

Gaussian Process Classification for Galaxy Blend Identification

James Buchanan, Michael Schneider, Robert Armstrong,
Amanda Muyskens, Benjamin Priest, Ryan Dana

Jan 24, 2022

LLNL-PRES-830243



LLNL-PRES-XXXXXX

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Status update

- Last talk in BL WG: April 26

- Since then:

Accepted by ApJ on Nov 1, published ten days ago

[doi:10.3847/1538-4357/ac35ca](https://doi.org/10.3847/1538-4357/ac35ca)

- Today: Review of the finished paper + DESC project proposal

Detection

- Common deblenders – e.g. SCARLET – require an initial detection step on coadds: estimate of how many objects are in a blend, and where those objects are
- Currently, default detection method is to construct footprints and find peaks in those footprints

Detection

- Common deblenders – SCARLET – require an initial detection step on coadds: estimate of how many objects are in a blend, and where those objects are
- Currently, default detection method is to construct footprints and find peaks in those footprints
- Can we do better?
- Looks like the answer is yes, and with a finite amount of time and effort

Detection

- Common deblenders – SCARLET – require an initial detection step on coadds: estimate of how many objects are in a blend, and where those objects are
- Currently, default detection method is to construct footprints and find peaks in those footprints
- Can we do better?
- Looks like the answer is yes, and with a finite amount of time and effort
- Can we do more?
- Answer: Probabilistic detections

Approach

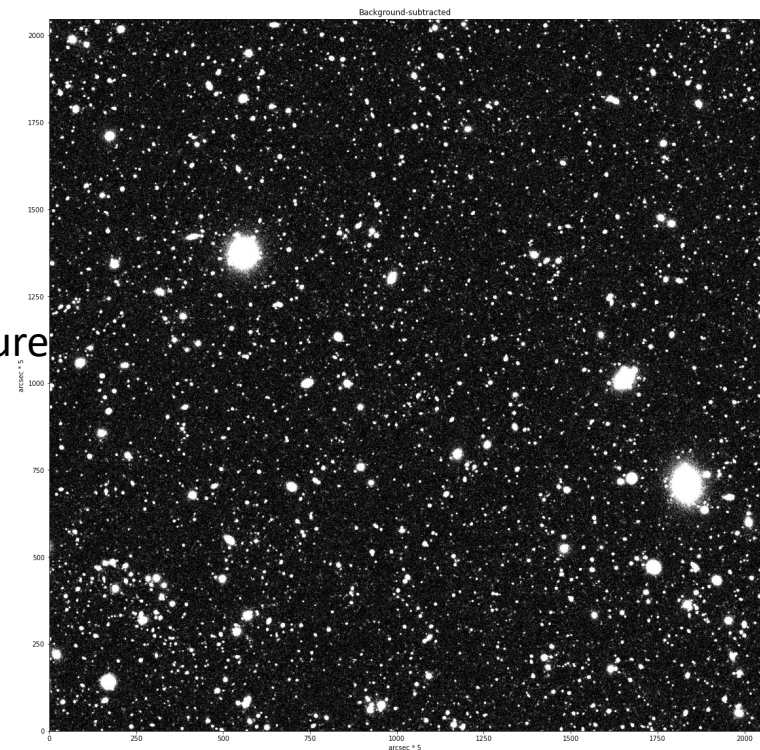
- Get a realistic, astronomically-relevant distribution of blends
- Make footprints
- Test different detection methods on footprints

Galaxy Distribution

- CosmoDC2 galaxy catalog
- Realistic number density of galaxies up through r-band mag 28, and many more up to mag 29
- Used for DESC DC2 campaign

Galaxy scene simulation

- **Sersic model bulge and disk for each galaxy**
 - Sérsic parameters, ellipticity components, relative component fluxes from **cosmoDC2 catalog**; overall flux in each band and lensed RA,Dec from DESC **DC2 truth catalog**
- **Weak lensing shear and magnification**
 - Gamma components and convergence from cosmoDC2 catalog
- **Kolmogorov PSF**
 - FWHM = 0.7 (+/- 10% per exposure)
- **Random sub-pixel-scale scene offset ('dither') per exposure**
- **Photon shooting**
- **Silicon sensor**
 - 'lsst_itl_32' in galsim
- **Sky background**
 - Dark sky magnitudes from smtn-002.lsst.io
 - +/- 5% mean flux per exposure
 - Poisson noise in each pixel
- **100 separate exposures simulated, then added together**



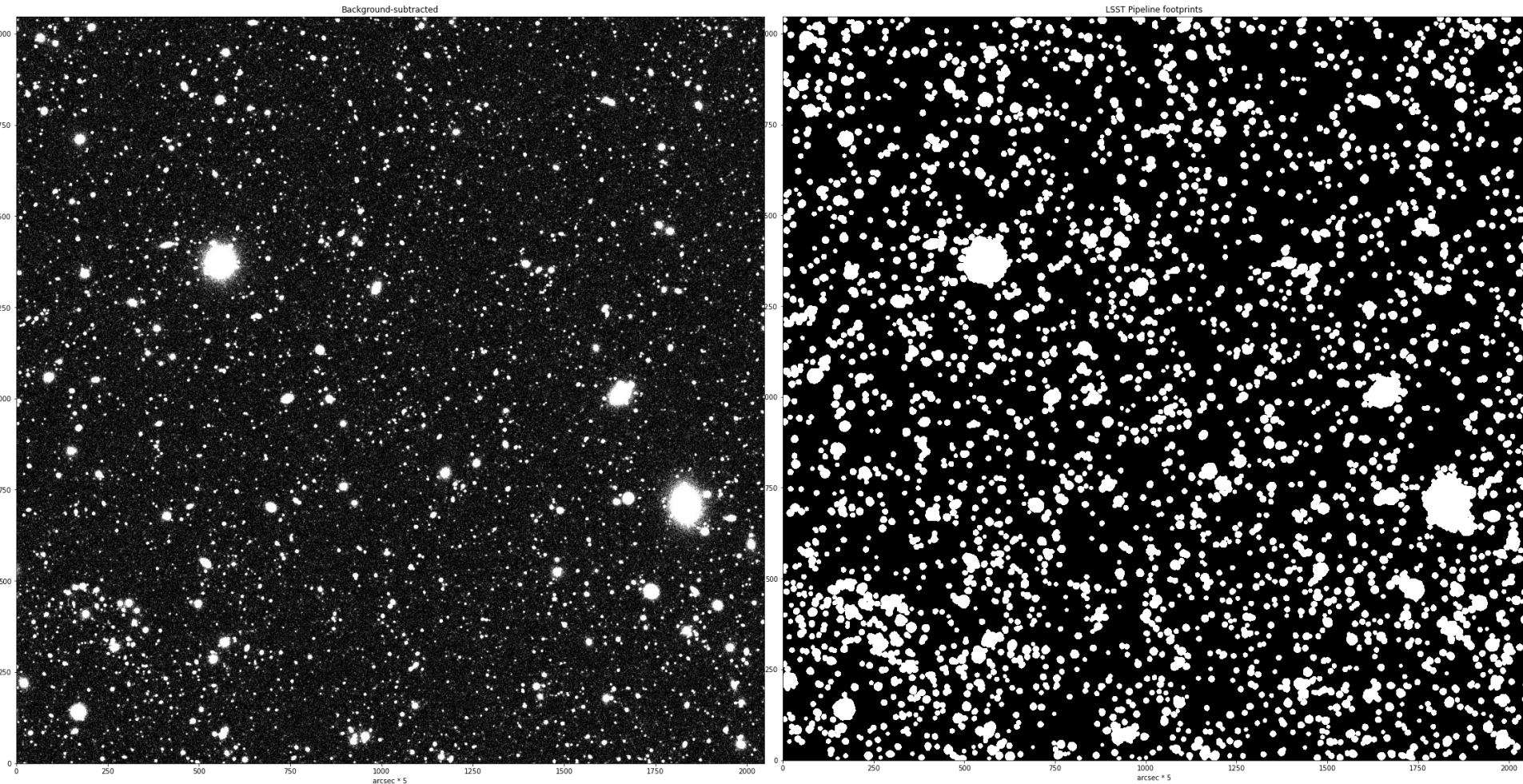
i-band, 2048² pixels
(409.6² arcsec)

