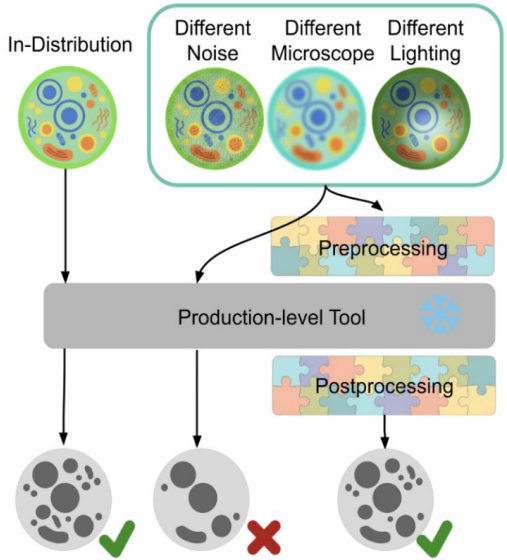
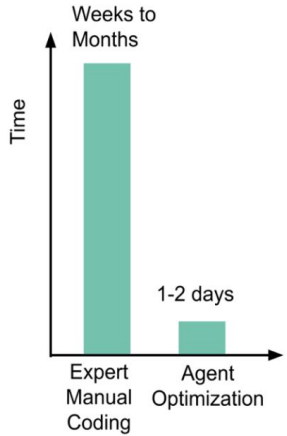
Simple Agents Outperform ***Experts*** in Biomedical Imaging Workflow Optimization

Xuefei (Julie) Wang[1], Kai A. Horstmann[2], Ethan Lin[2], Jonathan Chen[2], Alexander R. Farhang[1], Sophia Stiles[1], Atharva Sehgal[3], Jonathan Light[4], David Van Valen[1], Yisong Yue[1], Jennifer J. Sun[2]

Caltech[1], Cornell[2], UT Austin[3], Rensselaer Polytechnic Institute[4]

Motivation

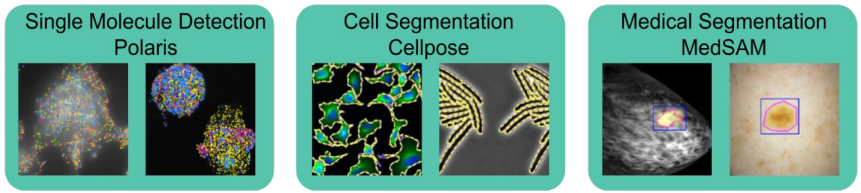




- Adapting production-level tools to new, *out-of-distribution* scientific datasets is a critical **"last-mile" bottleneck.**
- We consider using **AI agents** to automate this task.

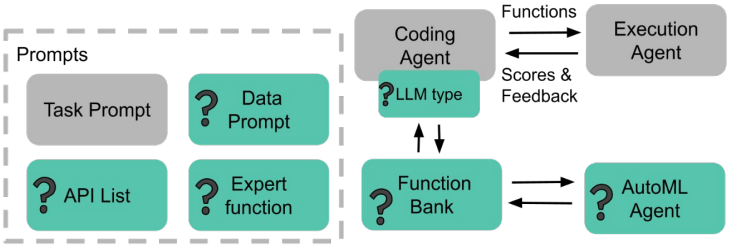
Framework

Tasks



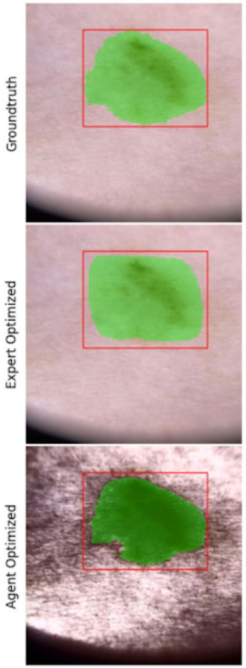
Scale	Task	Tool
Molecular	Single molecule detection	Polaris
Cellular	Cell segmentation	Cellpose
Macroscopic	Medical segmentation	MedSAM

Agent Design Space



Results

	Expert Baseline	Base Agent
Polaris (F1)	0.841	0.867
Cellpose (AP@IoU 0.5)	0.402	0.409
MedSAM (NSD+DSC)	0.820	0.971



