Data-centric AI workflow based on compressed raw images

Marco Aversa[1,2], Ziad Malik[1], Phillip Geier[1,3], Fabien Droz[4], Andres Upegui[3], Roderick Murray-Smith[2], Christoph Clausen[1], Bruno Sanguinetti[1,*]

[1]Dotphoton AG, Zug, Switzerland
[2]University of Glasgow, Glasgow, United Kingdom
[3]HEPIA, Geneva, Switzerland
[4]CSEM, Neuchatel, Switzerland
[*]bruno.sanguinetti@dotphoton.com

26 September, 2022

Abstract

In order to extract the full potential of the high volume of image data coming from earth observation, image compression is needed for transfer and storage, and artificial intelligence (AI) is needed for analysis. The promise of AI is to perform complex operations with low programming effort, naturally shifting the focus of the development of machine learning systems from the code, i.e. the implementation of the neural network, to the training process, and in particular to the acquisition, selection and preparation of training data. Lossy compression (like many other image processing methods), however, was developed primarily to compress already processed images for visual inspection, not regarding damage to invisible image properties which play an important role in machine-learning, such as higher order statistics, correlations and bias. The Jetraw image format, in contrast, was designed to compress raw image data, preserving its statistics and embedding camera calibration profile and noise model. These features facilitate the generation of accurate raw synthetic data. They allow for "Jetraw functions" to take a Jetraw image as an argument and return another Jetraw image, complete with its newly computed calibration profile and noise model. Several of these functions can be chained to build complex operations while always maintaining metrologically correct data, i.e. values that have independent errors, are unbiased and have a well-defined noise model. Jetraw images and functions may be used in end-to-end models to generate synthetic data with statistics matching those of genuine raw images, and play an important role in data-centric AI methodologies. Here we show how these features are used for a machine-learning task: the segmentation of cars in an urban, suburban and rural environment. Starting from a drone and airship image dataset in the Jetraw format (with calibrated sensor and optics), we use an end-to-end model to emulate realistic satellite raw images with on-demand parameters. First, we study the effect of various satellite parameters on the task's performance as well as on the compressed image size. These parameters are satellite mirror size, focal length, pixel size and pattern, exposure time and atmospheric haze. Then, we discuss characterising and improving the performance and tolerances of the neural network through the use of on-the-fly generation of data that accurately reflects the statistics of the target system.

1 Introduction

The promise of Machine Learning For a long time, image analysis has relied either on traditional machine vision algorithms or human analysts. Traditional algorithms are effective at identifying simple patterns in a very well-defined context (e.g. measuring the fill level of a test tube), but are generally less effective in more general tasks, such as object classification and segmentation (e.g. vehicles, crops etc.). Machine learning (ML) promises to match or even surpass human accuracy in these tasks as well as being able to process the vast quantities of data produced by modern satellites. The ML algorithms rely on a rather generic implementation that is adapted to a specific task through a training process. This shifts the focus from the implementation of the algorithm to the training data collection, selection and preparation.

Machine learning use in critical systems ML is already pervasive. For example, you can search your smartphone's image library for an object, text or person, and it will be found immediately. However, applying ML to critical systems, e.g. medical diagnostics, autonomous vehicles, or aerospace, still presents challenges [7]: the events to be identified are typically rare (as you would hope, for illnesses, accidents and disasters), the structures are small (little information

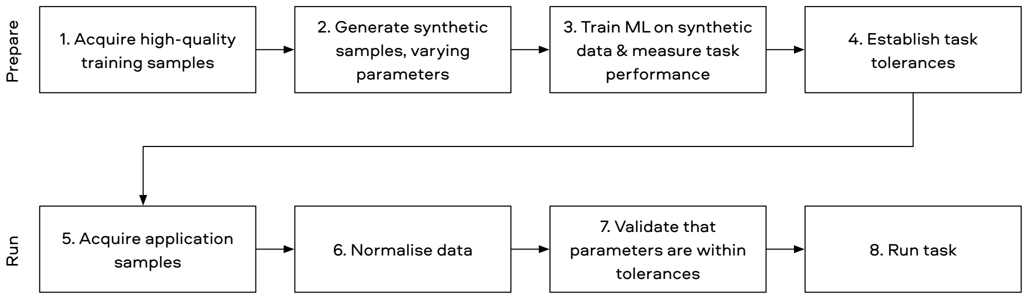

Prepare
1. Acquire high-quality training samples
2. Generate synthetic samples, varying parameters
3. Train ML on synthetic data & measure task performance
4. Establish task tolerances
Run
5. Acquire application samples
6. Normalise data
7. Validate that parameters are within tolerances
8. Run task

Figure 1: Proposed workflow: during a first development stage, a source image set is acquired and used to generate synthetic target sets that emulate the system under a number of engineering parameters and running conditions. Each target set is used to train the ML task and measure its performance, therefore allowing to translate task performance requirements to engineering requirements and tolerances. At runtime, this allows to validate that the images parameters are within the requirements of the application, and if possible to normalize them to optimal parameters.

