Computer Vision for Surgical Applications

-Project Report-

Submitted by:

Shai Kikozashvily – 206202384

Michael Fishman - 318962099

Phase 1: Synthetic Data Generation

- **Domain Gap Analysis:**
  - In this section, we compared synthetic surgical images generated through our pipeline with real surgical images. Key differences were identified in several areas, including:
    - Lighting: Synthetic images tend to have uniform lighting conditions, while real surgical images show variable lighting due to reflections from instruments and surroundings. Shadows in synthetic images were often less pronounced than in real-world scenarios.
    - Environmental Factors: The surgical environment in synthetic images lacks the complexity of real settings.
    - Objects placement and interaction with the surroundings: the objects in the real world are changing their positions, angels and visibility according to their surroundings and interactions with the surgeons, which is difficult to mimic in the code for generating the synthetic data.

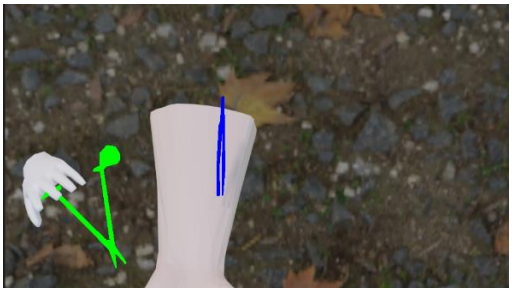
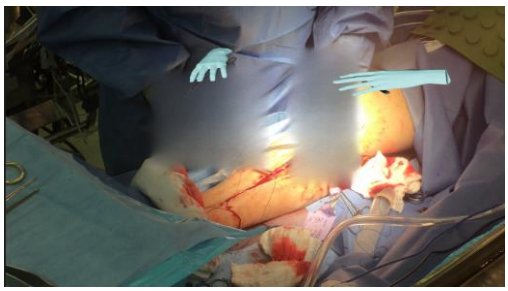
- **Data Generation Approach and Creative Variations:**

  We implemented a pipeline for synthetic data generation, designed to produce a diverse set of images mimicking surgical environments. The key components of our pipeline are:

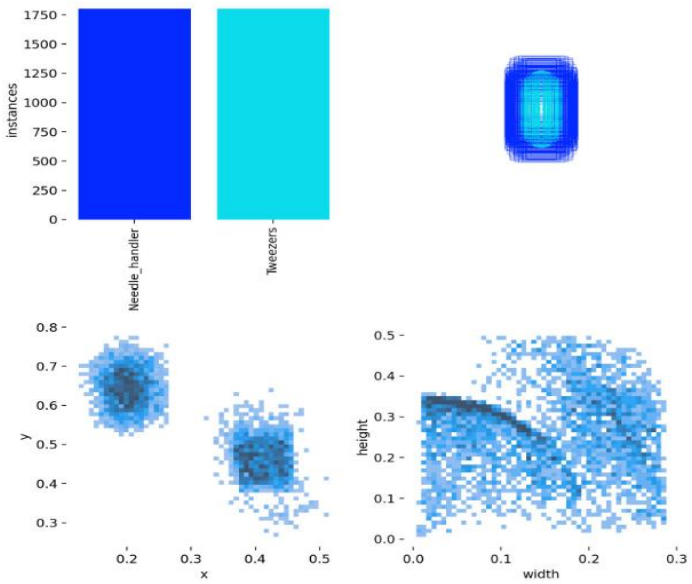
    - Instrument Positioning and Orientation: We utilized randomization techniques to vary the position and orientation of surgical instruments across frames.
    - Lighting Conditions: Lighting conditions were varied to simulate different surgical room settings, including bright overhead lights, side lighting, and focused lighting on instruments.
    - Background Variation: we tried many background variations including the coco backgrounds and hdr files we were given.
    - First Creative Variation: we added hands objects to the scene to mimic the holding of the instruments by the surgeons.
    - Second Creative Variation: we added an object to mimic the leg that the surgeons are operating on.
    - Third Creative Variation: we used the object detection model from HW1 to cut out the bounding boxes of the surgical tools leaving a black rectangle behind.
    - Fourth creative variation: The previous variation left big black bounding boxes behind leading to increased domain gap between the real images and the synthetic ones. So we used simple-lama-impainting model to blur the black rectangles and thus slightly bridge the gap and make the images more realistic.

