Deep Learning based microscopy image analysis pipelines

Conor L. Evans

Wellman Center for Photomedicine
Massachusetts General Hospital
Harvard Medical School

Team Central Mission

We are focused on problem-driven research developing optics, photonics, and computational based-solutions to understand and treat human disease

Acknowledgments

Alice Chao
James Chen
Avery Goss
Dan Greenfield
Helen Kashishian
Alex Ho
Xiaolei Li
Dr. Juan Cascales
Dr. Sinyoung Jeong
Dr. Ben Kuzma
Dr. Haley Marks
Dr. Matthias Muller
Dr. John Nguyen
Dr. Isaac Pence
Dr. Fei Peng
Dr. Manolis Rousakis
Dr. Lillian Witthauer
Dr. German Gonazles
Dr. Amin Feizpour

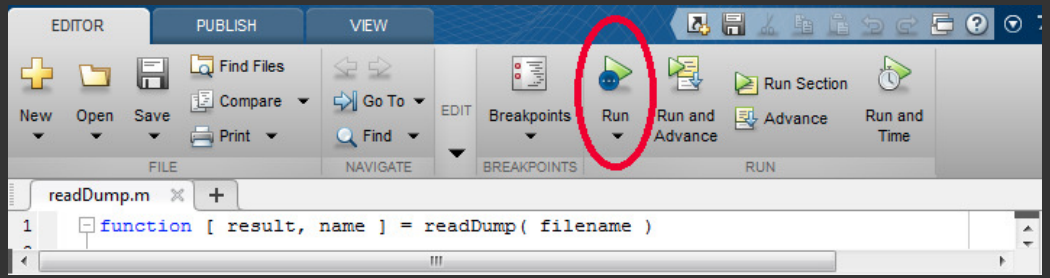
Acknowledgments

Problem: Data =/= Information

- Biomedical imaging experiments can generate huge amounts of data
  - Optical coherence tomorgraphy: 50GB per image volume
  - Multiphoton time-lapse imaging: 10's GB per stack
  - DiSPIM: 1-2 TB per image volume
- Very little of this data is information; the key information is contained within images and needs analysis and quantification

Current Inefficiencies

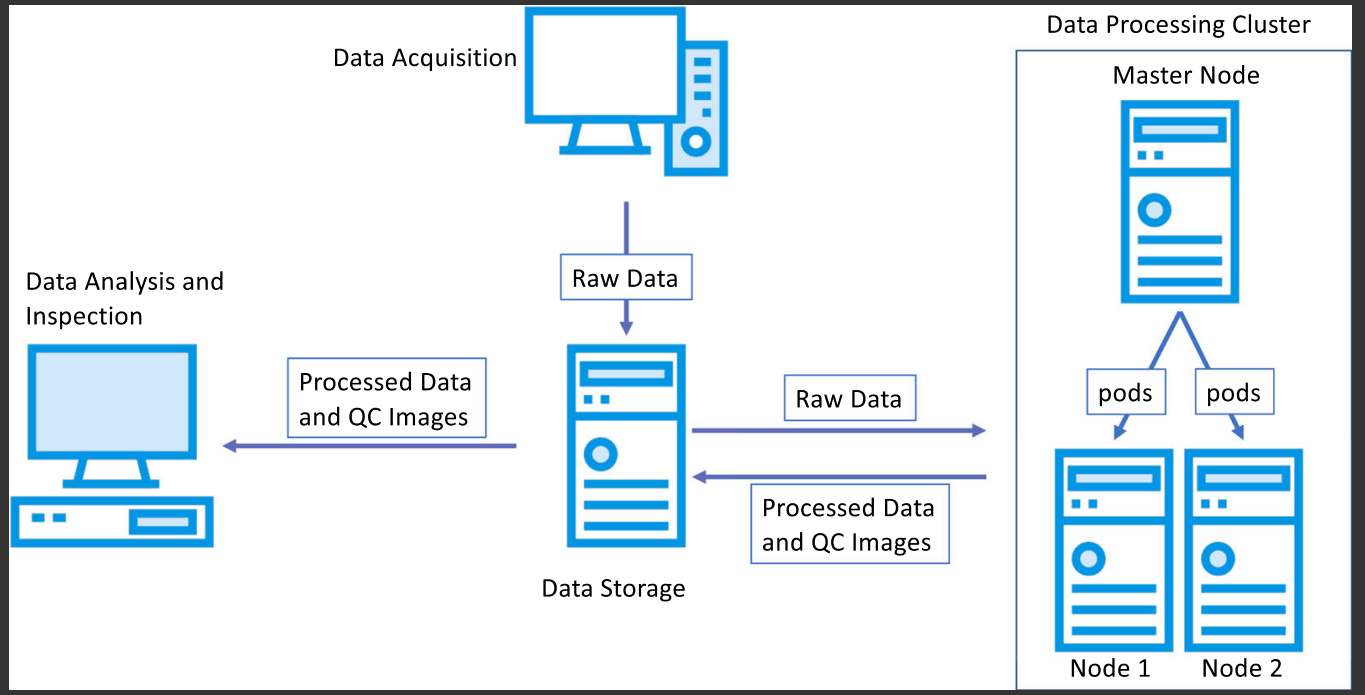
- Most students and fellows in my field are trained in small-scale, matlab-based analysis
- These skills do not scale well to medium-to-large datasets
- Typical pattern: One day of experimentation requires a week or more of manual analysis



Goal: “Cognitive Offloading”

- Much of image analysis involves comparison of signals (e.g. brightness) and pattern matching.
- Can we remove this burden from the researcher and move it to a fully automated stack?
- The key motivation is to make these analysis *background* processes that run overnight and deliver actionable information
- Shorten the acquisition-analysis cadence and give students/fellows more opportunities to experiment and innovate

Modular Image Analysis Pipeline



Gonzales and Evans, BioEssays, 2019, 190004

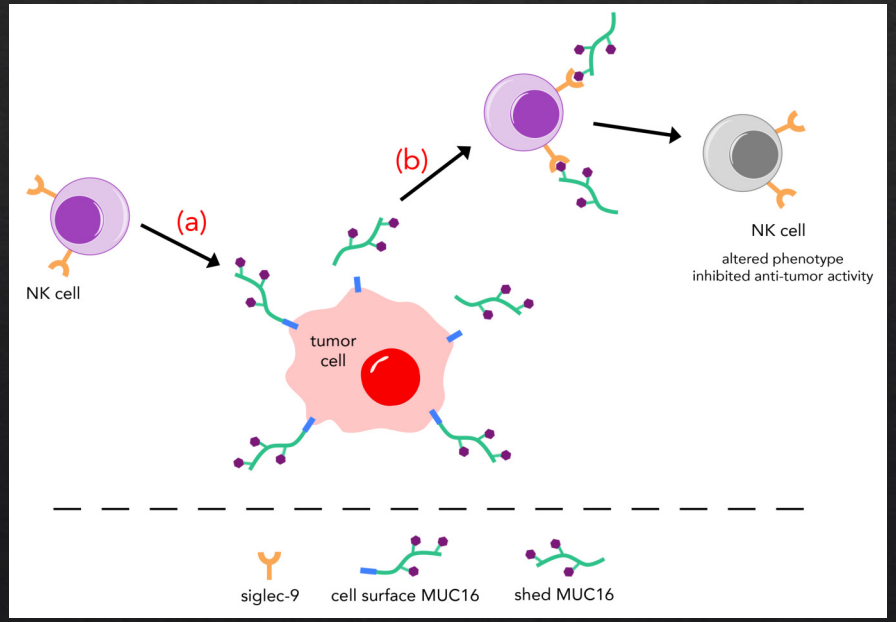
Modular Pipelines with Deep Learning

Example 1: Digitial Cytometry to Detect Low Levels of Leukocye-Antigen Binding

Example 2: Dermal Pharmacokinetics Measured *in situ* with Coherent Raman Imaging