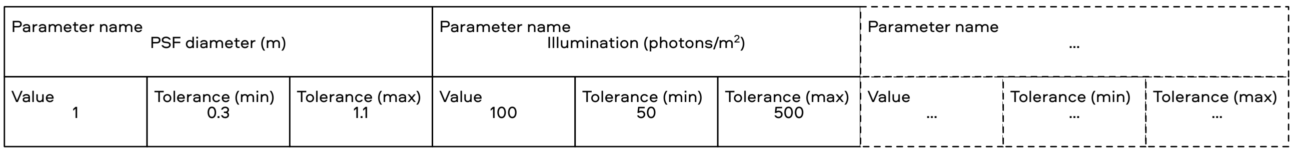
redundancy) and are at the limits of the instrument's resolution. In this situation, minor differences in pixel statistics can play an important role in the machine learning algorithm's performance. In particular, image processing, including compression, may alter these statistics and contribute to "dataset drift" [5]. While image processing can enhance details for the human eye to see, it is usually irreversible and reduces the information content of the output data. With access to the raw data, machine learning algorithms can optimize processing for their own application, either as part of a neural network directly or as a parameterized processing pipeline [5]. Depending on the application, the performance of specific image processing techniques may be mutually exclusive. For example, sharpening an image will typically result in more noise, and denoising would typically result in a less sharp image [10].

What is raw data, and why is it valuable for ML? For the rest of the discussion, we define what is intended as raw image data, and which properties of such data are important for machine learning. In practice, data from all current sensors is processed first in the analogue and then in the digital domain. This processing, however, if done with care, does not detract from the "rawness" of the data. We identified the following properties as being characteristic of raw data, which should therefore be preserved by processing techniques if the data is to be denoted as "raw". A more precise term would be "metrologically accurate" data:

1. The statistical uncertainty (also referred to as error or noise) of a given pixel is uncorrelated to other pixels.
2. Each sample arises from a well-defined statistical distribution.
3. Pixels are unbiased, i.e. the mean pixel values accurately represent the average amount of incident light.

The above properties are necessary to apply standard statistical methods to data, e.g. averaging, error propagation, etc. However, most processing algorithms (including compression) do not satisfy the above criteria, introduce systematic errors, and modify pixel statistics. Subsequent algorithms then cannot make any assumptions on each sample's random and systematic errors, resulting in significantly impaired performance. A few practical examples are: Summing lossy compressed frames typically will result in a decrease in signal-to-noise ratio rather than an increase. Shading correction increases the relative noise on the edges of the frame, decreasing the performance of object detection by thresholding in these areas. A simple linear fit to a gradient may improve accuracy by $10\times$ when using the appropriate uncertainty information. When vertically summing the pixels on a grating spectrometer, even a minimal bias at the pixel level may result in a measurable systematic error on the spectrum.

From Jetraw's "steganographic compression" to synthetic data for robust ML Lossless compression requires significant computational resources to achieve even modest compression ratios. On the other hand, lossy compression typically does not satisfy the above requirements to maintain the quality of raw data. Jetraw is a compression codec which is engineered to maintain the above-mentioned qualities of the raw data while achieving the compression ratio and resource efficiency of lossy codecs. Jetraw is a two-step compression algorithm with design principles derived from steganography. The first "preparation" step replaces random noise present in the image with pseudorandom noise and embeds a calibration profile in the image. Then, a compression step can compress prepared images efficiently in a lossless manner. Although not lossless, the preparation step is engineered to satisfy the requirements for metrological accuracy stated above and to



Parameter name PSF diameter (m)			Parameter name Illumination (photons/m^2)			Parameter name ...		
Value 1	Tolerance (min) 0.3	Tolerance (max) 1.1	Value 100	Tolerance (min) 50	Tolerance (max) 500	Value ...	Tolerance (min) ...	Tolerance (max) ...

Figure 2: The parameter vector is a metadata structure that defines an optimal value and a tolerance for each objectively measurable image parameter relevant to the application. Before the image is fed to an ML model for inference, its properties are checked against this parameter vector.

produce an image which is not distinguishable from an authentic raw image in a steganographic sense, i.e. by any algorithm, under the assumption that the sensor characteristics are correctly modelled.

Besides compression, the fact that the data is metrologically well-defined, and contains all the necessary calibration parameters, gives Jetraw images additional powers: it makes it straight-forward to implement functions that take as an input a Jetraw image arising from sensor A, and process it to output an image indistinguishable from one arising from a different sensor B, with different specifications. From a set of primitive functions that propagate both the image and calibration data, it is possible to build advanced pipelines for, for example, image correction or end-to-end simulation, data augmentation, normalisation etc. and to guarantee that at each step the metrological accuracy (the "rawness") of the data is maintained. These features can then be put to good use when implementing data-centric workflows in robust ML applications.

We used these features in the first place to establish the performance of our compression algorithm on image data emulating that coming from earth observation systems. As our compression algorithm effectively distils the information content of the image, it is particularly sensitive to the amount of real information present in the image, i.e. compression ratio increases with motion blur or point-spread function size and decreases with illumination. These testing methods, however, can also be applied to evaluate the performance of a ML task.

2 Metrologically accurate Data-centric workflow

Engineering machine-learning systems in critical applications is currently challenging. These algorithms typically operate on a large input parameter space, and hence require an even larger amount of samples to evaluate their performance. This is even more challenging for systems targeting a minimisation of false positives and false negatives on rare events. The current preferred method is to increase the odds for rare events by using extreme volumes of data. We propose a more data-centric approach, where we believe that higher performance may be achieved by a deeper understanding of the properties of the data. As a pre-requirement for these techniques to work, however, the image data must be metrologically well-defined, so that standard statistical techniques apply and each sample arises from a well-defined (physical) parameter space.