- Fifth creative variation: Applying the same rotation for the surgical tool and the hand model that is holding it. Which resulted in more realistic positions of the tool relative to the hand.
- Sixth creative variation: Using the object detection model from hw1 as a preprocessing step for the segmentation model. This means that first we cut out the bounding box, and use it as a region proposal for the segmentation model. This allows the segmentation model to focus only on the relevant part. Once we got the correct segmentation, we can paste it back in the proposed region and place it in the original frame.

Examples of generated images:

- **Synthetic Dataset Statistics:**



- **What Worked Well and Challenges Encountered:**
  - Worked well:
    - The pipeline generated diverse images by varying instrument positions, and backgrounds, which helped create a more realistic dataset.
    - Adding hands and a leg to mimic a surgeon operating enhanced the realism of the synthetic scene.
  - Challenges:
    - Synthetic images lacked the variable lighting seen in real surgery.
    - The synthetic surgical environment was much simpler than the real-world setting, lacking dynamic object interactions and environmental factors that are hard to model in code.

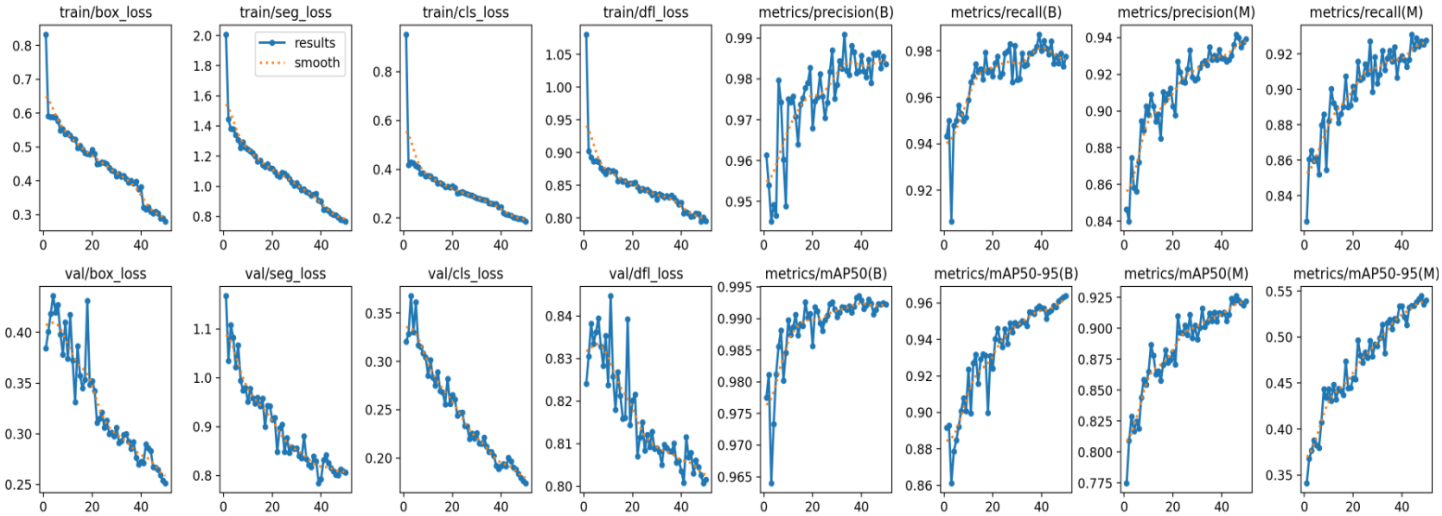
Phase 2: Segmentation Model Development

In this and in the next phase we tried 2 models: YOLO V8- Segmentation Model and FCN (fcn_resnet50 model which we adjusted its segmentation head for our task).