MUC16/CA125 has an Immunomodulatory Role

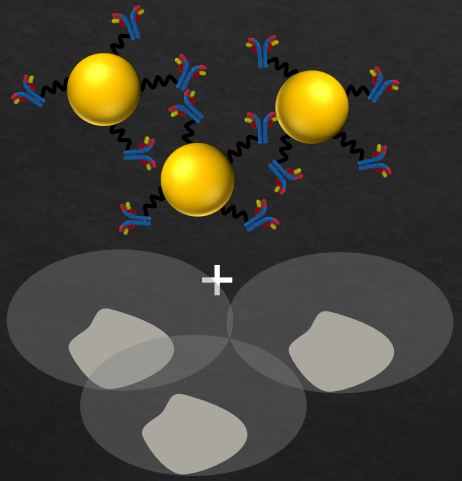
CA125-immune cell interactions suggest that the antigen binds to immune cells, blunting immune activity and potentially altering immune recognition of cancer



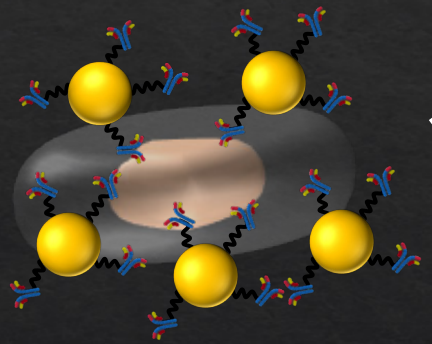
Key Challenge: only 10-100 estimated binding events *per cell*

Digital Cytometric PNP Immunoassay

Antibody conjugated
gold nanoparticles (PNPs)

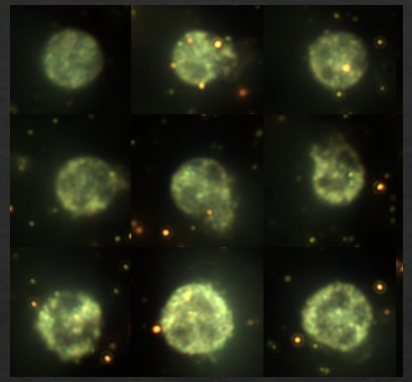


Peripheral blood
mononuclear cells (PBMCs)

