Data Models for Dataset Drift Controls in Machine Learning With Optical Images

Luis Oala [*] [1] [2] Marco Aversa [*] [2] [3] Gabriel Nobis [1] Kurt Willis [1] Yoan Neuenschwander [4] Michèle Buck [5] Christian Matek [6] Jérôme Extermann [4] Enrico Pomarico [4] Wojciech Samek [1] Roderick Murray-Smith [3] Christoph Clausen [2] Bruno Sanguinetti [2]

Abstract

This study addresses robustness concerns in machine learning due to dataset drift by integrating physical optics with machine learning to create explicit, differentiable data models. These models illuminate the impact of data generation on model performance and facilitate drift synthesis, precise tolerancing of model sensitivity (drift forensics), and beneficial drift creation (drift optimization). Accompanying the study are two datasets, Raw-Microscopy and Raw-Drone, available at https://github.com/aiaudit-org/raw2logit.

Camera image data has played a crucial role in advancing the field of machine learning and also features heavily in important application domains including medicine and geospatial modeling. Unfortunately, machine learning systems can fail depending on their inputs, making robustness essential to handle variations in input data [88; 75; 83]. This study illustrates the use of explicit data models for images to manage dataset drift in machine learning workflows, facilitating the creation of reliable validation protocols for critical domains. We define dataset drift through $(\boldsymbol{X}_{RAW}, Y) : \Omega \rightarrow \mathbb{R}^{H,W} \times \mathcal{Y}$, the raw sensor data generating random variable on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Raw inputs $\boldsymbol{x}_{\text{RAW}}$ undergo a data model $\boldsymbol{\Phi}_{\text{Proc}} : \mathbb{R}^{H,W} \rightarrow \mathbb{R}^{C,H,W}$, providing a processed view $\boldsymbol{v} = \boldsymbol{\Phi}_{\text{Proc}}(\boldsymbol{x}_{\text{RAW}})$ for training a task model $\boldsymbol{\Phi}_{\text{Task}} : \mathbb{R}^{C,H,W} \rightarrow \mathcal{Y}$. A different data model $\tilde{\boldsymbol{\Phi}}_{\text{Proc}}$ creates a new view $\tilde{\boldsymbol{V}} = \tilde{\boldsymbol{\Phi}}_{\text{Proc}}(\boldsymbol{X}_{RAW})$ of $\boldsymbol{X}_{RAW}$, inducing dataset drift

$$\mathcal{D}_s = \mathbb{P} \circ (\tilde{\boldsymbol{V}}, Y)^{-1} \neq \mathcal{D}_t. \tag{1}$$

Dataset drift can stem from variations in camera types or settings. Robustness validation is not just an engineering task but also mandated by quality standards [93; 27; 61]. Failure to do so has obstructed the application of machine learning technology in impactful fields [8; 85; 23; 52]. Hence, real-world robustness validation of image machine learning systems is not just an intellectual exercise but a necessity for successfully integrating machine learning research with real-world infrastructures and data. Image dataset drift validation employs either augmentation testing, which applies perturbations like Gaussian noise to images [31; 16; 54], or catalogue testing, which collects diverse camera datasets [40; 2; 55; 45]. Augmentation can produce unfaithful drift artifacts, limiting its physical accuracy [90; 96; 35]. Conversely, catalogue testing assures physically faithful samples but requires extensive data collection. The data models of images have received minimal focus in machine learning robustness studies, often treated as a black-box despite their importance [77]. This neglect is surprising, as explicit data models are crucial in optics, metrology, and industry applications [7; 73; 89; 13; 28; 64; 101; 99; 38; 74]. For a comprehensive taxonomy of related work please refer to Appendix B.4. We bridge machine learning and physical optics to generate explicit, differentiable data models for flexible, physically faithful drift controls. Our primary contributions are [1]:

① **Drift synthesis:** Physically faithful drift test case synthesis (Section 2.1). ② **Drift forensics:** Gradient propagation from the task model to the data model for precise data forensics (Section 2.2). ③ **Drift optimization:** Gradient connection adjustment between data and task model for better model performance (Section 2.3).

1. Methods

Advanced image data models necessitate raw sensor data, common in fields like microscopy, biomedicine, and autonomous vehicles. Modern digital camera systems, including smartphones, provide access to this data. We explain its procurement from optical hardware and share two datasets

[*] Equal contribution [1] Fraunhofer HHI, Berlin, Germany [2] Dotphoton AG, Zug, Switzerland [3] University of Glasgow, Glasgow, United Kingdom [4] HEPIA/HES-SO, Geneva, Switzerland [5] Klinikum rechts der Isar, Munich, Germany [6] Helmholtz Zentrum Munich, Munich, Germany. Correspondence to: Luis Oala <luis.oala@dotphoton.com>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023. Copyright 2023 by the author(s).

[1] This workshop manuscript highlights the most important components of our data models approach. For the full paper please see our publication in the *Transactions on Machine Learning Research* [60] at https://openreview.net/forum?id=I4IkGmgFJz.

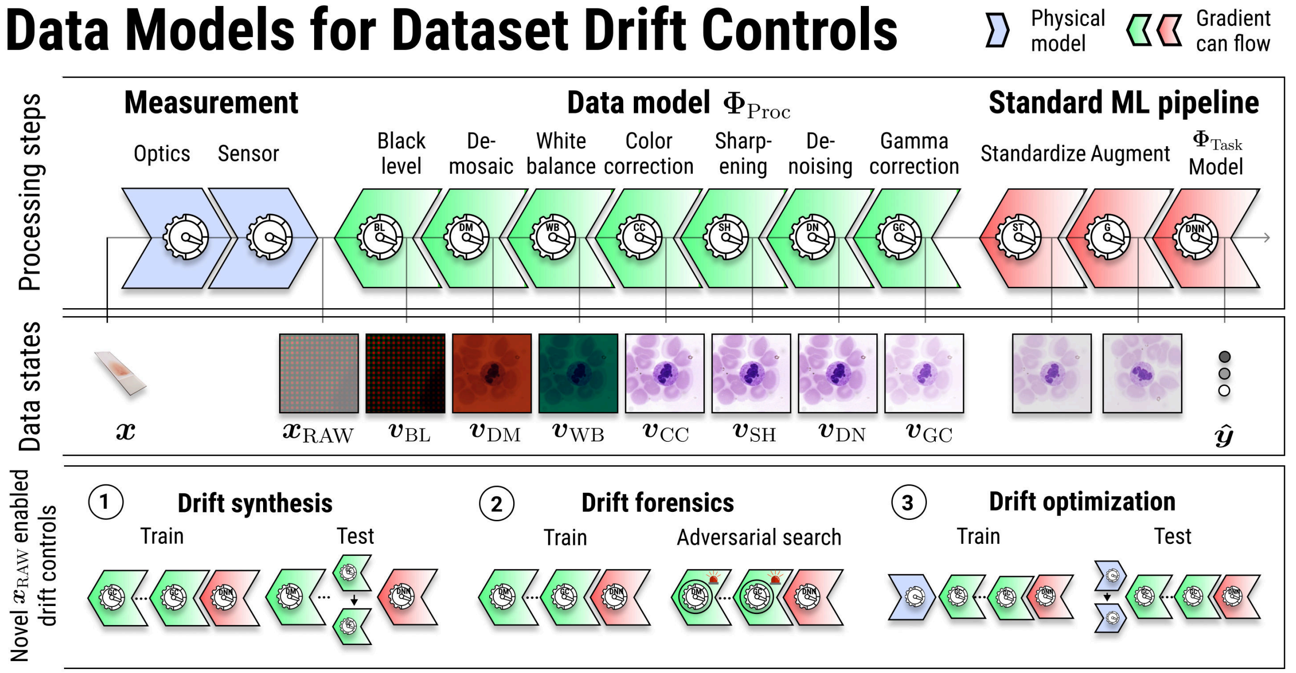

Data Models for Dataset Drift Controls
Physical model
Gradient can flow
Processing steps
Measurement
Data model Φ_{Proc}
Standard ML pipeline
Optics
Sensor
Black level
De-mosaic
White balance
Color correction
Sharp-ening
De-noising
Gamma correction
Standardize
Augment
Φ_{Task} Model
BL
DM
WB
CC
SH
DN
GC
ST
G
DNN
Data states
$\boldsymbol{x}$
$\boldsymbol{x}_{\text{RAW}}$
$\boldsymbol{v}_{\text{BL}}$
$\boldsymbol{v}_{\text{DM}}$
$\boldsymbol{v}_{\text{WB}}$
$\boldsymbol{v}_{\text{CC}}$
$\boldsymbol{v}_{\text{SH}}$
$\boldsymbol{v}_{\text{DN}}$
$\boldsymbol{v}_{\text{GC}}$
$\hat{\boldsymbol{y}}$
Novel $\boldsymbol{x}_{\text{RAW}}$ enabled drift controls
1
Drift synthesis
Train
Test
2
Drift forensics
Train
Adversarial search
3
Drift optimization
Train
Test

Figure 1: Illustration of an optical imaging pipeline and raw-enabled drift controls. The measurement process outputs raw data $\boldsymbol{x}_{\text{RAW}}$ which undergoes image signal processing (ISP) $\boldsymbol{\Phi}_{\text{Proc}}$. The processed data is consumed by a machine learning task model $\boldsymbol{\Phi}_{\text{Task}}$ that outputs $\hat{\boldsymbol{y}}$. Combining raw data, machine learning pipeline, and a differentiable data model allows drift controls: ① drift synthesis (creation of faithful drift test cases), ② drift forensics (specifying data environments to avoid), and ③ drift optimization (using task model gradient to optimize data generation).

for machine learning tasks at https://zenodo.org/record/5235536.

1.1. Raw dataset acquisition

We introduce two raw datasets, Raw-Microscopy and Raw-Drone, to fill the gap of calibrated, labelled raw data. Raw-Microscopy contains annotated blood smear microscope images, and Raw-Drone includes annotated drone car images. These diverse datasets representing classification and regression tasks underscore the importance of robustness and drift controls in high-stakes scenarios. The datasets' details are in Appendix B.5.1 and Figure 13 [24].

1.2. Data models: Image signal processing $\boldsymbol{\Phi}_{\text{Proc}}$

Image data transitions from raw state $\boldsymbol{x}_{\text{RAW}}$ to processed image $\boldsymbol{v}$ through image signal processing $\boldsymbol{\Phi}_{\text{Proc}}\colon \mathbb{R}^{H,W} \to \mathbb{R}^{C,H,W}$ [73]. Raw sensor images differ from typical machine learning input due to transformations such as correction, denoising, and sharpening that produce an RGB image $\boldsymbol{v}$ [7; 44; 26; 92]. These transformations result in varied images contributing to dataset drift, as shown in visual abstract on page 1.

Let $(\boldsymbol{X}_{RAW}, Y) : \Omega \to \mathbb{R}^{H,W} \times \mathcal{Y}$ be the raw sensor data generating random variable on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with $\mathcal{Y} = \{0,1\}^K$ for classification and $\mathcal{Y} = \{0,1\}^{H,W}$ for segmentation. Let $\boldsymbol{\Phi}_{\text{Task}} : \mathbb{R}^{C,H,W} \to \mathcal{Y}$ be the task model determined during training. The inputs that are given to the task model $\boldsymbol{\Phi}_{\text{Task}}$ are the outputs of the data model $\boldsymbol{\Phi}_{\text{Proc}}$. We distinguish between the raw sensor image $\boldsymbol{x}_{\text{RAW}}$ and a *view* $\boldsymbol{v} = \boldsymbol{\Phi}_{\text{Proc}}(\boldsymbol{x}_{\text{RAW}})$ of this image, where $\boldsymbol{\Phi}_{\text{Proc}}\colon \mathbb{R}^{H,W} \to \mathbb{R}^{C,H,W}$ models the transformation steps applied to the raw sensor image during processing.