The target is to define specific image properties that are objectively quantifiable and that achieve the specified performance in the ML task if they are within specification. We show the proposed workflow in Figure 1. It consists of the following steps:

1. A source set of images is acquired with an instrument providing higher quality than the instrument in the final application. These high-quality images will be used to emulate a target (lower quality) satellite image set. For example, to test the concept, we acquired images with a drone and with an airship, with high-quality optics and a long exposure time.

2. Target sets of images are generated by degrading the source set in a metrologically accurate manner using an end-to-end model. Each target set corresponds to a parameter combination that is relevant with respect to the specifications of the ML task.

3. For each target set of specific parameter values, the ML task is trained, and its performance is measured.

4. We can now establish the parameters and tolerances that satisfy the task's requirements and engineer the application's system to satisfy these requirements.

5. In the deployed application, images are acquired to satisfy or exceed the required parameters.

6. If images exceed the required parameter range, it is often possible to normalize them to the required parameter range. For example, an image of too high resolution may underperform if high-resolution images were not used during training. It is then possible to artificially degrade the resolution while increasing the task's performance.

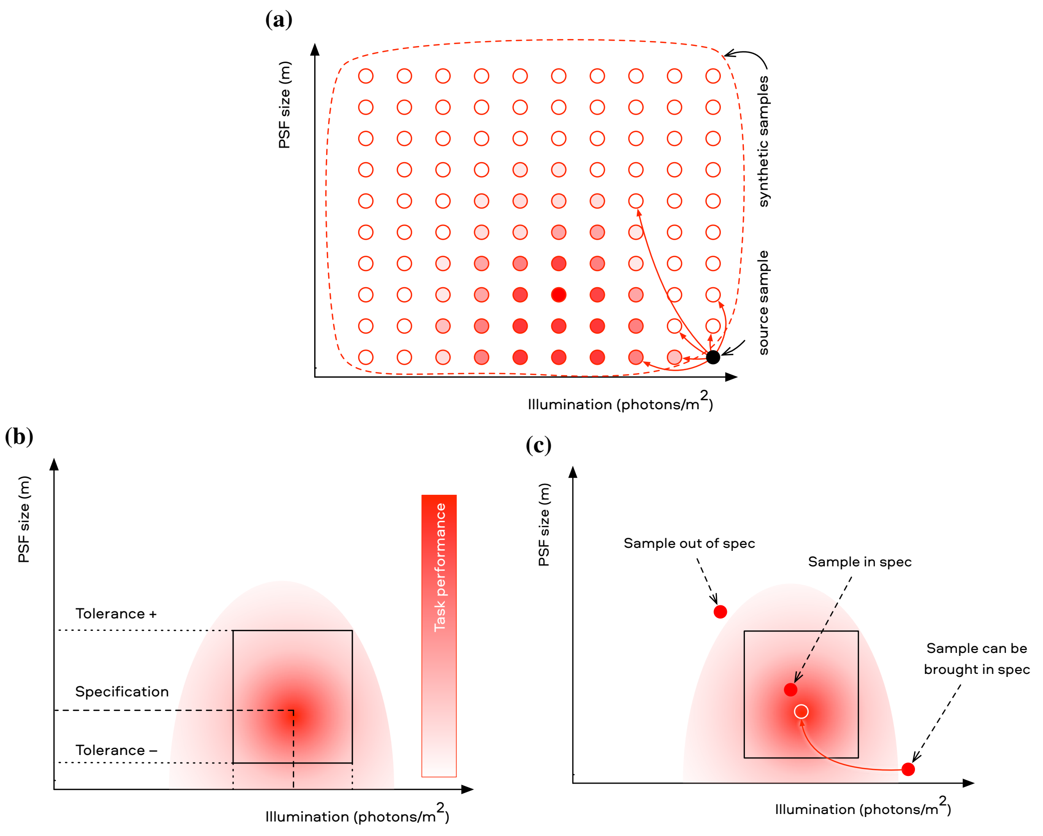


Figure 3: Illustration of the method used to define algorithm tolerances. For simplicity, only two parameters are used: the size of the point-spread-function (PSF) and the illumination. (a) Synthetic target samples are generated by metrologically accurate degradation of high-quality source samples in order to cover a desired parameter range. (b) The ML task is trained and its performance measured using the synthetic samples. Based on the result, a specification with tolerances is established. (c) After deployment, the task is run normally on images within specification. Images outside of specification are normalised before running the task to improve performance, if possible, or rejected right away.

7. After any normalization step has taken place, all parameters are compared to the requirements, and if they are within the tolerances, they are passed to the task.

Image specification vector Figure 2 gives an example of what the image parameter requirement's vector of the application may look like. Depending on the application, the requirements vector will contain a number of image parameters. Associated with each parameter is an optimal value, for which the ML task produces the best results, and a tolerance range, for which the task still produces acceptable results. Importantly, each parameter represents an objectively quantifiable metrological property of the image that can be stated and verified.

Defining algorithm tolerances Figure 3 illustrates the method by which tolerances are established. Fig. 3a illustrates how a set of target samples (open circles) is generated from a source sample (filled circle), covering the desired parameter space (dashed line). For each specific combination of parameter values, an ML algorithm is trained to perform the desired task, and its performance measured. This allows us to establish the relation between each parameter value combination and task performance, as indicated by the red gradient in Fig. 3b, and translate our desired task performance requirement into tolerances for each specific parameter (dashed lines). Fig. 3c shows how, after deployment, each input image is checked against the specification. In some cases, out-of-specification images can be brought within specification via normalisation. In other cases, they are irrecoverably out of specification and are not expected to give results with the desired performance.