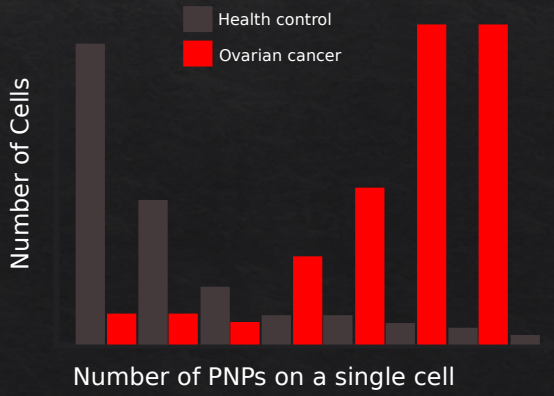


PNP-bound PBMCs

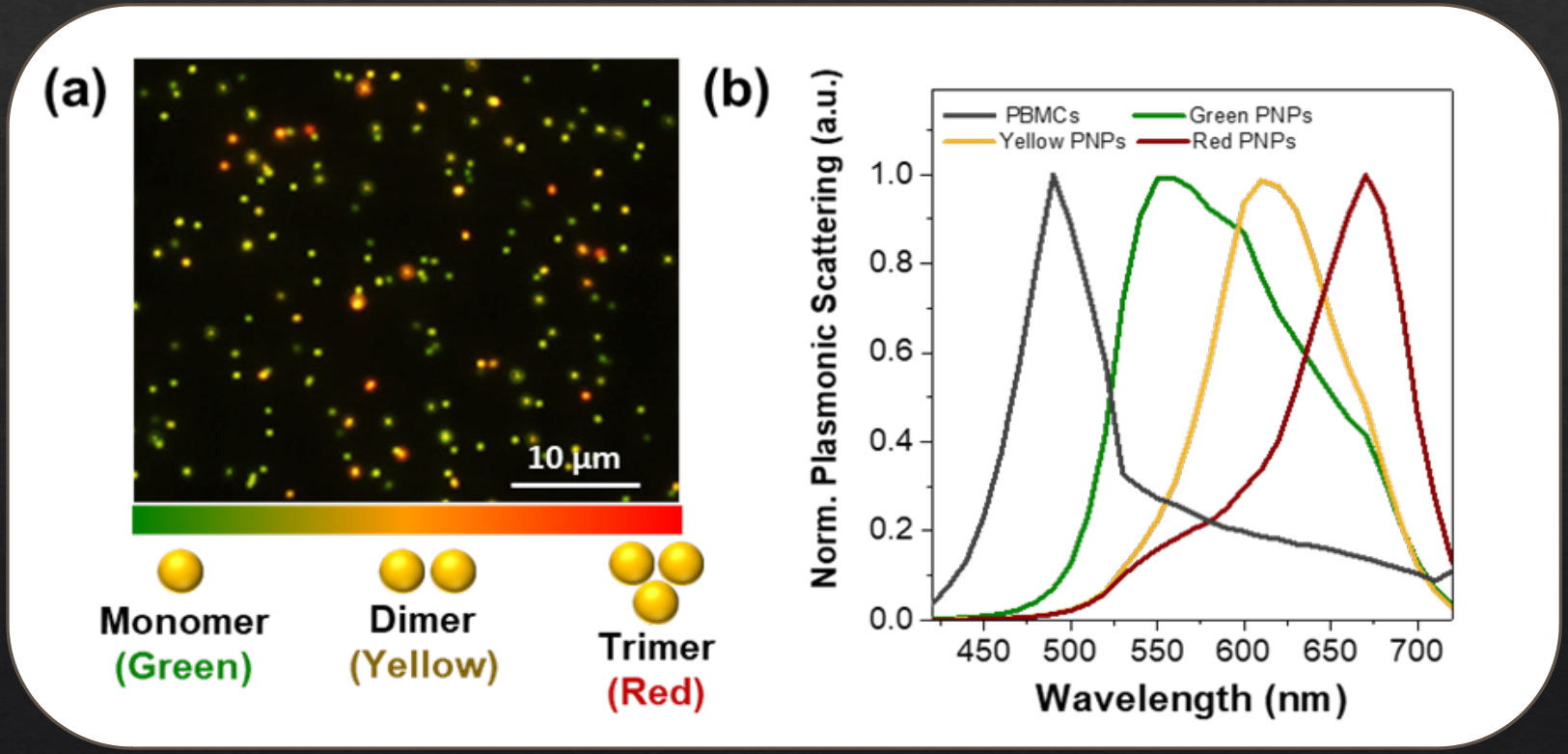
Dark-field Microscopic imaging



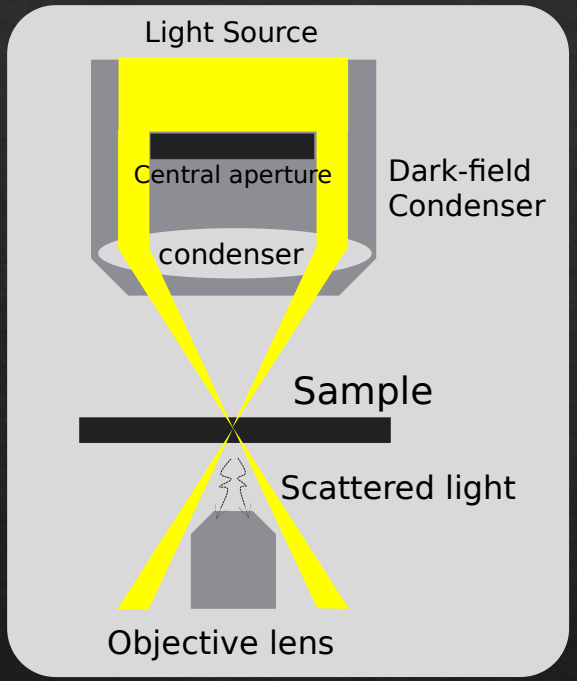
Digital counting

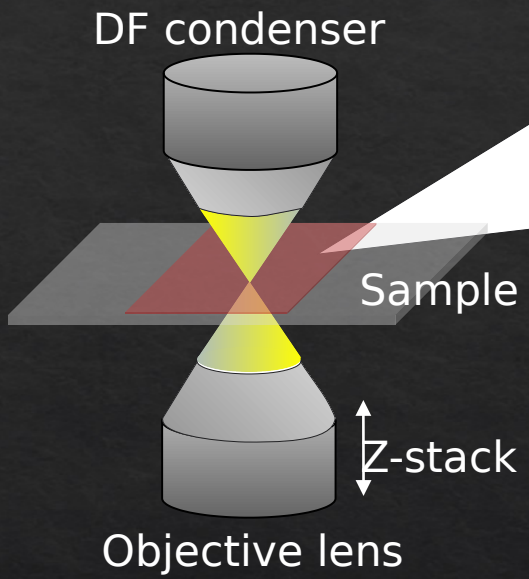


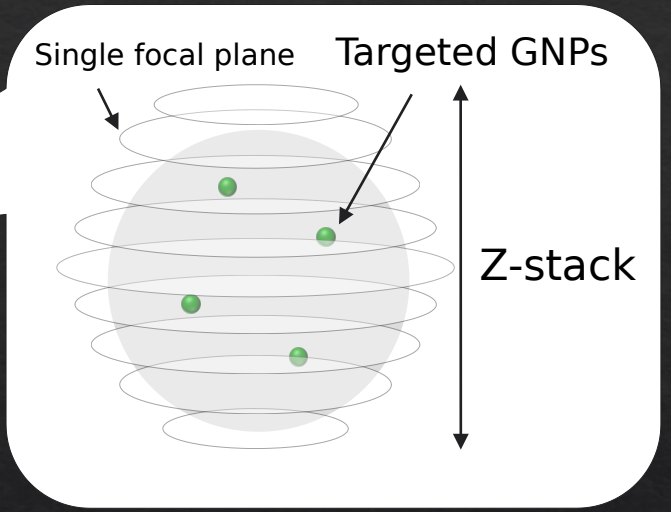
Nanoparticle Color Encodes Quantity



Selectively Imaging PNPs with Dark Field





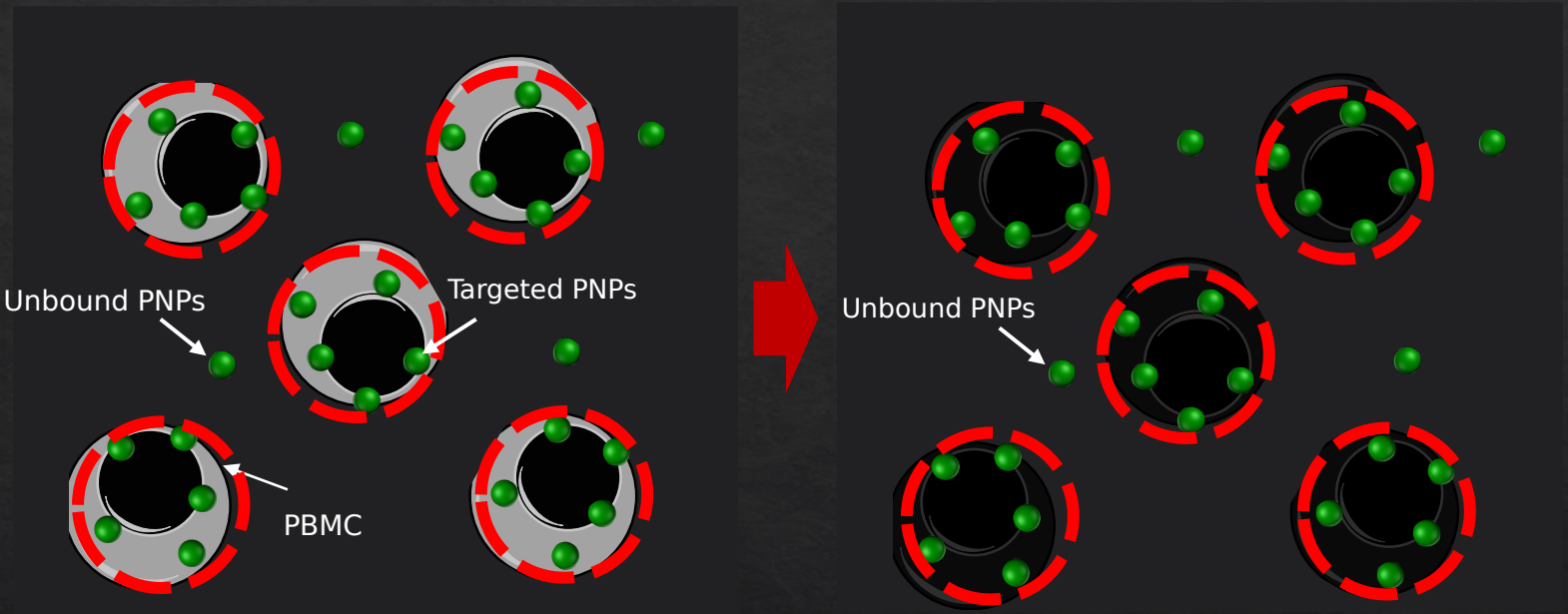


Dark-field microscopic 3D imaging

Challenge: 10's of TB of high resolution, 3D cell data

Solution: Fully Automated Imaging processing stack incorporating Deep Learning

Digital Cytometry



Step 1: Recognize Cell Boundaries
- Autoencoder

Step 2: Detect indiviual NPs
--> Laplacian of Gaussian
→ Unet

Digital Counting method is the cytometric equivalent to photon counting. Nonspecific binding should be Poissonian noise!