Expert optimized - preprocessing

```python
import numpy as np
def preprocess_images(images, is_rgb=False):
    resized_imgs = images
    for i in range(len(resized_imgs)):
        img_np = resized_imgs[i]
        if is_rgb:
            resized_imgs[i] = np.uint8((img_np - img_np.min()) / (np.max(img_np) - np.min(img_np)) * 255.0)
        else:
            lower_bound, upper_bound = np.percentile(img_np[img_np > 0], 0.5), np.percentile(img_np[img_np > 0], 99.5)
            img_np_pre = np.clip(img_np, lower_bound, upper_bound)
            img_np_pre = (img_np_pre - np.min(img_np_pre)) / (np.max(img_np_pre) - np.min(img_np_pre)) * 255.0
            img_np_pre[img_np == 0] = 0
            resized_imgs[i] = np.uint8(img_np_pre)
    return resized_imgs
```

Percentile-based Normalization

Expert optimized - postprocessing

```python
import torch.nn.functional as F
def postprocess_preds(preds):
    upsampled = F.interpolate(preds, size=(512, 512), mode='bilinear', align_corners=False)
    upsampled = upsampled.squeeze(1)
    seg = (upsampled > 0.5).int()
    return seg
```

Bilinear Interpolation

Agent optimized - preprocessing

```python
import numpy as np
import cv2
def preprocess_images(images):
    processed_images = []
    for img_array in images:
        img_array = np.copy(img_array)
        min_val = img_array.min()
        ptp_val = np.ptp(img_array)
        img_norm = (img_array - min_val) / (ptp_val + 1e-8)
        img_8bit = (img_norm * 255).astype(np.uint8)
        img_ycrcb = cv2.cvtColor(img_8bit, cv2.COLOR_RGB2YCrCb)
        y, Cr, Cb = cv2.split(img_ycrcb)
        y = cv2.equalizeHist(y)
        img_ycrcb_eq = cv2.merge([y, Cr, Cb])
        img_eq = cv2.cvtColor(img_ycrcb_eq, cv2.COLOR_YCrCb2RGB)
        img_blur = cv2.medianBlur(img_eq, 3)
        img_blur = img_blur.astype(np.float32)/255.0
        processed_images.append(img_blur)
    return processed_images
```

Min-Max Normalization

Brightness-only Histogram Equalization

Median Blurring

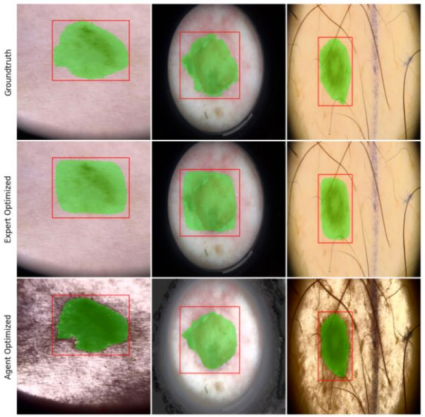
Agent optimized - postprocessing

```python
import torch
import numpy as np
import cv2
def postprocess_preds(preds):
    target_size = (512, 512)
    preds_copy = preds.clone().cpu().numpy()
    N, C, H, W = preds_copy.shape
    processed_masks = np.empty((N, 1, *target_size), dtype=np.float
    for idx in range(N):
        pred_mask = preds_copy[idx, 0]
        upsampled = cv2.resize(pred_mask, target_size, interpolation=cv2.INTER_LINEAR)
        binary = (upsampled > 0.45).astype(np.uint8)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))
        closed = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)
        processed_masks[idx, 0] = closed
    processed_masks = torch.from_numpy(processed_masks)
    processed_masks = processed_masks.squeeze(1)
    return processed_masks
```

