DOE Award #: DE-SC0021555

AI-Based Stabilization of Sample Environments

M. Henderson*, J. Edelen, M. Kilpatrick, and I. Pogorelov

*mhenderson@radiasoft.net

EPICS Collaboration

Meeting

April 24 – 28, 2023

RadiaSoft LLC | Boulder, Colorado USA |
radiasoft.net

Outline

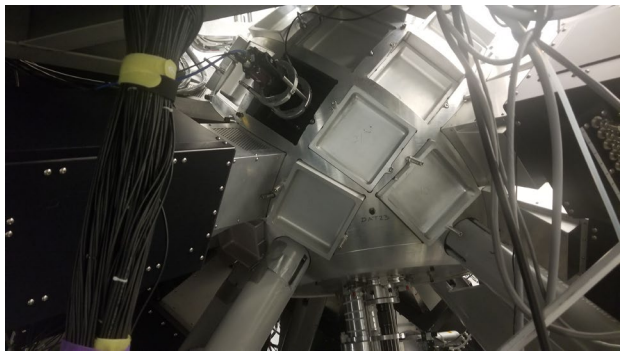
- Motivation & problem definition
- Controls framework
- Machine learning solutions
- Results & ongoing work

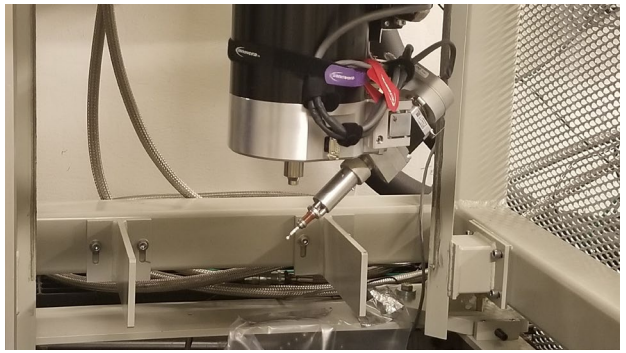
Motivation & problem definition

Beamlines, sample alignment, and computer vision

The Beamlines: TOPAZ

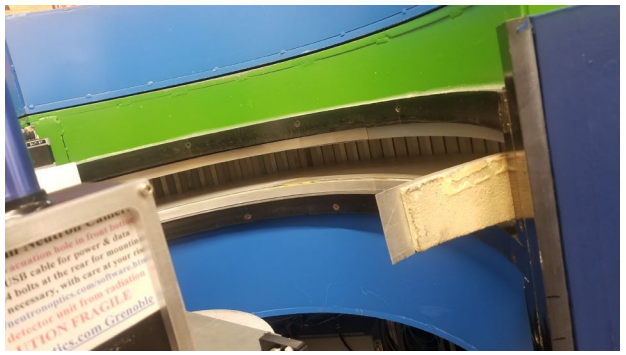
- Fed by the Spallation Neutron Source (SNS)
- Sample in a chamber
  - Sample arm with 3 translational, 2 rotational axes
  - Neutron detectors, cameras, & environmental controls
- Mature EPICS system
  - Plenty of control channels (PVs already exist)
  - Pre-existing IOC software
    - Includes point-and-click sample alignment
    - Must consider interactions with our framework

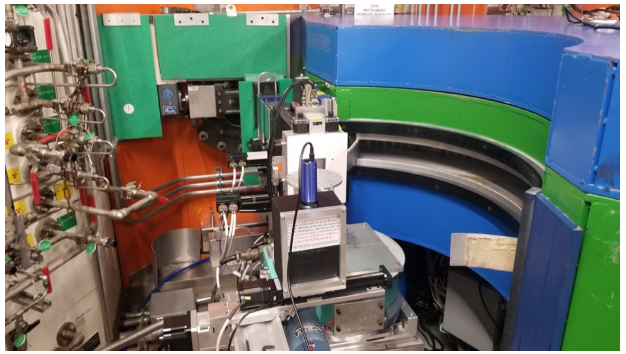




The Beamlines: HB2A

- Fed by the High-Flux Isotope Reactor (HFIR)
- Samples in containers (cans) on a stage
  - 3 translational axes, stage & sample rotation axes
  - Partially encircled by a neutron detector
  - Diagnostics done with a **neutron** camera
- Controls executed with SPICE (no EPICS)
  - Need a “bridge” between software to use EPICS
  - Sample alignment completely manual