Footprint Construction

- Estimate pixel noise level via clipped variance of coadd pixels
 - Correlated noise correction
- Subtract estimated sky background
- Convolve with Gaussian approximation of PSF
- Threshold each pixel at point-source $S/N > 5$ to get initial footprints
- Expand these initial footprints by $2.4 * \text{PSF width}$
- Merge any expanded footprints that overlap

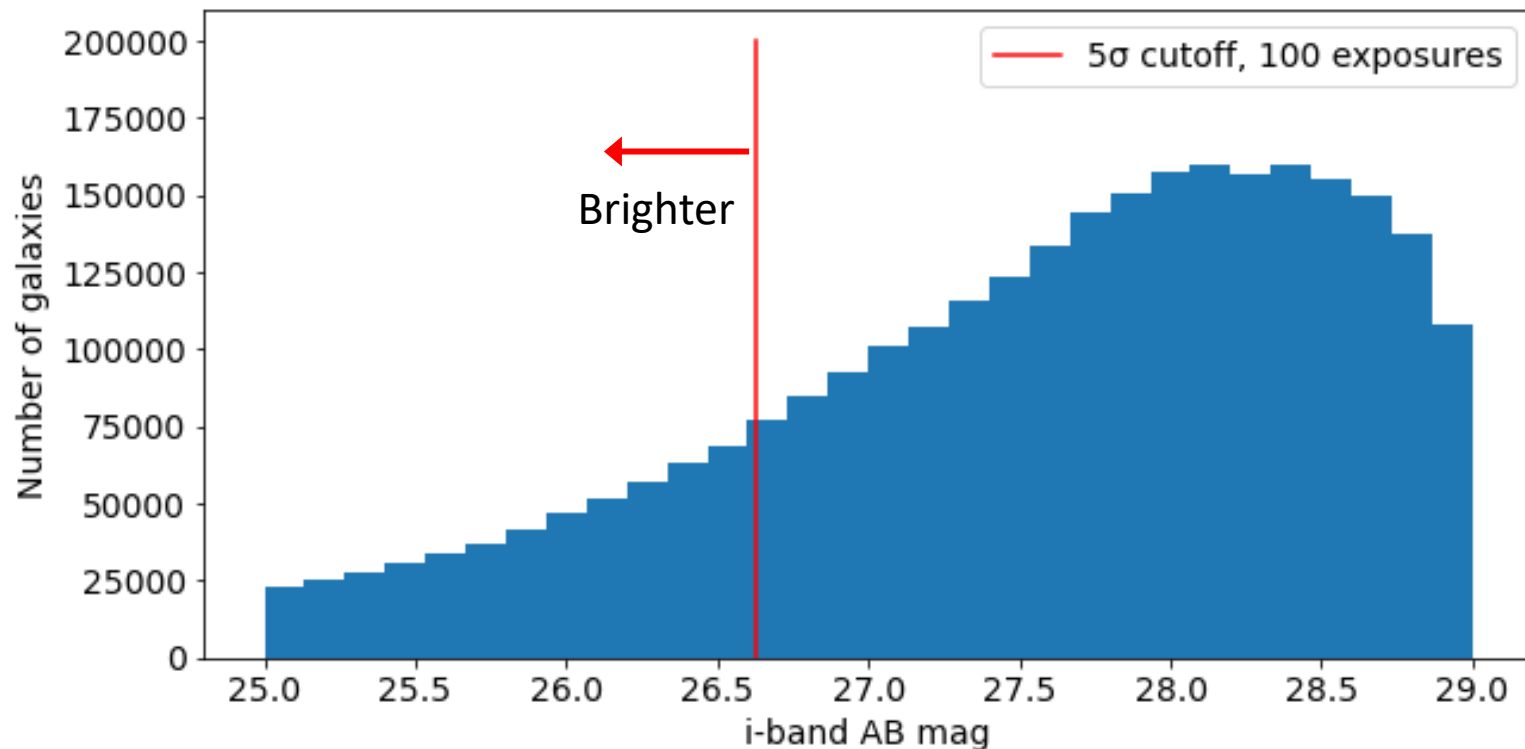
Simulated scene

LSST Pipeline footprints



Dataset

- Define an i-band footprint as **blended** if it contains the center of > 1 galaxy with “significant” (5-sigma) i-band flux