The objective in supervised machine learning is to learn a task model $\boldsymbol{\Phi}_{\text{Task}} : \mathbb{R}^{C,H,W} \to \mathcal{Y}$ within a fixed class of task models $\mathcal{H}$ that minimizes the expected loss wrt. the loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, that is to find $\boldsymbol{\Phi}^{\star}_{\text{Task}}$ such that

$$\inf_{\boldsymbol{\Phi}_{\text{Task}} \in \mathcal{H}} \mathbb{E}[\mathcal{L}(\boldsymbol{\Phi}_{\text{Task}}(\boldsymbol{V}), Y)] \qquad (2)$$

is attained. Towards that goal, $\boldsymbol{\Phi}_{\text{Task}}$ is determined during training such that the empirical error

$$\frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left(\boldsymbol{\Phi}_{\text{Task}}(\boldsymbol{v}_n), y_n\right) \qquad (3)$$

is minimized over a sample $\mathcal{S} = ((\boldsymbol{v}_1, y_1), ..., (\boldsymbol{v}_N, y_N))$ of views. Modelling in the conventional machine learning setting begins with the image data generating random variable $(\boldsymbol{V}, Y) = (\boldsymbol{\Phi}_{\text{Proc}}(\boldsymbol{X}_{RAW}), Y)$ and the target distribution $\mathcal{D}_t = \mathbb{P} \circ (\boldsymbol{V}, Y)^{-1}$. Given a dataset drift $\mathcal{D}_s = \mathbb{P} \circ (\tilde{\boldsymbol{V}}, Y)^{-1} \neq \mathcal{D}_t$, as specified in Equation (1), without a data model we have little recourse to disentangle reasons for performance drops in $\boldsymbol{\Phi}_{\text{Task}}$. To alleviate this underspecification, an explicit data model is needed. We consider two such models in this study: a static model $\boldsymbol{\Phi}^{\text{stat}}_{\text{Proc}}$ and a parametrized model $\boldsymbol{\Phi}^{\text{para}}_{\text{Proc}}$. Following common steps in ISP, the *static data model* is defined as the composition

$$\mathbf{\Phi}^{\text{stat}}_{\text{Proc}} = \mathbf{\Phi}_{\text{GC}} \circ \mathbf{\Phi}_{\text{DN}} \circ \mathbf{\Phi}_{\text{SH}} \circ \mathbf{\Phi}_{\text{CC}} \circ \mathbf{\Phi}_{\text{WB}} \circ \mathbf{\Phi}_{\text{DM}} \circ \mathbf{\Phi}_{\text{BL}}, \quad (4)$$

mapping a raw sensor image to a RGB image. We note that other data model variations, for example by reordering or adding steps, are feasible. The static data models allow the controlled synthesis of different, physically faithful views from the same underlying raw sensor data by manually changing the configurations of the intermediate steps. Fixing the continuous features, but varying $\mathbf{\Phi}_{\text{DM}}$, $\mathbf{\Phi}_{\text{SH}}$ and $\mathbf{\Phi}_{\text{DN}}$ results in twelve different views for the configurations considered here. The *parametrized data model*, $\mathbf{\Phi}^{\text{para}}_{\text{Proc}}$, maps from a parameter space, Θ, to an RGB image from a raw sensor image. Each processing step of this model is differentiable with respect to parameters $\boldsymbol{\theta}$, enabling backpropagation of the gradient through the model for drift forensics and adjustments. This data model is differentiable in $\boldsymbol{\theta}$, fulfilling

$$\mathbf{\Phi}^{\text{stat}}_{\text{Proc}} = \mathbf{\Phi}^{\text{para}}_{\text{Proc}}\left(\cdot, \boldsymbol{\theta}^{\text{stat}}\right) \quad (5)$$

for a specific parameter set $\boldsymbol{\theta}^{\text{stat}}$ and static pipeline configuration $\mathbf{\Phi}^{\text{stat}}_{\text{Proc}}$. Using the components specified earlier, the parametrized processing model is defined as

$$\mathbf{\Phi}^{\text{para}}_{\text{Proc}} : [0,1]^{3,H,W} \times \Theta \to [0,1]^{3,H,W}, (\boldsymbol{x}_{\text{RAW}}, \boldsymbol{\theta}) \mapsto \boldsymbol{v} \quad (6)$$

by composing:

$$\boldsymbol{v} = \left(\mathbf{\Phi}^{\text{para}}_{\text{GC}}\left(\cdot, \boldsymbol{\theta_7}\right) \circ \mathbf{\Phi}^{\text{para}}_{\text{DN}}\left(\cdot, \boldsymbol{\theta_6}\right) \circ \mathbf{\Phi}^{\text{para}}_{\text{SH}}\left(\cdot, \boldsymbol{\theta_5}\right) \circ \mathbf{\Phi}^{\text{para}}_{\text{CC}}\left(\cdot, \boldsymbol{\theta_4}\right) \circ \\ \mathbf{\Phi}^{\text{para}}_{\text{WB}}\left(\cdot, \boldsymbol{\theta_3}\right) \circ \mathbf{\Phi}^{\text{para}}_{\text{DM}}\left(\cdot, \boldsymbol{\theta_2}\right) \circ \mathbf{\Phi}^{\text{para}}_{\text{BL}}\left(\cdot, \boldsymbol{\theta_1}\right) (\boldsymbol{x}_{\text{RAW}}). \quad (7)$$

The operations used above are differentiable except for the clipping operation in the GC that is *a.e.*-differentiable[2], if the set $\{0,1\}$ of non-differentiable points has measure zero. Hence, assuming that $\mathbb{P}\left((v_{\text{DN}})_{c,h,w} \in \{0,1\}\right) = 0$ holds true for the entries of $\boldsymbol{v}_{\text{DN}}$ results in an $a.e.$-differentiable processing model. We further say that $\mathbf{\Phi}^{\text{para}}_{\text{Proc}}$ is differentiable, noting that this holds only $a.e.$ under the aforementioned assumption.

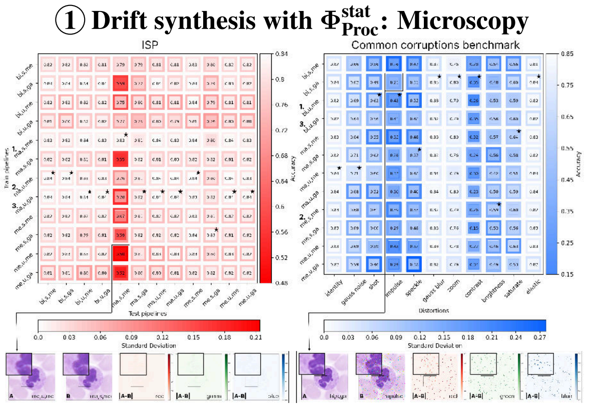

① Drift synthesis with $\mathbf{\Phi}^{\text{stat}}_{\text{Proc}}$: Microscopy

Figure 2: The figure depicts 5-fold cross-validation results for Raw-Microscopy drift synthesis experiments. Each cell indicates the average accuracy with a standard deviation border. Task models were trained on vertical axis data models and tested on the horizontal axis processed data. Numbers 1-3 left to the vertical axis rank task models by average accuracy across all test pipelines, with stars marking the training pipeline with the best performance for each test pipeline/corruption. Full ranking results are available in Appendix B.3. Top-left: Data model variations cause mild performance drops. Top-right: Comparison to a corruption benchmark shows a 13x higher average performance drop. Bottom: Visual inspection of worst-case train/test pipelines.

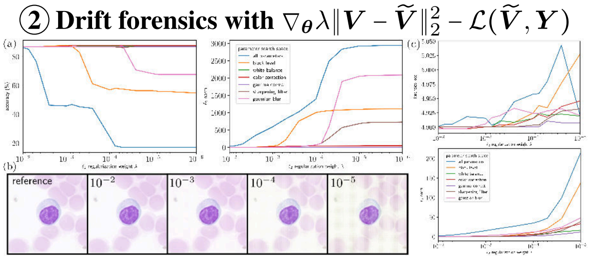

② Drift forensics with $\nabla_{\boldsymbol{\theta}} \lambda \|\boldsymbol{V} - \widetilde{\boldsymbol{V}}\|^2_2 - \mathcal{L}(\widetilde{\boldsymbol{V}}, \boldsymbol{Y})$

Figure 3: (a): The figure shows test accuracy on the Raw-Microscopy test set after 20 epochs of adversarial search in the data model for different regularization weights λ. Left plot represents various pipeline parameter selections. Right plot displays ℓ_2-norm (of processed images between the adversarially trained $\widetilde{\mathbf{\Phi}}^{\text{para}}_{\text{Proc}}$ and the default $\mathbf{\Phi}^{\text{para}}_{\text{Proc}}$) versus attained task model accuracy. (b): Processed samples from the drift forensics after 20 epochs with various regularization weights λ. (c): Similar results for Raw-Drone. Lower regularization results in a larger adversarial optimization search space. Forensics loss refers to the binary cross entropy and Dice loss used for the segmentation task model.

2. Applications

With data models, raw data and task models in place we are now able to demonstrate the advanced dataset drift controls comprising ① drift synthesis, ② modular drift forensics and ③ drift optimization.

2.1. Drift synthesis

The static data model enables controlled synthesis of processed views from a single raw dataset, aiding in model validation against device-specific drift. In our experiments, we used twelve data models, training task models on one and testing them on the other eleven. Results are mean values over 5-fold cross-validation (see Appendix B.2 for setup details). The leukocyte classification model, shown in Figure 2, demonstrates robustness to processing-induced drift, except for the (ma,s,me) configuration. The segmentation model (Figure 7) shows varied performance across data model combinations, with average performance dropping from 0.82 to 0.8 for classification and from 0.71 to 0.65 for segmentation. Drift synthesis results were compared to the Common Corruptions Benchmark [31], showing more severe performance drops for common corruptions. Physical faithfulness of test cases greatly impacts model selection decisions, with no overlap observed between top-3 training data models for classification when comparing ISP and common corruptions (refer to numbers 1-3 in Figures 2 and 7). Following ideas developed by Krikamol Muandet's [57], data models could aid targeted generalization, serving task models with the appropriate data model for each deployment environment. Finally, drift synthesis permits validation without physical measurement, but requires access to raw data and knowledge of the data model specification.

[2] *a.e.* stands for almost everywhere

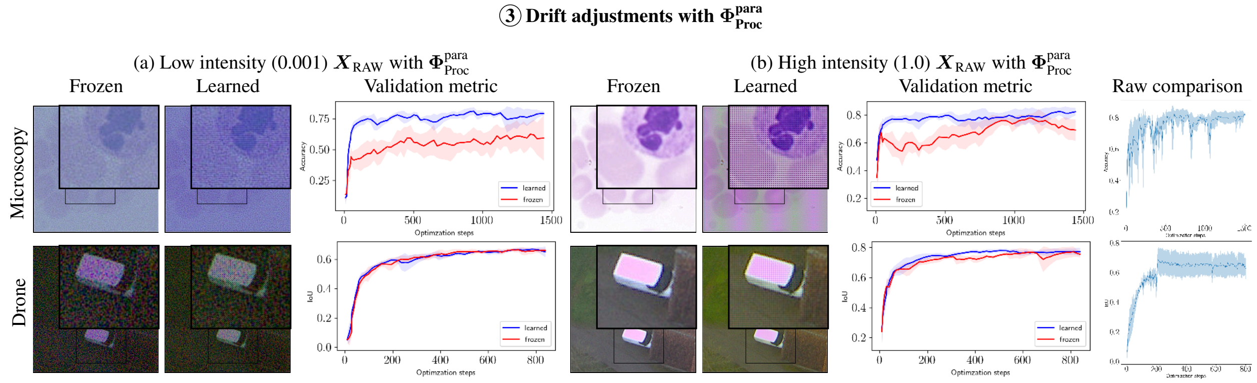


Figure 4: Low (a) and high (b) intensity images processed by a *frozen* and a *learned* pipeline. This type of drift optimization would not be possible with processed data. The plots columns three and six display the mean of validation metrics over five cross validation runs. Column seven shows additional results on raw data for comparison. Error bars are reported as one standard deviation. Optimization step 1439 and 915 correspond to epoch 60 into training.

2.2. Drift forensics

Products with machine learning components like medical devices or autonomous vehicles require clear specification of usage limitations. Our solution, $\Phi_{\text{Proc}}^{\text{para}}$, enables analysis of task model susceptibility to dataset drift using adversarial search, differentiating it from related work that applies gradient updates to individual images rather than data model parameters. We identify risky data model parameter configurations for task model operation, reflecting potential changes in data model parameters like altering camera ISPs. These parameters are kept within constraints of physical faithfulness while deteriorating task model performance. Sensitivity of the classification task model to changes in data model parameters is depicted in Figure 3. Interestingly, larger changes in the resulting RGB images don't necessarily lead to the most severe task model performance degradation. This emphasizes the need for precise data models for dataset drift validation. Practical use-cases of drift forensics include providing a forensic signature to a licensee detailing data model parameters that maintain task model performance. However, performing drift forensics requires access to raw data and data models.

2.3. Drift optimization

The previous section demonstrated how raw data and a differentiable data model can identify and test unfavorable data models. We extend this concept to optimize the data itself, creating a "beneficial drift". Two settings are considered: "learned", where both data and task model parameters are optimized, and "frozen", where only task model parameters are optimized. The "learned" model can increase accuracy by up to 25% with less variance (Figure 4 (a)). However, this was not seen in low-resolution tasks like segmentation. "Learned" models can also produce visual artifacts that may improve stability and generalization. Similarly positive results were observed under varying conditions (Figure 4 (b)). We've also investigated how parameterized data models can optimize drift under constraints, and the potential of learning directly on raw data for better task performance. Optimizing drift can enhance task model performance and adapt traditionally human-optimized imaging pipelines for ML models, beneficial in limited-resource situations. However, this doesn't work for all tasks and requires raw sensor access. The end goal may be to train on RAW data, with current methods serving as transitional solutions.