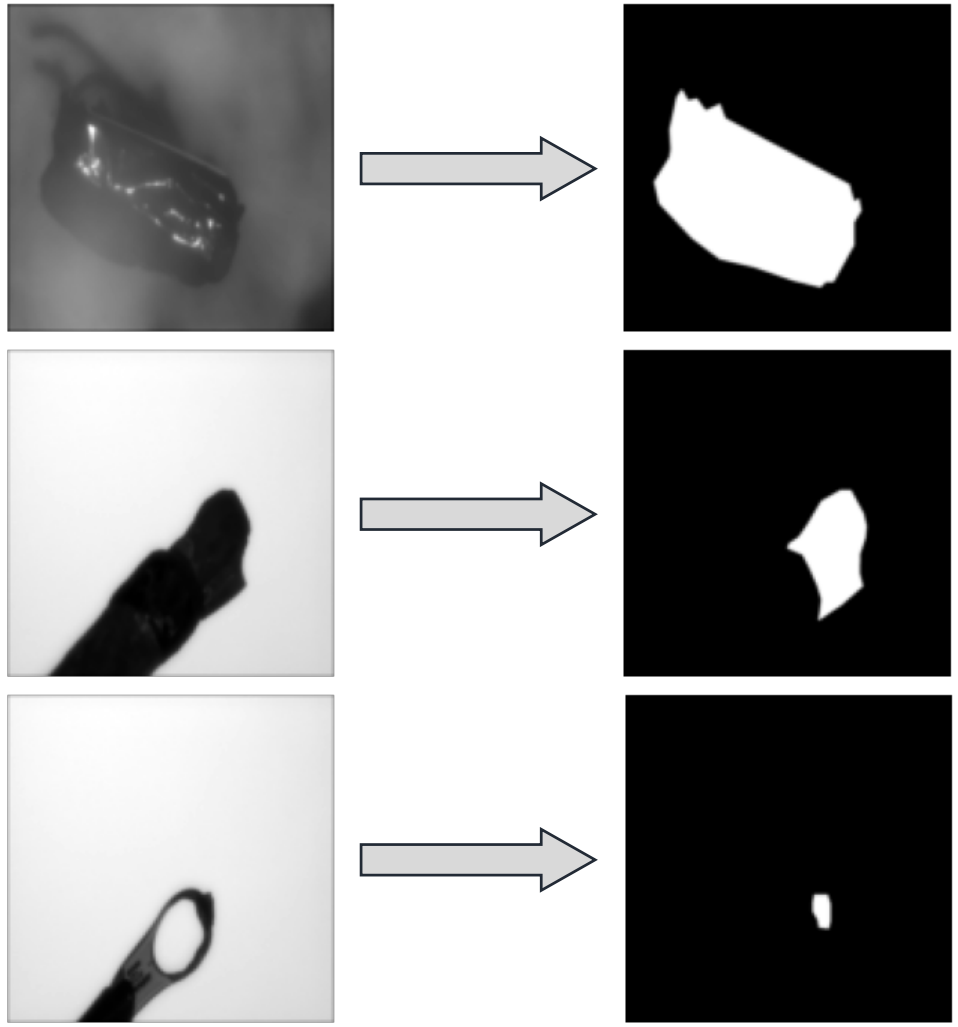


Background & Motivation

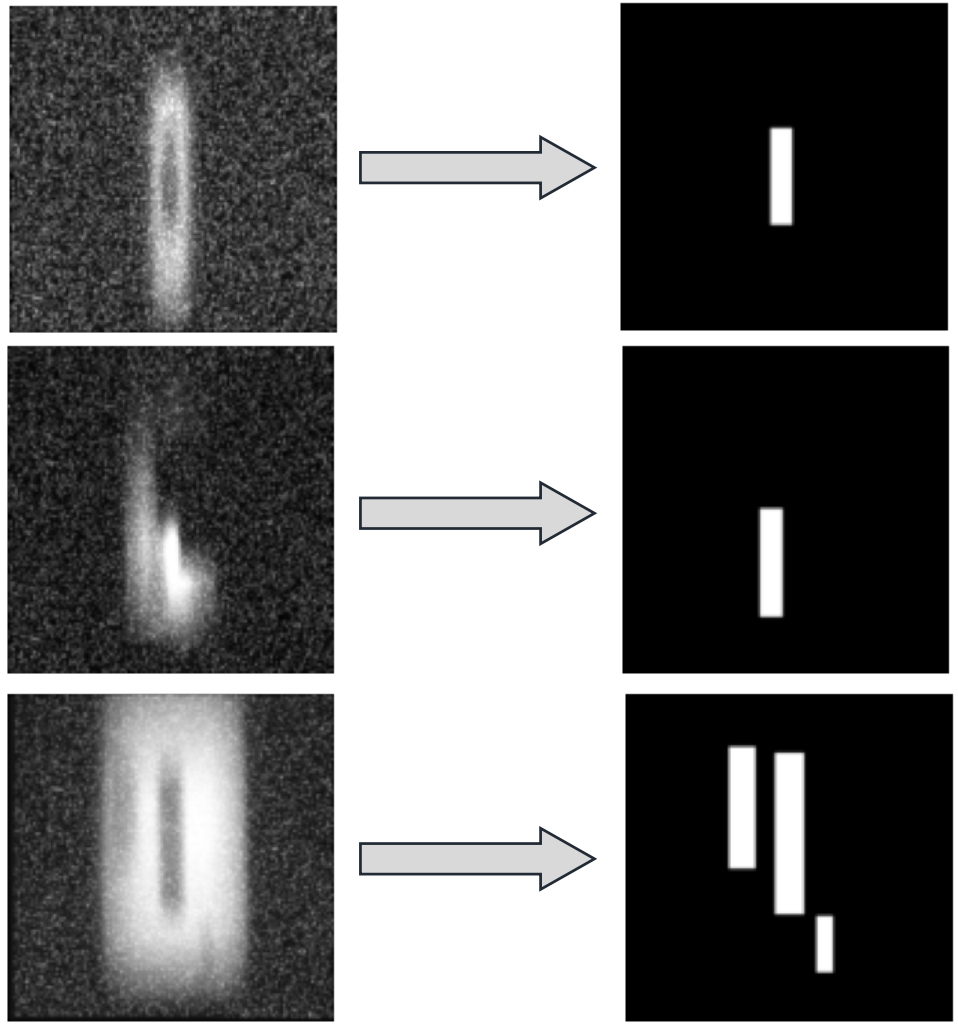
- Sample alignment is tedious, but critical
  - Currently requires human image processing
  - Limited neutron production time, schedule constraints
- Machine learning is a key automation tool
  - For computer vision, convolutional neural networks (CNNs)
- Alignment protocols are distinct to a beamline
  - Framework must be highly general & robust
  - Opportunity to deploy transfer learning