Dataset

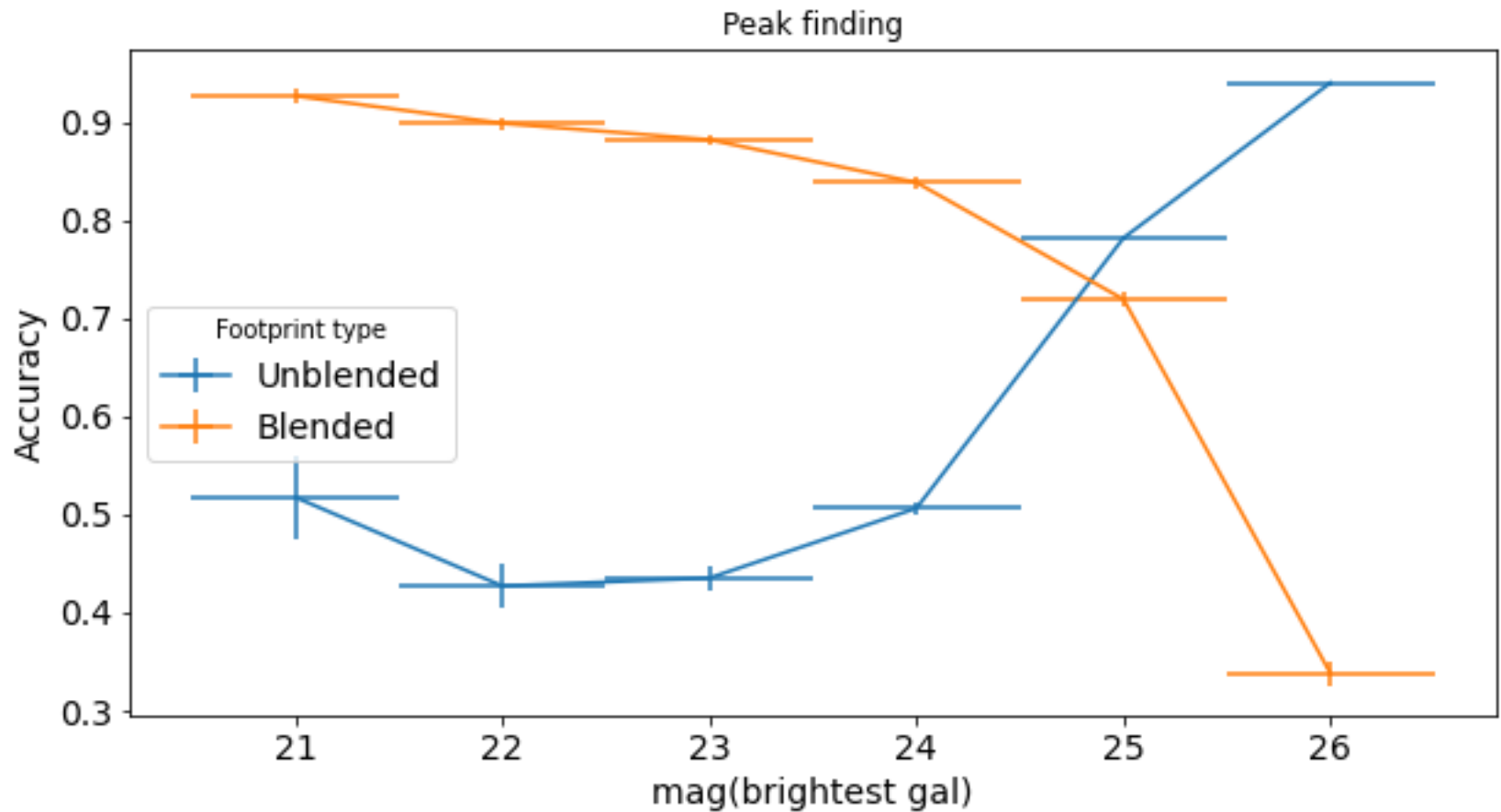
- **Define an i-band footprint as blended** if it contains the center of > 1 galaxy with “significant” (5-sigma) i-band flux

Across 20 total scenes:

- 134,493 total galaxies with i-band flux ≥ 5 sigma
- 66.01% of these galaxies are contained in i-band footprints
- 0.25% of footprints contain no 5-sigma galaxies
 - Ignoring these here
- 61.74% of footprints are blended
- 38.01% of footprints are unblended

Peak finding

- Overall: Unblended acc: **0.754**, Blended acc: **0.789**



Ways forward

- Can we do better?
- Consider other classification models: Logistic regression, CNN, Gaussian processes, ...
- Each of these models requires some additional preprocessing of the data
- Each of these models requires a separate training and testing set

Preprocessing

- Select an even mix of blended and unblended footprints for training

For each footprint:

- Make a square **cutout** of a fixed size, centered on that footprint
 - 23 pixels to a side seems to work well
- **Zero out** any pixels that aren't part of the footprint
- **Normalize** pixel values

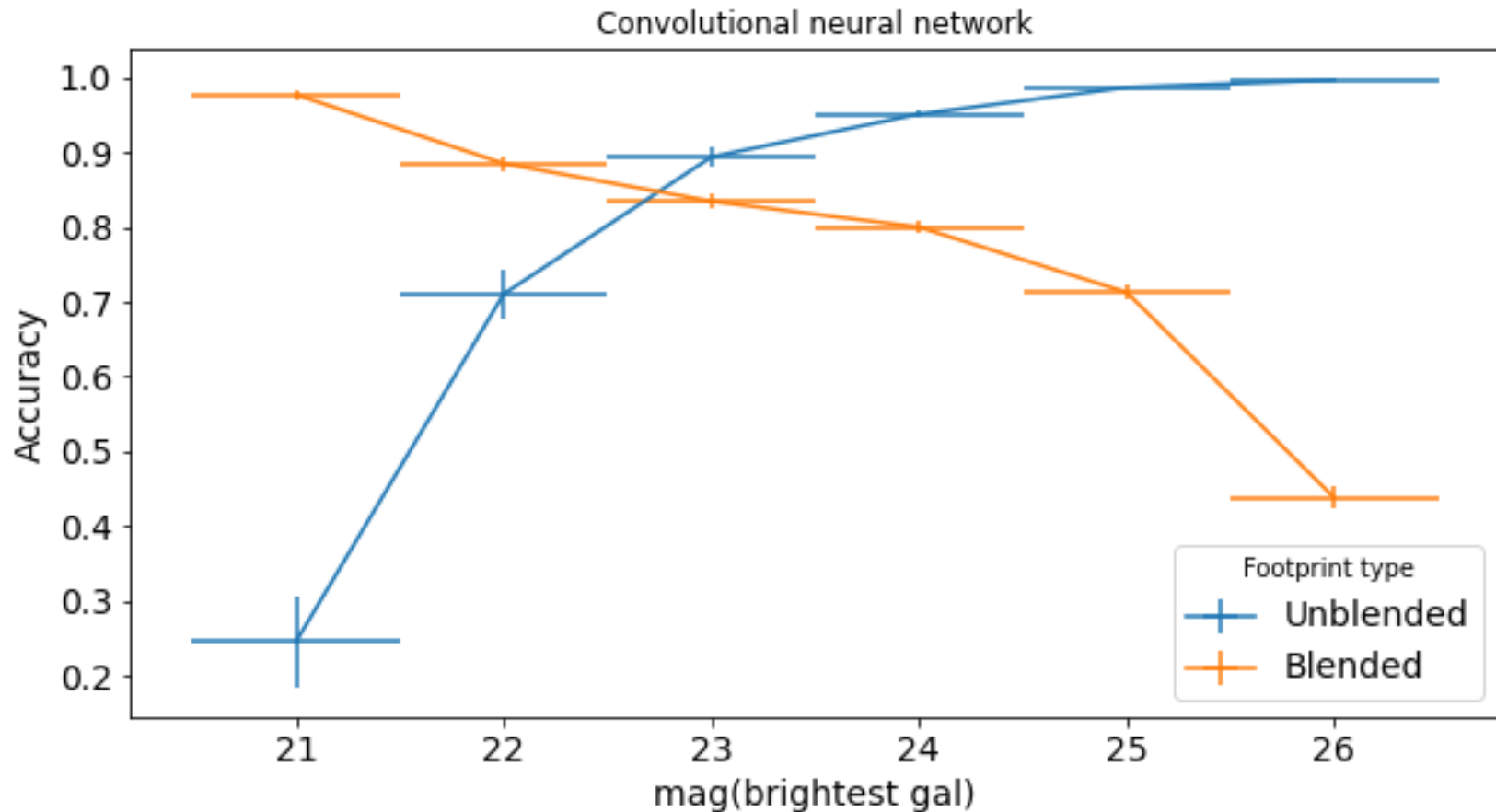
Convolutional neural network: Keras implementation

```
input_shape = (23, 23, 1)
num_classes=2
model = keras.Sequential()

model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(800,activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(400,activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(200,activation = 'relu'))
model.add(Dense(num_classes, activation="softmax"))
epochs=20
model.compile(loss="binary_crossentropy", optimizer='adam', metrics=["accuracy"])
time_start = time.time()
model.fit(np.reshape(trainxnorm,(len(trainx),23,23,1)), np.array(trainy),
          epochs=15,verbose=True,batch_size=200,validation_split = .1)
train_time = time.time()-time_start
```