3. Discussion

In this manuscript we studied the potential of differentiable data models for images, paired with raw data, to control dataset drift. This significant challenge affects numerous machine learning disciplines. Drift synthesis enables the creation of physically faithful drift test cases, leading to less severe performance drops. This allows model selection and new ways to think about generalization. Drift synthesis could be beneficial for various domains including data synthesis and precise data models [63; 62]. Drift forensics identify and document data model limitations, enabling precision to satisfy regulatory constraints [27; 93; 59]. Drift optimization with differentiable data models enhances stability and speed in learning, which could be valuable in areas such as federated learning or domain adaptation [79; 78; 97; 10]. However, it may not work across all tasks. Lastly, raw data usage needs to be more accessible to researchers for more physically faithful data models [28; 64; 101; 99; 38; 74]. We release two raw image datasets to aid this endeavor. Better APIs to optical hardware can also help in providing more accessible raw data.

How far we can push the gradient into the real world is an interesting future direction for data modelling. Including more parts of the data acquisition hardware into the data model and consequently the machine learning optimization pipeline appears feasible [98] and represents an important next step in aligning machine learning with real world data infrastructures.

Use of Personal Data and Human Subjects The microscopy slides were purchased from a commercial lab vendor (J. Lieder GmbH & Co. KG, Ludwigsburg/Germany) who attained consent. The drone dataset does not directly relate to people. Instances with potential PIIs such as faces or license plates were removed. Full datasheet documentation following [24] can be found in Appendix B.5.2.

Negative Societal Impact Machine learning risk management, such as the drift controls, can make ML deployment possible and safer. More deployment translates to increases in automation. A net risk-benefit analysis of automation is beyond the scope of this manuscript. What we do know is that steel can be cast into ploughs and swords. We are against the use of our findings for the latter purpose.

References

[1] A. Abdelhamed, S. Lin, and M. S. Brown. A high-quality denoising dataset for smartphone cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] B. Albertina, M. Watson, C. Holback, R. Jarosz, S. Kirk, Y. Lee, and J. Lemmerman. Radiology data from the cancer genome atlas lung adenocarcinoma [tcga-luad] collection. *The Cancer Imaging Archive*, 2016.

[3] G. Aresta, T. Araújo, S. Kwok, S. S. Chennamsetty, M. Safwan, V. Alex, B. Marami, M. Prastawa, M. Chan, M. Donovan, G. Fernandez, J. Zeineh, M. Kohl, C. Walz, F. Ludwig, S. Braunewell, M. Baust, Q. D. Vu, M. N. N. To, E. Kim, J. T. Kwak, S. Galal, V. Sanchez-Freire, N. Brancati, M. Frucci, D. Riccio, Y. Wang, L. Sun, K. Ma, J. Fang, I. Kone, L. Boulmane, A. Campilho, C. Eloy, A. Polónia, and P. Aguiar. Bach: Grand challenge on breast cancer histology images. *Medical Image Analysis*, 56:122–139, 2019.

[4] V. Ayyappan, A. Chang, C. Zhang, S. K. Paidi, R. Bordett, T. Liang, I. Barman, and R. Pandey. Identification and staging of b-cell acute lymphoblastic leukemia using quantitative phase imaging and machine learning. *ACS Sensors*, 5(10):3281–3289, 2020. PMID: 33092347.

[5] A. Azulay and Y. Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20:1–25, 2019.

[6] B. Bain. Diagnosis from the blood smear. *The New England journal of medicine*, 353 5:498–507, 2005.

[7] B. E. Bayer. Color imaging array, July 20 1976. US Patent 3,971,065.

[8] E. Beede, E. Baylor, F. Hersch, A. Iurchenko, L. Wilcox, P. Ruamviboonsuk, and L. M. Vardoulakis. A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[9] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.

[10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[11] S. Biswas and S. Barma. A large-scale optical microscopy image dataset of potato tuber for deep learning based plant cell assessment. *Scientific Data*, 7(1):1–11, 2020.

[12] H. Bolhasani, E. Amjadi, M. Tabatabaeian, and S. J. Jassbi. A histopathological image dataset for grading breast invasive ductal carcinomas. *Informatics in Medicine Unlocked*, 19:100341, 2020.

[13] D. L. Bongiorno, M. Bryson, D. G. Dansereau, and S. B. Williams. Spectral characterization of COTS RGB cameras using a linear variable edge filter. page 86600N, Burlingame, California, USA, Jan. 2013.

[14] T. A. Bubba, G. Kutyniok, M. Lassas, M. März, W. Samek, S. Siltanen, and V. Srinivasan. Learning the invisible: a hybrid deep learning-shearlet framework for limited angle computed tomography. *Inverse Problems*, 35(6):064002, 2019.

[15] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3291–3300, 2018.

[16] P. Chlap, M. Hang, N. Vandenberg, J. Dowling, L. Holloway, and A. Haworth. A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology*, 65, 06 2021.

[17] J. P. Cohen, M. Luck, and S. Honari. Distribution matching losses can hallucinate features in medical image translation. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*,

pages 529–536. Springer International Publishing, 09 2018.

[18] R. Conrad and K. Narayan. Cem500k, a large-scale heterogeneous unlabeled cellular electron microscopy image dataset for deep learning. *eLife*, 10:e65894, apr 2021.

[19] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. Raise: A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, page 219–224, New York, NY, USA, 2015. Association for Computing Machinery.

[20] M. Dejean-Servières, K. Desnos, K. Abdelouahab, W. Hamidouche, L. Morin, and M. Pelcat. Study of the impact of standard image compression techniques on performance of image classification with a convolutional neural network. Research Report hal-01725126, INSA Rennes; Univ Rennes; IETR; Institut Pascal, 2017.

[21] S. Dodge and L. Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2017.

[22] DroneDeploy. Segmentation Dataset. https://github.com/dronedeploy/dd-ml-segmentation-benchmark, 2019. Accessed: 2021-09-19.

[23] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[24] T. Gebru, J. H. Morgenstern, B. Vecchione, J. W. Vaughan, H. M. Wallach, H. Daumé, and K. Crawford. Datasheets for datasets. *arXiv*, 1803.09010, 2018.

[25] K. Goel, A. Gu, Y. Li, and C. Re. Model patching: Closing the subgroup performance gap with data augmentation. In *International Conference on Learning Representations*, 2021.

[26] B. Goyal, A. Dogra, S. Agrawal, B. Sohi, and A. Sharma. Image denoising review: From classical to state-of-the-art approaches. *Information Fusion*, 55:220–244, 2020.

[27] I. S. W. Group et al. Software as a medical device (samd): Application of quality management system, 2018.

[28] HAMAMATSU. *ORCA-Flash4.0 V3 Digital CMOS camera C13440-20CU - Technical note*. HAMAMATSU.

[29] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 35(6), 2016.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[31] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[32] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[33] L. Hou, R. Gupta, J. S. Van Arnam, Y. Zhang, K. Sivalenka, D. Samaras, T. M. Kurc, and J. H. Saltz. Dataset of segmented nuclei in hematoxylin and eosin stained histopathology images of ten cancer types. *Scientific data*, 7(1):1–12, 2020.

[34] R. W. G. Hunt and M. R. Pointer. *Measuring colour*. John Wiley & Sons, 2011.

[35] R. Jaroensri, C. Biscarrat, M. Aittala, and F. Durand. Generating training data for denoising real rgb images via camera pipeline simulation. *arXiv*, 1904.08825, 2019.

[36] X. Jia, Y. Cao, D. O'Connor, J. Zhu, D. C. Tsang, B. Zou, and D. Hou. Mapping soil pollution by using drone image recognition and machine learning at an arsenic-contaminated agricultural field. *Environmental Pollution*, 270:116281, 2021.

[37] Y.-Y. Jo, Y. S. Choi, H. W. Park, J. H. Lee, H. Jung, H.-E. Kim, K. Ko, C. W. Lee, H. S. Cha, and Y. Hwangbo. Impact of image compression on deep learning-based mammogram classification. *Scientific Reports*, 11(1):1–9, 2021.

[38] A. Karpathy. Tesla ai day 2021.

[39] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 1412.6980, 2015.

[40] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga,

R. L. Phillips, I. Gao, T. Lee, E. David, I. Stavness, W. Guo, B. Earnshaw, I. Haque, S. M. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang. Wilds: A benchmark of in-the-wild distribution shifts. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR, 18–24 Jul 2021.

[41] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, T. Lee, E. David, I. Stavness, W. Guo, B. Earnshaw, I. Haque, S. M. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR, 2021.

[42] M. Kulbacki, J. Segen, W. Knieć, R. Klempous, K. Kluwak, J. Nikodem, J. Kulbacka, and A. Serester. Survey of drones for agriculture automation from planting to harvest. In *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*, pages 000353–000358. IEEE, 2018.

[43] R. D. Labati, V. Piuri, and F. Scotti. All-idb: The acute lymphoblastic leukemia image database for image processing. In *2011 18th IEEE International Conference on Image Processing*, pages 2045–2048, 2011.

[44] X. Li, B. Gunturk, and L. Zhang. Image demosaicing: A systematic survey. In *Visual Communications and Image Processing 2008*, volume 6822, page 68221J. International Society for Optics and Photonics, 2008.

[45] W. Liang and J. Zou. Metashift: A dataset of datasets for evaluating contextual distribution shifts and training conflicts. In *International Conference on Learning Representations*, 2022.

[46] Library of Congress. Camera Raw Formats (Group Description). https://www.loc.gov/preservation/digital/formats/fdd/fdd000241.shtml, Dec. 2016. Accessed: 2020-11-03.

[47] S. Longanbach, M. Miers, E. Keohane, L. Smith, and J. Walenga. Rodak's hematology: Clinical principles and applications. 2016.

[48] Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:108–119, 2020.

[49] A. Maier, H. Köstler, M. Heisig, P. Krauss, and S. H. Yang. Known operator learning and hybrid machine learning in medical imaging—a review of the past, the present, and the future. *Progress in Biomedical Engineering*, 2022.

[50] A. Maier, H. Köstler, M. Heisig, P. Krauss, and S. H. Yang. Known operator learning and hybrid machine learning in medical imaging — a review of the past, the present, and the future. *arXiv*, 2108.04543, 2021.

[51] A. K. Maier, C. Syben, B. Stimpel, T. Würfl, M. Hoffmann, F. Schebesch, W. Fu, L. Mill, L. Kling, and S. Christiansen. Learning with known operators reduces maximum error bounds. *Nature machine intelligence*, 1(8):373–380, 2019.

[52] M. Maimaitijiang, V. Sagan, P. Sidike, S. Hartling, F. Esposito, and F. B. Fritschi. Soybean yield prediction from uav using multimodal data fusion and deep learning. *Remote Sensing of Environment*, 237:111599, 2020.

[53] A. Marcu, D. Costea, V. Licaret, and M. Leordeanu. Towards automatic annotation for semantic segmentation in drone videos. *arXiv*, 1910.10026, 2019.

[54] C. Matek and C. Marr. Robustness evaluation of a convolutional neural network for the classification of single cells in acute myeloid leukemia. In *ICLR 2021, RobustML workshop*, 2020.

[55] C. Matek, S. Schwarz, C. Marr, and K. Spiekermann. A single-cell morphological dataset of leukocytes from aml patients and non-malignant controls (aml-cytomorphology_lmu). *The Cancer Imaging Archive (TCIA)*, 2019.

[56] C. Matek, S. Schwarz, K. Spiekermann, and C. Marr. Human-level recognition of blast cells in acute myeloid leukaemia with convolutional neural networks. *Nature Machine Intelligence*, 1(11):538–544, 2019.

[57] K. Muandet. Impossibility of collective intelligence, 2022.

[58] R. Nguyen, D. K. Prasad, and M. S. Brown. Raw-to-raw: Mapping between image sensor color responses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3398–3405, 2014.

[59] U. NSTC. Ensuring american leadership in automated vehicle technologies: Automated vehicles 4.0. *Las Vegas. Recuperado el*, 25:2020–02, 2020.

[60] L. Oala, M. Aversa, G. Nobis, K. Willis, Y. Neuenschwander, M. Buck, C. Matek, J. Extermann, E. Pomarico, W. Samek, R. Murray-Smith, C. Clausen, and B. Sanguinetti. Data models for dataset drift controls in machine learning with optical images. *Transactions on Machine Learning Research*, 2023.

[61] L. Oala, J. Fehr, L. Gilli, P. Balachandran, A. W. Leite, S. Calderon-Ramirez, D. X. Li, G. Nobis, E. A. M. Alvarado, G. Jaramillo-Gutierrez, et al. Ml4h auditing: From paper to practice. In *Machine learning for health*, pages 280–317. PMLR, 2020.