Beamline Image Masks

TOPAZ



HB2A

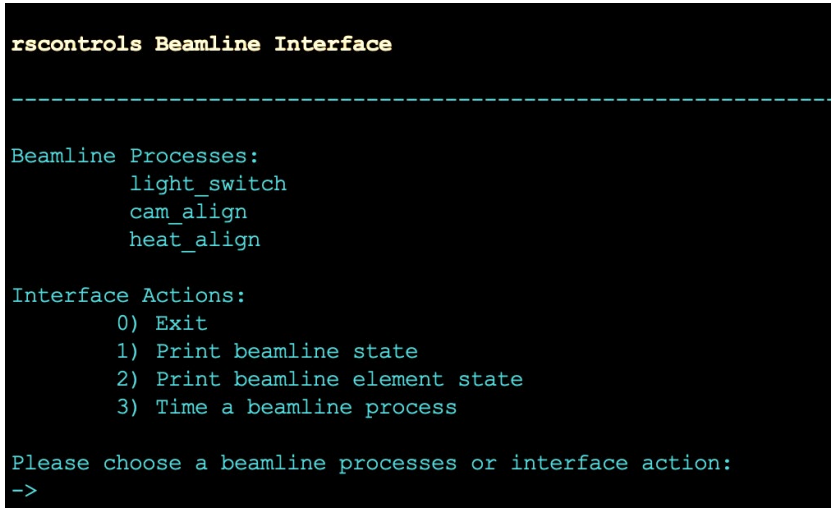


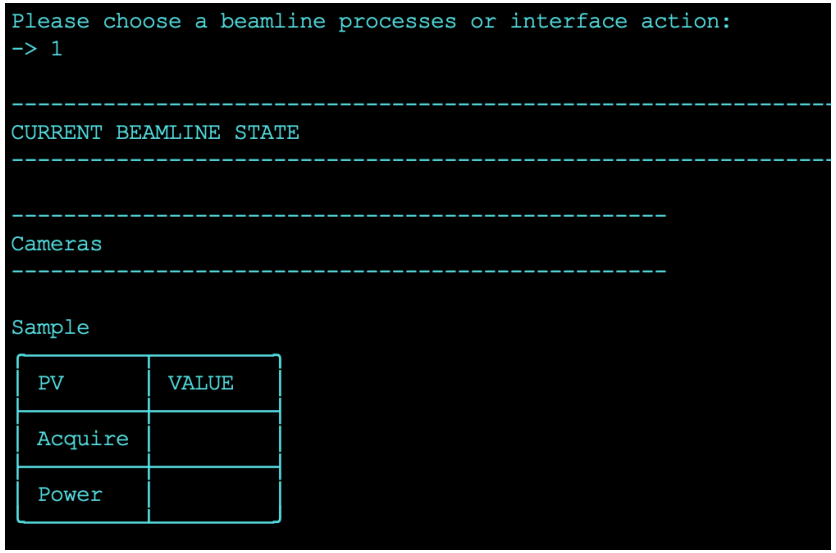
Controls framework

Implementing an EPICS interface with embedded automation tools

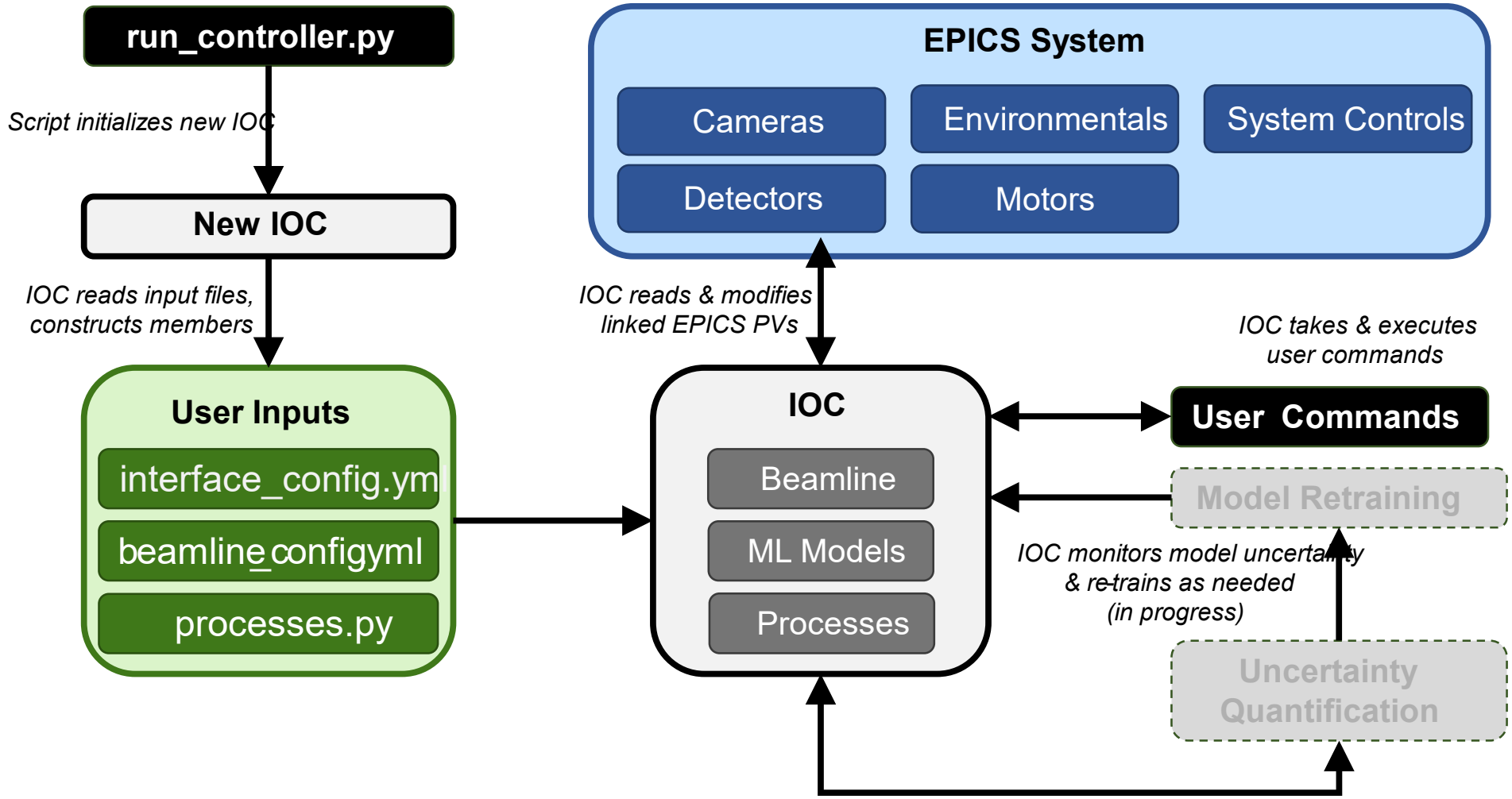
Controls Framework

- Top-level controller
  - Handles elements & models
  - Executes control processes
- Virtual beamline elements
  - Cameras, motors, detectors, etc.
  - Methods & properties linked to EPICS PVs
- ML models
  - User-provided, passed to control processes
- Command-line interface for testing





Python “IOC” Workflow



Machine learning solutions

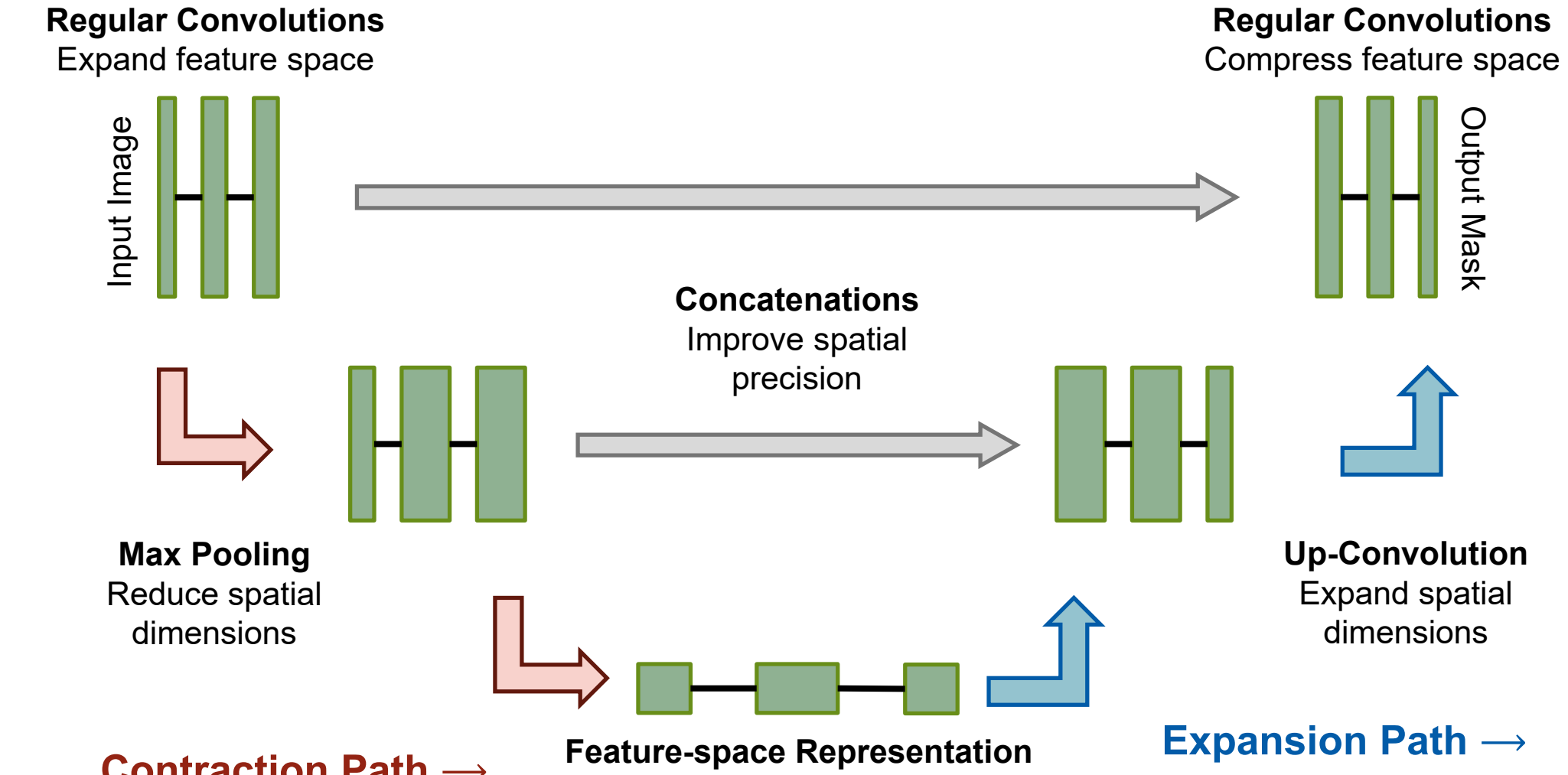
Developing ML models for beamline computer vision tasks

Network Architecture

- Convolutional neural network (CNN)
  - Convolution layers extract features
  - Many different network types available
- Adopt U-Net architecture
  - Originally designed for medical imaging tasks[1]
  - Features compression/extraction scheme
    - Also a form of encoder/decoder, useful for denoising

[1] *Ronneberger, Fischer, and T. Brox (2015)*

Network Architecture



Neutron Image Denoising

- Diagnostics at HB2A done by neutron camera
  - Aging hardware, noisy images
  - Want to train on images with transferable features
- Images denoised for training
  - Must denoise during live operations
- Consider quality & time-cost of options
  - Use same denoising protocol used for training images
  - Train an additional network to denoise images on the fly

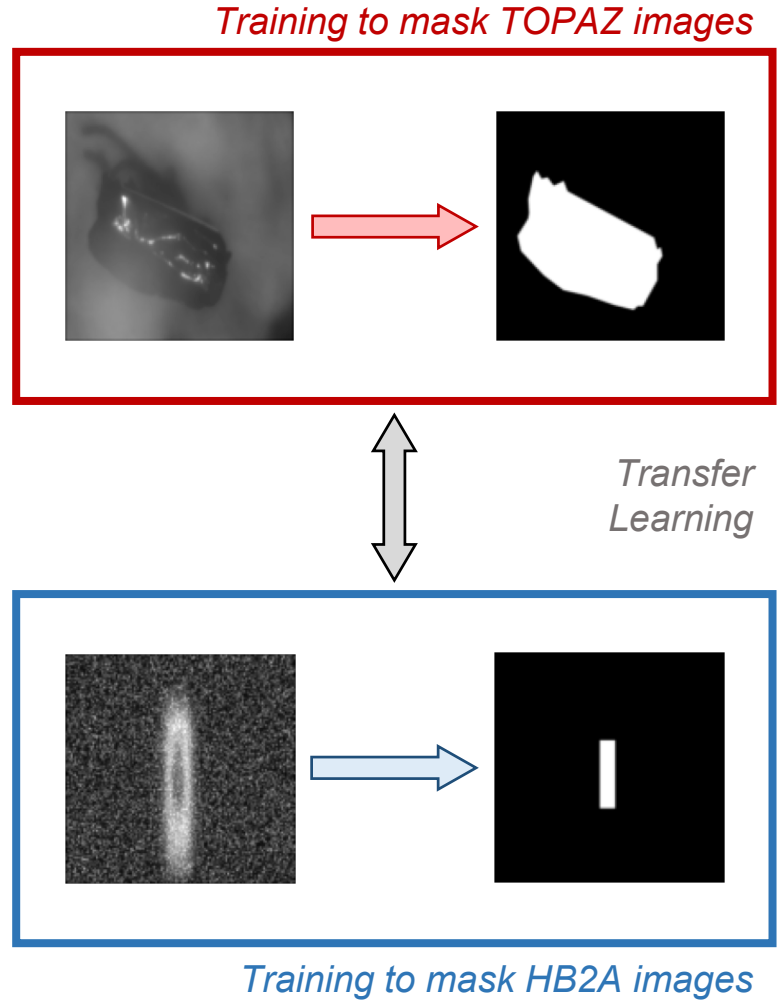
Uncertainty Quantification (UQ)