Building a Pipeline from Scratch

Considerations

1. Metadata, filename, and folder naming conventions
    - Early in the process developed a metadata format and consistent file naming convention to encode important information
    - Expandable, so that could store unanticipated values later
    - File naming system stored additional time, date, user, program, sample values
    - Folder system for separating datasets that supported automated data import
2. Data storage for medium-side laboratory scale analysis
    - Coded human samples as part of IRB study required (at the time) in-house storage
    - QNAP NAS ~50 TB at start with expansion potential
    - Purchased higher-end Intel i7 version so that NAS could compute
3. Data upload and backup
    - Created syncronized folders on acquisition PC that uploaded after each experiment
    - NAS continuously synchronized with off-site cloud storage

Building a Pipeline from Scratch

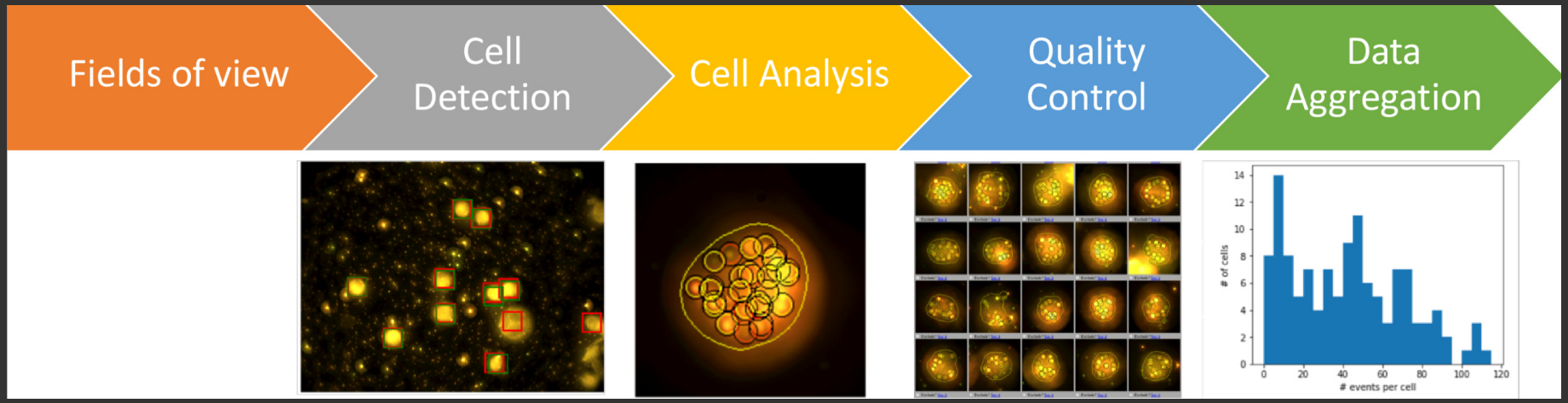
Key Implementation Steps

1. Reading in and converting each file
2. Identifying each cell in a field of view
3. Identifying the boundary of each cell
4. Classifying the type of each cell
5. Finding and counting the number of bound nanoparticles per cell, per cell type
6. Generating quality control output for each analyzed image
7. Creating a CSV file output that summarizes the findings
8. Moving to the next image and starting again

Blue steps are those that began manual and transitioned to machine learning

Manual labeling key in generating the “ground truth” needed for deep learning training

Modular Pipeline: Step by Step



Phase I	“Cell Shooter”	“PNP Picker”	Manual Inspection	Jupyterlab
Phase II	“Cell Shooter”	Unet	Djanjo	Jupyterlab
Final	Autoencoder	Unet	Djanjo	Automated

Details

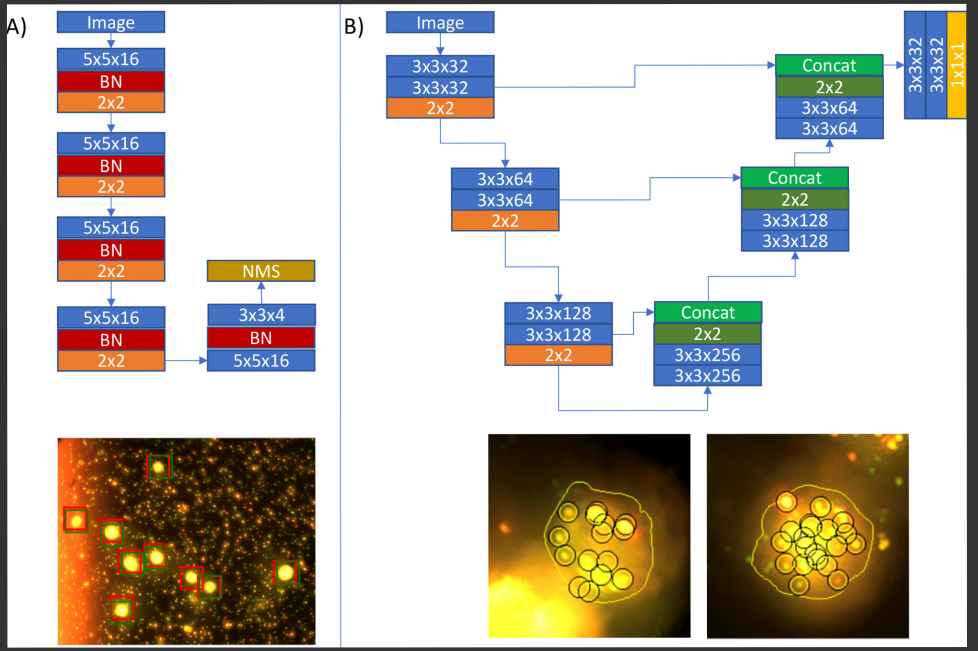
Pipeline coded in Python 2.7 with numpy, imageio, tifffile, keras, and tensorflow libraries

Scripting was carried out using bashscript, with timing controlled by the cron

The “cell shooter” and “PNP picker” applications were written in python and tk/tcl; the pipeline would run until it exhausted each step.

Deep learning-based image analysis carried by:

- An autoencoder for cell and cell boundaries
- A 3-payer 3D Unet for detecting PNPs on cells



Translating Code to Containers

Each major module was encapsulated in a Docker container:

1. File Parsing, Cell Location, Cell Boundary Detection
2. PNP Counting, Cell Lineage, QC compilation
3. Djanjo server for QC inspection

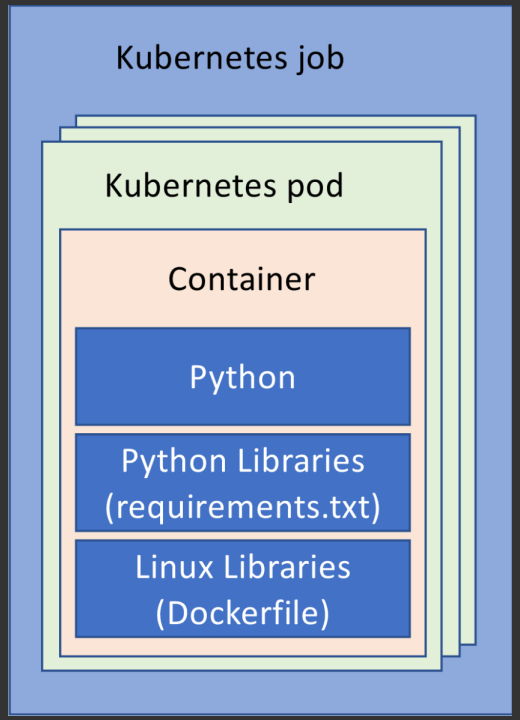
Docker containers were run using the “Container Station” application on the QNAP NAS

Eventually *all* calculations ran on the NAS, including model inference

Entire platform can be easily pushed to a cloud provider (GCP, Azure)

Speeding up: K8's

We experimented with using Kubernetes container orchestration to accelerate data analysis. Though we first used GCP, we eventually built out our own K8 cluster (1 master and 3 nodes)



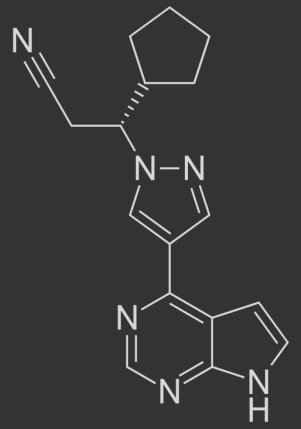
Each pod contained all the libraries needed for processing a given image, with a pod dispatched for each new image

5X speed increase over serial containers

Example 2: Skin Inflammation

Collection of skin diseases that cause itch, redness and rash, also known as **eczema**.