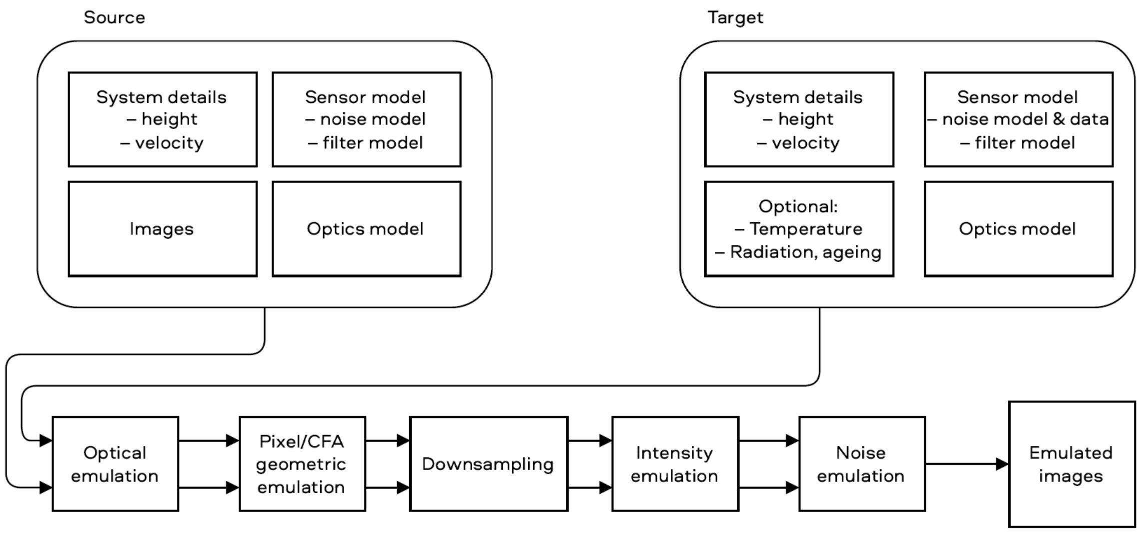


Figure 4: The emulation setup takes optical and sensor parameters of both the source (drone) and target (emulated satellite) systems as input. Images collected with a drone are processed through a physics-based pipeline in order to obtain a synthetic target image.

3 Image Emulation

The proposed method emulates the acquisition process of an imagery satellite payload starting from images collected with a drone, as shown in Fig. 5a.

As a use case, the end-to-end emulation pipeline was built to mimic the STREEGO satellite's sensor and optical acquisition properties [9, 1]. 480 raw image patches acquired with a DJI Mavic 2 Pro Drone, equipped with a Hasselblad L1D-20c camera [5], serve as our source dataset. We provided a car segmentation mask for every raw drone image, manually annotated. During the emulation process, the spatial transform is applied to the image and the mask. In the following, the satellite system will be referred to as the *target* and the drone system as the *source*. The emulation pipeline takes the optical model, the system dynamics details and the sensor properties of both the source and the target system. Fig. 4 shows a schematic representation of the end-to-end process. Given these ingredients, the emulation process follows the forward target model from the object, through the optics, and up to the sensor. The model is composed of several steps (see Fig. 5b):

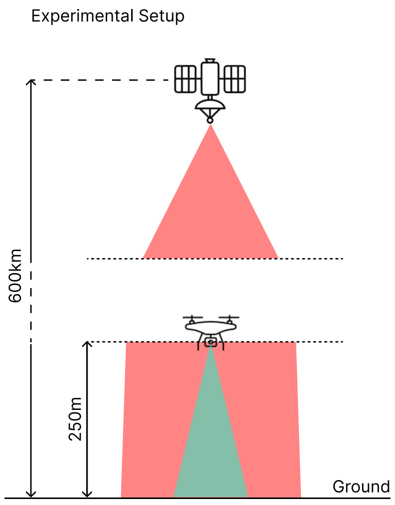
Optics As a starting point, the target optical transfer function has been simulated using the POPPY framework [6]. Since there is a substantial gap between the source and target heights, the source image is approximated with an infinitesimal spatial resolution. Therefore, the emulated point spread function depends on the height and pixel size difference between the target and the source system, and directly on the target's optical properties.

Motion blur Once the optical setup is simulated, the relative motion between satellite and Earth is introduced. In the approximation of the drone standing still with respect to the Earth, the velocity discrepancy introduces a directional motion blur across the entire image.