[62] L. Oala, C. Heiß, J. Macdonald, M. März, G. Kutyniok, and W. Samek. Detecting failure modes in image reconstructions with interval neural network uncertainty. *International Journal of Computer Assisted Radiology and Surgery*, 16(12):2089–2097, 2021.

[63] A. Oliver, A. Odena, C. Raffel, E. Cubuk, and I. Goodfellow. Realistic evaluation of semi-supervised learning algortihms. In *International conference on Learning Representations*, pages 1–15, 2018.

[64] PerkinElmer. *TotalChrom Workstation User's Guide - Volume I.* PerkinElmer.

[65] B. Phan, F. Mannan, and F. Heide. Adversarial imaging pipelines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16051–16061, 2021.

[66] E. Pomarico, C. Schmidt, F. Chays, D. Nguyen, A. Planchette, A. Tissot, A. Roux, L. Batti, C. Clausen, T. Lasser, et al. Statistical distortion of supervised learning predictions in optical microscopy induced by image compression. *Scientific reports*, 12(1):1–10, 2022.

[67] M. Poyser, A. Atapour-Abarghouei, and T. P. Breckon. On the impact of lossy image and video compression on the performance of deep convolutional neural network architectures. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2830–2837. IEEE, 2021.

[68] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *Acm Sigplan Notices*, 48(6):519–530, 2013.

[69] S. Ratnasingam. Deep camera: A fully convolutional neural network for image signal processing. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3868–3878, Los Alamitos, CA, USA, oct 2019. IEEE Computer Society.

[70] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020.

[71] M. Ronchetti. Torchradon: Fast differentiable routines for computed tomography. *arXiv*, 2009.14788, 2020.

[72] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[73] A. Rowlands. *Physics of digital photography*. IOP Publishing, 2017.

[74] G. Rudolph and U. Voelzke. Three sensor types drive autonomous vehicles, 2017.

[75] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[76] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, dec 2015.

[77] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021. Association for Computing Machinery.

[78] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek. On the byzantine robustness of clustered federated learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8861–8865. IEEE, 2020.

[79] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

[80] F. Schiffers, Z. Yu, S. Arguin, A. Maier, and Q. Ren. Synthetic fundus fluorescein angiography using deep neural networks. In A. Maier, T. M. Deserno, H. Handels, K. H. Maier-Hein, C. Palm, and T. Tolxdorff, editors, *Bildverarbeitung für die Medizin 2018*, pages 234–238, Berlin, Heidelberg, 2018. Springer Berlin Heidelberg.

[81] F. Schill. pyraw. https://github.com/fschill/pyraw, 2015.

[82] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Z. Fulé, and E. Blasch. Aerial imagery pile burn detection using deep learning: The flame dataset. *Computer Networks*, 193:108001, 2021.

[83] A. Subbaswamy, R. Adams, and S. Saria. Evaluating model robustness and stability to dataset shift. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2611–2619. PMLR, 13–15 Apr 2021.

[84] A. Subbaswamy, R. Adams, and S. Saria. Evaluating model robustness and stability to dataset shift. In *International Conference on Artificial Intelligence and Statistics*, pages 2611–2619. PMLR, 2021.

[85] S. M. Swetter. Artificial intelligence may improve melanoma detection. *Dermatology Times*, 41(9):36, 2020.

[86] C. Syben, M. Michen, B. Stimpel, S. Seitz, S. Ploner, and A. K. Maier. Pyro-nn: Python reconstruction operators in neural networks. *Medical physics*, 46(11):5110–5115, 2019.

[87] N.-S. Syu, Y.-S. Chen, and Y.-Y. Chuang. Learning deep convolutional networks for demosaicing. *arXiv*, 1802.03769, 2018.

[88] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[89] S. Tani, Y. Fukunaga, S. Shimizu, M. Fukunishi, K. Ishii, and K. Tamiya. Color Standardization Method and System for Whole Slide Imaging Based on Spectral Sensing. *Analytical Cellular Pathology*, 35(2):107–115, 2012.

[90] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[91] D. Tellez, G. Litjens, P. Bándi, W. Bulten, J.-M. Bokhorst, F. Ciompi, and J. van der Laak. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical Image Analysis*, 58:101544, 2019.

[92] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020.

[93] A. TIR57. Principles for medical device security—risk management. *Arlington, VA: Association for the Advancement of Medical Instrumentation*, 2016.

[94] E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide. Differentiable compound optics and processing pipeline optimization for end-to-end camera design. *ACM Trans. Graph.*, 40(2), June 2021.

[95] TU Graz. ICG - DroneDataset. https://www.tugraz.at/index.php?id=22387, 2019. Accessed: 2021-05-06.

[96] G. Verhoeven. It's all about the format–unleashing the power of raw aerial photography. *International Journal of Remote Sensing*, 31(8):2009–2042, 2010.

[97] H. Wang, Z. Kaplan, D. Niu, and B. Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 1698–1707, 2020.

[98] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549–555, 2022.

[99] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6), 2021.

[100] F. G. Zanjani, S. Zinger, B. Piepers, S. Mahmoudpour, P. Schelkens, and P. H. N. de With. Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images. *Journal of Medical Imaging*, 6(2):1 – 9, 2019.

[101] ZEISS. *Exporting Images and Movies in ZEN Blue*. ZEISS.

[102] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492, 2018.

[103] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (01):1–1, Oct. 2021.

A. Appendices

B. Preliminaries: a data model for images

Before proceeding with a description of the methods we use to obtain the data models $\mathbf{\Phi}_{\text{Proc}}$ in this study, let us briefly review the distinction between raw data $\boldsymbol{x}_{\text{RAW}}$, processed image $\boldsymbol{v}$ and the mechanisms $\mathbf{\Phi}_{\text{Proc}}: \mathbb{R}^{H,W} \to \mathbb{R}^{C,H,W}$ by which image data transitions between these states[3]. Image acquisition has traditionally been optimized for the human perception of a scene [34; 73]. Human eyes detect only the visible spectrum of electromagnetic radiation, hence imaging cameras in different application domains such as medical imaging or remote sensing are usually calibrated to aid the human eye perform a downstream task. This process that gives rise to optical image data, which ultimately forms the backbone for any machine learning downstream, is rarely considered in the machine learning literature. Conversely, most research to date has been conducted on processed RGB image representations. The *raw sensor image* $\boldsymbol{x}_{\text{RAW}}$ obtained from a camera differs substantially from the processed image that is used in conventional machine learning pipelines. The $\boldsymbol{x}_{\text{RAW}}$ state appears like a grey scale image with a grid structure (see $\boldsymbol{x}_{raw}$ in Figure 1). This grid is given by the Bayer color filter mosaic, which lies over sensors [7]. The final *RGB image* $\boldsymbol{v}$ is the result of a series of transformations applied to $\boldsymbol{x}_{\text{RAW}}$. For many steps in this process different possible algorithms exist. Starting from a single $\boldsymbol{x}_{\text{RAW}}$, all those possible combinations can generate an exponential number of possible images that are slightly different in terms of colors, lighting and blur - variations that contribute to dataset drift. In Figure 1 a conventional pipeline from $\boldsymbol{x}_{\text{RAW}}$ to the final RGB image $\boldsymbol{v}$ is depicted. Here, common and core transformations are considered. Note that depending on the application context it is possible to reorder or add additional steps. The symbol $\mathbf{\Phi}_{\text{i}}$ is used to denote the i^{th} transformation and $\boldsymbol{v}_{\text{i}}$ (*view*) for the output image of $\mathbf{\Phi}_{\text{i}}$. The first step of the pipeline is *black level* correction $\mathbf{\Phi}_{\text{BL}}$, which removes any constant offset. The image $\boldsymbol{v}_{\text{BL}}$ is a grey image with a Bayer filter pattern. A *demosaicing* algorithm $\mathbf{\Phi}_{\text{DM}}$ is applied to construct the full RGB color image [44]. Given $\boldsymbol{v}_{\text{DM}}$, intensities are adjusted to obtain a neutrally illuminated image $\boldsymbol{v}_{\text{WB}}$ through a *white balance* transformation $\mathbf{\Phi}_{\text{WB}}$. By considering color dependencies, a *color correction* transformation $\mathbf{\Phi}_{\text{CC}}$ is applied to balance hue and saturation of the image. Once lighting and colors are corrected, a *sharpening* algorithm $\mathbf{\Phi}_{\text{SH}}$ is applied to reduce image blurriness. This transformation can make the image appear more noisy. For this reason a *denoising* algorithm $\mathbf{\Phi}_{\text{DN}}$ is applied afterwards [26; 92]. Finally, *gamma correction*, $\mathbf{\Phi}_{\text{GC}}$, adjusts the linearity of the pixel values. For a closed form description of these transformations see Section 1.2. Compression may also take place as an additional step. It is not considered here as the input image size is already small. Furthermore, the effect of compression on downstream task model performance has been thoroughly examined before [20; 37; 100; 67; 66]. However, users of our code can add this step or reorder the sequence of steps in the modular processing object class per their use case needs[4].

B.1. Data models details

The second ingredient to this study are the data models of image processing. Image data transitions from a raw state $\boldsymbol{x}_{\text{RAW}}$ to processed image $\boldsymbol{v}$ via image signal processing $\mathbf{\Phi}_{\text{Proc}}: \mathbb{R}^{H,W} \to \mathbb{R}^{C,H,W}$ [73]. Raw sensor images ($\boldsymbol{x}_{\text{RAW}}$) from cameras are different from conventional machine learning input, appearing as a grey scale image with a grid structure due to the Bayer color filter mosaic [7]. These images undergo multiple transformations to form the final RGB image $\boldsymbol{v}$, generating numerous slightly different images, contributing to dataset drift. The process is illustrated in Figure 1, including transformations such as black level correction, demosaicing, white balance, color correction, sharpening, denoising, and gamma correction [44; 26; 92]. Compression, not discussed here, is also applicable and can be added or reordered in the modular processing object class as per use-case requirements.

Let $(\boldsymbol{X}_{RAW}, Y) : \Omega \to \mathbb{R}^{H,W} \times \mathcal{Y}$ be the raw sensor data generating random variable on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with $\mathcal{Y} = \{0,1\}^K$ for classification and $\mathcal{Y} = \{0,1\}^{H,W}$ for segmentation. Let $\mathbf{\Phi}_{\text{Task}} : \mathbb{R}^{C,H,W} \to \mathcal{Y}$ be the task model determined during training. The inputs that are given to the task model $\mathbf{\Phi}_{\text{Task}}$ are the outputs of the data model $\mathbf{\Phi}_{\text{Proc}}$. We distinguish between the raw sensor image $\boldsymbol{x}_{\text{RAW}}$ and a *view* $\boldsymbol{v} = \mathbf{\Phi}_{\text{Proc}}(\boldsymbol{x}_{\text{RAW}})$ of this image, where $\mathbf{\Phi}_{\text{Proc}}: \mathbb{R}^{H,W} \to \mathbb{R}^{C,H,W}$ models the transformation steps applied to the raw sensor image during processing.

The objective in supervised machine learning is to learn a task model $\mathbf{\Phi}_{\text{Task}} : \mathbb{R}^{C,H,W} \to \mathcal{Y}$ within a fixed class of task models $\mathcal{H}$ that minimizes the expected loss wrt. the loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, that is to find $\mathbf{\Phi}^{\star}_{\text{Task}}$ such that

$$\inf_{\mathbf{\Phi}_{\text{Task}} \in \mathcal{H}} \mathbb{E}[\mathcal{L}(\mathbf{\Phi}_{\text{Task}}(\boldsymbol{V}), Y)] \tag{8}$$

[3] We recommend [73] for a good introduction to the physics of digital optical imaging.

[4] See `pipeline_torch.py` and `pipeline_numpy.py` in our code.