- Live ML applications require robust UQ
  - Need to know which ML results to trust
  - No access to ground-truth
- Often provided by ensemble statistics
  - Variance of predictions by an ensemble of models
- Aleatoric vs. epistemic uncertainty[2]
  - Statistical fluctuations vs. lack of information

[2] *Hüllermeier and Waegeman (2021)*

Transfer Learning[3,4]

- Retrain on data from other beamline
- Test effect of training pipeline
  - Hyperparameter optimization
  - Number of rounds
  - Order of training
- Requires training *many* models
  - Especially if training ensembles



[3]Caruna (1997)

[4]Pan and Yang (2010)

Results & ongoing work

Phase 2, year 1 accomplishments and tasks for year 2

Results & Current Status

- Neutron camera images denoised
  - Comparison between ML & regular filter solutions completed
- Robust mask generating CNNs developed
  - Training & architecture optimization
  - Uncertainty quantification
  - Transfer learning
- Extensive testing carried out at TOPAZ
  - Successful alignment procedure completed
- Infrastructure developments at HB2A well underway
  - Began testing EPICS/SPICE bridge & motor controls

Denoising Results

Image denoising, regular filter

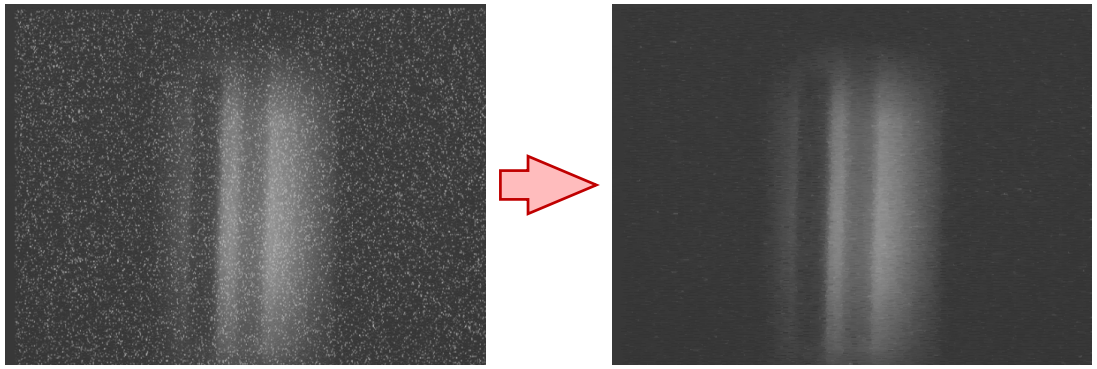
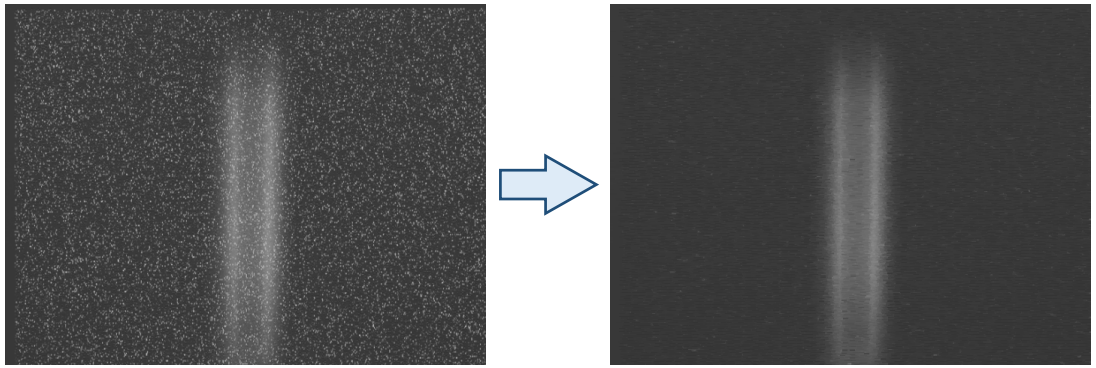
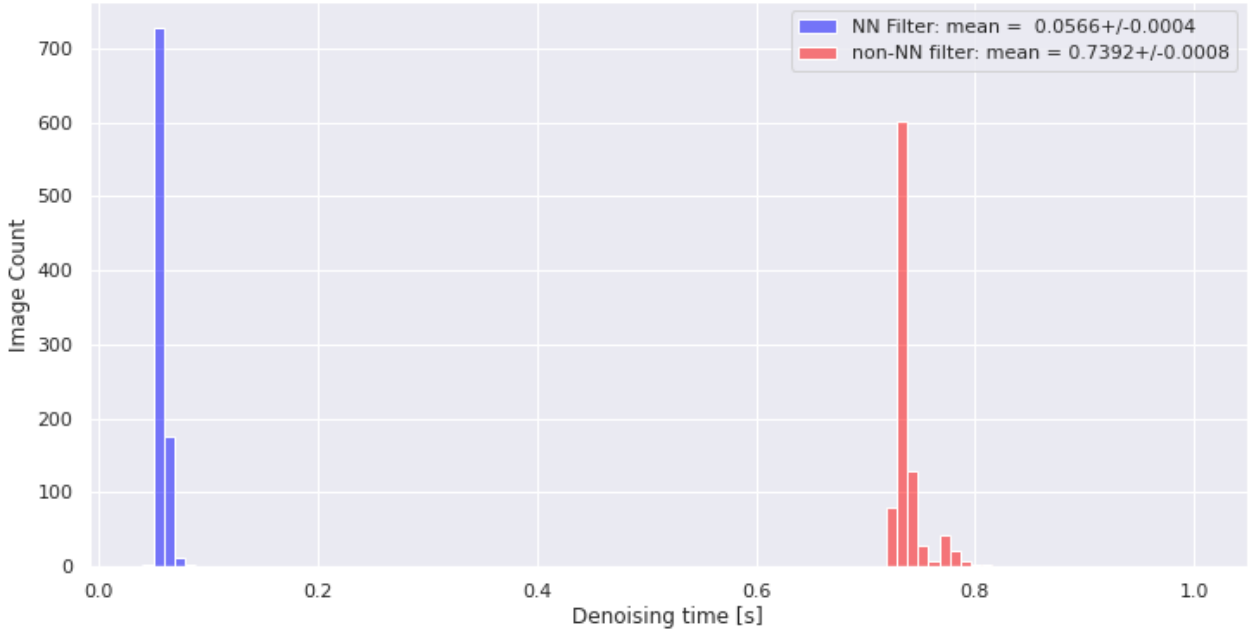