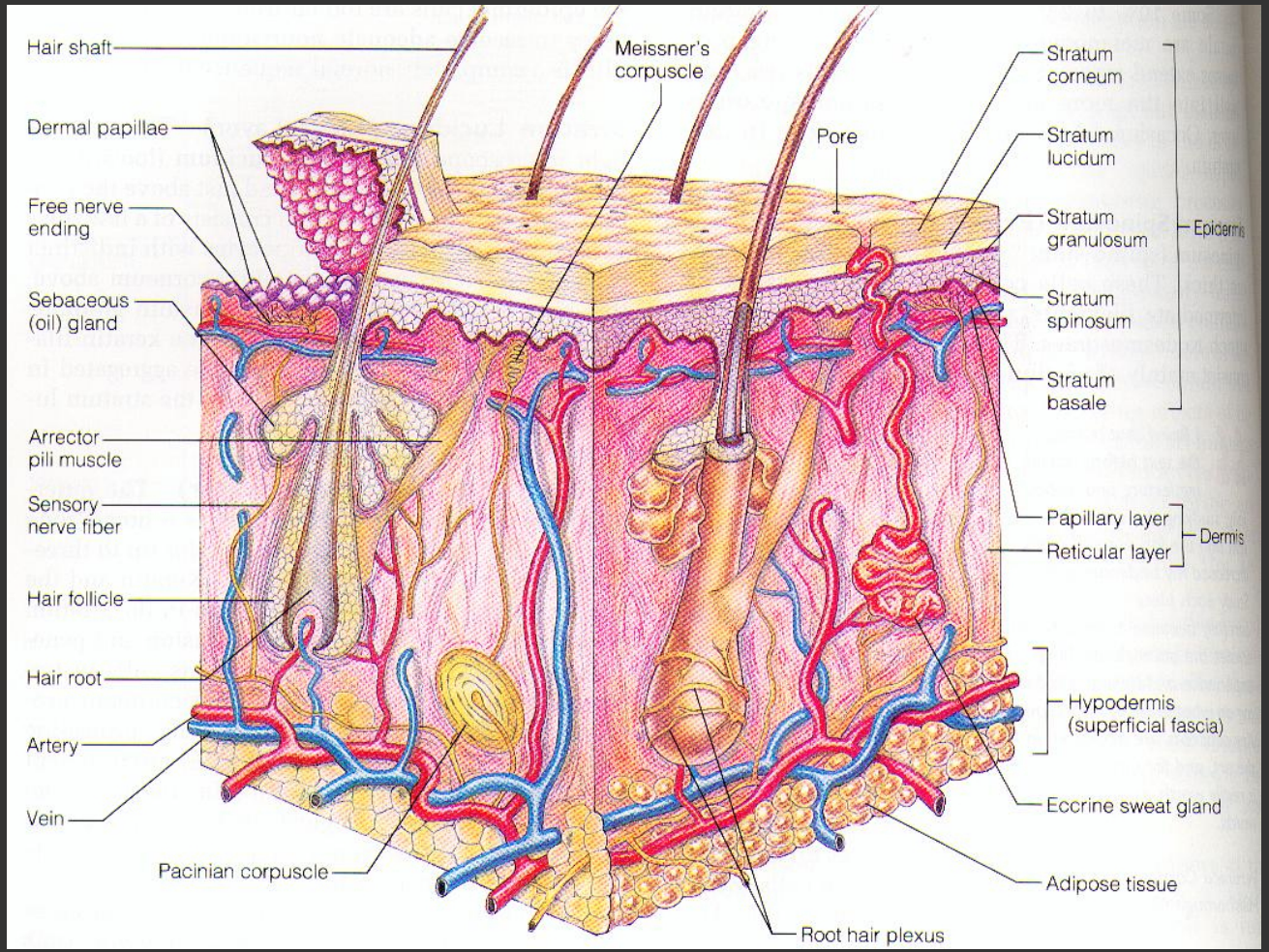
Affects >245 Million people worldwide, **atopic dermatitis** being the most common.
Atopic dermatitis affects up to 30% of people in the US.



New treatment approach is to target the **JAK** pathways to control inflammation

Ruxolitinib

Skin Structure



Coherent Raman Scattering (CRS) Microscopy

Imaging based on intrinsic vibrational contrast

Two Colors: ω_p “Pump”

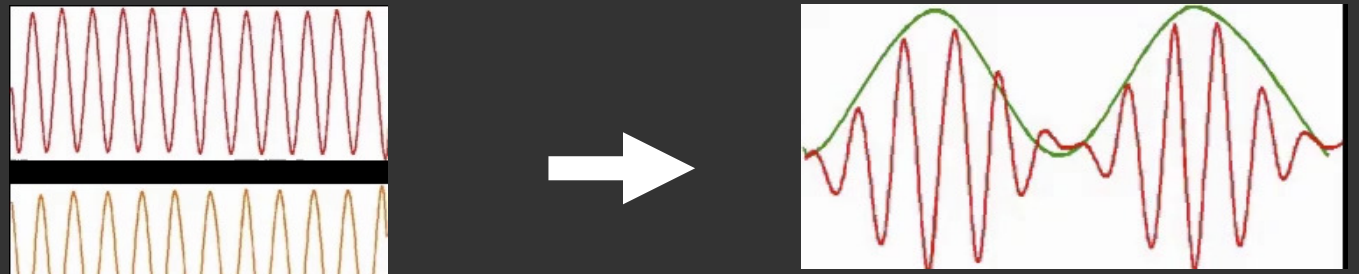
ω_s “Stokes”

Coherent Raman Scattering (CRS) Microscopy

Imaging based on intrinsic vibrational contrast

Two Colors: ω_p "Pump"

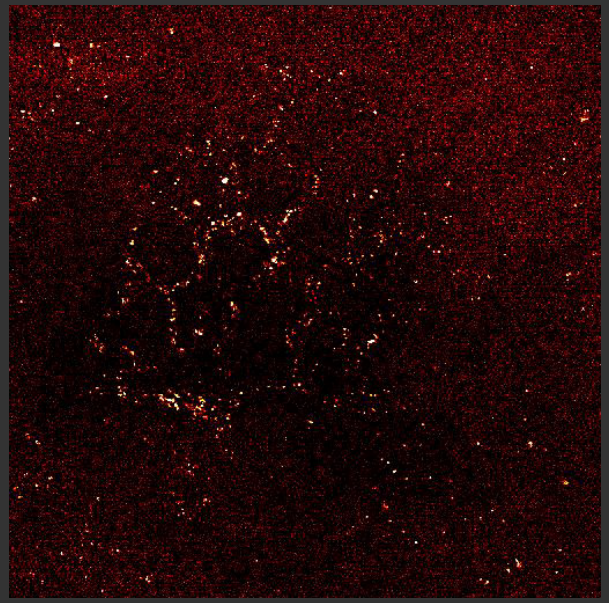
ω_s "Stokes"