Data models	Used functions		
bi,s,me	$\mathbf{\Phi}_{\text{DM}}^{\text{Bil}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{SF}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{MD}}$
bi,s,ga	$\mathbf{\Phi}_{\text{DM}}^{\text{Bil}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{SF}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{GD}}$
bi,u,me	$\mathbf{\Phi}_{\text{DM}}^{\text{Bil}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{UM}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{MD}}$
bi,u,ga	$\mathbf{\Phi}_{\text{DM}}^{\text{Bil}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{UM}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{GD}}$
me,s,me	$\mathbf{\Phi}_{\text{DM}}^{\text{Men}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{SF}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{MD}}$
me,s,ga	$\mathbf{\Phi}_{\text{DM}}^{\text{Men}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{SF}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{GD}}$
me,u,me	$\mathbf{\Phi}_{\text{DM}}^{\text{Men}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{UM}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{MD}}$
me,u,ga	$\mathbf{\Phi}_{\text{DM}}^{\text{Men}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{UM}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{GD}}$
ma,s,me	$\mathbf{\Phi}_{\text{DM}}^{\text{Mal}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{SF}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{MD}}$
ma,s,ga	$\mathbf{\Phi}_{\text{DM}}^{\text{Mal}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{SF}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{GD}}$
ma,u,me	$\mathbf{\Phi}_{\text{DM}}^{\text{Mal}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{UM}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{MD}}$
ma,u,ga	$\mathbf{\Phi}_{\text{DM}}^{\text{Mal}}$	$\mathbf{\Phi}_{\text{SH}}^{\text{UM}}$	$\mathbf{\Phi}_{\text{DN}}^{\text{GD}}$

Table 1: Abbreviations of the twelve configurations of the static data model $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$ used in the drift synthesis experiments.

is attained. Towards that goal, $\mathbf{\Phi}_{\text{Task}}$ is determined during training such that the empirical error

$$\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(\mathbf{\Phi}_{\text{Task}}(\boldsymbol{v}_n), y_n\right) \tag{9}$$

is minimized over a sample $\mathcal{S} = ((\boldsymbol{v}_1, y_1), ..., (\boldsymbol{v}_N, y_N))$ of views. Modelling in the conventional machine learning setting begins with the image data generating random variable $(\boldsymbol{V}, Y) = (\mathbf{\Phi}_{\text{Proc}}(\boldsymbol{X}_{RAW}), Y)$ and the target distribution $\mathcal{D}_t = \mathbb{P} \circ (\boldsymbol{V}, Y)^{-1}$. Given a dataset drift $\mathcal{D}_s = \mathbb{P} \circ (\tilde{\boldsymbol{V}}, Y)^{-1} \neq \mathcal{D}_t$, as specified in Equation (1), without a data model we have little recourse to disentangle reasons for performance drops in $\mathbf{\Phi}_{\text{Task}}$. To alleviate this underspecification, an explicit data model is needed. We consider two such models in this study: a static model $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$ and a parametrized model $\mathbf{\Phi}_{\text{Proc}}^{\text{para}}$.

In the following, we denote by $\boldsymbol{x}_{\text{RAW}} \in [0,1]^{H,W}$ the normalized raw image, that is a grey scale image with a Bayer filter pattern normalized by $2^{16} - 1$, i.e.

$$\boldsymbol{x}_{\text{RAW}} = \begin{bmatrix} \boldsymbol{A}_{1,1} & \cdot & \cdot & \cdot & \boldsymbol{A}_{1,\frac{W}{2}} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ \boldsymbol{A}_{\frac{H}{2},1} & \cdot & \cdot & \cdot & \boldsymbol{A}_{\frac{H}{2},\frac{W}{2}} \end{bmatrix} \quad \textit{with} \quad \boldsymbol{A}_{h,j} = \begin{bmatrix} r_{2h+1,2w+1} & g_{2h+1,2w} \\ g_{2h,2w+1} & b_{2h,2w} \end{bmatrix}, \tag{10}$$

where the values $r_{2h+1,2w+1}, g_{2h+1,2w}, g_{2h,2w+1}, b_{2h,2w}$ correspond to the values measured through the different sensors and normalized by $2^{16} - 1$. We provide here a precise description of the transformations that we consider in our static model $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$, followed by a description how to convert this static model into a differentiable, parametrized model $\mathbf{\Phi}_{\text{Proc}}^{\text{para}}$.

B.1.1. The static data model $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$

Following common steps in ISP, the *static data model* is defined as the composition

$$\mathbf{\Phi}_{\text{Proc}}^{\text{stat}} = \mathbf{\Phi}_{\text{GC}} \circ \mathbf{\Phi}_{\text{DN}} \circ \mathbf{\Phi}_{\text{SH}} \circ \mathbf{\Phi}_{\text{CC}} \circ \mathbf{\Phi}_{\text{WB}} \circ \mathbf{\Phi}_{\text{DM}} \circ \mathbf{\Phi}_{\text{BL}}, \tag{11}$$

mapping a raw sensor image to a RGB image. We note that other data model variations, for example by reordering or adding steps, are feasible. The static data models allow the controlled synthesis of different, physically faithful views from the same underlying raw sensor data by manually changing the configurations of the intermediate steps. Fixing the continuous features, but varying $\mathbf{\Phi}_{\text{DM}}$, $\mathbf{\Phi}_{\text{SH}}$ and $\mathbf{\Phi}_{\text{DN}}$ results in twelve different views for the configurations considered here. Samples for each of the twelve data models are provided in Figure 5. The individual functions of the composition $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$ can be found in Appendix B.1.3.

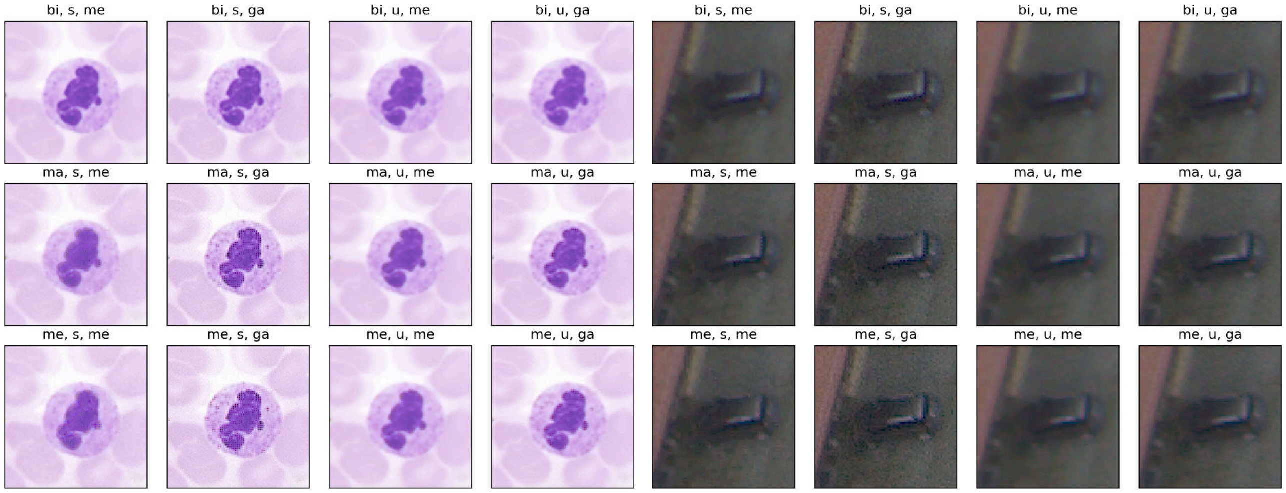


Figure 5: Samples for both datasets, Raw-Microscopy and Raw-Drone, from all twelve static data models $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$ used for the drift synthesis experiments in Section 2.1. A version with higher resolution is omitted here to save space and can instead be found in Figure 6 in the appendices.

An overview of the data model configurations and their corresponding abbreviations can be found alongside processed samples in Table 1 and Figure 5.

B.1.2. The parametrized data model $\mathbf{\Phi}_{\text{PROC}}^{\text{PARA}}$

For a fixed raw sensor image, the *parametrized data model* $\mathbf{\Phi}_{\text{Proc}}^{\text{para}}$ maps from a parameter space Θ to a RGB image. It is similar to the static data model with the notable difference that each processing step is differentiable wrt. its parameters $\boldsymbol{\theta}$. This allows for backpropagation of the gradient from the output of the task model $\mathbf{\Phi}_{\text{Task}}$ through the data model $\mathbf{\Phi}_{\text{Proc}}$ all the way back to the raw sensor image $\boldsymbol{x}_{\text{RAW}}$ to perform drift forensics and drift adjustments. Hence, we aim to design a data model $\mathbf{\Phi}_{\text{Proc}}^{\text{para}} : \mathbb{R}^{H,W} \times \Theta \to \mathbb{R}^{C,H,W}$ that is differentiable in $\boldsymbol{\theta} \in \Theta$ satisfying

$$\mathbf{\Phi}_{\text{Proc}}^{\text{stat}} = \mathbf{\Phi}_{\text{Proc}}^{\text{para}} \left(\cdot, \boldsymbol{\theta}^{\text{stat}}\right) \tag{12}$$

for some choice of parameters $\boldsymbol{\theta}^{\text{stat}}$ and some fixed configuration of the static pipeline $\mathbf{\Phi}_{\text{Proc}}^{\text{stat}}$. Using the individual functional components specified in Appendix B.1.4, we define for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_7) \in \Theta$ the parametrized processing model

$$\mathbf{\Phi}_{Proc}^{\text{para}} : [0,1]^{3,H,W} \times \Theta \to [0,1]^{3,H,W}, (\boldsymbol{x}_{\text{RAW}}, \boldsymbol{\theta}) \mapsto \boldsymbol{v} \tag{13}$$

by the composition

$$\boldsymbol{v} = \left(\mathbf{\Phi}_{\text{GC}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_7}\right) \circ \mathbf{\Phi}_{\text{DN}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_6}\right) \circ \mathbf{\Phi}_{\text{SH}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_5}\right) \circ \mathbf{\Phi}_{\text{CC}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_4}\right) \circ \mathbf{\Phi}_{\text{WB}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_3}\right) \circ \mathbf{\Phi}_{\text{DM}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_2}\right) \circ \mathbf{\Phi}_{\text{BL}}^{\text{para}}\left(\cdot, \boldsymbol{\theta_1}\right)\right)\left(\boldsymbol{x}_{\text{RAW}}\right). \tag{14}$$

The operations used above are differentiable except for the clipping operation in the GC that is *a.e.*-differentiable[5], since the set $\{0,1\}$ of non-differentiable points has measure zero. Assuming in addition that $\mathbb{P}\left(\left(v_{\text{DN}}\right)_{c,h,w} \in \{0,1\}\right) = 0$ holds true for the entries of $\boldsymbol{v}_{\text{DN}}$ results in an $a.e.$-differentiable processing model. We further say that $\mathbf{\Phi}_{\text{Proc}}^{\text{para}}$ is differentiable, noting that this holds only $a.e.$ under the aforementioned assumption.

B.1.3. Static data model $\mathbf{\Phi}_{\text{PROC}}^{\text{STAT}}$

If not stated otherwise, writing the equation $v_{c,h,w} = a_{c,h,w} + b_{c,h,w}$ defines $v_{c,h,w}$ for all $1 \leq c \leq 3$, $1 \leq h \leq H$ and $1 \leq h \leq W$.

Black level correction (BL) removes thermal noise and readout noise generated from the camera sensor. The transformation is given by

$$\mathbf{\Phi}_{\text{BL}} : [0,1]^{H,W} \to [0,1]^{H,W}, \boldsymbol{x}_{\text{RAW}} \mapsto \boldsymbol{v}_{\text{BL}}, \tag{15}$$

[5] *a.e.* stands for almost everywhere

with

$$
\begin{aligned}
(v_{\mathrm{BL}})_{2h+1,2w+1} &= x_{2h+1,2w+1} - bl_1 \\
(v_{\mathrm{BL}})_{2h,2w+1} &= x_{2h,2w+1} - bl_2 \\
(v_{\mathrm{BL}})_{2h+1,2w} &= x_{2h+1,2w} - bl_3 \\
(v_{\mathrm{BL}})_{2h,2w} &= x_{2h,2w} - bl_4,
\end{aligned}
$$

By design of $\boldsymbol{bl} \in \mathbb{R}^4$, black level correction ensures that $\boldsymbol{v}_{\mathrm{BL}}$ is again an element of $[0,1]^{H,W}$.