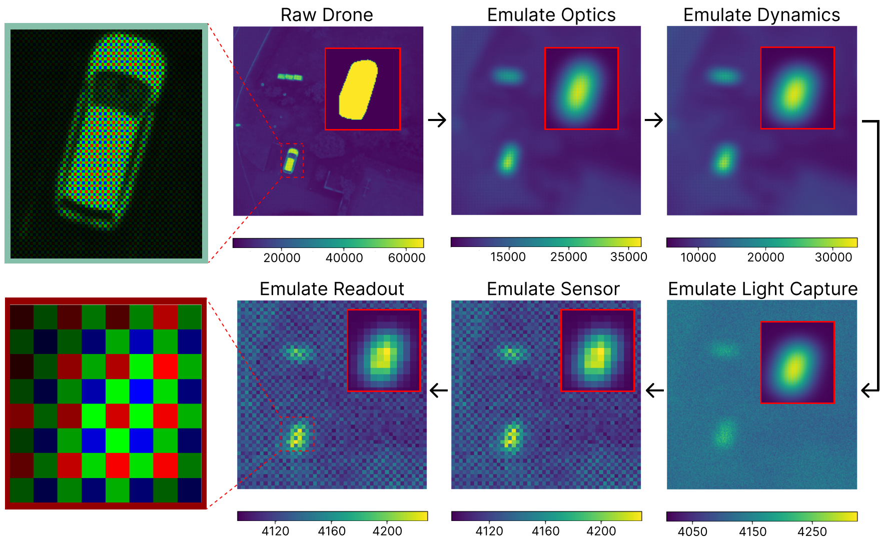
Intensity The pixel values are rescaled at this stage, according to the selected emulated exposure time and optical collection efficiency of the target system, including optical setup and sensor properties. *Sensor Geometry* The height difference corresponds to different ground sampling distances (GSDs). The area on the ground covered by the target system is broader than that of the source. Therefore, a batch of pixels in a target's image corresponds to a single pixel, depending on the GSD difference. Here, the colour filter array (CFA) sensor geometry is modelled to be compatible with this spatial discrepancy. During the sensor geometry emulation, the image can be downsampled approximately 10 to 20 times.

Sensor Readout The above image transformations introduce spatial correlation. This final step decouples pixels and guarantees a metrologically correct noise distribution. Given the target noise model estimated via calibration data, the final output image is obtained by replacing the noise distribution with a calibrated pseudo-random noise.

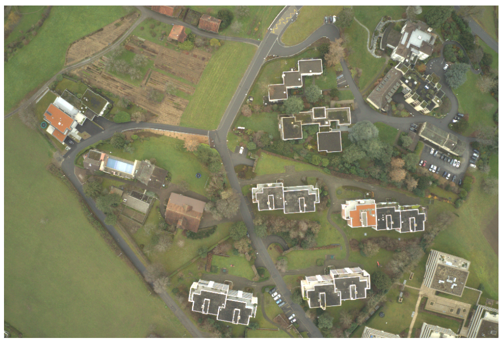
(a)



(b)



(c)



(d)

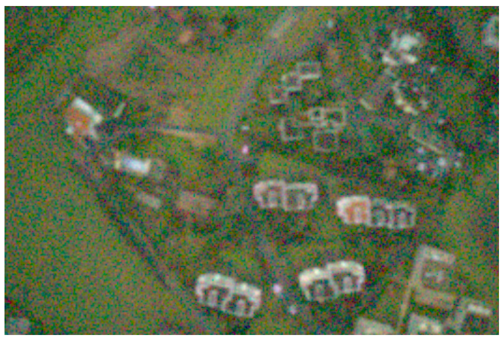
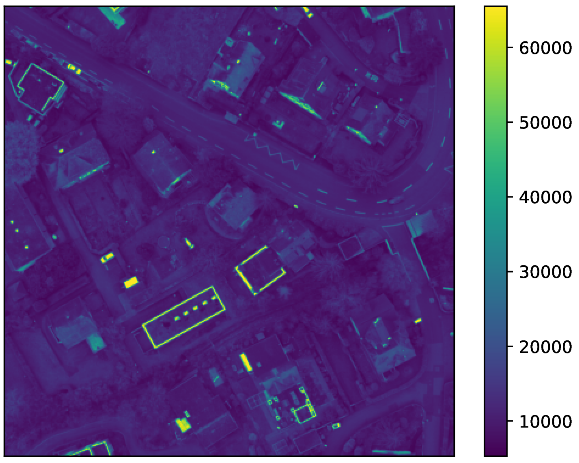


Figure 5: Emulation of satellite images. a) Schematic representation of the experimental setup. Raw images are collected with a drone flying at $250\,\mathrm{m}$. The emulated system is a payload satellite at the height of $600\,\mathrm{km}$. b) Sample false-color images and segmentation masks (insets) at various stages of the emulation pipeline. The input image from the drone (top) is sharp, highly resolved and has good contrast, and the car (Bayer-pattern zoom on the top left) is easily recognized. Throughout the pipeline (images to the right and below), the image is more and more degraded, and only a featureless blob remains of the car (Bayer-pattern zoom on the bottom left) c) Drone raw image. d) End-to-end emulated satellite image.

(a)



(b)

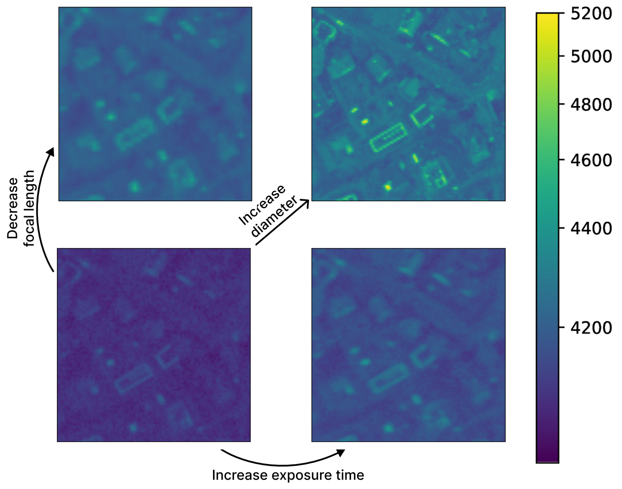


Figure 6: The images shown are subsampled to one of the green pixels in the Bayer pattern. a) Raw drone image collected at the height of $250\,\mathrm{m}$. b) Image emulation distortions are shown for different parameter configurations. (bottom-left) Image emulated with $f = 1.6\,\mathrm{m}$, diameter $d = 0.2\,\mathrm{m}$, and exposure time $t = 180\,\mu\mathrm{s}$. (top-left) Focal length decreased from $f = 1.6\,\mathrm{m}$ to $0.8\,\mathrm{m}$. (top-right) Diameter increased from $d = 0.2\,\mathrm{m}$ to $0.5\,\mathrm{m}$. (bottom-left) Exposure time increased from $t = 180\,\mu\mathrm{s}$ to $500\,\mu\mathrm{s}$.