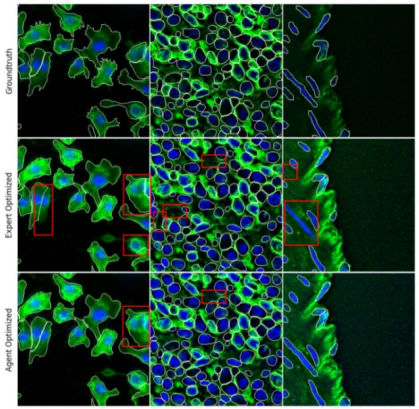
Bilinear Interpolation

Morphological Closing

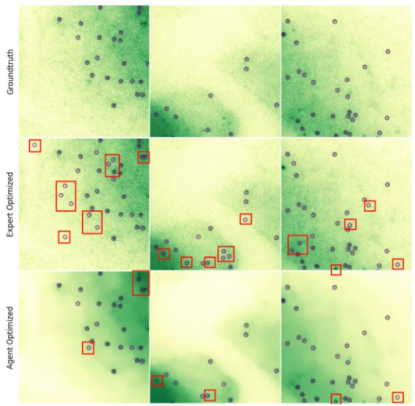
Results



Spot Detection - Polaris / DeepCell
F1 Score - Expert: 0.841 Agent: 0.929



Cell Segmentation - Cellpose 3 (cyto3)
AP @ IoU 0.5 - Expert: 0.402 Agent: 0.417



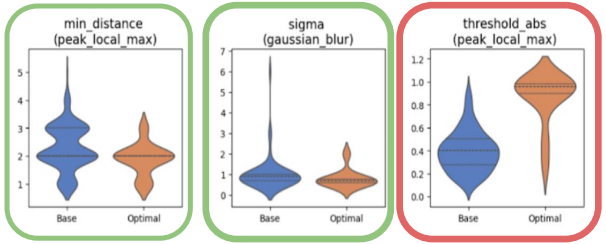
Medical Segmentation - MedSAM
NSD + DSC: Expert: 0.820 **Agent**: 1.037

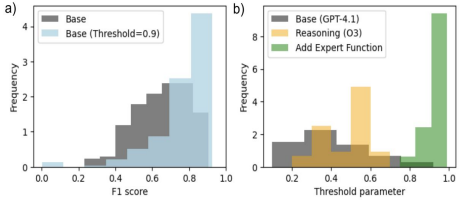
	Expert Baseline	Base Agent
Polaris (F1)	0.841	0.867
Cellpose (AP@IoU 0.5)	0.402	0.409
MedSAM (NSD+DSC)	0.820	0.971

Add Expert Function	Add Function Bank	Reasoning LLM	Small LLM	No Data Prompt	No API List
0.929	0.889	0.844	0.805	0.856	0.868
0.410	0.416	0.412	0.397	0.406	0.417
0.888	0.943	1.020	0.918	0.952	1.037

Solution Space Analysis

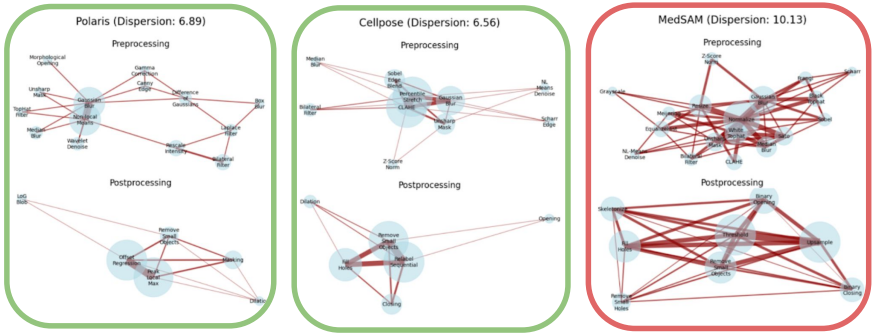
Parameter Space Analysis – Why does Reasoning LLM hurt Polaris performance?





- when LLM's prior on the parameter selection diverges from the optimal range, agent fails. Reasoning LLMs is more limited in its prior.

API Space Analysis – Why does Expert Function and Function Bank hurt MedSaM performance?



- When the optimal solution space is more diverse, presenting expert or prior solutions limits agent's exploration.

Search Space Summary

	API Space	Parameter Space
Polaris	Concentrated	Hard-to-optimize
Cellpose	Concentrated	Easy-to-optimize
MedSAM	Dispersed	Easy-to-optimize

Ablation Studies

Does AutoML Module help?

	Non-Agentic AutoML	Agent w/ FuncBank	Agent w/ FuncBank + AutoML
Polaris	0.844	0.889	0.877
Cellpose	0.373	0.416	0.417
MedSAM	0.879	0.943	1.014

- AutoML alone is not useful.
- AutoML + Agent can be useful if not overfitting.

Does complicated Tree Search Agent help?

	Base Agent	Base Agent w/ FuncBank	AIDE Agent (Tree Search)
Polaris	0.867	0.889	0.872
Cellpose	0.409	0.416	0.414
MedSAM	0.971	0.943	0.971

- No obvious advantages from Tree Search Agent.

Insight & Impact