Convolutional neural network

- Overall: Unblended acc: **0.977**, Blended acc: **0.753**



Preprocessing

- *Select an even mix of blended and unblended footprints for training*

For each footprint:

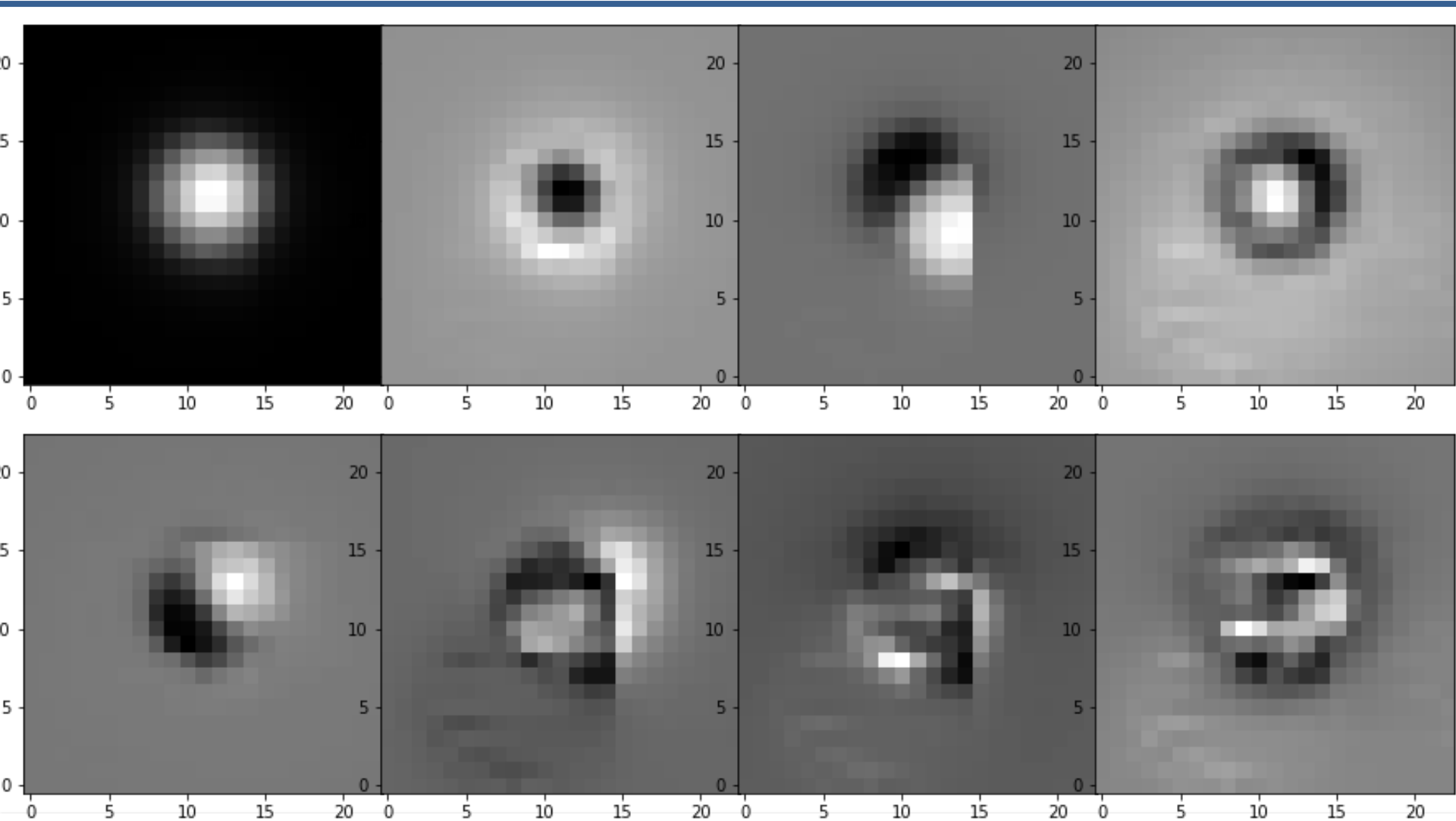
- *Make a square **cutout** of a fixed size, centered on that footprint*
 - *23 pixels to a side seems to work well*
- ***Zero out** any pixels that aren't part of the footprint*
- ***Normalize** pixel values*

For logistic regression and Gaussian process models:

- **Flatten** the 2D pixel array (size=23x23) into a 1D vector (size=529)
- **PCA embedding** to reduce dimensionality: 529 → 8