In Fig. 5c,5d we show respectively an example of a drone raw acquisition and the final emulated image obtained from it. The emulation takes advantage of the CUDA cores of an Nvidia GeForce RTX 3090 and can be integrated into any data augmentation framework.

3.1 Emulated optical distortions

In the previous section, we described the emulation process that mimics the imaging payload of a satellite. We can control the output image quality through a set of parameters. Fixing the height and pixel size of a satellite, we investigated how the image properties are affected by the focal length of the optical system, the mirror's diameter and the exposure time. We compare the image quality for different parameter in Fig. 6b. The mirror diameter d affects the point spread function and the collection efficiency of the satellite. The diameter of the point spread function is inversely proportional to d, and the collection efficiency scales as d^2. A compromise must be made when choosing the exposure time. A longer exposure time improves the signal-to-noise ratio but introduces some motion blur due to satellite motion. As a result, a point source is stretched out along the satellite trajectory. We chose an exposure time in the range between 100 to $500\,\mu\mathrm{s}$, in accordance with existing satellite parameters. At the end of the synthesis process, the emulated sensor collects approximately 70 photons per pixel with a signal-to-noise ratio around $8\,\mathrm{dB}$.

In a typical satellite, the balance between the object magnification and the field of view depends on the focal length. Shortening focal lengths lead to a lower magnification and a broader angle of view. Since we can cover only a limited area with the drone, we estimated the corresponding downsampling required to obtain the sub-image of the broader area covered by the satellite. At the end of the emulation process, we upsampled the image with a bilinear interpolation for training purposes. Moreover, varying the focal length changes the image intensity. Given a fixed diameter, the image intensity scales inversely proportional to the square of the focal length.

4 Neural Network Tolerancing

Deploying a neural network model for critical and technical applications requires a well-defined protocol to obtain a product that is reliable and accurate. Once the model architecture is chosen, data collection and maintenance play a key role in developing industrial machine learning. While neural network calibration has been investigated in recent years [12, 2], a standard for data collection has not yet been defined. This work introduces a neural network tolerancing approach based on synthetic data. The emulation process introduced in section 3 is based on physics insight about the optical setup and calibrated sensor properties. Physics-based synthetic data provides various advantages in the data collection process

