Title: Semantic Segmentation of Activated Muscles During Exercise Using Custom UNH Gym Dataset

Team Members: Gururaj Somanna and Harsha Keladi Ganapathi

Objective: The objective of this study is to develop a U-Net-based semantic segmentation model that detects and segments only the activated muscle regions during different exercises.

1. Introduction:

Muscle segmentation from images is a crucial task in biomedical imaging and sports science. It aids in applications such as injury detection, fitness tracking, and anatomical analysis. Traditional methods rely on manual observation, which can be intrusive and time-consuming. In recent years, much significant research has been conducted to detect and analyze human motion and muscle contraction using Computer vision and deep learning techniques [1]. One notable study introduced by *Iman et al.* called the Neural Muscle Activation Detection (NMAD) framework, which employs deep learning to detect muscle activation based on surface electromyography (sEMG) signals [2]. While previous research has primarily focused on EMG-based detection or pose estimation, there has been limited work on direct visual segmentation of activated muscles during exercise.

In this project, we focus on fine-tuning a semantic segmentation model by utilizing a pre-trained U-Net model to identify and segment different muscle regions, such as the triceps, calves, and others, from a custom dataset collected from the University of New Haven (UNH) gym. We performed different hyperparameter tuning and validation to achieve an optimized model that generalizes well on unseen data and ensures robust generalization across all classes.

2. Methodology:

We used a U-Net architecture with a ResNet-34 backbone as the encoder for semantic segmentation. U-Net is a popular choice in medical and muscle segmentation tasks due to its ability to capture fine-grained spatial details via skip connections and its encoder-decoder design [3].

Architecture Overview:

1. Encoder (ResNet-34 Backbone):

The encoder compresses spatial information while increasing the receptive field using convolutional layers, batch normalization, ReLU activations, and max pooling. Specifically:

- **Initial Block**:
  - Conv2d (3, 64, kernel_size=7, stride=2, padding=3)
  - Batch Norm 2d (64)
  - ReLU
  - MaxPool2d(kernel_size=3, stride=2, padding=1)
- **Residual Layers**:
  - **Layer1**: 3 x Basic Block (64 → 64)
  - **Layer2**: 4 x Basic Block (64 → 128) *(first block has stride 2 for down sampling)*
  - **Layer3**: 6 x Basic Block (128 → 256)

- **Layer4**: 3 x Basic Block (256 → 512)

Each Basic Block includes two 3x3 convolutions with batch normalization and ReLU, with optional down-sampling via 1x1 convolutions.

2. Decoder (U-Net-style):

The decoder up-samples the features and combines them with corresponding encoder features through skip connections. This helps restore spatial resolution and enhances localization.
Each decoder block typically includes:

- **Upsampling**
- **Concatenation** with encoder features
- **Two 3x3 Conv → BN → ReLU blocks**

Final layers:

- Conv 2d (filters, num_classes, kernel_size=1) to reduce channel dimensions
- SoftMax activation for multi-class output

3. Output:

- 4 output channels corresponding to the background and 3 muscle classes (e.g., bicep,calf, tricep, etc.)
- Output shape matches the input image size (e.g., 256×256)

Loss Function

We used **Cross Entropy Loss** for multi-class segmentation, which is well-suited for problems where each pixel belongs to one of N classes.

Optimizer and Training Details

- **Optimizer**: Adam and SGD
- **Learning Rate Scheduler**: StepLR (step size = 5, gamma = 0.5)
- **Epochs**:2,10, 30
- **Batch Size**: 4,8,16
- **Validation Split**: 20%

3. Data set Description

We have utilized the custom dataset for the semantic segmentation. To collect the dataset, samples of UNH students with different disparity are chosen and undergoing them to perform different exercises through zym equipment's present in UNH rec-centre and RGB based images are captured while performing these workouts and annotated using Roboflow tool to mask the classes by pixel wise. 200 images are captured with the image size of 256*256 consisting of four classes such as:

- **Label Classes:**
  - 0: Background
  - 1: Activated Biceps
  - 2: Activated Triceps
  - 3: Activated Calf

Each class is labelled with a unique integer (e.g., 0 for background, 1–3 for muscles).

Image Characteristics

- **Image Dimensions**: All images and masks were resized to **256×256 pixels** for uniform input to the model.
- **Color Channels**: Grayscale input (1 channel) was converted to a 3-channel format to align with ResNet-34's expectations.
- **Mask Format**: Single-channel images with values $\in \{0, 1, 2, 3, 4\}$, representing the background and muscle classes.