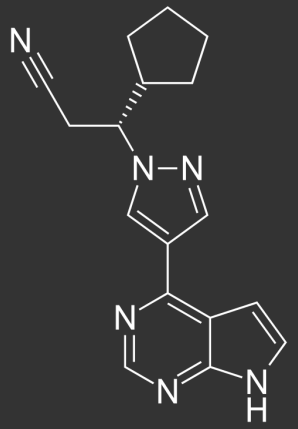


$$\omega_p - \omega_s = \Omega_{vib}$$

Drug Uptake Dynamics in the Stratum Corneum

SRS Microscopy
Nitrile Stretch: 2250 cm-1
100% resonant signal
120 min



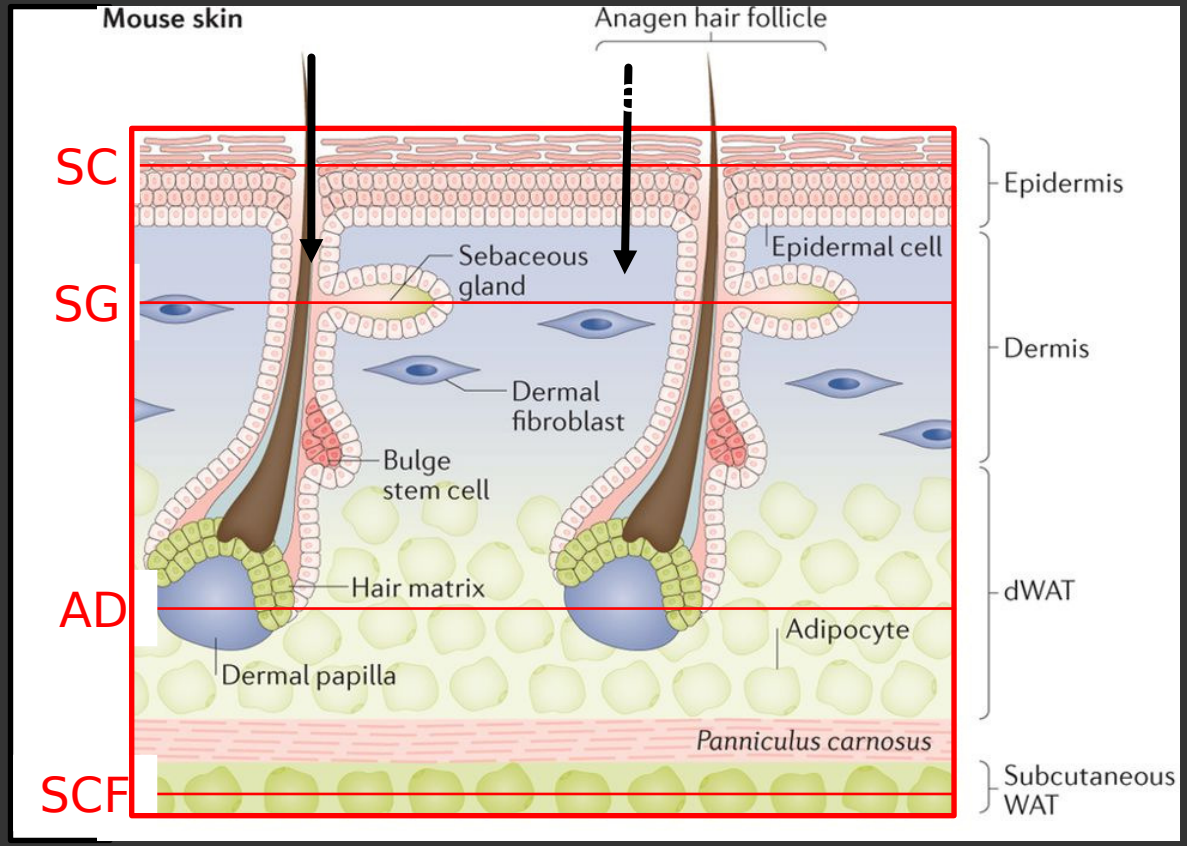


Direct visualization of Rux depositing
on the surface of skin without
background signal

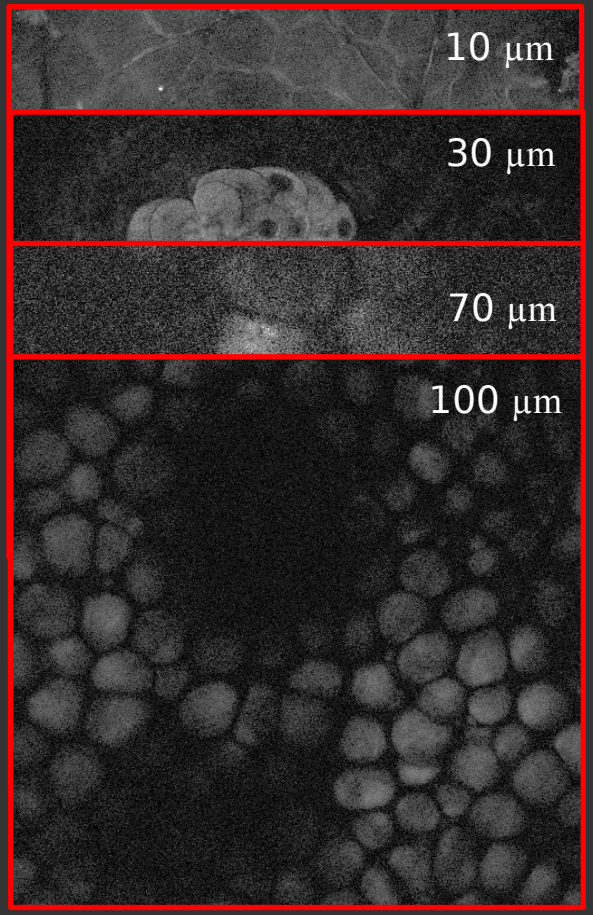
Image-based PK Challenges

- Understanding skin PK requires that the time-evolving drug images be analyzed in the context of skin biology and structure
- **BUT**: Skin is highly heterogeneous and complex, meaning that standard feature identification algorithms have unacceptable failure rates.
- Poor analytical methods can introduce bias, which would skew results and deliver incorrect PK information

Multidepth Drug Tracking



Nature Reviews Endocrinology **volume14**, pages163–173 (2018)

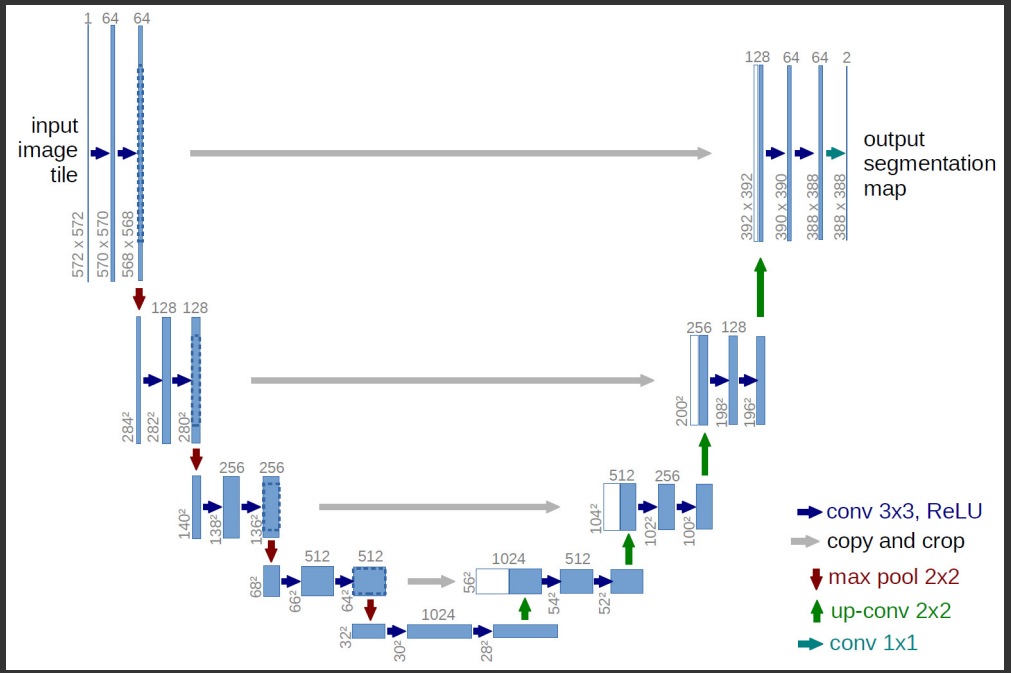


Quantifying PK with Deep Learning

Detecting skin compartments can be challenging: irregular size, shape, connectivity, etc