Image denoising, NN filter

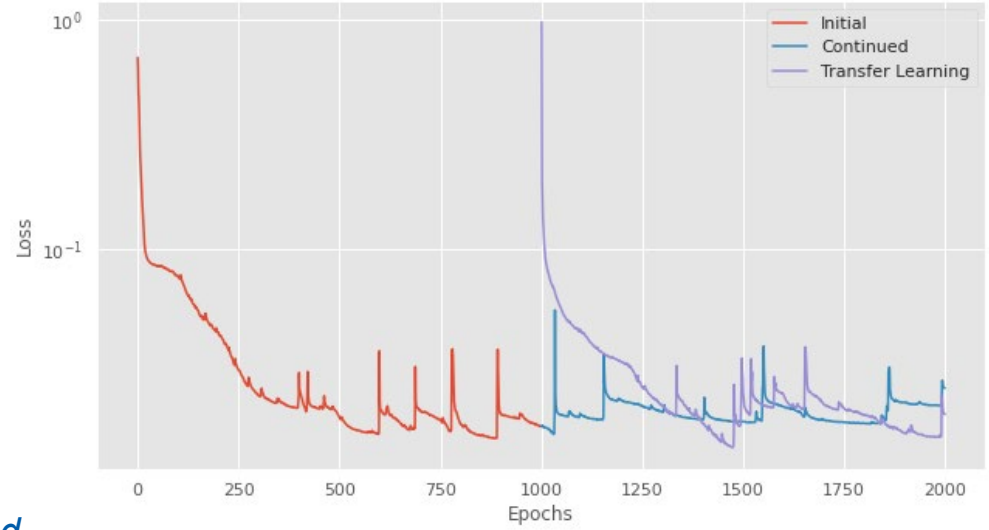


Execution time, regular vs. NN filter

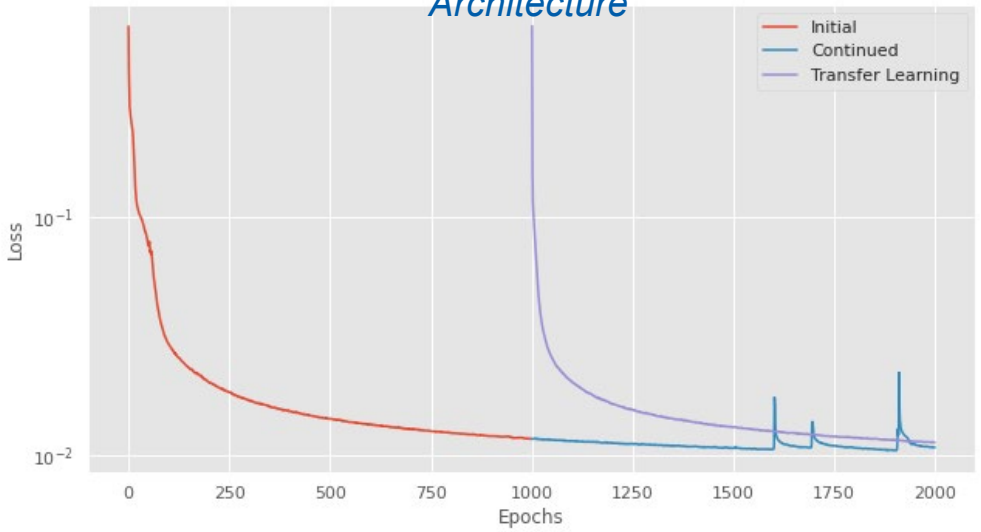


Transfer Learning Results

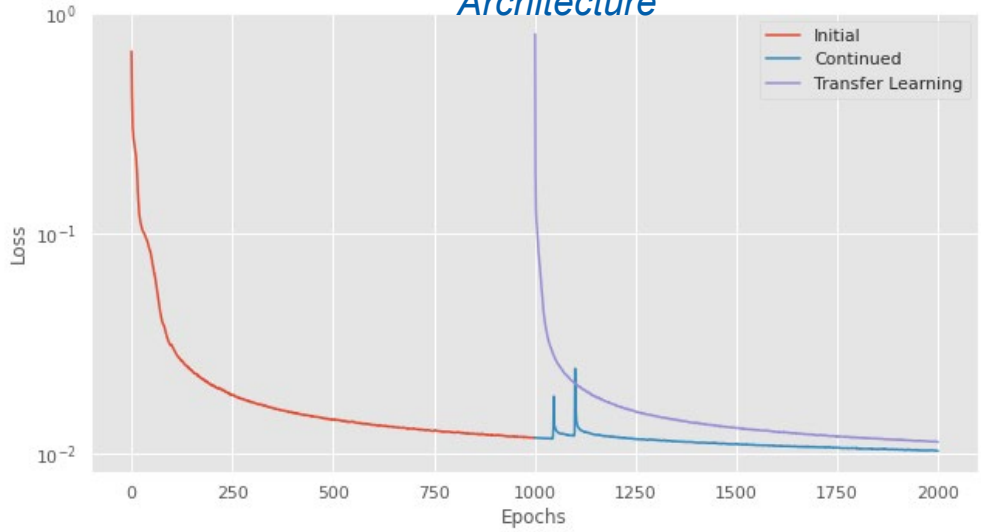
Unoptimized Architecture



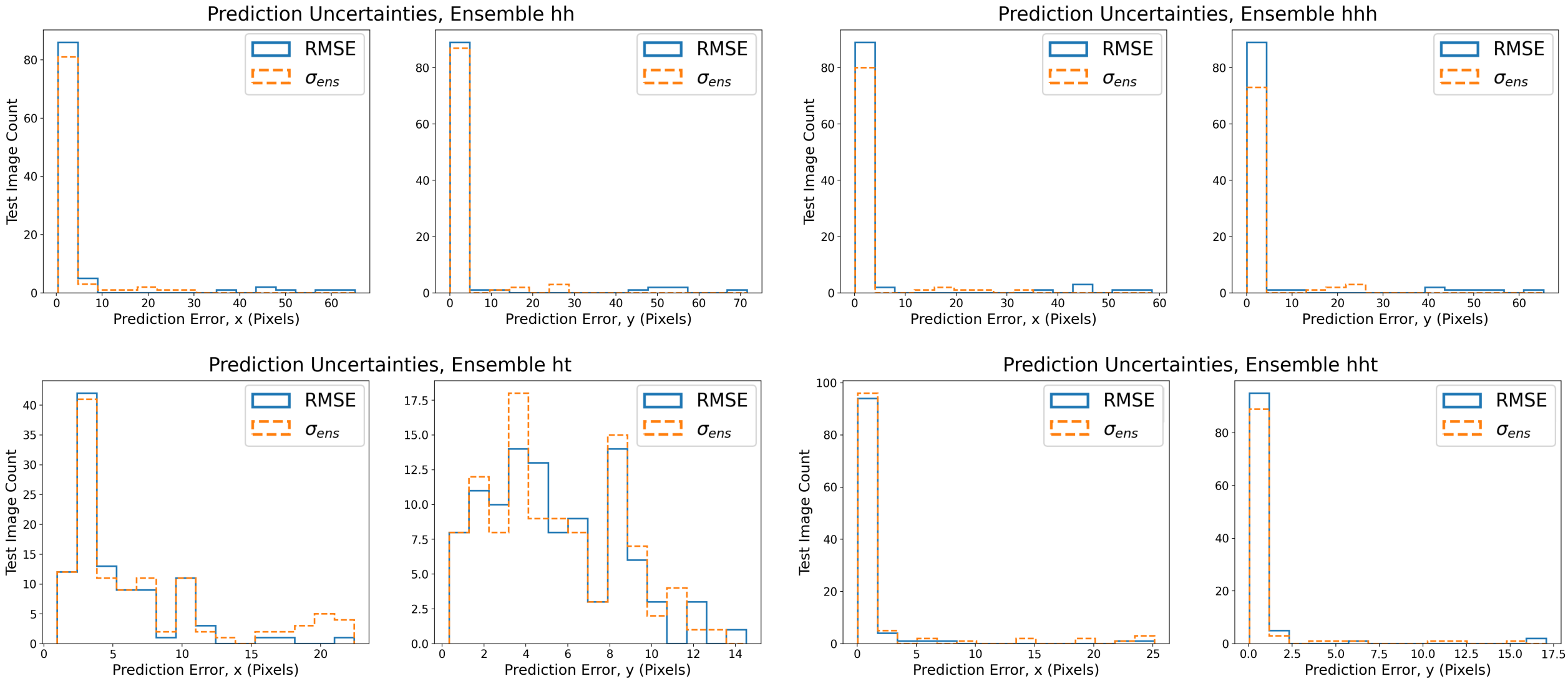
HB2A-Optimized Architecture



TOPAZ-Optimized Architecture

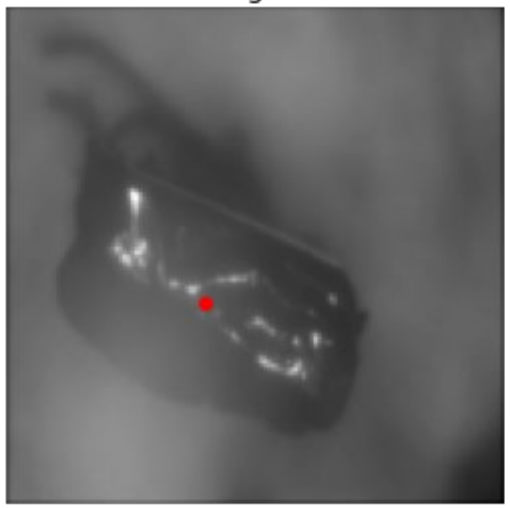


Ensemble Uncertainties: Epistemic vs. Aleatoric

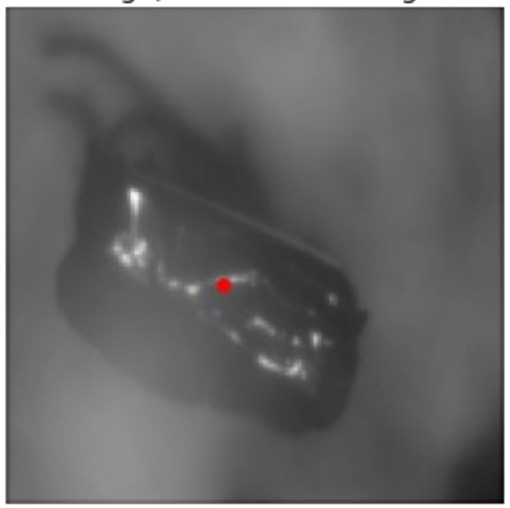