Demosaicing (DM) is applied to reconstruct the full RGB color image through interpolation. We use one out of the three demosaicing algorithms BayerBilinear ($\boldsymbol{\Phi}_{\mathrm{DM}}^{\mathrm{Bil}}$), Menon2007 ($\boldsymbol{\Phi}_{\mathrm{DM}}^{\mathrm{Men}}$) and Malvar2004 ($\boldsymbol{\Phi}_{\mathrm{DM}}^{\mathrm{Mal}}$) from the Python package `color-demosaicing` and denote this transformation by the map

$$\boldsymbol{\Phi}_{\mathrm{DM}} : [0,1]^{H,W} \to [0,1]^{3,H,W}, \boldsymbol{v} \mapsto \boldsymbol{v}_{\mathrm{DM}}. \tag{16}$$

White balance (WB) is applied to obtain a neutrally illuminated image. The transformation is given by

$$\boldsymbol{\Phi}_{\mathrm{WB}} : [0,1]^{3,H,W} \to [0,1]^{3,H,W}, \boldsymbol{v} \mapsto \boldsymbol{v}_{\mathrm{WB}}, \tag{17}$$

where $\boldsymbol{wb} \in [0,1]^3$ adjusts the intensities by

$$(v_{\mathrm{WB}})_{c,h,w} = wb_c \cdot (v_{\mathrm{DM}})_{c,h,w}. \tag{18}$$

Color correction (CC) balances the saturation of the image by considering color dependencies. Let $\boldsymbol{M} \in \mathbb{R}^{3,3}$ be the color matrix. The transformation is defined by

$$\boldsymbol{\Phi}_{\mathrm{CC}} : [0,1]^{3,H,W} \to \mathbb{R}^{3,H,W}, \boldsymbol{v} \mapsto \boldsymbol{v}_{\mathrm{CC}}, \tag{19}$$

where

$$\boldsymbol{v}_{\mathrm{CC}} = \begin{bmatrix} (v_{\mathrm{CC}})_{1,h,w} \\ (v_{\mathrm{CC}})_{2,h,w} \\ (v_{\mathrm{CC}})_{3,h,w} \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} (v_{\mathrm{WB}})_{1,h,w} \\ (v_{\mathrm{WB}})_{2,h,w} \\ (v_{\mathrm{WB}})_{3,h,w} \end{bmatrix}. \tag{20}$$

The entries of the resulting $\boldsymbol{v}_{\mathrm{CC}}$ are no longer restricted to $[0,1]$.

Sharpening (SH) reduces the blurriness of an image. We use the two methods sharpening filter ($\boldsymbol{\Phi}_{\mathrm{SH}}^{\mathrm{SF}}$) and unsharp masking ($\boldsymbol{\Phi}_{\mathrm{SH}}^{\mathrm{UM}}$) that are applied after a transformation of the view $\boldsymbol{v}_{\mathrm{CC}}$ to the YUV-color space. To convert the view to the YUV-color space we use the `skimage.color` function `rgb2yuv` ($\boldsymbol{\Phi}_{YUV}$). The sharpening filter

$$SF : \mathbb{R}^{3,H,W} \to \mathbb{R}^{3,H,W}, \tag{21}$$

is defined by a channel-wise convolution

$$(SF(\boldsymbol{v}))_{c,h,w} = ((\boldsymbol{v}_c \star \boldsymbol{k})_{h,w})_c \quad with \quad \boldsymbol{k} := \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{22}$$

of the view

$$\boldsymbol{v} = \boldsymbol{\Phi}_{YUV}(\boldsymbol{v}_{\mathrm{CC}}). \tag{23}$$

For unsharp masking we use the `ski.filters` function `unsharp_mask` modeled by UM. To formally define the sharpening we write

$$\boldsymbol{\Phi}_{\mathrm{SH}} : \mathbb{R}^{3,H,W} \to \mathbb{R}^{3,H,W}, \boldsymbol{v} \mapsto \boldsymbol{v}_{\mathrm{SH}} \tag{24}$$

where

$$\boldsymbol{v}_{\mathrm{SH}} = algo \circ \boldsymbol{\Phi}_{YUV}(\boldsymbol{v}_{\mathrm{CC}}) \quad with \quad algo \in \{SH, UM\}. \tag{25}$$

Denoising (DN) reduces the noise in an image that is (partly) introduced by SH and transforms the YUV-color space view back to the RGB-color space. For the latter transformation, the `skimage.color` function `yuv2rgb` ($\mathbf{\Phi}_{YUV}^{-1}$) is used. We apply one out of the two methods Gaussian denoising ($\mathbf{\Phi}_{\mathrm{DN}}^{\mathrm{GD}}$) and Median denoising ($\mathbf{\Phi}_{\mathrm{DN}}^{\mathrm{GD}}$). For Gaussian denoising, we apply a Gaussian filter (GF) with standard deviation of $\sigma = 0.5$ from the `scipy.ndimage` package. For median denoising we apply a median filter (MF of size 3 from the `scipy.ndimage` package. Formally, this reads as

$$\mathbf{\Phi}_{\mathrm{DN}} : \mathbb{R}^{3,H,W} \to \mathbb{R}^{3,H,W}, \boldsymbol{v} \mapsto \boldsymbol{v}_{\mathrm{DN}} \tag{26}$$

where

$$\boldsymbol{v}_{\mathrm{DN}} = \mathbf{\Phi}_{YUV}^{-1} \circ algo(\boldsymbol{v}_{\mathrm{SH}}) \quad with \quad algo \in \{GF, UM\}. \tag{27}$$

Gamma correction (GC) equilibrates the overall brightness of the image. First, the entries of the view $\boldsymbol{v}_{\mathrm{DN}}$ are clipped to $[0,1]$ leading to

$$(v_{CP})_{c,h,w} = (v_{\mathrm{DN}})_{c,h,w}\, \mathbb{1}_{\{0 \le (v_{\mathrm{DN}})_{c,h,w} \le 1\}} + \mathbb{1}_{\{(v_{\mathrm{DN}})_{c,h,w} > 1\}}. \tag{28}$$

Second, the brightness adjusting transformation is defined by

$$\mathbf{\Phi}_{\mathrm{GC}} : \mathbb{R}^{3,H,W} \to [0,1]^{3,H,W}, \boldsymbol{v} \mapsto \boldsymbol{v}_{\mathrm{GC}} = (\boldsymbol{v}_{CP})^{\frac{1}{\gamma}} \tag{29}$$

for some $\gamma > 0$ applied element-wise. Note that zero-clipping is necessary for $\boldsymbol{v}_{\mathrm{GC}}$ to be well-defined.

In total, we define the composition

$$\mathbf{\Phi}_{Proc}^{\mathrm{stat}} : [0,1]^{H,W} \mapsto [0,1]^{3,H,W} \tag{30}$$

of the above steps

$$\mathbf{\Phi}_{\mathrm{Proc}}^{\mathrm{stat}} := \mathbf{\Phi}_{\mathrm{GC}} \circ \mathbf{\Phi}_{\mathrm{DN}} \circ \mathbf{\Phi}_{\mathrm{SH}} \circ \mathbf{\Phi}_{\mathrm{CC}} \circ \mathbf{\Phi}_{\mathrm{WB}} \circ \mathbf{\Phi}_{\mathrm{DM}} \circ \mathbf{\Phi}_{\mathrm{BL}} \tag{31}$$

and call $\mathbf{\Phi}_{\mathrm{Proc}}^{\mathrm{stat}}$ the *static pipeline*.

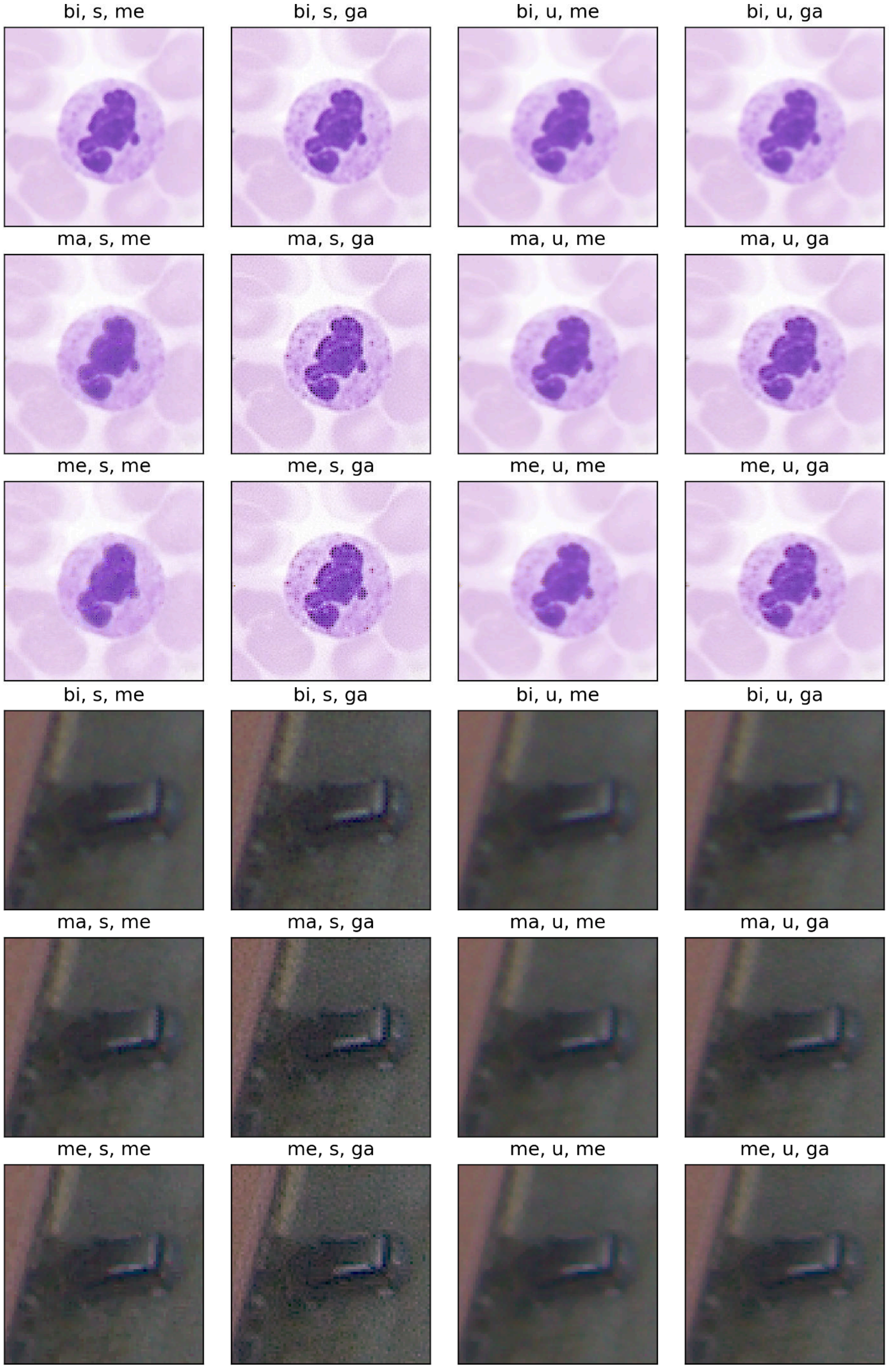


Figure 6: Samples for both datasets, Raw-Microscopy and Raw-Drone, from all twelve pipelines used in the drift synthesis experiments. The legend for abbreviations can be found in Table 1.

B.1.4. Parametrized data model $\boldsymbol{\Phi}^{\text{PARA}}_{\text{PROC}}$

Black level correction (BL) For the parametrized black level correction define the map

$$\boldsymbol{\Phi}^{\text{stat}}_{\text{BL}} : [0,1]^{H,W} \times \mathbb{R}^4 \to \mathbb{R}^{H,W}, (\boldsymbol{x}_{\text{RAW}}, \boldsymbol{\theta}_1) \mapsto \boldsymbol{v}_{\text{BL}} = \boldsymbol{\Phi}_{\text{BL}}(\boldsymbol{x}_{\text{RAW}})|_{\boldsymbol{bl}=\boldsymbol{\theta}_1}. \tag{32}$$

and set $\Theta_1 := \mathbb{R}^4$.

Demosaicing (DM) We first convert $\boldsymbol{v}_{\text{BL}}$ to a three channel image $[\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}] \in \mathbb{R}^{3,H,W}$ where the entries of $\boldsymbol{R}, \boldsymbol{G}$ and $\boldsymbol{B}$ are zero except

$$\begin{aligned} R_{2h+1,2w+1} &= \boldsymbol{v}_{BL_{2h+1,2w+1}}, \quad & B_{2h,2w} &= v_{BL_{2h,2w}}, \\ G_{2h+1,2w} &= v_{BL_{2h+1,2w}}, \quad & G_{2h,2w+1} &= v_{BL_{2h,2w+1}}. \end{aligned}$$

To parametrize $\boldsymbol{\Phi}^{\text{Bil}}_{\text{DM}}$ define the map

$$\boldsymbol{\Phi}^{\text{para}}_{\text{DM}} : [0,1]^{H,W} \times \mathbb{R}^{3,3,3} \to \mathbb{R}^{3,H,W}, (\boldsymbol{v}_{\text{BL}}, \boldsymbol{\theta}_2) \mapsto \boldsymbol{v}_{\text{DM}} \tag{33}$$

with $\boldsymbol{\theta}_2 = [\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3]$, where the kernels $\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3 \in \mathbb{R}^{3,3}$ are separately applied to each color channel resuling in

$$\begin{aligned} \boldsymbol{v}_{DM_{1,h,w}} &= (R \star \boldsymbol{k}_1)_{h,w} \\ \boldsymbol{v}_{DM_{2,h,w}} &= (G \star \boldsymbol{k}_2)_{h,w} \\ \boldsymbol{v}_{DM_{3,h,w}} &= (B \star \boldsymbol{k}_3)_{h,w}. \end{aligned}$$

The source code of BayerBilinear shows that the parameter choice

$$\boldsymbol{k}_1 = \boldsymbol{k}_3 = \begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 1 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix} \quad \textit{and} \quad \boldsymbol{k}_2 = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} \tag{34}$$

leads to

$$\boldsymbol{\Phi}^{\text{Bil}}_{\text{DM}} = \boldsymbol{\Phi}^{\text{para}}_{\text{DM}}(\cdot, \boldsymbol{\theta}_2). \tag{35}$$

Towards the definition of the parameter space set $\Theta_2 := \mathbb{R}^{3,3,3} \times \Theta_1$.