The results of the FCN on the real data were quite bad, the trained model could not generalize well on the real-world videos, although it succeeded well on the synthetic data.

The results of the YOLO model on the real data were far better than the FCN, but they were not so great either. We will now provide details for both models, but we will focus on the performance metrics of the YOLO model.

- **Data Preprocessing and Augmentation:**
  - Preprocessing: Input images were resized and normalized. Instruments were labeled, and ground truth masks were generated. The code for rendering using blenderproc outputs coco annotations so we had to process it to make it suitable for the models. For the FCN model we created mask images with 3 labels for each class: 0 for the background, 1 for the needle holder and 2 for the tweezers. For the YOLO model we created polygons masks for each object.
  - Augmentation techniques used: Random rotations, flips, brightness/contrast adjustments were applied.
- **Training Strategy:**
  - For the FCN model the largest batch size which we were able to use was 8 due to memory constraints of the GPU. For the YOLO model it was 32.
  - The learning rate used was 0.001.
  - The optimizer used was Adam.
- **Training logs and performance metrics on synthetic data (YOLO model):**



Both models performed well on synthetic data, achieving a high IoU score.
Here are some images of the results for the FCN model on the synthetic data:

And for the YOLO model:

- **What Worked Well and Challenges Encountered:**
  - Worked well:
    - YOLO performed significantly better than the FCN model, especially on real data, with higher accuracy in segmentation
    - Techniques like random rotations, flips, brightness adjustments were successful in improving model robustness during training on synthetic data.
  - Challenges:
    - The FCN model struggled to generalize to real-world data despite good performance on synthetic data.
    - Both models, especially FCN, could not bridge the domain gap between synthetic and real surgical videos, leading to poor performance on real-world images.
    - The FCN model's training was limited by the GPU's memory, restricting batch size and potentially hindering model optimization.

Phase 3: Domain Adaptation

- **Domain Adaptation Strategy:**
  In this phase we implemented Domain-Adversarial Neural Networks (DANN) – one for each model (DANN with FCN as the feature extractor and segmentation head, and DANN with YOLO instead of the FCN). Since the results on the real-world's data were really bad for the FCN model, and better but not so great either for the YOLO model, we tried to find a way to improve it. We researched for a suitable technique for our task and came across this technique of unsupervised learning which is good for our task since we do not have labels for the real-world data. The goal of this technique is to minimize the gap between the source domain (synthetic or labeled data) and the target domain (real-world or unlabeled data) by aligning their feature distributions. In DANN, this is achieved by training a shared feature extractor that learns domain-invariant features. DANN consists of three main components:

1. Feature Extractor: This is the shared backbone of the network that extracts representations from both the source and target domains. We used the backbone layers of the FCN and YOLO models.
2. Label Predictor: A classifier that operates on the extracted features to predict labels for the source domain data, as it has ground truth annotations.
3. Domain Classifier: This part distinguishes between the features extracted from the source domain and the target domain. The domain classifier aims to classify whether the input data comes from the source or target domain, while the feature extractor learns to generate domain-agnostic features that confuse the domain classifier.

The key idea behind DANN is the use of adversarial training, where the domain classifier and the feature extractor play a minimax game. The domain classifier tries to correctly identify the domain of the input data (source or target), while the feature extractor learns to generate features that make the domain classifier's job difficult. This forces the feature extractor to learn features that are indistinguishable between the source and target domains, leading to better generalization on the target (unlabeled) data.

To implement the adversarial aspect, we introduced a gradient reversal layer (GRL) between the feature extractor and the domain classifier. During backpropagation, this layer multiplies the gradient by a negative constant, which forces the feature extractor to learn domain-invariant features by confusing the domain classifier.

The total loss for the DANN model consists of:

- Classification Loss: This is the standard loss (cross-entropy) on the labeled source domain data.
- Domain Loss: This is the binary cross-entropy loss used to train the domain classifier. The feature extractor tries to minimize the classification loss while maximizing the domain loss (via the gradient reversal layer), whereas the domain classifier tries to minimize the domain loss.
- The combined loss function can be written as:

$$loss_{total} = loss_{classification} + \lambda \cdot loss_{domain}$$

λ is a hyperparameter that controls the trade-off between the two losses.