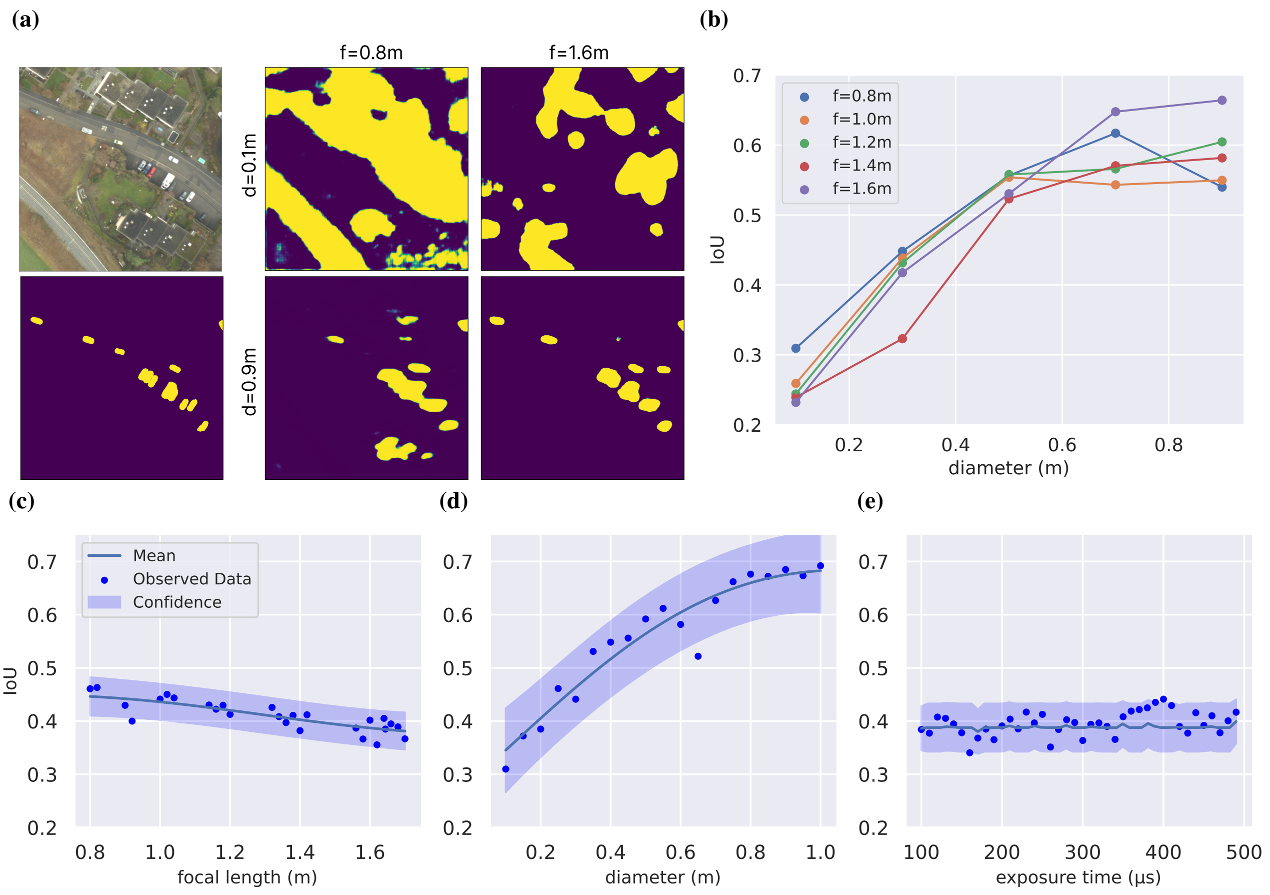


Figure 7: Neural network car segmentation results. a) (left) Unlabelled drone raw image with corresponding segmentation mask prediction. The model was trained on the original images. (right) Prediction segmentation mask for different data model configuration. The exposure time is 250 µs. b) IoU values for a large set of focal lengths and diameters. Exposure time is fixed to $t = 250\,\mu\text{s}$. IoU trend concerning c) focal length variation, fixed $d = 0.2\,\text{m}$ and $t = 180\,\mu\text{s}$; d) diameter variation, fixed $f = 1.644\,\text{m}$ and $t = 180\,\mu\text{s}$; e) exposure time variation, fixed $f = 1.644\,\text{m}$ and $d = 0.2\,\text{m}$. Mean and confidence estimated via a gaussian process. Error bars are reported as two standard deviations.

[4]. First, the model can be evaluated with different optical setups without the cost of building actual prototypes. Second, the model is guaranteed to be robust for input data that satisfies the tolerance requirements. In addition, if the model has already been trained for a specific optical setup with synthetic data it is possible to fine-tune the model to adapt to a different imaging system [3].

4.1 Segmentation on synthetic data

For a chosen model architecture [8], we evaluated model performances under different end-to-end emulated pipelines. Range of emulation parameters used are described in Fig. 7. The model was trained for 500 epochs for each emulation with a learning rate of 10^{-4}. In order to focus on the effect of a different emulation data model, hyperparameters were kept constant across all runs. We chose the same random seed for training/testing dataset splitting on every run. This allowed us to compare diverse kinds of data emulation. Trained models were evaluated with the intersection over union (IoU) metrics. The model trained on high-resolution drone raw images has an IoU score of $\sim 70\%$. With this IoU value, the model can segment most cars in the dataset, but the prediction covers a slightly bigger area around each car. An example on the drone dataset is shown in Fig. 7a. We explored the model performances by varying three emulation parameters. For every parameter, we trained different models in a physical acceptable range of values while we fixed the other remaining two parameters.

Image scale For different focal lengths, we investigated how slight changes in the image scale affect model performances.

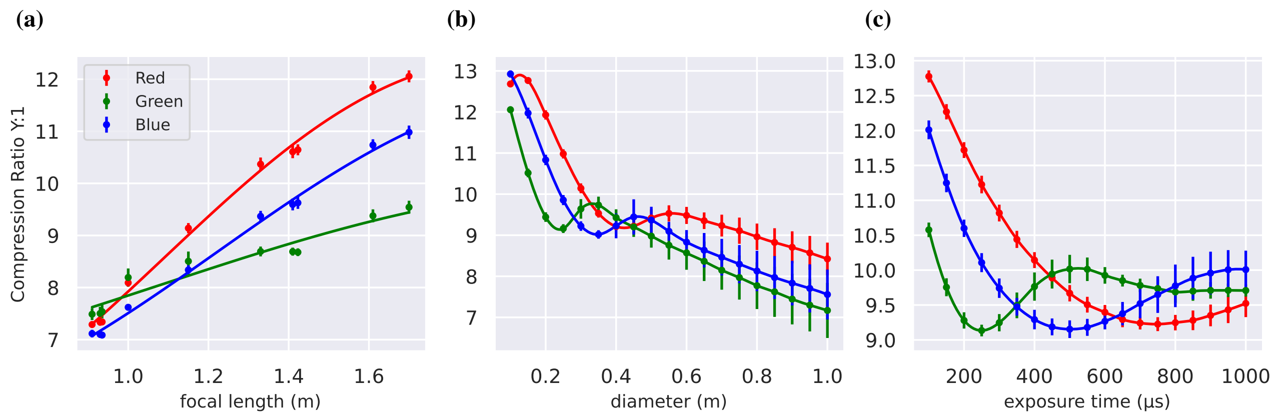


Figure 8: Compression ratio trend with respect to emulation parameters. Trend lines are used to approximate the general trend of the scatter plots. Error bars are reported as one standard deviation. Parameters investigated are: a) focal length, fixed d=0.2m, t=180μs; b) exposure time, fixed f=1.644m, d=0.2m; c) lens' diameter, fixed f=1.644m, t=180μs.