White balance (WB) For the parametrized white balance define the map

$$\boldsymbol{\Phi}^{\text{para}}_{\text{WB}} : \mathbb{R}^{3,H,W} \times \mathbb{R}^3 \to \mathbb{R}^{3,H,W}, (\boldsymbol{v}_{\text{DM}}, \boldsymbol{\theta}_3) \mapsto \boldsymbol{v}_{\text{WB}} = \boldsymbol{\Phi}_{\text{WB}}(\boldsymbol{v}_{\text{DM}})|_{\boldsymbol{wb}=\boldsymbol{\theta}_3} \tag{36}$$

and set $\Theta_3 := \mathbb{R}^3 \times \Theta_2$.

Color correction (CC) For the parametrized color correction define the map

$$\boldsymbol{\Phi}^{\text{para}}_{\text{CC}} : \mathbb{R}^{3,H,W} \times \mathbb{R}^{3,3} \to \mathbb{R}^{3,H,W}, (\boldsymbol{v}_{\text{WB}}, \boldsymbol{\theta}_4) \mapsto \boldsymbol{v}_{\text{CC}} = \boldsymbol{\Phi}_{\text{CC}}(v_{\text{WB}})|_{\boldsymbol{M}=\boldsymbol{\theta}_4} \tag{37}$$

and set $\Theta_4 := \mathbb{R}^{3,3} \times \Theta_3$

Sharpening (SH) We parametrize the sharpening filter configuration of the static pipeline, by using the entries of $\boldsymbol{k} \in \mathbb{R}^{3,3}$ defined in (22) as parameters leading to

$$\boldsymbol{\Phi}^{\text{para}}_{\text{SH}} : \mathbb{R}^{3,H,W} \times \mathbb{R}^{3,3} \to \mathbb{R}^{3,H,W}, (\boldsymbol{v}_{\text{CC}}, \boldsymbol{\theta}_5) \mapsto \boldsymbol{v}_{\text{SH}} = \boldsymbol{\Phi}_{\text{SH}}(v_{\text{CC}})|_{\boldsymbol{k}=\boldsymbol{\theta}_5} \tag{38}$$

and $\Theta_5 := \mathbb{R}^{3,3} \times \theta_4$.

Denoising (DN) We parametrize the configuration where the Gaussian denoising method is applied. Applying the Gaussian filter from `scipy.ndimage` with $\sigma = 0.5$ is equivalent to a convolution of the view in the YUV-color space with a specific $\boldsymbol{k}_{gauss} \in \mathbb{R}^{5,5}$. For the specific values of $\boldsymbol{k}_{gauss}$ see `K_BLUR` at the code of the parametrized pipeline. Therefore, to parametrize DN we define the map

$$\boldsymbol{\Phi}^{\text{para}}_{\text{DN}} : \mathbb{R}^{3,H,W} \times \mathbb{R}^{5,5} \to \mathbb{R}^{3,H,W}, (\boldsymbol{v}_{\text{SH}}, \boldsymbol{\theta}_6) \mapsto \boldsymbol{v}_{\text{DN}} = \boldsymbol{\Phi}_{\text{DN}}(v_{\text{SH}})|_{\boldsymbol{k}_{gauss}=\boldsymbol{\theta}_6} \tag{39}$$

and set $\Theta_6 := \mathbb{R}^{5,5} \times \Theta_5$

Gamma correction (GC) Define the parametrized gamma correction by

$$\boldsymbol{\Phi}^{\text{para}}_{\text{GC}} : \mathbb{R}^{3,H,W} \times \mathbb{R} \to [0,1]^{3,H,W}, (\boldsymbol{v}_{\text{DN}}, \boldsymbol{\theta_7}) \mapsto \boldsymbol{v} = \boldsymbol{v}_{\text{GC}} = \boldsymbol{\Phi}_{\text{GC}}(\boldsymbol{v}_{\text{DN}})|_{\gamma=\boldsymbol{\theta_7}}. \tag{40}$$

The following values were used to initialize $\boldsymbol{\Phi}^{\text{para}}_{\text{Proc}}$ (both "Frozen" and "Learned") in experiment Section 2.3:

```python
class ParametrizedProcessing(nn.Module):
    """Differentiable processing pipeline via torch transformations

    Args:
        camera_parameters (tuple(list), optional): applies given camera parameters in processing
        track_stages (bool, optional): whether or not to retain intermediary steps in processing
        batch_norm_output (bool, optional): adds a BatchNorm layer to the end of the processing
    """

    def __init__(self, camera_parameters=None, track_stages=False, batch_norm_output=True):
        super().__init__()
        self.stages = None
        self.buffer = None
        self.track_stages = track_stages

        if camera_parameters is None:
            camera_parameters = DEFAULT_CAMERA_PARAMS

        black_level, white_balance, colour_matrix = camera_parameters

        self.black_level = nn.Parameter(torch.as_tensor(black_level))
        self.white_balance = nn.Parameter(torch.as_tensor(white_balance).reshape(1, 3))
        self.colour_correction = nn.Parameter(torch.as_tensor(colour_matrix).reshape(3, 3))

        self.gamma_correct = nn.Parameter(torch.Tensor([2.2]))

        self.debayer = Debayer()

        self.sharpening_filter = nn.Conv2d(1, 1, kernel_size=3, padding=1, bias=False)
        self.sharpening_filter.weight.data[0][0] = K_SHARP.clone()

        self.gaussian_blur = nn.Conv2d(1, 1, kernel_size=5, padding=2, padding_mode=’reflect’, bias=False)
        self.gaussian_blur.weight.data[0][0] = K_BLUR.clone()

        self.batch_norm = nn.BatchNorm2d(3, affine=False) if batch_norm_output else None

        self.register_buffer(’M_RGB_2_YUV’, M_RGB_2_YUV.clone())
        self.register_buffer(’M_YUV_2_RGB’, M_YUV_2_RGB.clone())

        self.additive_layer = None
```

where

```python
K_G = torch.Tensor([[0, 1, 0],
                    [1, 4, 1],
                    [0, 1, 0]]) / 4

K_RB = torch.Tensor([[1, 2, 1],
                     [2, 4, 2],
                     [1, 2, 1]]) / 4

M_RGB_2_YUV = torch.Tensor([[0.299, 0.587, 0.114],
                            [-0.14714119, -0.28886916, 0.43601035],
                            [0.61497538, -0.51496512, -0.10001026]])
M_YUV_2_RGB = torch.Tensor([[1.0000000000e+00, -4.1827794561e-09, 1.1398830414e+00],
                            [1.0000000000e+00, -3.9464232326e-01, -5.8062183857e-01],
                            [1.0000000000e+00, 2.0320618153e+00, -1.2232658220e-09]])

K_BLUR = torch.Tensor([[6.9625e-08, 2.8089e-05, 2.0755e-04, 2.8089e-05, 6.9625e-08],
                       [2.8089e-05, 1.1332e-02, 8.3731e-02, 1.1332e-02, 2.8089e-05],
                       [2.0755e-04, 8.3731e-02, 6.1869e-01, 8.3731e-02, 2.0755e-04],
                       [2.8089e-05, 1.1332e-02, 8.3731e-02, 1.1332e-02, 2.8089e-05],
                       [6.9625e-08, 2.8089e-05, 2.0755e-04, 2.8089e-05, 6.9625e-08]])
K_SHARP = torch.Tensor([[0, -1, 0],
                        [-1, 5, -1],
                        [0, -1, 0]])
DEFAULT_CAMERA_PARAMS = (
    [0., 0., 0., 0.],
    [1., 1., 1.],
    [1., 0., 0., 0., 1., 0., 0., 0., 1.],
)
```

Note that the camera parameters are camera, and conversely in our case dataset, dependent and defined in the dataset classes.

B.2. Description of the task models Φ_{Task}

Two task models are employed in the experiments. For the classification task on the Raw-Microscopy dataset a 18-layer residual net (ResNet18) [30] was used as reference task model. To segment cars from the Raw-Drone dataset the convolutional neural network proposed in [72] (U-Net) was used. Both task models were trained using common data augmentations on processed views $\boldsymbol{v}$ of the image measurements to avoid naive robustness failures. A detailed description of the task models and their hyperparameters is given below.

	Classification	**Segmentation**
Φ_{Task}	ResNet18 based on [30] trained with Adam [39] for 100 epochs learning rate: 10^{-4} mini-batch size: 128	U-Net++ based on [72] trained with Adam for 100 epochs learning rate: $7.5 \cdot 10^{-5}$ mini-batch size: 12

Table 2: Summary of the training procedure for both task models.

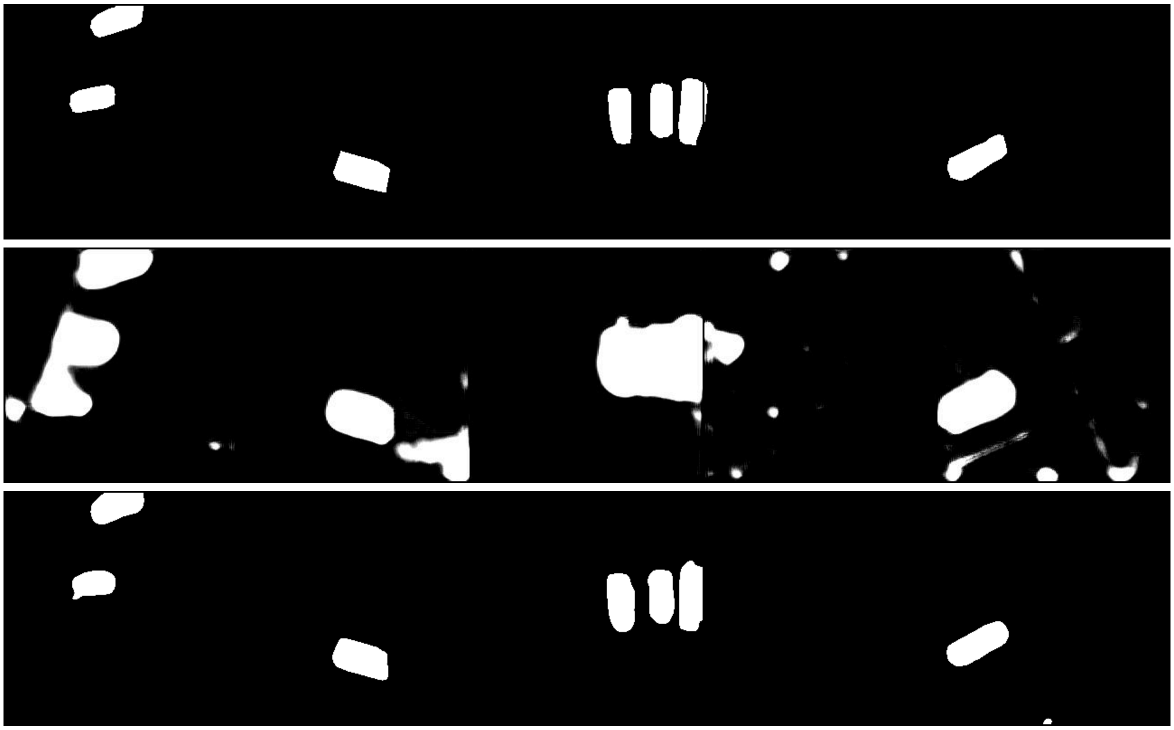


Table 3: A set of random test samples for the segmentation task under learned processing. Top row: Targets, middle row: predictions of the task model after the first epoch, last row: predictions of the task model after the last epoch.

ResNet18 This model is designed to classify images from ImageNet [76] and has therefore an output dimension of 1000. In order to use the model to classify images from Raw-Microscopy, we changed the output dimension of the fully-connected layer to nine. The model was trained for 100 epochs using pre-trained ResNet features. Hyperparameters were kept constant across all runs to isolate the effect of varying image processing pipelines. For implementation the code provided at `https://pytorch.org/hub/pytorch_vision_resnet/` was used. The model consists of 34 layers with approximately 11.2 million trainable parameters. The storage size of the model is 44.725 MB.

