

Manuscript Number: ASCOM-D-20-00107

A Machine Learning Approach to the Detection of Ghosting and Scattered Light Artifacts in Dark Energy Survey Images

Dear Dr. Wang,

Thank you for submitting your manuscript to Astronomy and Computing.

I have completed my evaluation of your manuscript. The reviewers recommend reconsideration of your manuscript following major revision. I invite you to resubmit your manuscript after addressing the comments below. Please resubmit your revised manuscript by Feb 26, 2021.

When revising your manuscript, please consider all issues mentioned in the reviewers' comments carefully: please outline every change made in response to their comments and provide suitable rebuttals for any comments not addressed. Please note that your revised submission may need to be re-reviewed.

To submit your revised manuscript, please log in as an author at

[https://urldefense.proofpoint.com/v2/url?u=https-3A\\_\\_www.editorialmanager.com\\_ascom\\_&d=DwIGaQ&c=gRgGjJ3BkIsb5y6s49QqsA&r=zFEkHHAPxXtVxNbgRXmC\\_Q&m=IQThViUIY1fNdqBcEdA4fDhN9wks2OvpSVduMGA6ZVM&s=xGZaGqmqPfxV2QgTky1e3sRtklReSepQqTZHPHvi2sM&e=](https://urldefense.proofpoint.com/v2/url?u=https-3A__www.editorialmanager.com_ascom_&d=DwIGaQ&c=gRgGjJ3BkIsb5y6s49QqsA&r=zFEkHHAPxXtVxNbgRXmC_Q&m=IQThViUIY1fNdqBcEdA4fDhN9wks2OvpSVduMGA6ZVM&s=xGZaGqmqPfxV2QgTky1e3sRtklReSepQqTZHPHvi2sM&e=), and navigate to the "Submissions Needing Revision" folder.

Astronomy and Computing values your contribution and I look forward to receiving your revised manuscript.

Kind regards,

Jessi Cisewski-Kehe, Ph.D.

Editor

Astronomy and Computing

Editor and Reviewer comments:

Reviewer #1: The authors proposed a Neural Network architecture to classify artifacts in astronomical images. The paper is concise, direct and clear. The problem is relevant, and it is indeed a typical case where a Deep Learning algorithm could perform a fast and accurate assessment. However, I would recommend significant corrections in the present paper. As this paper presents a proof-of-concept, I understand that these corrections are needed to clarify and explore the results, limitations, and knowledge obtained.

*We thank the referee for a careful reading and many insightful comments. We have responded below and have highlighted changes to the text in blue.*

At the beginning of your introduction, the authors mention that They are dealing with a specific type of artifact. Why are the others less relevant?

*Other imaging artifacts are indeed important; however, we choose to attack the problem of ghosts/scattered light because: (1) They can contaminate large areas of individual images, (2) We have an automated approach that is inadequate, (3) We have a large data set that has*

undergone visual inspection. This provides us both the need (to correct the algorithm) and the large data set needed to train a deep learning algorithm.

The authors present a Deep learning architecture with no further motivation. Could you please motivate your choice? Did the authors try a different design? The architecture is not innovative but is also not a famous design as Vgg, Inception, Resnet and others. Thus, to justify a choice and not to be simplistic, I would recommend some motivation on the design chosen.

From the text, it seems that your network outputs two numbers, why if this is a binary classification? This is a possible choice, but this leaves room for ambiguity. For instance, there might be cases in which both classes' probabilities are minimal. Did the authors find such cases? It was also unclear how the authors define the confusion matrix, i.e. there was a threshold, or the method picks the highest of the two output probabilities? Please clarify that in the text.

Our design choice was motivated by the primary goal of providing a proof-of-principle demonstration for using ML methods in ghost/scattered light artifact detection. This meant the main qualities we were looking for in the architecture were simplicity, ease of implementation in existing frameworks, low computational resource requirements for easy training, and decent performance on standard image classification datasets. The CNN architecture we used is essentially AlexNet, differing from it only in terms of hyperparameters. We have added new text at the beginning of Section 3.1 to touch on the motivation behind the choice of this architecture.

The two SoftMax outputs always sum to 1, and is thus effectively a single output (as you have correctly identified). To avoid any ambiguity, we have added text in the 2nd paragraph of Section 3.1 (right after describing the SoftMax outputs) describing how these two outputs can be interpreted. Providing this detail should make it clear that this is equivalent to requiring a threshold of  $>0.5$  on the "ghosts" output to indicate the possible presence of a ghost in the image.

I would strongly suggest that the authors add value to the paper by presenting examples. For instance, the authors could present cases where the method completely misclassify. This is a helpful practice to give insights into the data. Is there a pattern of poorly classified objects? The authors could also add high probabilities and low probabilities examples.