Ensemble Prediction Results (Unoptimized)

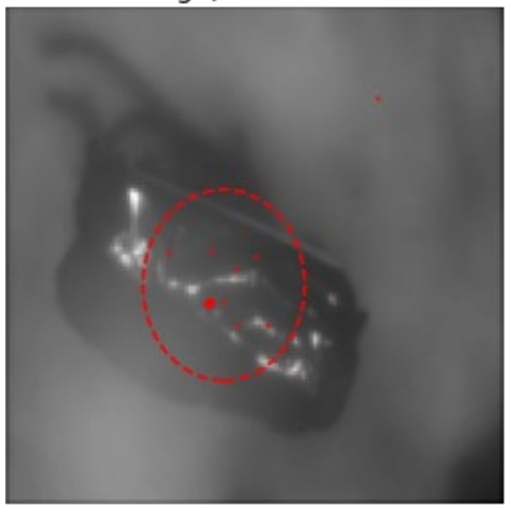
Test Image & CoM



Test Image, Ensemble Average CoM

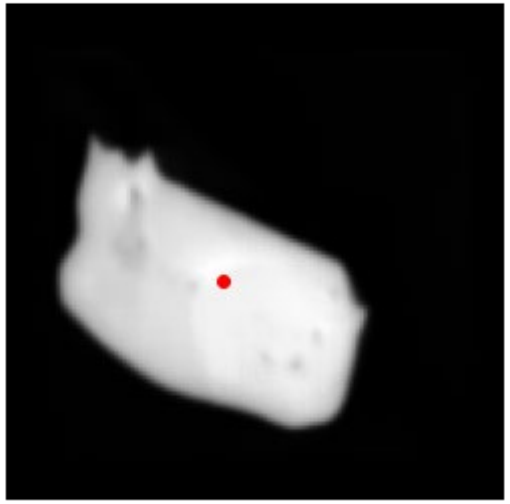


Test Image, Ensemble CoMs

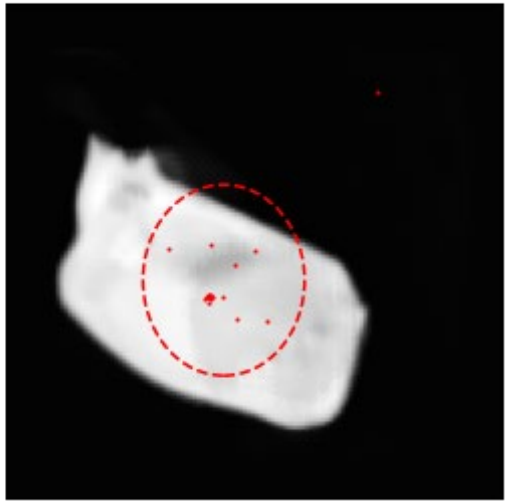


Test Mask & CoM

Ensemble Average Mask & CoM

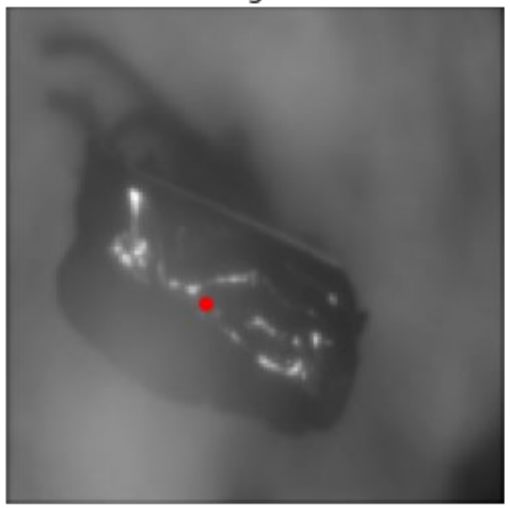


Ensemble Mask Variance & CoMs

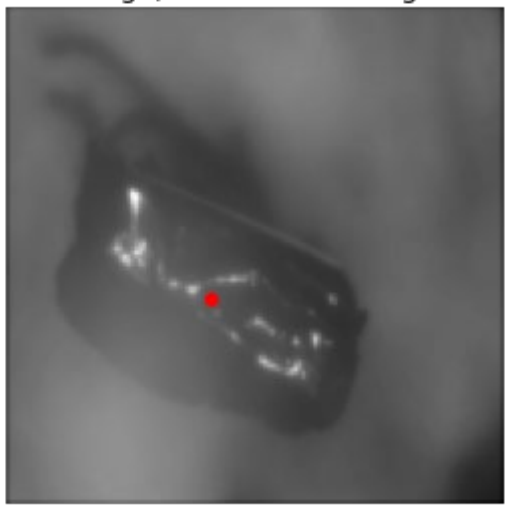


Ensemble Prediction Results (Optimized)