Data Augmentation:

To increase dataset diversity and improve model generalization, the following augmentations were applied . Each image generated 3 augmented variants on average. The transformations included:

- Horizontal and Vertical Flip
- Shear: ±15° (both horizontal and vertical)
- Grayscale: Applied to 25% of the images
- Gaussian Blur: Up to 1.2px
- Random Noise: Up to 0.22% of pixels

These augmentations helped simulate variations in orientation, lighting, and imaging conditions often present in real-world data collection.

Preprocessing Steps

1. **Resizing**: All input images and masks were resized to (256, 256) using bilinear (images) and nearest-neighbor (masks) interpolation.
2. **Normalization**: Pixel values were normalized to the range [0, 1] for input images.
3. **Tensor Conversion**: Images and masks were converted into PyTorch tensors.

Train-Validation Split

- **Total Images**: 520 samples
- **Split**:
    - **Training Set**: 80% - 480 images
    - **Validation Set**: 20% - 40 images
- The split was performed randomly with a fixed seed to ensure reproducibility
- **Directory structure:**

```
dataset/
train /    images/
           masks/
Val /      images/
           masks/
Test /     images/
           masks
```

4. Experiments and Results:

a) Hyperparameter Tuning and Validation

To optimize segmentation performance, we conducted hyperparameter tuning over several key parameters as shown in Table 1.

Table 1: It depicts the overview of hyperparameters utilized for experimentation

Hyperparameter	**Range Tested**	**Best Value Found**
Learning Rate	0.01,0.0005,0.001, 0.0001	**0.0001**
Batch Size	4, 8, 16	**16**
Optimizer	Adam, SGD	**Adam**
Epochs	10, 20, 30	**30**
Pretrained Backbone	ResNet34	Used with the encoder

Each combination was trained for 2–3 epochs and evaluated based on validation loss and mean Intersection over Union (mIoU). Among these, Adam optimizer consistently outperformed SGD, and lower learning rates led to better convergence.

Table 2: It depicts the different hyperparameter with its respective metrics

Learning Rate	**Batch Size**	**Optimizer**	**Val Loss**	**Val mIoU**	**Epochs**
0.001	4	Adam	0.0874	0.3006	2
0.001	4	SGD	0.1065	0.2451	2
0.001	8	Adam	0.0895	0.2451	2
0.001	8	SGD	0.1331	0.2451	2
0.001	16	Adam	0.0917	0.2451	2
0.001	16	SGD	0.1632	0.2451	2
0.0005	4	Adam	0.0761	0.2451	2
0.0005	4	SGD	0.1386	0.2451	2
0.0005	8	Adam	0.1154	0.2451	2
0.0005	8	SGD	0.1894	0.2450	2
0.0005	16	Adam	0.1628	0.3000	2
0.0005	16	SGD	0.2746	0.2449	2
0.0001	16	Adam	**0.0411**	**0.5165**	**30**

After initial experimentation, we trained the best-performing configuration — learning rate = 1e-4, batch size = 16, Adam optimizer — for 30 full epochs.

Training vs Validation Metrics

During the training process, both loss and accuracy were monitored. Below is a typical outcome from one of the best trainings runs as shown in Table 3:

Table 3: It describes the metrics evaluated during model training.

Metric	**Training Set**	**Validation Set**
Loss	0.13	0.17
Pixel Accuracy	94.5%	92.3%
mIoU	0.58	0.51

Loss Curves

- The training loss showed a **smooth downward trend**, indicating good convergence.
- Validation loss also decreased, showing no significant overfitting.