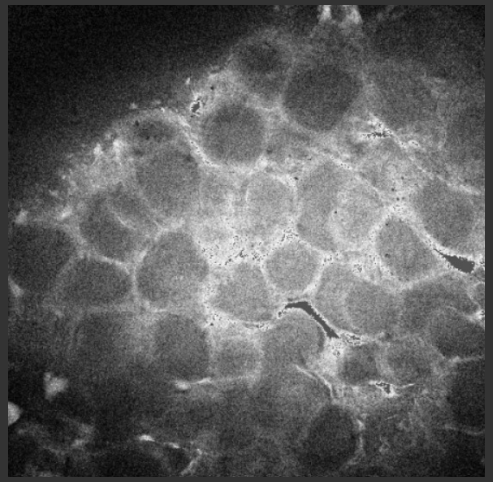
Machine learning provides a robust method for consistent feature detection

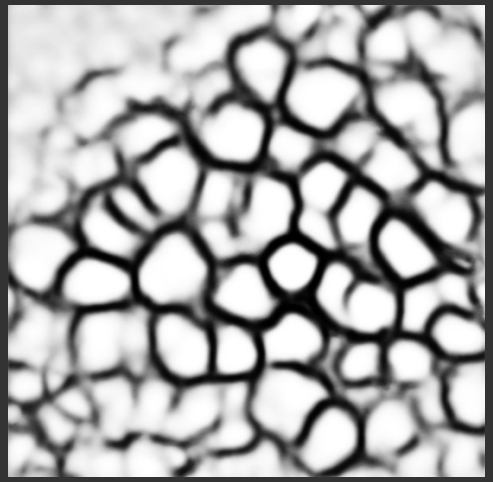
We use a Unet full-convoluted Convoluted Neural Network (CNN) architecture that won the ISBI challenge

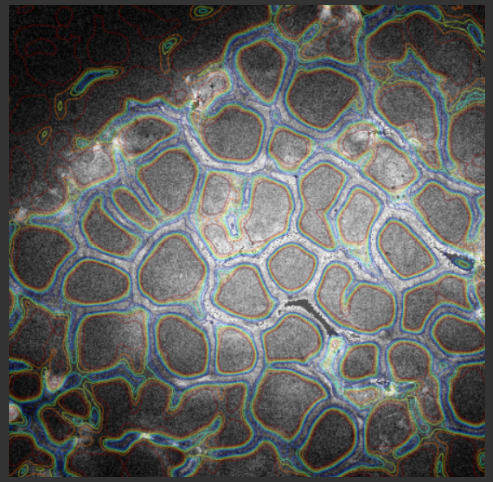


Quantifying PK with Deep Learning

- Annotations to each image were provided by image experts
- Annotations were hand-drawn in either Photoshop of GIMP,
- Trained individual Unets for 2D images, even when using 3D data
  - This minimized the memory required and simplified training
  - Avoided issues of uneven sampling in z-axis
  - Not all 3D datasets contain features of interest. Required “balancing the set” to make sure feature-containing images are not in the minority







New Fully Automated Pipeline

- Coded workflow in Python with OpenCV and Tensorflow
- Runs nightly, finds new data, carries out full automated analysis
  - Finds new, unanalyzed data
  - Reads in data into database
  - Runs conversion and deep learning analysis
  - Calculates PK profiles on in-memory data
  - Runs Quality Control each step
  - Saves all output to standard numpy files for later parsing

Interactive Data Analysis

- Built a Juypterlab/Python based interactive environment
- Parses directories, loads in numpy saved files
- Builds an in-memory dataframe using Pandas that can be queried to find specific experiment details (e.g. drug, formulation, depth, experiment type, etc)
- Queries can be used to parse specific data types to extract images, composite data, QC outputs
- PK analysis then carried out on queried data output using R-based noncompartmental analysis

Details and Containers

Hardware: System76 Leopard WD Xeon-2145 equipped with (1) Quadro P6000

Pipeline image analysis in Python 3 with numpy, javabridge, python-bioformats, keras, and tensorflow libraries

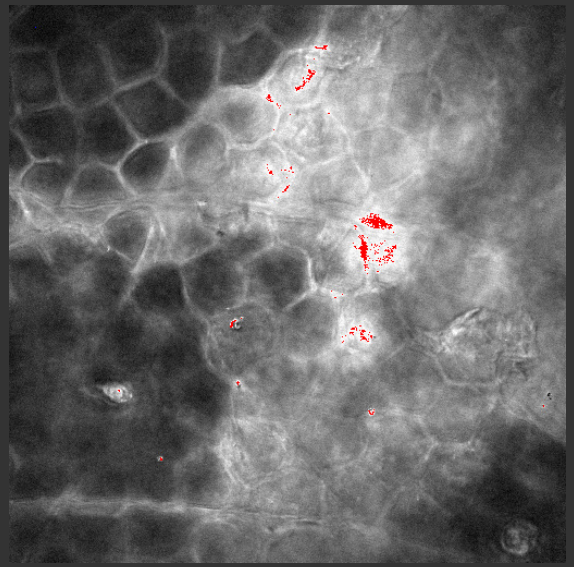
PK analysis carried out using R using rpy2; R needed for specific pharmacokinetic analysis packages

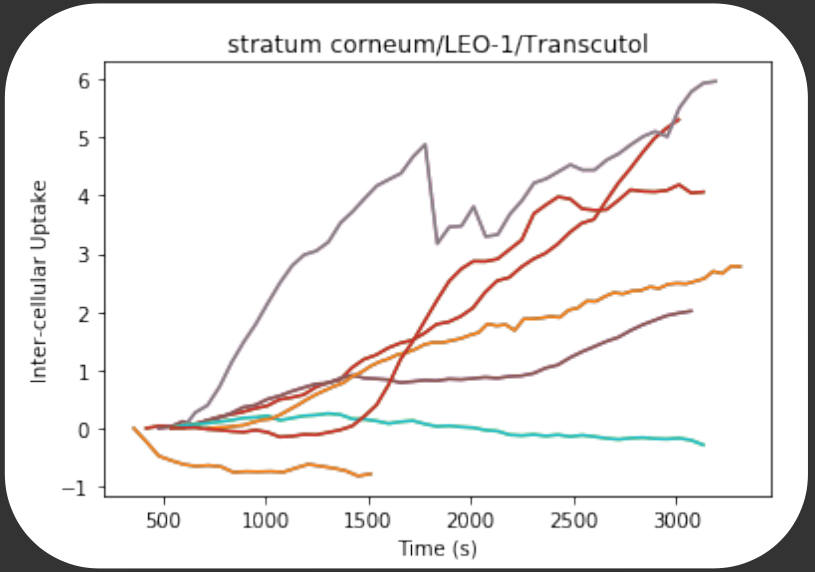
Scripting was carried out using bashscript, with timing controlled by the systemd timers