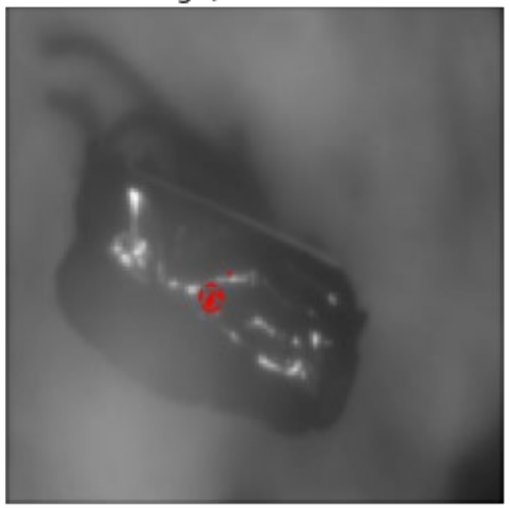
Test Image & CoM



Test Image, Ensemble Average CoM

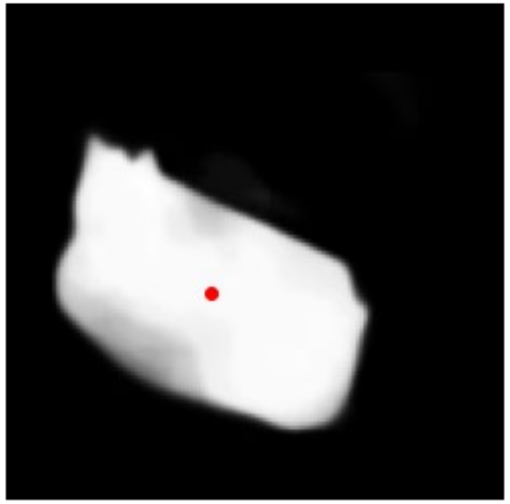


Test Image, Ensemble CoMs

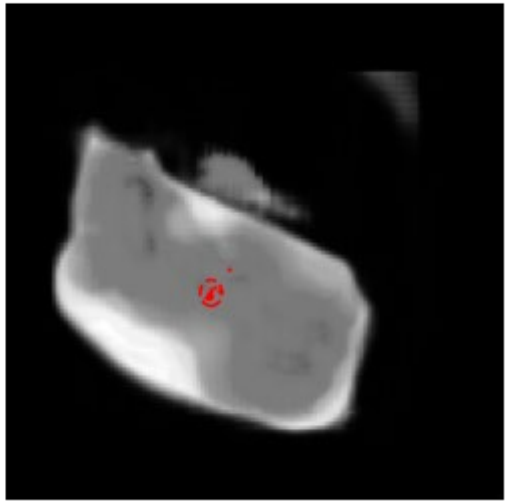


Test Mask & CoM

Ensemble Average Mask & CoM

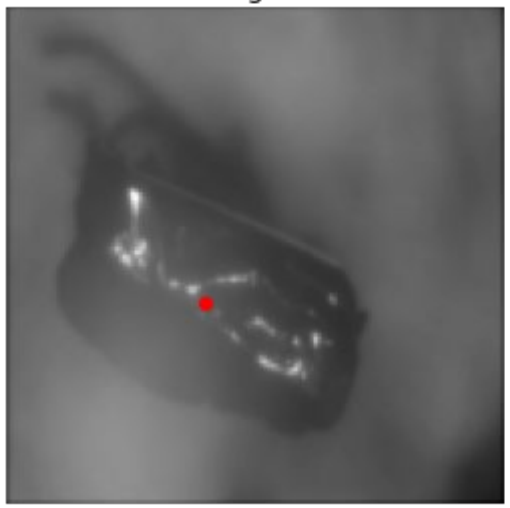


Ensemble Mask Variance & CoMs

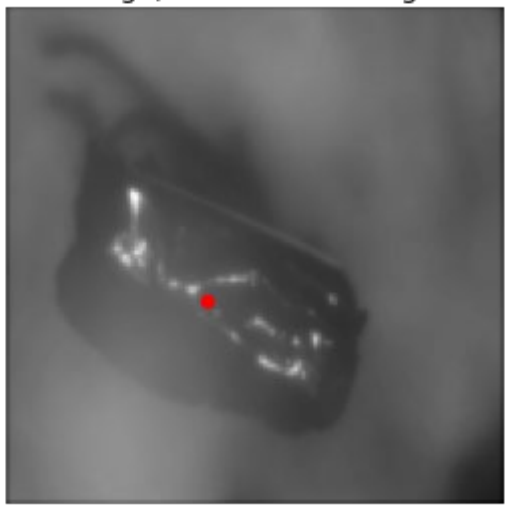


Ensemble Prediction Results (Transferred)

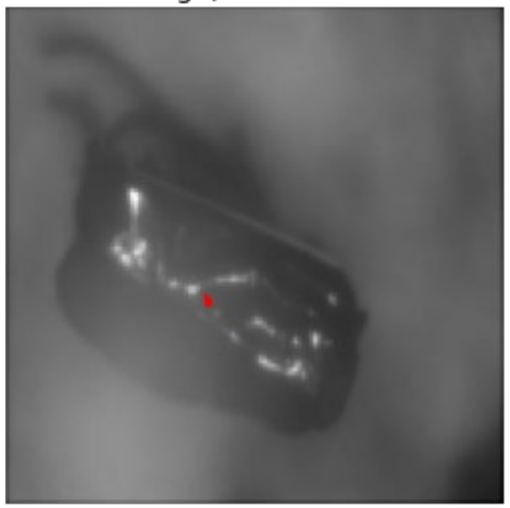
Test Image & CoM



Test Image, Ensemble Average CoM

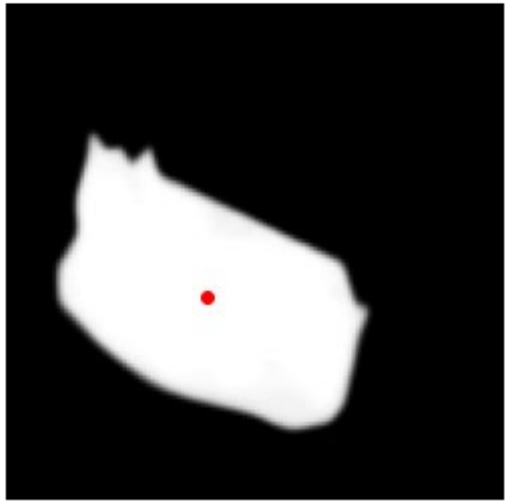


Test Image, Ensemble CoMs

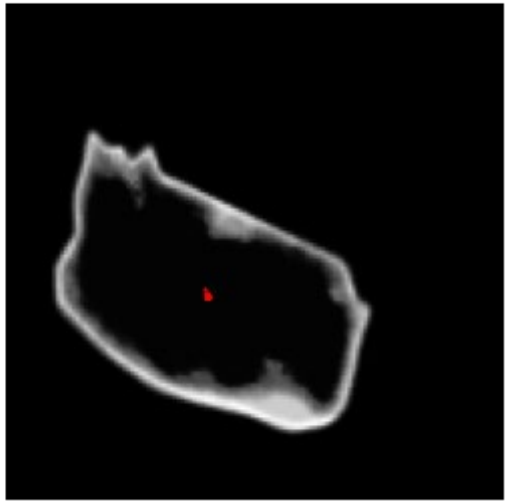


Test Mask & CoM

Ensemble Average Mask & CoM



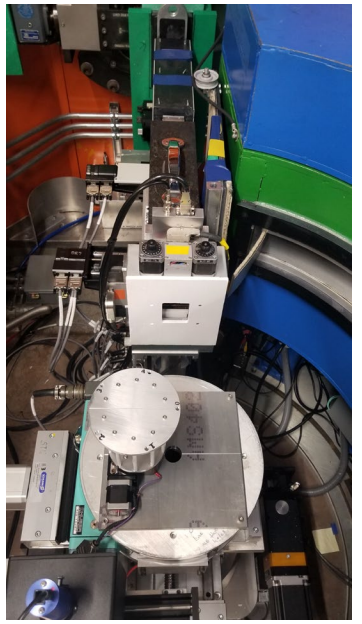
Ensemble Mask Variance & CoMs

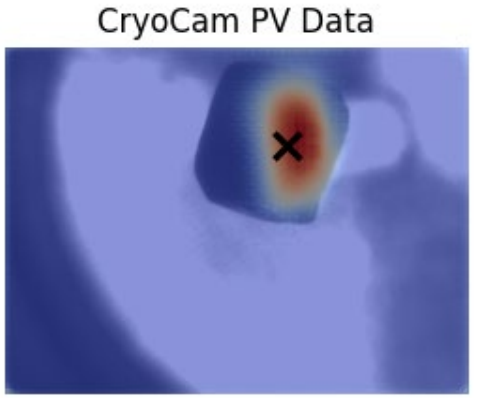


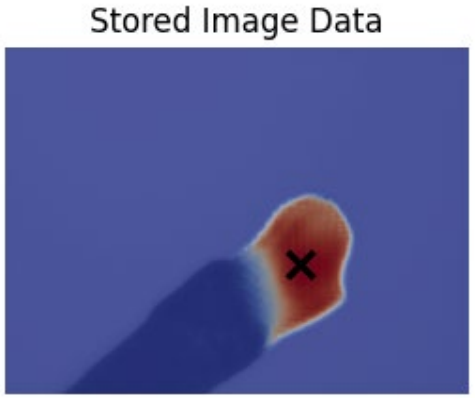
Latest Testing Round, TOPAZ

Ongoing Work

- Address shift at progress PV change on TOPAZ
- Start implementing alignment processes on HB2A
  - Initial controls testing completed (as of 04/21/2023)
  - Testing initial alignment approaches in simulated settings
- Extra training on cryo-mode images with artefacts
  - Need only identified through live testing
- Move to detector-driven alignment
  - Optimize data value at TOPAZ
  - Critical alignment tool at HB2A







Thank you!