We have added typical examples of false positives and false negatives found by the model in the test data sample used in the evaluation phase. We have also included a brief description/discussion of these examples in Section 3.3.

The presented method used png files. This seems an unnecessary limitation. Is there a reason for that?

The downsampled PNG files were generated automatically as part of the DES data processing. The full imaging data is substantially larger (DECam is a 570 megapixel camera), so some downsampling is required prior to application of the CNN

The ROC and the confusion matrix presented in figure 4 contains training and validation. This was unclear in text. This ROC has a few information, so I suggest you plot together with the other from the test. With the two ROCs in the sample plot, it would be easier to compare them and have insights on how the model loses performance in the test set. I understand why the authors separated them, but the way the figure is, it did not enrich the paper.

The ROC curve shown in Figure 4 was actually only for the validation set. We have now replaced Figures 4 and 5 with a single plot (Fig. 4) that shows the ROCs separately for the training, validation, and test sets, with the AUCs associated with each curve indicated in the legend.

Could you provide the numbers such as precision, recall, AUC in a table for all samples? Please make sure you present these numbers on the validation separated from the training.

A table (Table 1) has been added that shows the Accuracy, Precision, Recall, and AUCs of the training, validation, and test samples.

I do not understand the precision estimate in the section 3. The authors mentioned 716 false positives and 241 true positives. Thus this would give a precision of  $241/(716+241) = 25,18\%$ . This is a much smaller precision compared to your validation sample and the number presented as precision in this test sample ( $\sim 78\%$ ). Please verify these numbers and in case I am missing something, my suggestion is that you give a more detailed explanation of these numbers. If the precision drops dramatically in this test sample, please add a discussion on the probable reason on how to address them. It is worth noticing that this is an estimation of precision as you don't verify all the samples you assume that the ray-tracing algorithm did not miss any ghosts that could also be missed by your method. Please make this clear in the text. If this is the case, I would not name the estimates you use in this sample as a confusion matrix.

We apologize if this was not clear in the paper. The total number of images in the Y5 sample was 23,755. Out of this total, 3,285 were identified by the ML model as "positives". Of these positives, 241 were in common with those found by the ray-tracer and the rest (3,044) were found only by the ML model. All 241 in the overlap region with the ray-tracer were TPs while  $2328 = 3,044 - 716$  of the 3,044 exclusively found by the ML model were TPs. So the precision in the case of the Y5 sample is  $p = (2328 + 241) / [(2328 + 241) + 716] = 2569 / 3285 = 0.78$ . We have reworded the text to make things clearer.

Also, a short description of how ray-tracing method works would be useful for the reader, myself included, that is not familiar with how artifacts are detected in astronomical surveys. Please enrich the discussion by commenting on the future steps on developing a precise, accurate tool to ghost detection in future surveys. What are the requirements to push the tool beyond the proof-of-concept stage?

We have added a short section after the introduction describing the ray tracing program.

We have also updated the last paragraph of the Introduction to mention follow-up work that will not only detect the presence of artifacts but will allow their localization within the image down to the pixel level.

Reviewer #2: I have two general comments and some smaller ones:

It would be good to include more information about computational resources that were used. What hardware was the code run on? How long did it take to train using what resources? How much time does it take to run the CNN (train+test) versus the ray-tracing? Can this model be run at LSST scale?

The CNN we used has a fairly lightweight architecture relative to many other deep neural network models commonly used. It does not require significant computing resources to train on our ~5k training+validation set. In fact, the training over 30 epochs took less than 10 minutes on a laptop computer with a mobile GPU. Running the same training in Google Cloud takes even less time (a factor of 4 less). The process of performing inference takes ~2ms per image on the laptop GPU with our unoptimized network.

Regarding the ray tracing program, most of the time is spent querying the bright star catalog around each exposure. The actual process of ray tracing itself takes negligible time -- on the order of a few ms per image similar to the CNN performance above. However, we should reiterate that the CNN we used for this proof-of-concept study has not been heavily optimized. There now exist various ML toolkits that can automatically optimize trained networks for speed through techniques like pruning nodes with low weights, and by using lower precision. Also available now are high level synthesis tools that allow ML networks to be implemented in programmable hardware like FPGAs for critical real-time applications. Performing inference for LSST should not be an issue, since LSST has allocated 60s to process an image and generate alerts.

We have added a separate section (Section 5) that describes the computational resources used in more detail.