- **<u>Challenges:</u>**
  In our efforts to apply Domain-Adversarial Neural Networks (DANN) to improve the generalization of both the FCN and YOLO models on real-world data, we encountered several significant challenges that ultimately impacted the effectiveness of the adaptation:

1. One of the primary challenges was the large domain discrepancy between the synthetic source data and the real-world target data. While DANN aims to align the feature distributions between the two domains, it became noticeable that the gap was too wide for the models to bridge effectively. In the case of the FCN, despite learning domain-invariant features, the predictions on real-world data remained poor. The segmentation outputs lacked precision, and the model struggled to generalize to the textures, lighting, and object appearances in the real-world images that were not present in the synthetic dataset.

2. Surprisingly, for the YOLO model, domain adaptation using DANN led to worse results on the real-world data compared to the non-adapted baseline. This degradation was likely due to the adversarial training process, which may have

overly focused on confusing the domain classifier rather than preserving features critical for the segmentation.

3. Another challenge was the instability inherent in adversarial training. The balance between the feature extractor and the domain classifier is delicate, and during training, we observed that small changes in hyperparameters such as the learning rate and the gradient reversal factor could drastically affect model performance. If the domain classifier became too strong, the feature extractor would overly focus on making the domains indistinguishable at the cost of degrading task-specific features, which we believe contributed to the issues seen with the YOLO model.

4. The feature extractors used in both the FCN and YOLO models might not have been sufficiently flexible to capture the complex variations in the real-world data. Even with domain adaptation, the models were unable to capture the diversity in object shapes, lighting conditions, and backgrounds present in the target domain, leading to poor generalization.

5. Not enough time: the training of the FCN model with the adversarial setup takes a lot of time and therefore we were not able to try a sufficient amount of hyperparameters. Maybe checking more options would have led to better results.

6. Ultimately, our goal was to use the trained models to produce pseudo-labels like we did in HW1, but since the models did not perform well on the real data, we were unable to do so.

- **<u>Recommendations for Further Refinement of the Domain Adaptation Process:</u>**
    - Improving Synthetic Data Quality: One of the key reasons for the domain gap is the difference in quality and realism between the synthetic and real-world datasets. To reduce this gap, we recommend generating synthetic data that better simulates the real-world conditions.

    - A possible way to improve the realism of the synthetic generated data is pasting the hand model alongside the surgical tool it holds (note that the tweezers are always in the left hand, and therefore located on the right of the needle handler). The position of the pasted models should correspond to the removed bounding box. This should help the model understand where to look for the desired tool

    - Feature Alignment at Multiple Levels: The DANN approach aligns the feature distributions at a single point in the network, but real-world variations often occur at multiple levels of representation (low-level textures, mid-level object parts, high-level object semantics). A more advanced approach could involve multi-level feature alignment, where domain adaptation occurs at different layers of the network. For example, aligning both low-level (e.g., edges, textures) and high-level (e.g., object categories) features may provide more comprehensive adaptation.

    - Alternative Domain Adaptation Methods: While DANN is a popular approach for unsupervised domain adaptation (as far as our small research can tell), other techniques may be more suited to our specific task. For instance, CycleGAN or other image-to-image translation methods could be used to transform the synthetic data to look more like real-world data, thereby reducing the domain gap at the image level.

Summary:

Although our results on real-world data were not as expected, this project was a great learning experience. We learned how to generate synthetic data using BlenderProc and experienced the challenges of domain adaptation firsthand when applying models like FCN and YOLO to real-world data. This highlighted the domain gap problem and the limitations of current unsupervised learning methods like Domain-Adversarial Neural Networks.

Training the models using PyTorch taught us how to adjust architectures and optimize them for our task. We now understand that the domain gap was too wide for adversarial learning to be effective, probably mostly because of not good enough synthetic data.

Despite the challenges, this project gave us deep insights into domain adaptation and model training, and we look forward to building on this knowledge in our future studies.