multi-class classification, as it internally applies the Softmax activation to the model's logits before computing the negative log-likelihood.

During inference, we noticed a **significant class imbalance** in the requested output, where the background dominated, encouraging the model to guess it way more often than the other colors.
As you can see in the following image, where the radius was semgneted correctly, while the rest of the image was segmeneted as background.



To counteract this imbalance, we incorporated **class weighting** into the loss function. The new weighting strategy is:

- **Background (Class 0)**: Receives a reduced weight of **0.1**. This prevents the model from trivially predicting the abundant background pixels, forcing it to focus on foreground muscle structures.
- **Muscle Groups (Classes 1-7)**: Receive the **standard weight of 1.0**.

## Validation Strategy: Subject-Based Split

To accurately test the model's ability to generalize, a **subject-based train/validation split** was chosen as the validation strategy (Instead of randomly sample a fixed percent of the frames as validation).

This ensures that videos from the same individual never appear in both the training and validation sets, but may cause variation in the train-validation ratio (Because different subjects had videos of different lengths)

This step is crucial because we noticed the ultrasound images of the forearm differ significantly between individuals due to variations in anatomy or probe placement.

By testing on entirely unseen subjects, the validation loss truly measures the model's real-world performance and its ability to overcome the generalization limitations of the original motion prediction model.

The dataset is split with a **90% training ratio** and **10% validation**.
The model is checkpointed and the best model is selected based on the **lowest average validation loss** achieved across all epochs.


## The configuration

Adjustable using a config.json file or command-line parameters.

- Learning Rate: Set to a conservative $10^{-5}$ for stable fine-tuning of the pre-trained encoder.
- Batch Size: 10.
- Epochs: 10, with monitoring to save the best model.
- Data Loaders: Utilize 10 parallel workers to maximize throughput by overlapping data loading with GPU computation.

## Results

We conducted a series of experiments to systematically evaluate the impact of different augmentation strategies and training durations on our model's performance.
Our experiments progressively introduced complexity, starting from a minimal baseline and incrementally adding geometric and intensity-based augmentations, while also exploring the effects of extended training periods.

### Metrics

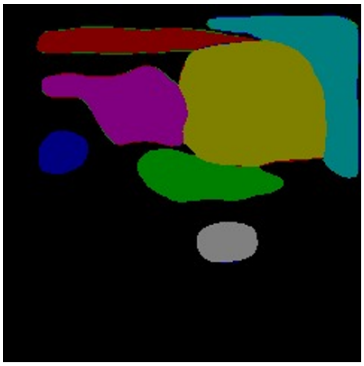
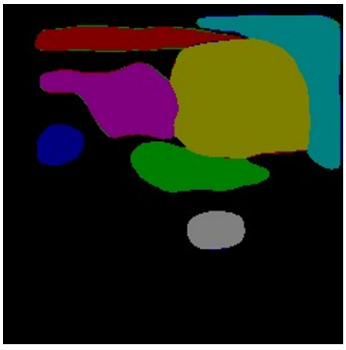
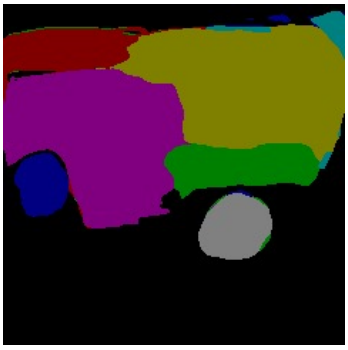
To evaluate our results more precisely, we also introduced a couple of metrics to compare over the four experiments, in addition to the loss function we used in the training.
The metrics were calculated for each class separately and then averaged.

- Accuracy: $\frac{\text{correctly predicted pixels}}{\text{total number of pixels}} = \frac{True\ Positives + True\ Negatives}{Total\ Pixels}$
- Precision: $\frac{\text{correctly identified class x pixels}}{\text{all pixels the model } predicted \text{ as class x}} = \frac{True\ Positives}{True\ Positives + False\ Positives}$
- Recall: : $\frac{\text{correctly identified class x pixels}}{\text{all pixels belong to class x}} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$

## Visual results

Here we can see the results of the model's segmentation prediction on both training and validation subjects across the different experiments (Along with a short description on their differences from the previous experiment)

| | 1 | | 2 | | 3 | |
|---|---|---|---|---|---|---|
| Description | 1 epoch | Augmentations: None | 10 epochs | Augmentations: None | 10 epochs | Augmentations:<br>- Flip<br>- Crop & Resize |
| Training |  | |  | |  | |
| Validation |  | |  | |  | |