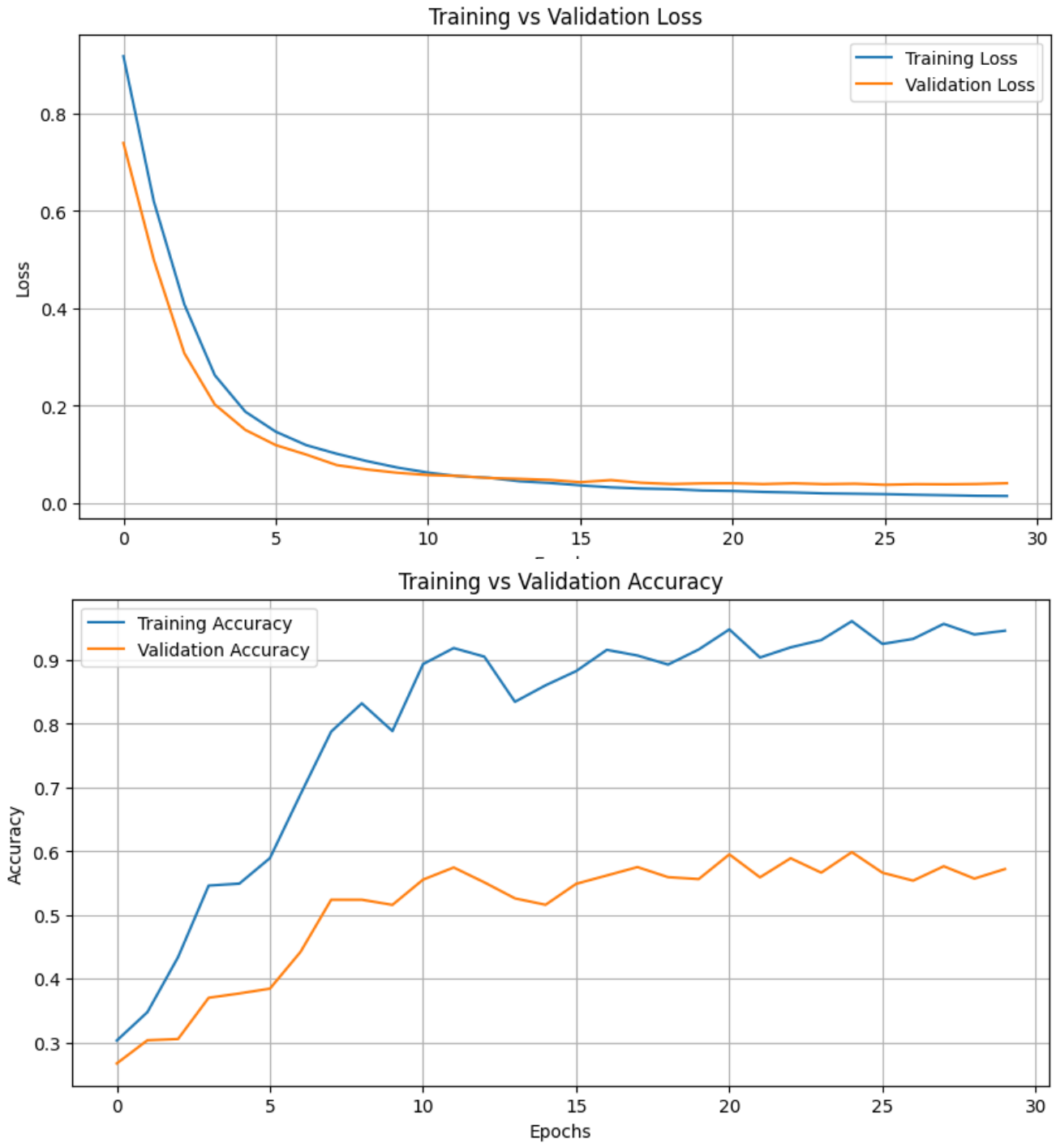

Figure 1: It describes the overall performance of the trained model

b) Pre-training vs Post-training Analysis

Aspect	Before Training (Pretrained on ImageNet)	After Fine-Tuning (Our Dataset)
Class Activation	Weak, generalized textures	Clear focus on muscle boundaries
Mask Predictions	Poor alignment with true regions or no masked assigned.	Sharp, anatomically consistent masks
Class Mislabeling	Frequent confusions (e.g., calf ↔ tricep) and mostly predicted background class.	Greatly reduced misclassifications
mIoU	**~0.35**	**~0.51**

Qualitative Results

- Sample predictions show tight overlay between ground truth and model predictions.
- Below are examples of results on triceps , bicep and calf images:

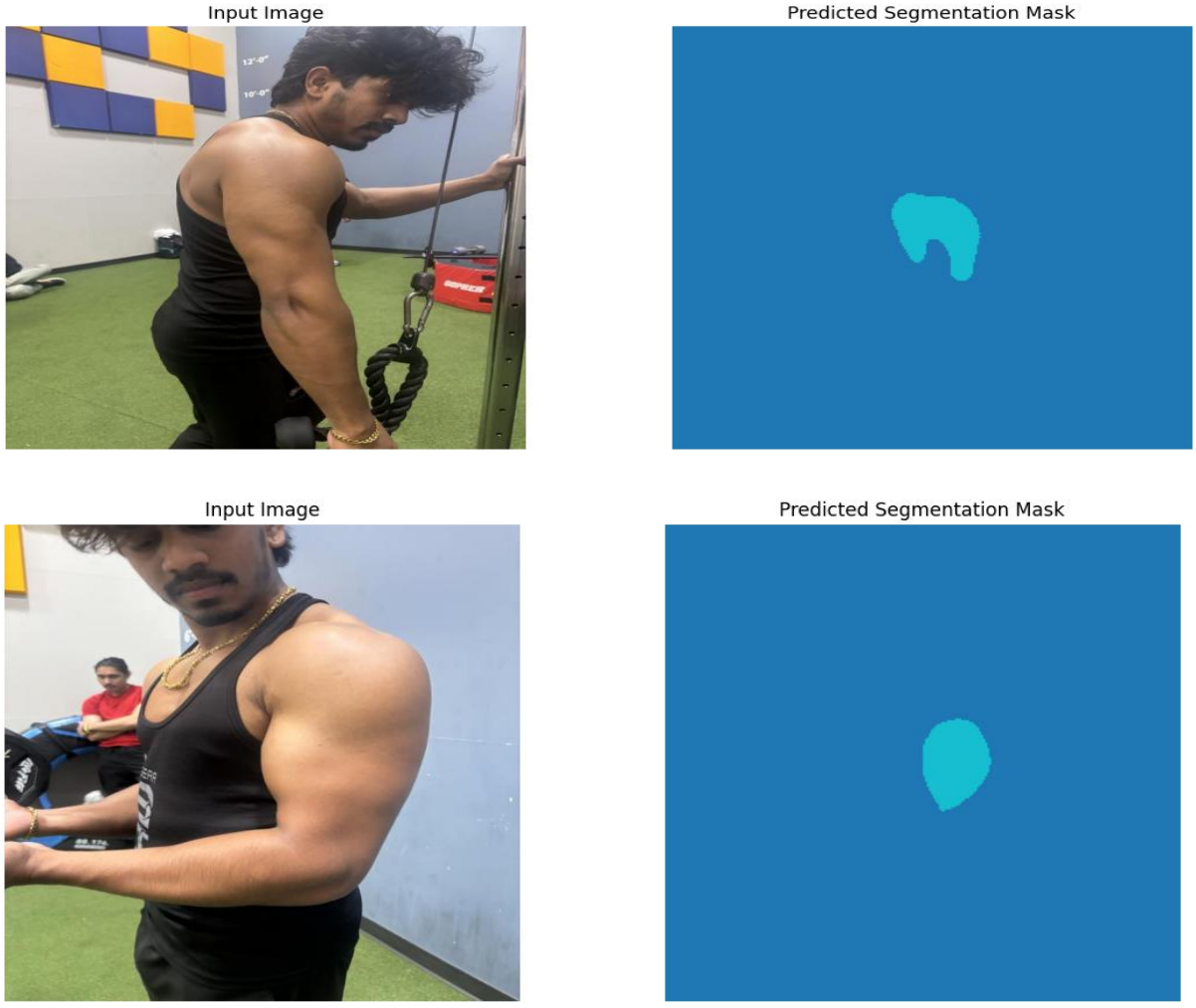

Figure 2: It shows the model's Predicted Images

5. Conclusion

In this project, we implemented a semantic segmentation model based on the pre-trained model called U-Net architecture with a ResNet34 encoder, fine-tuned on a custom-annotated dataset containing muscle groups such as biceps, triceps, and calf muscles. We conducted structured hyperparameter tuning across various learning rates, batch sizes, and optimizers to identify the optimal configuration.

Although the training pixel accuracy reached 94.61%, the validation accuracy and mIoU were relatively modest, primarily due to the limited dataset size and the quality of muscle boundary annotations. Many of the annotated images contained poorly defined or partially visible muscle specimens, making it difficult for the model to learn consistent spatial features for segmentation.

Despite these challenges, the model was successfully deployed using Gradio, allowing interactive testing on new images and video inputs. With access to more clearly annotated data and better-defined muscle samples, the model's performance could be further improved.

References

[1] J. W. Kim, J. Y. Choi, E. J. Ha, and J. H. Choi, "Human Pose Estimation Using MediaPipe Pose and Optimization Method Based on a Humanoid Model," *Appl. Sci.*, vol. 13, no. 4, 2023, doi: 10.3390/app13042700.

[2] I. Akef Khowailed and A. Abotabl, "Neural muscle activation detection: A deep learning approach using surface electromyography," *J. Biomech.*, vol. 95, p. 109322, 2019, doi: 10.1016/j.jbiomech.2019.109322.

[3] W. Weng and X. Zhu, "INet: Convolutional Networks for Biomedical Image Segmentation," *IEEE Access*, vol. 9, pp. 16591–16603, 2021, doi: 10.1109/ACCESS.2021.3053408.

Code Link:

https://github.com/mateuszbuda/brain-segmentation-pytorch

Annotated Dataset link:
https://unhnewhavenmy.sharepoint.com/:f:/g/personal/hkela1_unh_newhaven_edu/EsW8gl8TIhpHs1p9sV2zj1YBc6Cx5SAxWU1_mtmdUxLf_Q?e=3btCIb