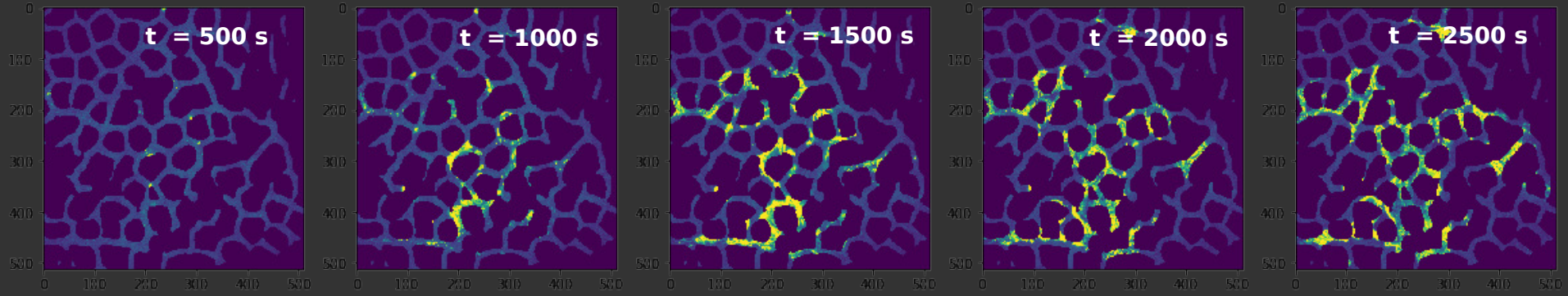
Jupyterlab server chosen over Jupyter Notebooks as it provided a better interactive coding environment

Entire platform encapsulated with a Docker container (Ubuntu 18.04) for easy deployment

Heterogenous intercellular uptake of ruxolitinib in transcutol through the SC

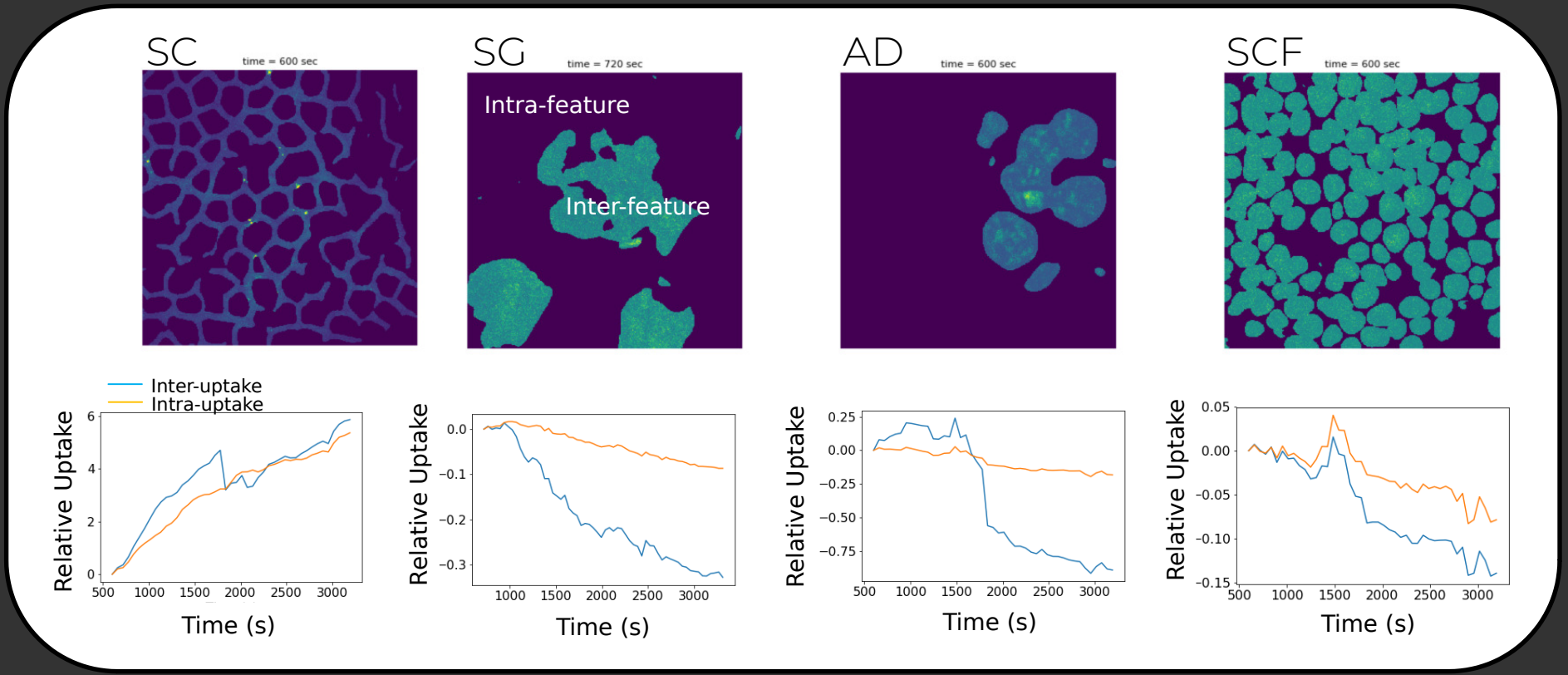






Following Drug Diffusion Through Skin Compartments

4 different layers of mouse skin. Drug: Ruxolitinib, Vehicle: Transcutol

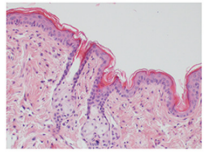
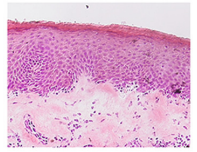


PK profiles show that primary route of entry is through the hair follicle and sebaceous gland when ruxolitinib is administered in transcutol

Can observe the downward “wave” of the drug at the t_{max} peaks in the AD and SCF

Translation to the Clinic

- Model systems do not exactly mimic native skin
  - *Ex vivo* human skin testing is established in field
    - Cadaver or biopsy required
    - Difficult to study disease state
  - Non-invasive *in vivo* human skin imaging <u>at disease sites, over time</u> will give unprecedented information about PK/PD for cutaneous drug & formulation development

	Mouse	Human
Histology		
Epidermal layers	2 - 5	16 - 18
Skin renewal	8 - 10 d	28 d
Hair follicle	High number & wider diversity	Significantly less than rodents
Cutaneous muscle layer	Panniculus carnosus	No
sweat glands	Eccrine sweat glands (exclusively in the pads)	Apocrine & eccrine
Melanocytes	Follicular location	Interfollicular location
T-cell population in the skin	Dendritic epidermal T-cells (DETC)	α/ß T-cells
Antimicrobial peptides	Tissue specific different from humans	Yes

Adapted from Löwa, et al., Alternatives to animal testing in basic and preclinical research of atopic dermatitis. Exp Dermatol. 2018; 27: 476– 483. https://doi.org/10.1111/exd.13498

Next Steps

- Generalizing the pipeline
  - Rewritten pipeline in Python 3
  - Working to create distinct plug-in structure where modules can be written/loaded by team members
- Scaling the pipeline
  - Data from clinical instruments will be 10X greater than current
  - Looking a cloud platforms (Azure, AWS, GCP) for scaling
- Container distribution for reproducible computing
  - So far have used Docker, but now looking at Singularity

Evans.conor@mgh.harvard.edu