PSF Modeling with Deep Learning for Next-Generation Survey Pipelines

Jonathan Carney, Hank Corbett, William Marshall, Nicholas Law, Shannon Fitton, Nathan Galliher, Ramses Gonzalez, Lawrence Machia, Thomas Procter, Alan Vasquez Soto

Argus Array

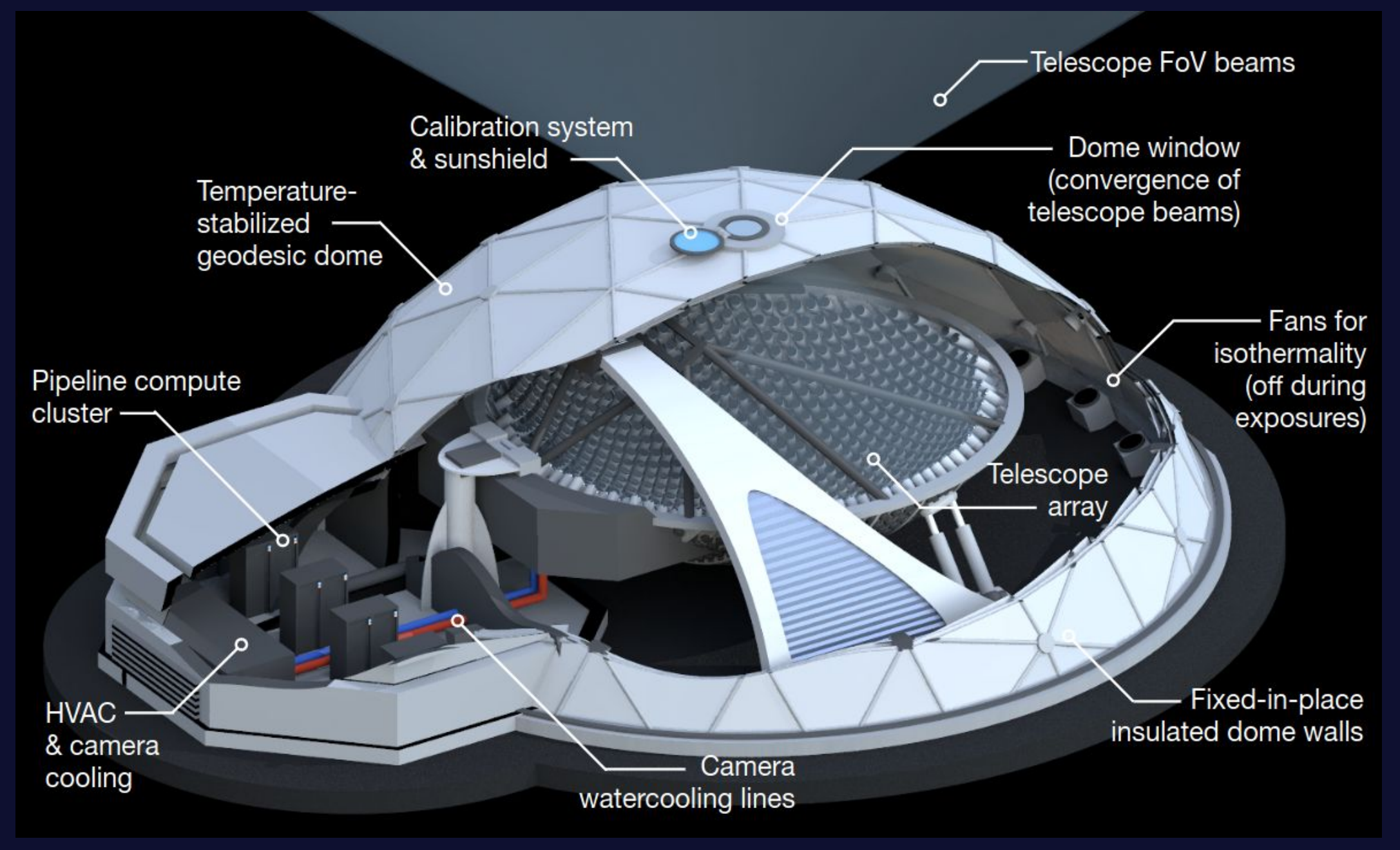
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