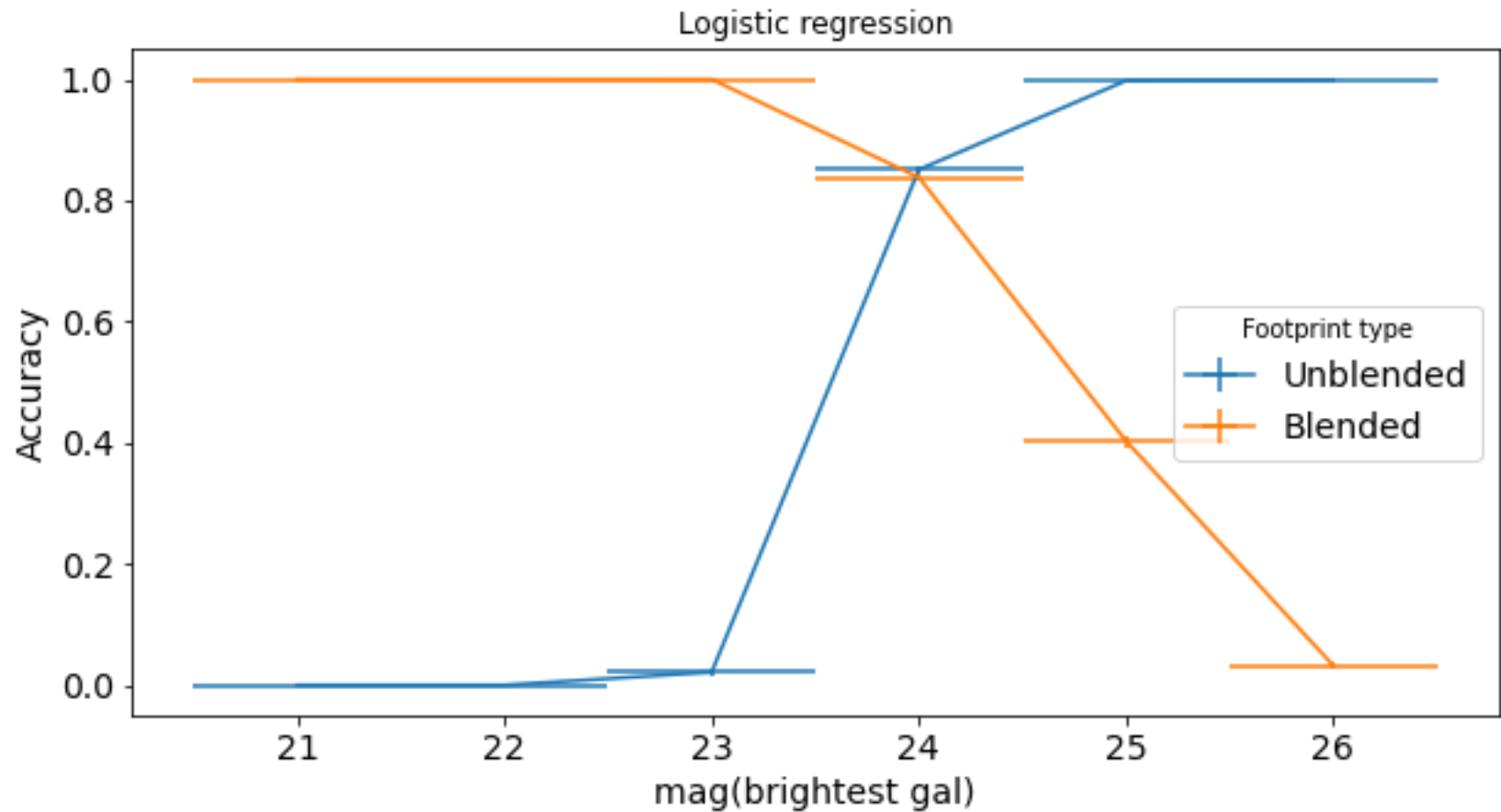
PCA components

*from an actual DC2 patch



Logistic regression

- Overall: Unblended acc: **0.895**, Blended acc: **0.716**



Gaussian Process model

- Gaussian process: An infinite collection of random variables, any finite subset of which is Gaussian-distributed
- The random variables: **For each possible value of the PCA-embedded footprint cutouts, yield a number specifying the “blendedness”**
- For each training example i , define **truth label $y_i = +1$ if blended, -1 if unblended**
- Let \mathbf{f}^* denote the **model-estimated blendedness** of test examples
 - If $\mathbf{f}^* > 0$, classify examples as blended; otherwise unblended

Some math

- Given the PCA encodings of train and test examples, assert Bayesian **prior** on the joint distribution of **blendedness of training and test sets**:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(0, \sigma^2 \begin{bmatrix} K_{\mathbf{ff}} + \tau^2 I_n & K_{\mathbf{f}*} \\ K_{*\mathbf{f}} & K_{**} \end{bmatrix} \right)$$

- Kernel matrices:

$$(K_{\mathbf{ff}})_{i,j} \equiv k(x_i^{\text{train}}, x_j^{\text{train}})$$

$$(K_{\mathbf{f}*})_{i,j} \equiv k(x_i^{\text{train}}, x_j^{\text{test}}) = (K_{*\mathbf{f}})_{j,i}$$

$$(K_{**})_{i,j} \equiv k(x_i^{\text{test}}, x_j^{\text{test}})$$

- Matérn kernel:

$$k_{\text{Matérn}}(\vec{x}, \vec{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\vec{x} - \vec{x}'\|_2}{\ell} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\|\vec{x} - \vec{x}'\|_2}{\ell} \right)$$

More math

- Additionally **given the actual blendedness of the training examples**, we can **analytically compute the posterior joint distribution of blendedness of test set**:

$$\mathbf{f}^* | X_{train}, X_{test}^*, y \sim \mathcal{N}(\bar{\mathbf{f}}^*, \sigma^2 C)$$

$$\bar{\mathbf{f}}^* \doteq K_{*\mathbf{f}}(K_{\mathbf{ff}} + \tau^2 I_n)^{-1} \mathbf{y}$$

$$C \doteq K_{**} - K_{*\mathbf{f}}(K_{\mathbf{ff}} + \tau^2 I_n)^{-1} K_{\mathbf{f}*}$$

- Classify test example as blended if $\bar{\mathbf{f}}^* > 0$

A Note on Training

- CNN and LR models trained to minimize cross-entropy loss
- Gaussian process model trained in two stages:

All hyperparameters other than sigma tuned to maximize balanced accuracy of class label predictions

Then, sigma (which doesn't affect class label predictions) tuned to minimize cross-entropy loss

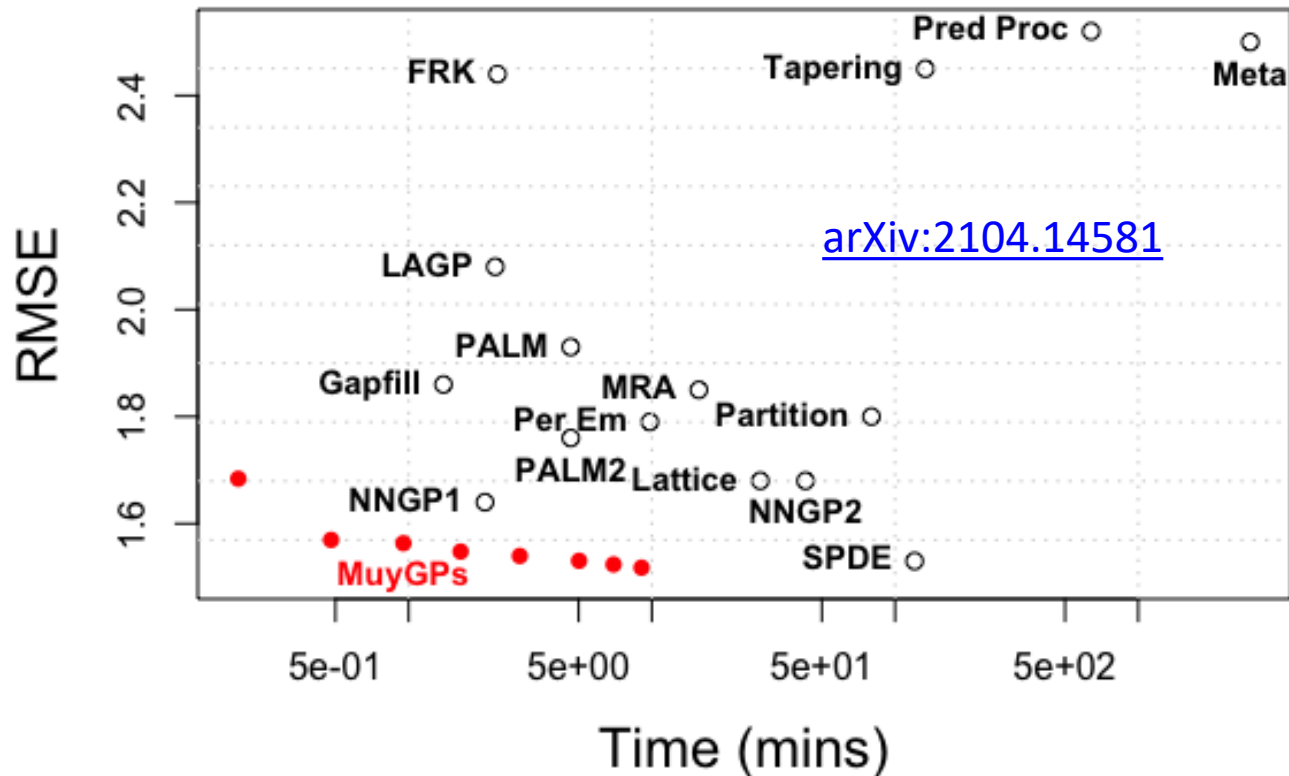
$$\mathbf{f}^* | X_{train}, X_{test}^*, y \sim \mathcal{N}(\bar{\mathbf{f}}^*, \sigma^2 C)$$