A little more information on the performance of the ray tracing program would be helpful. There is a need for ML in this space, but I would like to see more proof of this need. What are the statistics from the ray tracing program? How long does it take to run. My point is, when LSST comes online, if ray tracing is faster than the CNN, will ray tracing be good enough for the first round of data reductions that go out to the alert stream? Are the CNNs only feasible for the yearly data releases? Or is the CNN something we need and can do on each image as it comes in? This is touched upon in Section 3 but not explicitly stated.

We have created a new section (Section 2) to discuss the ray tracing program in more detail. The primary challenge of the ray-tracing algorithm is that it is a “predictive” model based on external measurements and observing metadata, but *does not* use the actual DECam image data. This leads to the following major limitations:

- the ray-tracing depends on the accuracy of the optical model and telescope pointing telemetry.
- the fluxes of bright stars that are reported in external catalogs that report measurements for different wavelengths and at different times than the actual image

With regards to LSST, the development of an analogous ray tracing code would depend on how fast and portable LSST optical model and ray-tracing algorithms are (we do not have access to that codebase) and whether a CNN algorithm can be trained effectively on commissioning data

in advance of the first survey observations. The CNN execution time should not be a limiting factor, assuming that suitable down-sampled focal plane images can be generated.

As mentioned in the response to the previous question, the performance of the unoptimized CNN in the task of inference (detailed in Section 5 of the updated manuscript), is already comparable to that of the ray-tracing program even with a mobile GPU on a laptop. Section 5 also mentions hardware acceleration methods that make it feasible to use a suitably trained CNN in real-time, online environments, to perform inference on images as they are acquired.

Introduction:

"In an unpublished report, a classical ML algorithm (i.e., a support vector machine) and a CNN were applied to DES images to identify artifacts belonging to 28 different classes"

-> This examples seems unfinished. Did the SVM do better or worse than the CNN? Did they both do well enough or poorly? Same for the HST example. Are ML techniques working to classify artifacts or are people unable to do them quickly or with a high accuracy?

We have added more details about the performance of the CNN on both previous examples. In general, the ML techniques are improving on what has been done previously.

Section 2.2.1:

How large is the final training set given that the data is augmented?

The size of the final training set actually remained the same after "augmentation". The procedure takes each batch of images used in training and randomly flips them horizontally and vertically. The original images in the batch are then replaced by the flipped images and this "augmented" or modified set is the set used to train the model. So this is not an additive procedure in which the flipped images are used together with the original images. This process of perturbing the original sample with some random "noise" aims to help make the model more generalizable. The term "augmentation" is often used in the field to refer to the procedure as described above and we agree this can be confusing. We have removed the incorrect phrase "... augment the training set size ..." in the 2nd paragraph of Section 2.2.1 and reworded the text to avoid this confusion. We also mention the use of the ImageDataGenerator class in Keras to perform this step, explicitly stating that the process involves in-place substitution where the total data size is kept constant. Please see the official keras page cited in the paper or the following page for more details:

<https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation>

Section 2.2.2:

How many false positives or negatives did you have in the training set for the ray tracing program? I would like to understand the performance of the ray-tracing program versus the ML. Doing a more direct comparison between the two would show the need for ML more convincingly.

The ray tracing program is not "trained" in the conventional sense used when discussing ML algorithms. The ray-tracing code was originally developed in the context of DECAM construction. The procedure for predicting the locations of ghosts the known as-built optical

design of the corrector and standard sequential ray tracing. Roughly 100 images from DES Science Verification were used to calibrate the predicted intensities of these ghosts. We have added this discussion to Section 2.

Section 2.2.3/2.3:

What is the probability threshold to determine TP, FP, TN, FN and all the associated statistics? We used a probability threshold of 0.5 to distinguish between “ghosts” or “no ghosts”. I.e.  $>0.5$  on the “ghosts” output of the network is considered a positive. We have added some text in the 2nd paragraph of Sect. 3.1 about how the two SoftMax outputs always sum up to 1 and how the larger of the two outputs is chosen to determine the model’s prediction. This is equivalent to applying a threshold of 0.5 on one of the outputs to determine the prediction.

Section 3:

How long did it take to perform inference on the Year 5 data?

Performing inference on the Y5 data set took 2ms per image. We have added this detail in a new section (Section 4) that discusses the computer resources used.

Is the higher false positive rate in the Y5 sample due to the nonrepresentativeness of the training set? The training set has 50/50 ghost and clean samples whereas the Y5 data presumably has a much lower occurrence of artifacts.

In Fig. 6 of the updated version of the manuscript, we have included representative examples of false positives identified by the ML model for the test data sample. The false positives in the Year 5 data set belong to these same classes or types. It would appear that Year 5 has more of these types of images.

Figure 6 is not referenced anywhere in the text.

A reference has been added in Section 3.

Conclusion:

There is either a missing word or too many words in the second sentence in the conclusion. This sentence has been reworded.