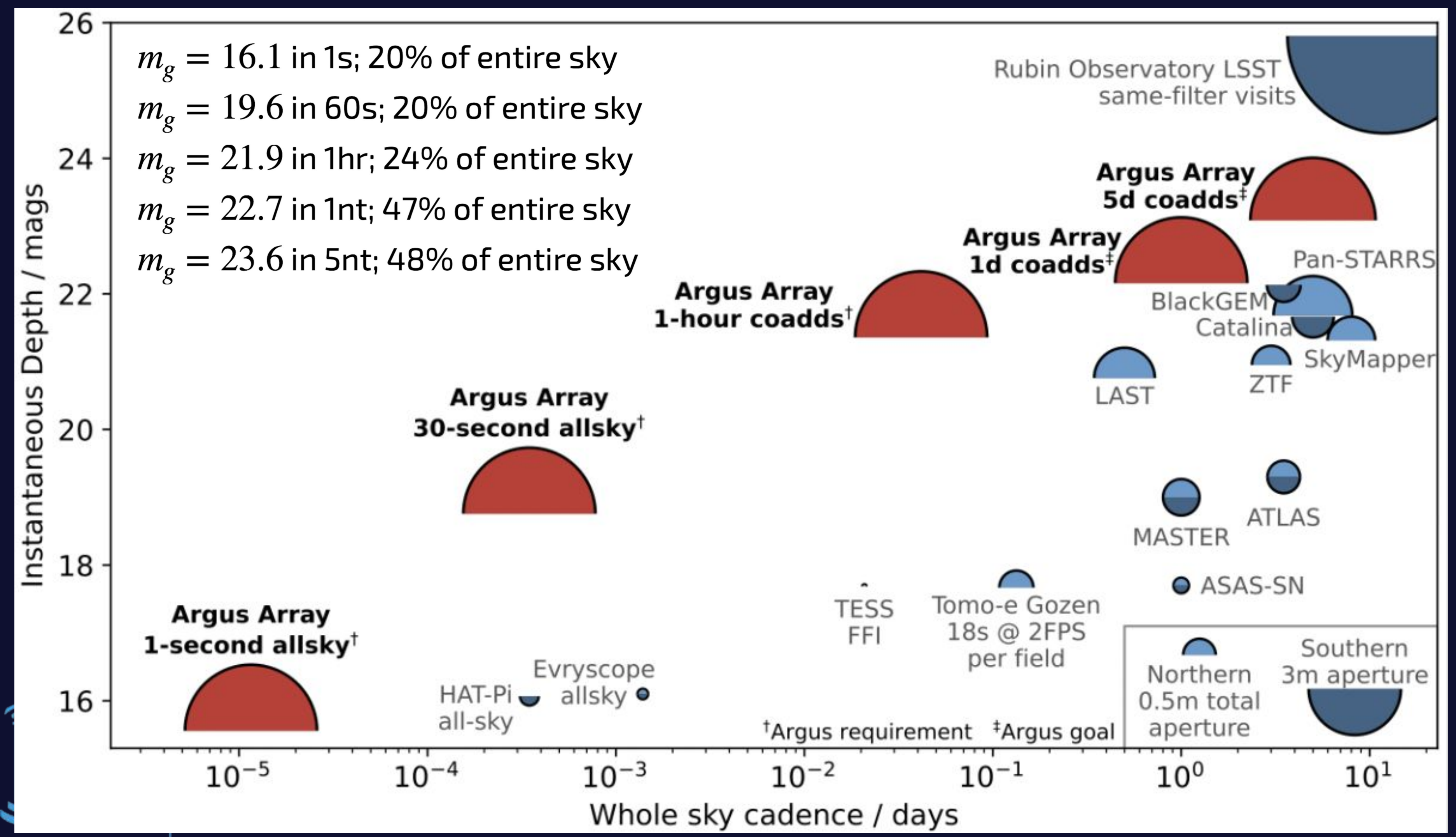
1. Training the Argus Pipeline with Simulated Transients

Machine learning systems for transient discovery and candidate vetting for the Argus Array are currently under development (Corbett+2023, 2022). We train these models by using large simulated datasets to build representations that can then be fine-tuned over a small, hand-labelled dataset of on-sky detections. Producing this training set from real transients in archival data at scale is impractical due to the work-hours involved, so instead we choose to augment small hand-labeled datasets with large, cheaply simulated counterparts. We generate training data by injecting simulated transient sources into archival data; however, point spread functions (PSFs) in astronomical images are variable due to changing atmospheric and instrument conditions. Existing analytic methods for generating high-fidelity instrumental PSFs are not scalable to low-latency data pipelining of O(10^9) images. Machine learning models provide an alternative fast enough to be of practical use for real-time data augmentation.