$$\bar{\mathbf{f}}^* \doteq K_{*\mathbf{f}}(K_{\mathbf{ff}} + \tau^2 I_n)^{-1} \mathbf{y}$$

$$C \doteq K_{**} - K_{*\mathbf{f}}(K_{\mathbf{ff}} + \tau^2 I_n)^{-1} K_{\mathbf{f}*}$$

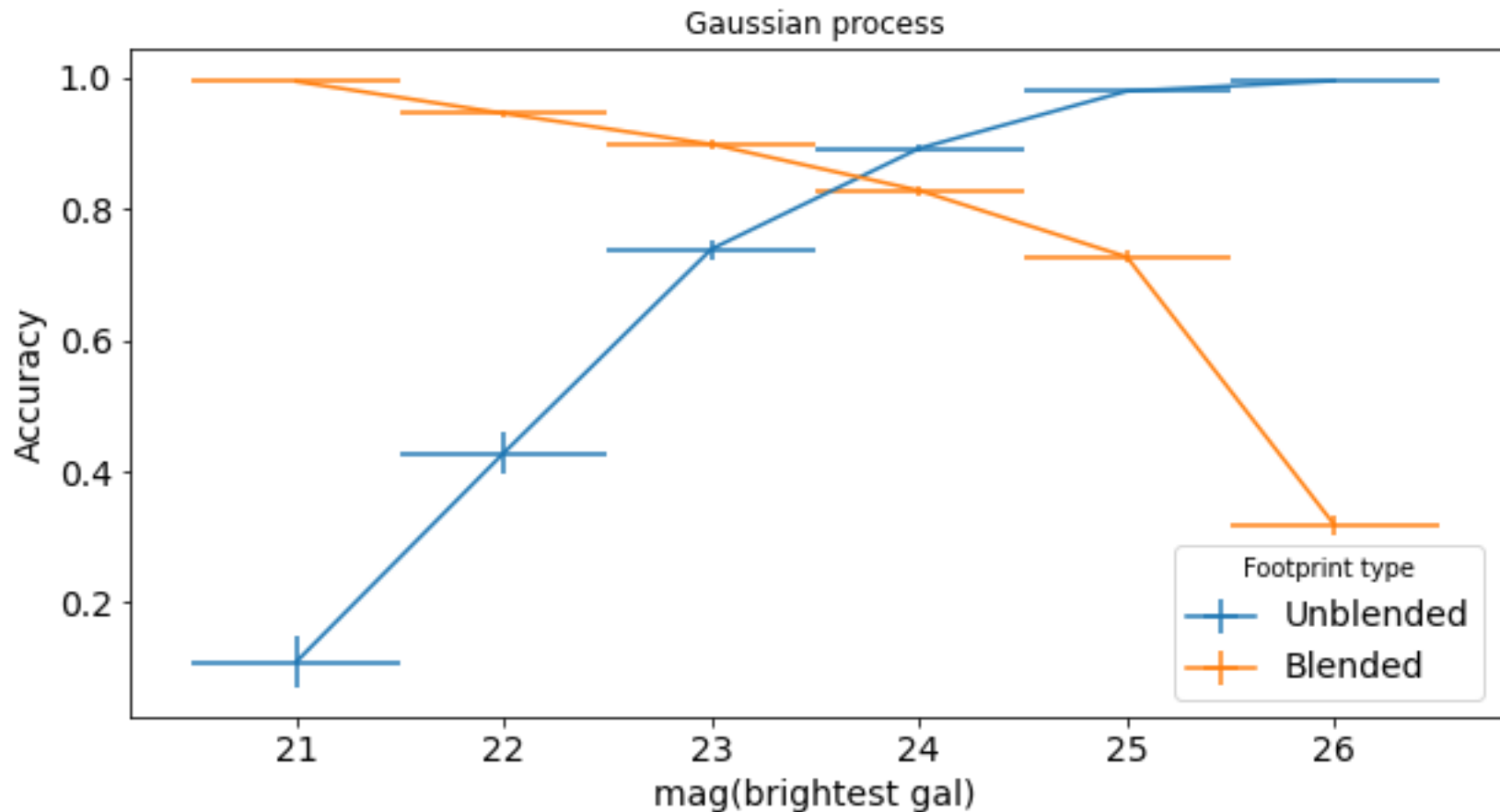
A Note on Speed

- MuyGPs: Very fast implementation of Gaussian process inference + hyperparameter tuning
- <https://github.com/LLNL/MuyGPyS>