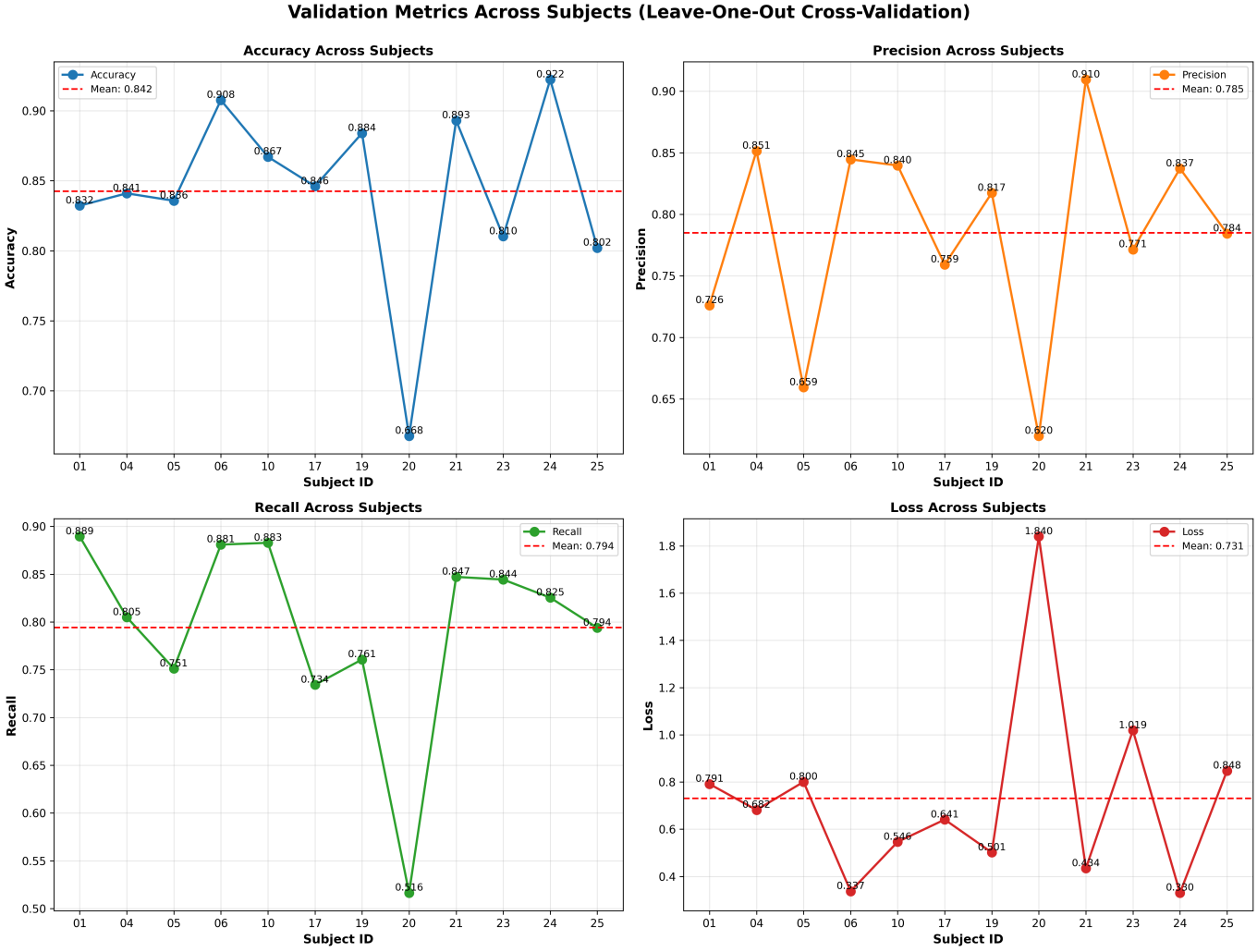
| | 4 | | 5 | |
|---|---|---|---|---|
| Description | 10 epochs | Augmentations:<br>- Flip<br>- Crop & Resize<br>- Speckle Noise<br>- Random Gamma | 50 epochs | Augmentations:<br>- Speckle Noise<br>- Random Gamma |
| Training |  | |  | |
| Validation |  | |  | |

Our experiments show that data augmentation was the key to getting accurate results. While increasing the training period from 1 to 10 epochs (Experiments 1 and 2) helped the model learn the training data, it wasn't enough to produce clean predictions on new validation images.

The best performance was achieved in Experiment 4, by adding Speckle Noise and Random Gamma adjustments on top of standard flipping and cropping.
This combination prevented the model from memorizing the data and forced it to learn the actual muscle shapes, resulting in the cleanest and most accurate segmentation masks.

## Leave one subject out

In this experiment, we wanted to emphasize the difference between the different subjects in the dataset, and their effect on the training.
For each subject X - we trained the model on the rest of the subject for a duration of one epoch with all 4 augmentations, then validated and inferred subject X.
We can see that subject 20 for instance shows a high distinction from the rest of the subject, which might label it as an outlier and compromise our training process.