For longer focal lengths, both the GSD and the SNR decrease. We found that for shorter focal lengths, the model is less robust (see Fig. 7c). Even with a better resolution, the noise corrupts the image such that the model cannot precisely detect the car's edges. Models are trained with fixed $d = 0.2\,\mathrm{m}$ and $t = 180\,\mu\mathrm{s}$.

Spatial resolution The strongest impact of varying the mirror diameter is through the the spatial resolution. When d is increased, smaller objects are more likely to be resolved and detected. In Fig. 7d, we notice that the model can detect a darker car due to the small improvement in spatial resolution. We obtained a steadily increasing IoU as a function of mirror diameter in the investigated range (see Fig. 7d). Models are trained with fixed $f = 1.644\,\mathrm{m}$ and $t = 180\,\mu\mathrm{s}$.

Intensity Image intensity can be increased by choosing a longer exposure time, at the cost of additional motion blur. The overall segmentation performance seems to slightly improve as a function of exposure time, as shown in Fig. 7e, but results have high fluctuation. The spatial distortion induced by the motion blur does not seem to significantly affect the model. Models are trained with fixed $f = 1.644\,\mathrm{m}$ and $d = 0.2\,\mathrm{m}$.

For a complete analysis, we run cross-simulations for a sub-set of the above parameters (see Fig. 7b). As described above, the IoU depends directly on the mirror diameter and inversely on the focal length. We obtained similar trends for different exposure times.

5 Image compression performance evaluation

We use this end-to-end model, and the synthetic data it generates, to estimate the compression performance of the Jetraw algorithm [11] on realistic raw image data from earth observation satellites. As described in the introduction, Jetraw compresses the noise component of the signal in a lossy manner, and the information component in a lossless manner. This effectively amounts to "distilling" the information content into the compressed file. As the amount of information capture will change depending on the optics, motion, illumination, sensor etc, it is useful to be able to plot compression ratio with respect to these parameters. In Fig. 8a, the compression ratio is shown to increase monotonically with the focal length. This is expected, as the size of the PSF on the sensor will increase with focal length, increasing the pixel-to-pixel correlations, and reducing the per-pixel information content. Furthermore, the effects of overall light intensity (green pixels are more efficient) and wavelength-dependence (shorter wavelength have a smaller PSF) are visible in this graph.

Fig. 8b) shows the effect of increasing aperture diameter. At small apertures, increasing aperture strongly increases the captured information (decreases compression ratio) due to a rapid increase in SNR. Beyond a diameter of 0.4m, the increase in information capture is more moderate, and is due to a mix of increase in optical resolution (smaller PSF) and an increase in illumination. he effect of varying exposure time is shown in Fig. 8c). The captured information content initially increases with SNR, however, beyond 200 µs motion blur reduces the captured information content, and results in higher compression ratios.

6 Conclusions

In this work, we used the metrological calibration profile that is embedded in Jetraw images, and that can be propagated through a physical model of the imaging system to build an end-to-end model taking drone images as an input and outputting raw images emulating those from a earth observation satellite. Next, used this technology to establish the system's tolerances that are needed to achieve a specific performance on a car segmentation task. These tolerances included optical, sensor and mission parameters. Finally, we used the end-to-end model to establish the compression ratio provided by the Jetraw algorithm while varying system parameters such as aperture, exposure time and focal length. These techniques demonstrate the value of having raw data to build end-to-end models, the possibility of compressing raw data by up to 10:1, and an engineering method to enhance the robustness of a machine learning tasks.

References

[1] Mojtaba Abolghasemi and Dariush Abbasi-Moghadam. "Design and performance evaluation of the imaging payload for a remote sensing satellite". In: *Optics Laser Technology* 44.8 (2012), pp. 2418–2426. ISSN: 0030-3992. DOI: `https://doi.org/10.1016/j.optlastec.2012.04.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0030399212001600`.

[2] Chuan Guo et al. "On calibration of modern neural networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[3] Kyu-Yul Lee and Jae-Young Sim. "Cloud Removal of Satellite Images Using Convolutional Neural Network With Reliable Cloudy Image Synthesis Model". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 3581–3585. DOI: `10.1109/ICIP.2019.8803666`.

[4] Sergey I Nikolenko. *Synthetic data for deep learning*. Vol. 174. Springer, 2021.

[5] Oala, Luis and Aversa, Marco et al. "Data Models for Dataset Drift Controls in Machine Learning With Images". In: under review.

[6] Marshall Perrin et al. "POPPY: physical optics propagation in PYthon". In: *Astrophysics Source Code Library* (2016), ascl–1602.

[7] Mark Pritt and Gary Chern. "Satellite Image Classification with Deep Learning". In: *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. 2017, pp. 1–7. DOI: `10.1109/AIPR.2017.8457969`.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[9] Massimiliano Rossi et al. "Fabrication and testing of STREEGO: a compact optical payload for earth observation on small satellites". In: *Optical Systems Design 2015: Optical Fabrication, Testing, and Metrology V*. Vol. 9628. SPIE. 2015, pp. 8–16.

[10] D Andrew Rowlands. *Physics of digital photography*. IOP Publishing, 2020.

[11] Bruno Sanguinetti et al. "Validated efficient image compression for quantitative and AI applications". In: OBPDC. 2020.

[12] Yuan Zhao, Jiasi Chen, and Samet Oymak. "On the role of dataset quality and heterogeneity in model confidence". In: *arXiv preprint arXiv:2002.09831* (2020).