Gaussian process

- Overall: Unblended acc: **0.942**, Blended acc: **0.800**

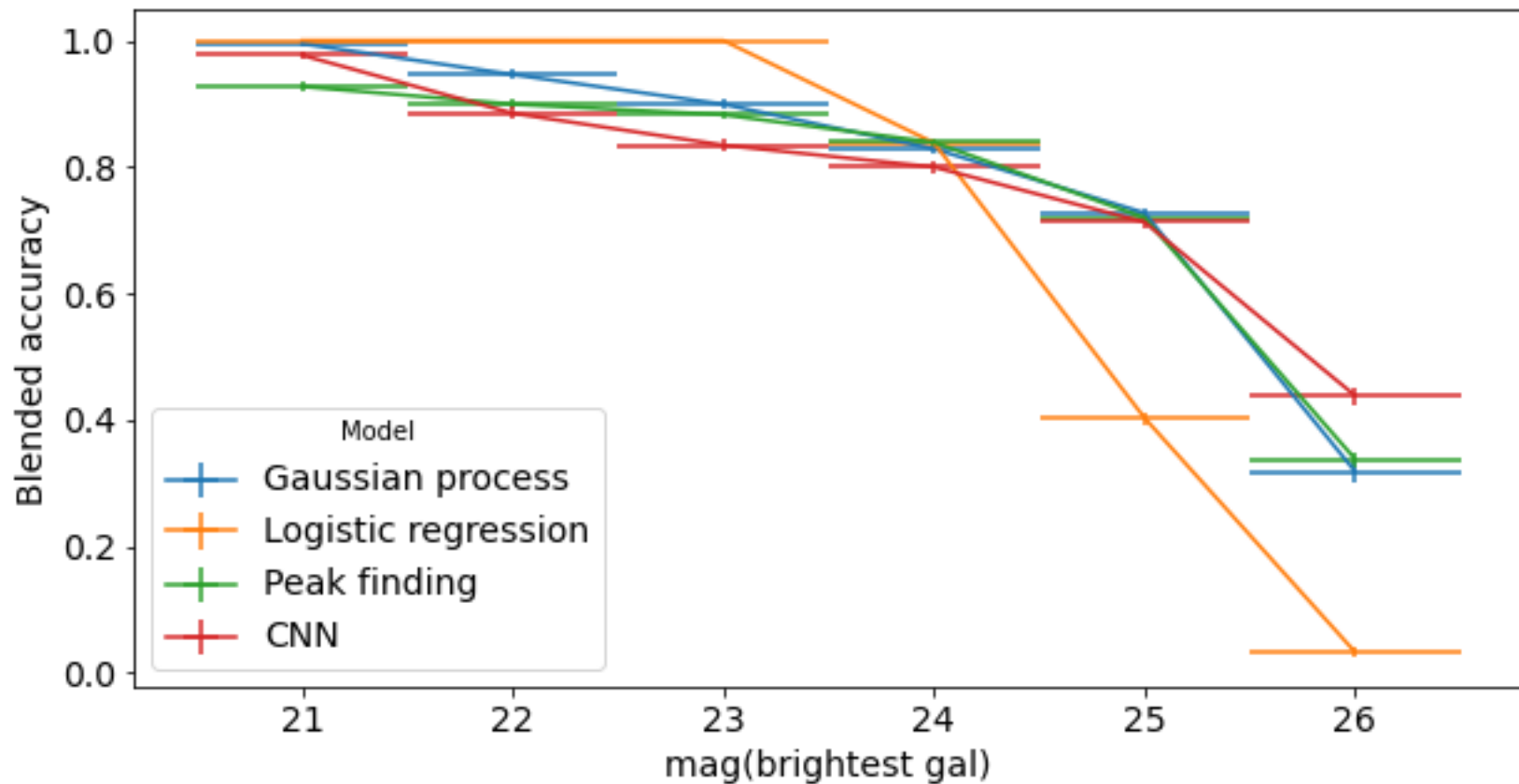


Model Comparison:

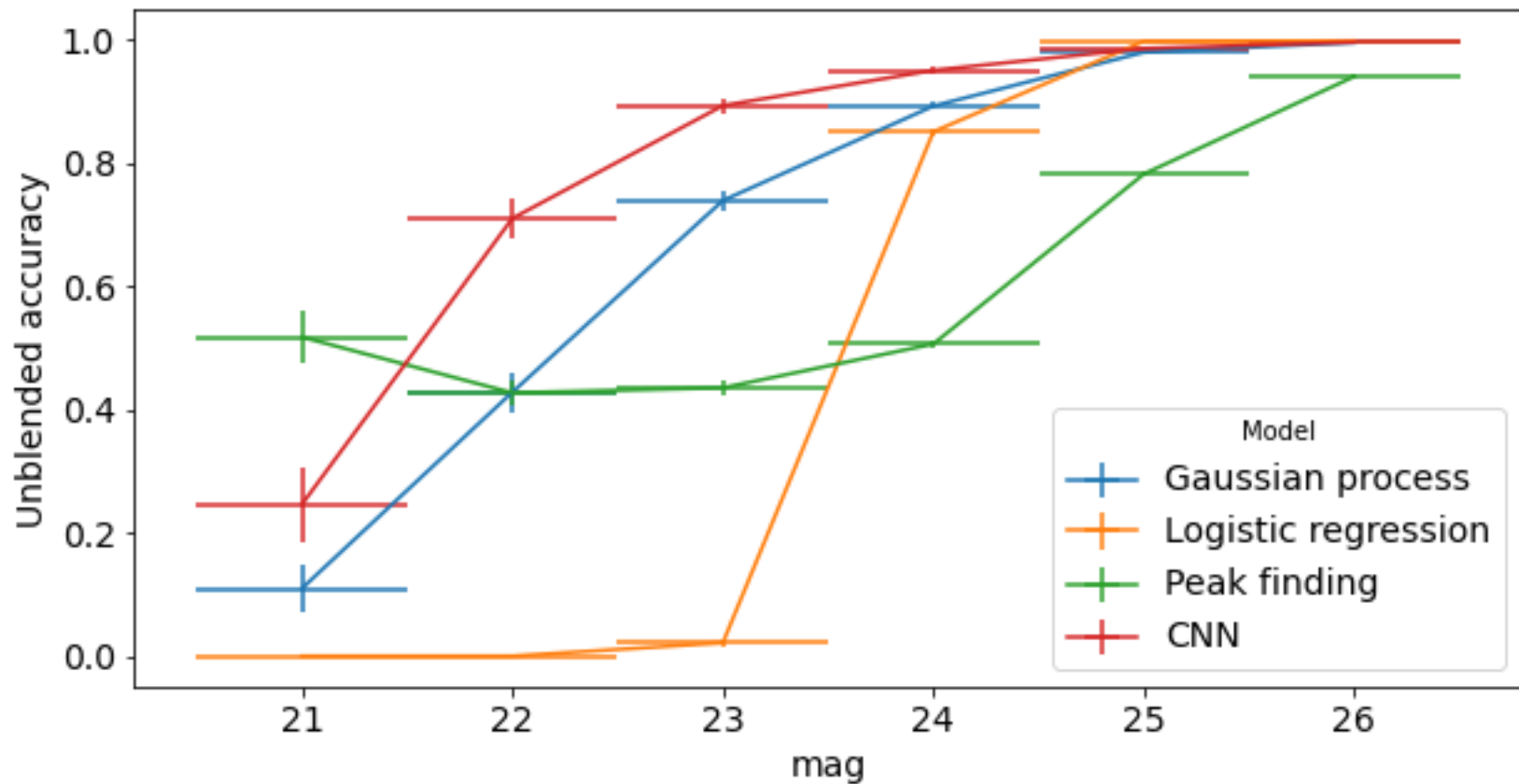
Overall performance

- **Peak counting**
 - Balanced accuracy = **0.771**
 - Unblended acc: **0.754**, Blended acc: **0.789**
- **Logistic regression** with l2 regularization
 - Balanced accuracy = **0.805**
 - Unblended acc: **0.895**, Blended acc: **0.716**
- **Convolutional neural network**
 - Balanced accuracy = **0.865**
 - Unblended acc: = **0.977**, Blended acc: **0.753**
- **Gaussian process**
 - Balanced accuracy = **0.871**
 - Unblended acc: **0.942**, Blended acc: **0.800**
- Mean cross entropy loss:
Logistic regression: 0.438, CNN: 0.301, Gaussian process: 0.324