2. Argus Array Overview

The Argus Array (Law+22) is a forthcoming synoptic all-sky survey that will record a 55-gigapixel movie of its 8000 deg^2 field of view at cadences as fast as 1 second. Argus data will be shared with the entire astronomical community through public transient alerts, images, and millions of million epoch stellar light curves.


Telescope FoV beams
Calibration system & sunshield
Dome window (convergence of telescope beams)
Temperature-stabilized geodesic dome
Fans for isothermality (off during exposures)
Pipeline compute cluster
Telescope array
HVAC & camera cooling
Fixed-in-place insulated dome walls
Camera watercooling lines



3. Machine Learning Model

The transient injection model is based on a modification of the UNet 3+ architecture (Huang+2020), with the addition of a fully connected layer in order to connect disparate spatial information. We used softmax activation in our output layer in order to normalize the output image in a manner that can be used probability distribution for our injection method. During injection small quanta of flux are stochastically added at locations drawn from this distribution. We train our model using archival data from the Evryscopes (Ratzloff+2019, Corbett+2023) and use the median PSF of the input image as a training target.

Key:

Modified UNet 3+

UNet 3+ without full upsampling. The network consists of a networks of nested encoder-decoder pathways connected by dense 2D convolutional blocks.

Downsampling $\xrightarrow{}$

more chanels, fewer features

Upsampling $\xrightarrow{}$

fewer channels, more features

Skip connections $\xrightarrow{}$

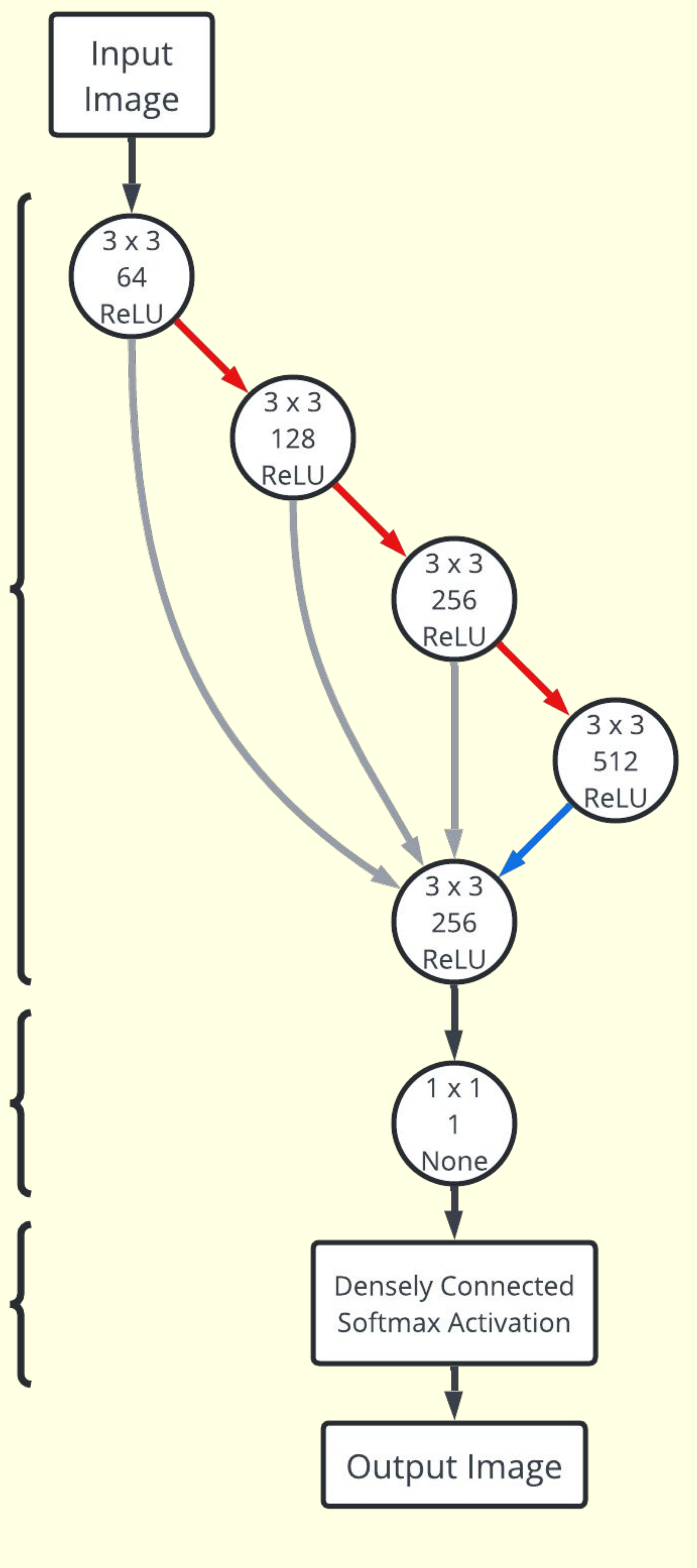
propogate early, high frequency information deeper into the network