(Q&A)

Additional Slides

References & unused materials

References

1. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention* (2015)

2. E. Hüllermeier and W. Waegeman, “Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods,” *Machine Learning* **110** (2021)

3. R. Caruana, “Multitask Learning,” *Machine Learning* **28** (1997)

4. S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering* **22** (2010)

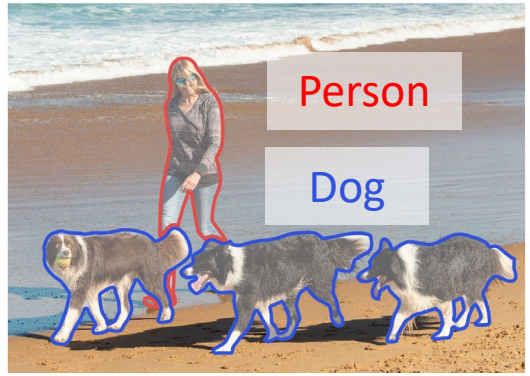
Types of Computer Vision Tasks

Image Classification

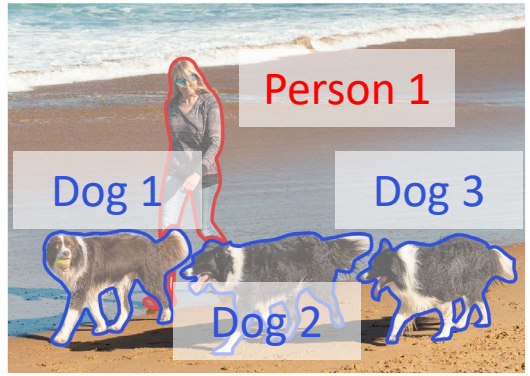
Classification with Localization

Object Detection

this project



Semantic Segmentation



Instance Segmentation

Images courtesy of Qualcomm Developer Network

Example Controller Configuration

```yaml
---
PATH : ../

BEAMLINE:
  MODES : [standard, special]
  PATH : config/beamline_config.yml

MLMODELS:
  UNet :
    TYPE : UNET
    MODEL_PATH: models/new-unet_weights.h5
    ARCH_PATH: models/new-unet_arch.pkl

PROCESSES:
  PATH : config/processes.py

  light_switch : [SampleLED]

  cam_align:
    standard: [SampleCam, SampleArm, CameraCentering, BeamState, UNet]
    special: [SpecialCam, SpecialArm, SpecialCentering, BeamState, UNet]

  heat_align :
    standard: [SampleCam, SampleArm, Thermostat, CameraCentering, BeamState, Unet]
    special: [SpecialCam, SpecialArm, Thermostat, SpecialCentering, BeamState, UNet]
```

Example Beamline Configuration

```
---
PREFIX : EX

CAMERAS :
 PREFIX : Cam

 SampleCam :
  MODE : standard
  PREFIX : SamCam
  IMAGE_DIMENSIONS : [1920, 1080]
  IMAGE_PV : SamCamArray
  STATE_PVS : [Acquire, Power]

 SpecialCam :
  Mode : special
  PREFIX : SpecCam
  IMAGE_DIMENSIONS : [1920, 1080]
  IMAGE_PV : SpecCamArray
  STATE_PVS : [Acquire, Power]

DETECTORS :
 PREFIX : Det

 NeutronDetector :
  PREFIX : Ndet
  DATA_DIMENSIONS : [12, 50]
  DATA_PV : NDArray
  STATE_PVS : [Acquire, Power]
```

```
MOTORS :
 PREFIX : Mot

 SampleArm :
  MODE : standard
  PREFIX : SamArm
  MOTOR_PVS : [X, Y, Z, phi, theta]
  TWEAK_VALS : [0.1, 0.1, 0.1, 5, 5]
  PROC_PVS : [Home, Th0, Th90, ThN90, Ph0, Ph90, Ph180]

 SpecialArm :
  MODE : special
  PREFIX : SpecArm
  MOTOR_PVS : [X, Y, Z, phi, omega]
  TWEAK_VALS : [0.1, 0.1, 0.1, 5, 15]
  PROC_PVS : [Home, Om0, Om90, OmN90, Ph0, Ph90, Ph180]

ENVIRONMENTALS :
 PREFIX : Env

 Thermostat :
  PREFIX : Thermo
  ENV_PVS : [Temp, dTTol, TargTemp, RampRate]
  PROC_PVS : [RampTemp, StopRamp, RoomTemp]

 SampleLED :
  PREFIX : SamLED
  ENV_PVS : [Power, Intensity]
```

```
CONTROLS :
 PREFIX : Ctrl

 BeamState :
  PREFIX : Beam
  CONTROL_PVS : [BX, BY, ToF]

 CameraCentering :
  PREFIX : CamCenter
  CONTROL_PVS : [XSam, YSam, XMMPP, YMMPP]

 SpecialCentering :
  PREFIX : SpecCenter
  CONTROL_PVS : [XSam, YSam, XMMPP, YMMPP]

 DetectorCentering :
  PREFIX : DetCenter
  CONTROL_PVS : [XSam, YSam, XMax, YMax, IMax]
```

Handling User-Defined Controls

- Provided in Python scripts
  - File imported by controller
  - Processes defined as functions
- Take inputs from controller
  - Beamline elements
  - ML models
- Execute gets/puts
- Call other processes

```
def light_switch(LED):
    """Changes the on/off state of an LED"""
    LED.put_pv('Power', int(not int(LED.get_pv('Power'))))

def cam_align(camera, sample_arm, cam_centering, beam_state, unet):
    """Aligns a sample using the sample camera"""

    # Get an image from the camera
    image = camera.get_image()

    # Pass the image through the UNet to get the sample CoM & read the beam CoM
    _, samCoM = unet(image)
    beamCoM = array([beam_state.get('BX'), beam_state.get('BY')])

    # Make the motor adjustments to align centers of mass
    mmpp = array([cam_centering.get('MMPP_X'), cam_centering.get('MMPP_Y')])
    px_move = mmpp*(beamCoM - samCoM)
    sample_arm.move('X', px_move[0])
    sample_arm.move('Y', px_move[1])

def heat_align(camera, sample_arm, thermostat, cam_centering, beam_state, unet):
    """Heats & aligns a sample using the sample camera"""

    # Get & apply a target temperature for the thermostat
    target_temp = input("\n\tTarget temperature: ")
    thermostat.put_pv('TARGETTEMP', target_temp)

    # Call the normal camera alignment process
    cam_align(camera, sample_arm, cam_centering, beam_state, unet)
```

Interoperating with Existing IOCs

- Mature EPICS systems can speed development
  - User needs & behaviors known *a priori*
  - Existing PVs provide many knobs to turn
- Existing IOCs pose a challenge for new framework
  - Constantly watching for changes to PVs
    - Possibly gatekeeping all read/write calls
  - Introduce an assortment of “virtual” PVs
- Need to work with existing IOCs whenever possible

While You're At It…

- Noise reduction becomes a service
  - Neutron cameras subject to significant noise
    - Denoised images used to train CNN
  - Adding dedicated denoising channel for HB2A
    - Read raw image PV, denoise, write to "clean" PV
- Clarifying functions of legacy software
  - Expertise with critical software diminishes
  - Interfacing with legacy code provides discovery process
    - Existing process replacement/integration at TOPAZ
    - Building the EPICS/SPICE bridge for HB2A
      - Constructed by Gary Taufer at ORNL