Model comparison: Blended accuracy

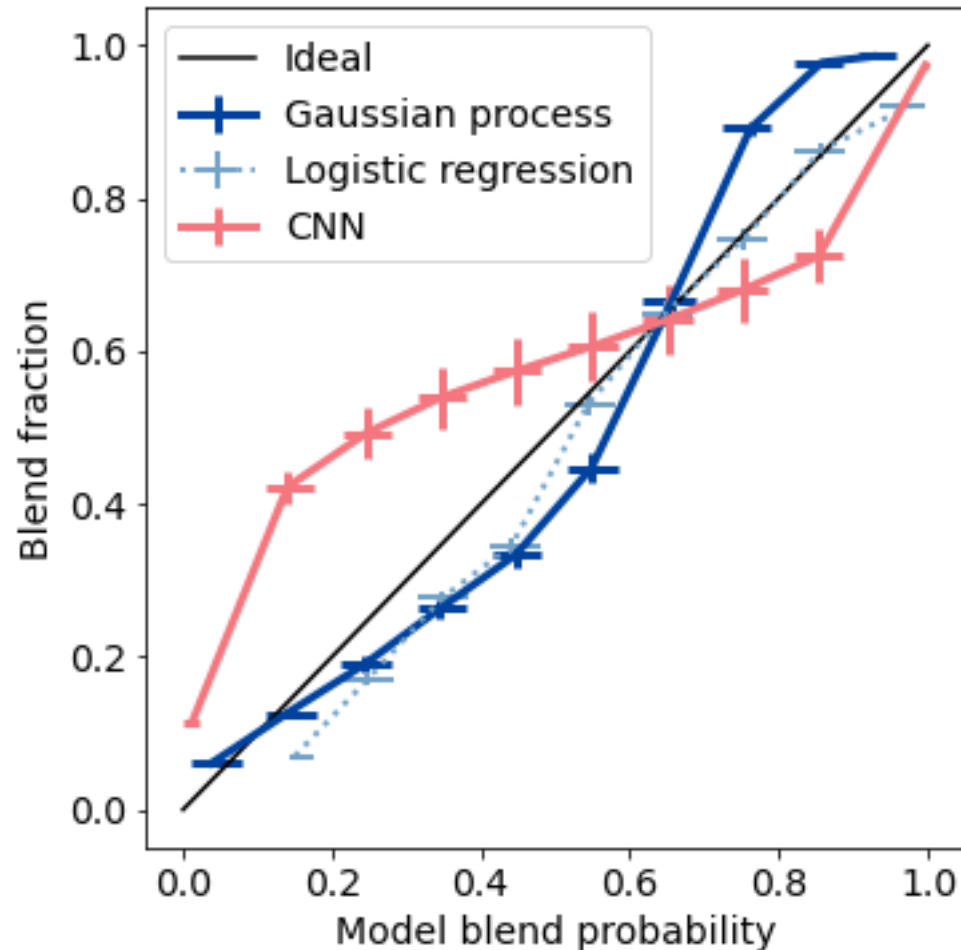


Model comparison: Unblended accuracy



Model comparison: Calibration

- How well can we interpret model score as a blend probability?



Model comparison: Confidence

- How confident are these models when they are right, when they are wrong, and when they disagree with each other?

Cutout	Truth	Peaks	GP	CNN	LR
(a)	Unblended	1	0.052	0.953	0.246
(b)	Blended	1	0.471	0.401	0.255
(c)	Unblended	1	0.510	0.147	0.341
(d)	Blended	1	0.307	0.939	0.346
(e)	Unblended	2	0.283	0.964	0.401
(f)	Blended	4	0.957	7.46e−11	0.974

Some topics for further study:

Hierarchical probabilistic detection

- Multi-class classification
 - This work describes a 1 vs. >1 classifier
 - Train additional 2 vs. >2, 3 vs. >3, ... classifiers
- Object localization
 - Train a regressor to predict x—y coords of each object
 - Separate regressors for footprints with 1 object, 2, 3, etc.
- If the # of objects and the object locations are all given as probabilities, the combined result is a fully probabilistic detection scheme starting from footprints

Some topics for further study:

Include all filter bands

- One option: Build separate classifiers for every band, then later combine results across bands using current Pipelines strategy
- Another option: Concatenate PCA embeddings across all bands, then train model to count the total number of objects that are bright in at least one band

Some topics for further study:

Prior mean

- In this study, GP model assumes a prior mean of 0 everywhere, and training set is evenly balanced between blended and unblended examples
- A priori, blend probability actually depends very strongly on footprint size
- Try using a more sophisticated prior mean estimate based on footprint size
- Reassess training set composition

Some topics for further study:

Impact on science

- Feed the outputs into a deblender that needs it (SCARLET)
- Check impact on, e.g., weak lensing shear inference

Some topics for further study:

Go big

- Test on as much data with as much realism as possible
- DC2
- NERSC

Some topics for further study:

Pipelines implementation

- Replace the peak set for each footprint with the GP detection+localization outputs
- Concretely, in findFootprints (afw/detection/FootprintSet.cc), load a trained GP model, and then for every footprint:
 - Ignore the existing call to findPeaks
 - Find instead the model output on that footprint – the number of objects and their locations
 - For each object, call foot.addPeak(x, y, sn)
 - “sn” should be the value at location x, y in the smoothed image
- From this point forward, all references to “peaks” in the Pipelines code actually refer to the GP model outputs
- This works most straightforwardly with the “separate classifier for every band” strategy
- Investigate temporary local background subtraction

Some topics for further study:

Model combination

- GP and CNN models seem to make different kinds of mistakes
- Maybe combining the two could cancel out some errors
- This would make the probabilistic interpretation trickier
- The GP model should perform about as well on slightly more realistic data, especially since it's based on a just few PCA components
 - This is true in a preliminary study of actual DC2 images
- But CNN models can suffer sharp performance drop outside training distribution – hard to say how well it would generalize to real images

Some topics for further study:

Calibration

- Apply calibration factors to posterior estimates, see if results are more reliable
- Try re-optimizing hyperparameters of calibrated model
 - Re-calibrate, re-optimize, etc.

Some topics for further study:

Object definition

- In this study, a footprint is “blended” if it contains more than one “object”, and an “object” is defined by $S/N > 5$ in the i-band
- May be more useful, for deblending, to detect objects down to a lower threshold, even if lower- S/N objects are not kept in a final analysis
- Peak finding can detect arbitrarily low S/N in the outer wings of footprints, down to ~ 4 in practice
- Preliminary study: GP classifier is about 1% less accurate if $S/N > 4$ is used
- This also reduces the number of “empty footprints”

Some topics for further study: S/N variability

- In the work above, consistent noise level and PSF width used for each coadd
- In real coadds, these can vary
- These affect S/N calculation, and hence the definition of whether a footprint is “blended”
- Proposed mitigation: Train a separate classifier model for each cell/patch, using the noise level and PSF width estimated in that cell to make training images



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.