1 x 1 Convolution

Reduces number of features with channel wise convolution

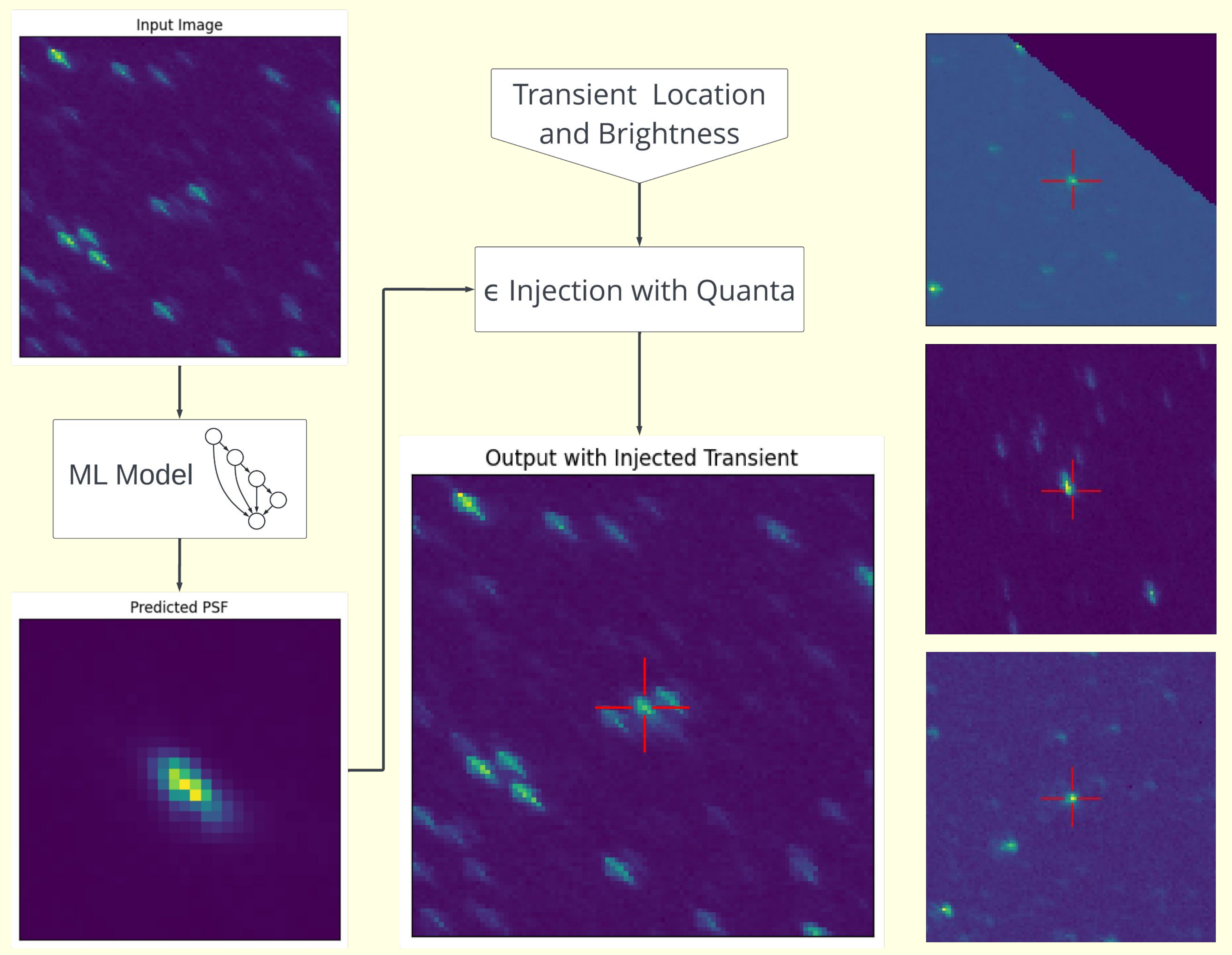
Added Fully Connected Layer

Connects spatially disparate features, outputs PSF as a probability distribution



4. Optically Limited Example from Evryscope South

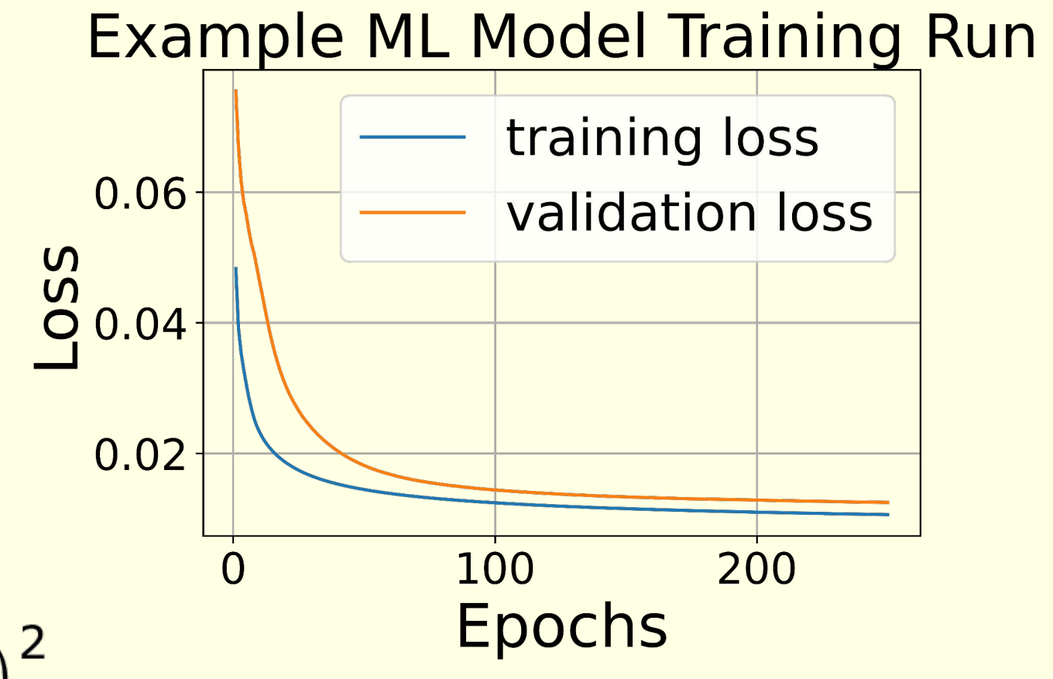
Before injection, the predicted PSF is masked and renormalized so that only pixels greater than an adjustable factor ϵ of the maximum value are included. This prevents a "postage stamp" effect around injected transients. We utilize Evryscope data as it provides a multi-million sample training set across a diversity of pointings, telescopes, and observing conditions, challenging the model beyond the planned Argus distribution. Below is an overview of transient injection and 4 example injections into Evryscope South images.


Input Image
Transient Location and Brightness
ϵ Injection with Quanta
ML Model
Output with Injected Transient
Predicted PSF

5. Hyperparameters and Training

Inspired by the Combo loss from Taghanaki+2018, we used a combination loss of the Kullback-Leibler divergence (Kullback+1951) and a pixel value weighted square difference loss. By tuning the value of β, we can adjust how much each loss component is emphasized.

Example ML Model Training Run



$$Loss = \underbrace{\beta \sum y_{true} log\left(\frac{y_{true}}{y_{pred}}\right)}_{Kullbach\ Leibler\ Divergence} + \underbrace{(1-\beta)\sum y_{true}\left(y_{true}-y_{pred}\right)^2}_{Pixel\ Weighted\ Square\ Difference}$$

We train with the AdamW optimizer (Loshchilov+2019) and a constant learning rate of 10^{-6}. The injections shown in the figures above were produced after training on 322k samples for 250 epochs, with β = 0.75 and ϵ = 0.005.