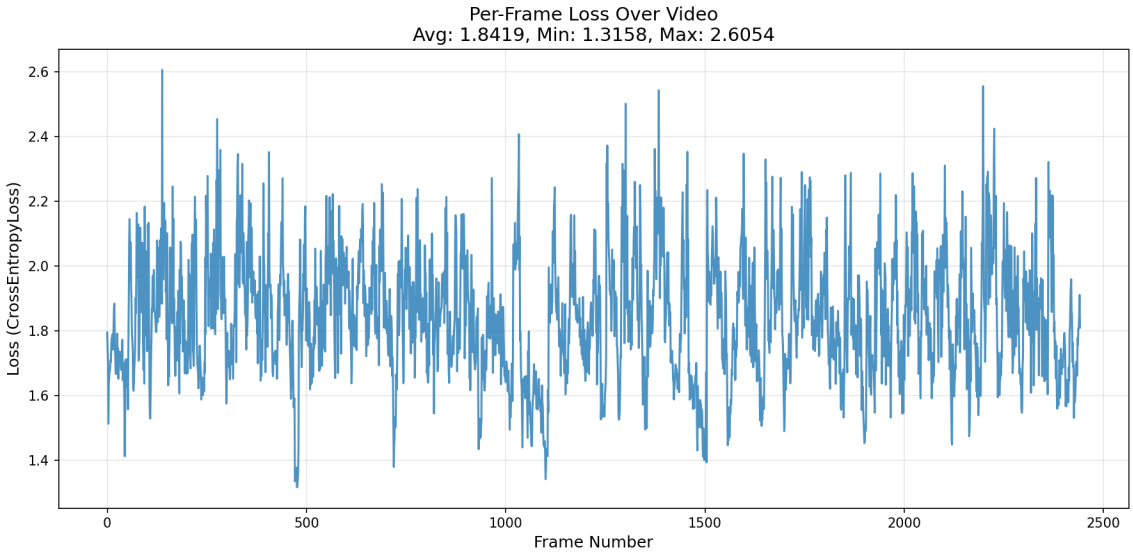


## XMEM++ effect on the prediction

Since our model relies heavily on the XMEM++' propagation output (After our corrections), we needed to confirm that the propagation process didn't accumulate errors that would obstruct our predictions.

To test this, we ran inference on a full video (Subject 06, video 01) and plotted the Cross-Entropy Loss for every single frame.

If XMEM++'s propagation failed over time, we would expect to see the loss steadily increase as the video progressed.




As can be seen in the graph - there is no clear upward trend in the loss across the 2500 frames. We observed the same flat trend on other subjects as well

Therefore, we can conclude XMEM++ did a good job of propagating our segmentations. It confirms that the automatic propagations were stable and did not introduce any cumulative errors that degrade the model's performance over the video duration.

## Discussion and Future Work

### Discussion

In this project, we were able to establish a training process that forces the model to look past the visual noise in the raw images and identify the true anatomical structure.
By simulating the grainy, inconsistent nature of live ultrasound scans, we ensured the model did not just memorize the training videos but actually learned to recognize distinct muscle groups across different subjects.
We believe this implementation will help generalize the current solution.
This means the motion prediction system can now interpret movement based on physiologic data rather than confusing pixel noise, effectively improving the generalization effort.

## Next Steps

### Ensure the train dataset is consistent

During the experiments, we noticed that some of the earlier segmentations were performed inconsistently.
To improve the reliability and consistency of the inference process, we recommend reviewing the existing results with an expert and correcting or excluding samples that do not meet the required standards.

### Integration into the full pipeline

The most immediate next step is to integrate the trained U-Net model into the motion prediction pipeline described in Dean's original article.
The segmented images would replace the raw ultrasound images as the input to the motion inference network. The performance should then be evaluated by comparing the generalization results (e.g., performance on unseen subjects) against the original model that only used raw ultrasound input.

### Enrich the dataset with additional motion tasks

The original article's dataset included videos of both piano playing and keyboard typing, while our segmentation work focused only on videos from the former task.
Following successful integration and verification of improved generalization, the same annotation and training pipeline should be run on the keyboard typing videos to further enrich the dataset and expose the model to a wider range of muscle deformation patterns, potentially boosting the robustness of the entire system.

### Experiment with hyperparameters and different backbones for optimal performance

Once the dataset is enriched – we'll double our original input and will be able to make our model more complex, while still avoiding overfitting the train samples.

Therefore, this step will include a necessary architecture overview and hyperparameter tuning, which includes:

- Testing higher learning rates.
- Evaluating different optimizers to ensure the most efficient minimization of the weighted Cross-Entropy Loss.
- Exploring different backbones and architectures to replace ResNet34 backbone with U-Net to potentially achieve higher segmentation accuracy and even better generalization across subjects.

## AI Usage

We used AI tools to assist with specific technical parts of this project.

### Report Writing (Gemini)

We used Gemini to improve our English phrasing and grammar, while the ideas, analysis, and conclusions were written by us.

### Code Assistance (Claude Code)

We used Claude Code to speed up the writing of utility scripts, specifically for data visualization and standard image augmentations.
The core logic, including the U-Net architecture, training loop design, and hyperparameter tuning, was written and refined by us.

## References

### Figures


1. https://arxiv.org/pdf/2202.05204
2. https://www.semanticscholar.org/paper/CHAPTER-51-Wrist-and-Forearm-Williams-Kim/4fae5348ebb116f876becdd5c3a4938a4d71b276/figure/31
3. https://teachmeanatomy.info/wp-content/uploads/Muscles-in-the-Superficial-Layer-of-the-Posterior-Forearm-824x1024.jpg


### Git Repository

https://github.com/adirbru/USPrediction