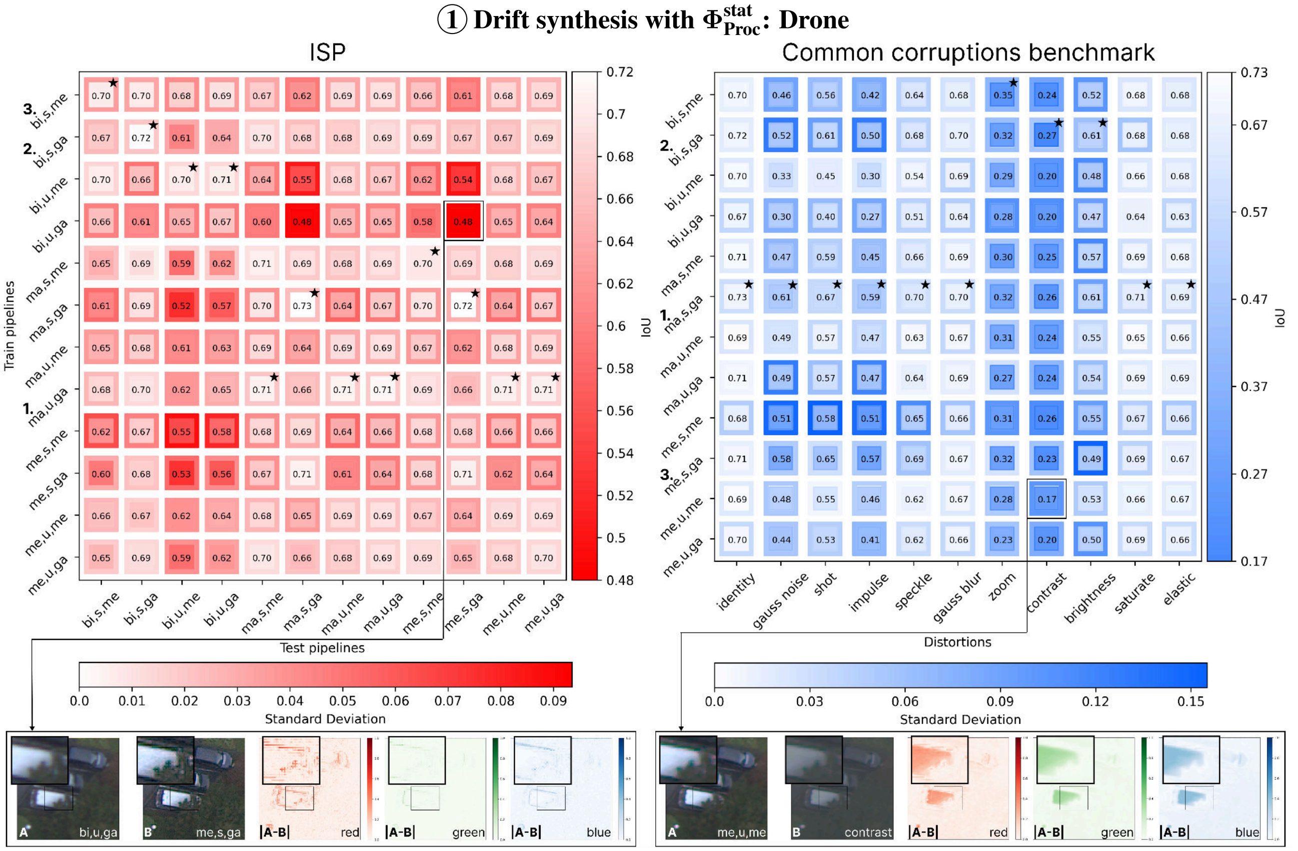
U-Net++ The model was trained for 100 epochs using pretrained ResNet features as the encoder of the U-Net++. Hyperparameters were kept constant across all runs to isolate the effect of varying image processing pipelines. For implementation we used the code provided at `https://github.com/qubvel/segmentation_models.pytorch`. The model has approximately 26.1 million trainable parameters. The storage size of the model is 104.315 MB.

Raw training In the drift optimization experiments of Section 2.3 the raw data is demosaiced using `class RawToRGB(nn.Module)` from `/processing/pipeline_torch.py` in the data model code. Then task models are tuned to raw data under the same regimes described above.

For a summary of the training procedure see Table 2.

B.3. Additional results

B.3.1. Drift synthesis



ISP

Train pipelines	bi,s,me	bi,s,ga	bi,u,me	bi,u,ga	ma,s,me	ma,s,ga	ma,u,me	ma,u,ga	me,s,me	me,s,ga	me,u,me	me,u,ga
3. bi,s,me	0.70 ★	0.70	0.68	0.69	0.67	0.62	0.69	0.69	0.66	0.61	0.68	0.69
2. bi,s,ga	0.67	0.72 ★	0.61	0.64	0.70	0.68	0.68	0.69	0.69	0.67	0.67	0.69
bi,u,me	0.70	0.66	0.70 ★	0.71 ★	0.64	0.55	0.68	0.67	0.62	0.54	0.68	0.67
bi,u,ga	0.66	0.61	0.65	0.67	0.60	0.48	0.65	0.65	0.58	0.48	0.65	0.64
ma,s,me	0.65	0.69	0.59	0.62	0.71	0.69	0.68	0.69	0.70 ★	0.69	0.68	0.69
ma,s,ga	0.61	0.69	0.52	0.57	0.70	0.73 ★	0.64	0.67	0.70	0.72 ★	0.64	0.67
ma,u,me	0.65	0.68	0.61	0.63	0.69	0.64	0.69	0.69	0.67	0.62	0.68	0.69
1. ma,u,ga	0.68	0.70	0.62	0.65	0.71 ★	0.66	0.71 ★	0.71 ★	0.69	0.66	0.71 ★	0.71 ★
me,s,me	0.62	0.67	0.55	0.58	0.68	0.69	0.64	0.66	0.68	0.68	0.66	0.66
me,s,ga	0.60	0.68	0.53	0.56	0.67	0.71	0.61	0.64	0.68	0.71	0.62	0.64
me,u,me	0.66	0.67	0.62	0.64	0.68	0.65	0.69	0.69	0.67	0.64	0.69	0.69
me,u,ga	0.65	0.69	0.59	0.62	0.70	0.66	0.68	0.69	0.69	0.65	0.68	0.70

Common corruptions benchmark

	identity	gauss noise	shot	impulse	speckle	gauss blur	zoom	contrast	brightness	saturate	elastic
bi,s,me	0.70	0.46	0.56	0.42	0.64	0.68	0.35 ★	0.24	0.52	0.68	0.68
2. bi,s,ga	0.72	0.52	0.61	0.50	0.68	0.70	0.32	0.27 ★	0.61 ★	0.68	0.68
bi,u,me	0.70	0.33	0.45	0.30	0.54	0.69	0.29	0.20	0.48	0.66	0.68
bi,u,ga	0.67	0.30	0.40	0.27	0.51	0.64	0.28	0.20	0.47	0.64	0.63
ma,s,me	0.71	0.47	0.59	0.45	0.66	0.69	0.30	0.25	0.57	0.69	0.68
1. ma,s,ga	0.73 ★	0.61 ★	0.67 ★	0.59 ★	0.70 ★	0.70 ★	0.32	0.26	0.61	0.71 ★	0.69 ★
ma,u,me	0.69	0.49	0.57	0.47	0.63	0.67	0.31	0.24	0.55	0.65	0.66
ma,u,ga	0.71	0.49	0.57	0.47	0.64	0.69	0.27	0.24	0.54	0.69	0.69
me,s,me	0.68	0.51	0.58	0.51	0.65	0.66	0.31	0.26	0.55	0.67	0.66
3. me,s,ga	0.71	0.58	0.65	0.57	0.69	0.67	0.32	0.23	0.49	0.69	0.67
me,u,me	0.69	0.48	0.55	0.46	0.62	0.67	0.28	0.17	0.53	0.66	0.67
me,u,ga	0.70	0.44	0.53	0.41	0.62	0.66	0.23	0.20	0.50	0.69	0.66

Figure 7: 5-fold cross-validation results of the Raw-Drone drift synthesis experiments. Each cell contains the average IoU with a color coded border for the standard deviation. Task models were trained on the data model on the vertical axis and then tested on processed data as indicated on the horizontal axis. Numbers 1-3 left to the vertical axis denote the ranking of task models according to their average IoU across all test pipelines respective corruptions. Stars denote the train pipeline under which the task model performed best on the respective test pipeline/corruption. Full ranking results can be found in Tables 5, 8 and 9 of Appendix B.3. Left: Varying the data model leads to mixed performance drops. Diagonal is $\mathbf{\Phi}_{\text{Proc}} = \tilde{\mathbf{\Phi}}_{\text{Proc}}$. Right: Comparison to the corruption benchmark at medium severity (level 3). The average performance drop is more than four times higher compared to data model variations. First column is $\mathbf{\Phi}_{\text{Proc}} = \tilde{\mathbf{\Phi}}_{\text{Proc}}$. Bottom: Visual inspection of worst case (globally worst scoring) train/test pipelines.

	Microscopy Average accuracy	**Drone** Average IoU
Learned (low)	0.75 ± 0.09	0.59 ± 0.05
Frozen (low)	0.54 ± 0.21	0.59 ± 0.05
Learned (high)	0.78 ± 0.08	0.74 ± 0.04
Frozen (high)	0.67 ± 0.14	0.71 ± 0.05
Direct raw	0.75 ± 0.07	0.60 ± 0.07

Table 4: Tabular summary of the drift optimization results. The average accuracy and standard deviations over cross-validation runs and training steps are displayed, summarizing both the stability and converge trajectory for each setting.

	Microscopy-ISP		Microscopy-CC		Drone-ISP		Drone-CC	
Rank	Train pipeline	Avg. score	Train pipeline	Avg. score	Train pipeline	Avg. score	Train pipeline	Avg. score
1	ma,s,me	0.83	bi,u,me	0.63	ma,u,ga	0.68	ma,s,ga	0.60
2	ma,u,me	0.83	me,s,me	0.63	bi,s,ga	0.68	bi,s,ga	0.57
3	ma,u,ga	0.82	bi,u,ga	0.62	bi,s,me	0.67	me,s,ga	0.57
4	bi,s,me	0.81	ma,s,me	0.62	ma,s,me	0.67	ma,s,me	0.55
5	bi,u,me	0.81	me,u,me	0.62	me,u,ga	0.67	me,s,me	0.55
6	me,s,me	0.81	ma,s,ga	0.62	me,u,me	0.67	ma,u,ga	0.55
7	bi,s,ga	0.81	ma,u,me	0.61	ma,u,me	0.66	bi,s,me	0.54
8	me,s,ga	0.80	me,s,ga	0.60	ma,s,ga	0.66	ma,u,me	0.54
9	me,u,me	0.80	bi,s,me	0.59	bi,u,me	0.65	me,u,me	0.53
10	ma,s,ga	0.80	ma,u,ga	0.59	me,s,me	0.65	me,u,ga	0.51
11	bi,u,ga	0.79	bi,s,ga	0.58	me,s,ga	0.64	bi,u,me	0.48
12	me,u,ga	0.79	me,u,ga	0.58	bi,u,ga	0.61	bi,u,ga	0.46

Table 5: Rankings of task models from Section 2.1 trained on different data models (columns 2, 4, 6, 8) according to their average accuracy or IoU (columns 3, 5, 7, 9) across all test pipelines respective corruptions. ISP corresponds to drift synthesis with physically faithful data models, CC corresponds to common corruptions.

	Microscopy-ISP											
Rank	bi,s,me	bi,s,ga	bi,u,me	bi,u,ga	ma,s,me	ma,s,ga	ma,u,me	ma,u,ga	me,s,me	me,s,ga	me,u,me	me,u,ga
1	ma,u,me	ma,u,me	ma,u,ga	ma,u,ga	ma,s,me	ma,u,ga	ma,u,ga	ma,u,ga	ma,u,me	me,s,ga	ma,u,ga	ma,u,ga
2	ma,u,ga	ma,u,ga	bi,s,ga	bi,s,ga	bi,s,me	me,s,ga	ma,s,me	ma,u,me	ma,s,me	ma,u,ga	ma,u,me	ma,u,me
3	bi,s,ga	bi,s,ga	ma,s,me	ma,s,me	bi,u,ga	ma,s,ga	ma,u,me	ma,s,me	bi,s,ga	ma,s,ga	ma,s,me	ma,s,me
4	ma,s,me	ma,s,me	ma,u,me	ma,u,me	ma,u,me	ma,s,me	bi,s,ga	me,u,me	me,s,ga	me,u,ga	me,u,me	me,u,me
5	bi,s,me	bi,u,me	me,u,me	me,u,me	bi,u,me	ma,u,me	me,u,me	ma,s,ga	bi,u,me	me,s,me	bi,s,ga	bi,s,ga
6	bi,u,me	me,u,me	bi,u,me	bi,u,me	ma,u,ga	me,s,me	me,s,ga	bi,s,ga	ma,u,ga	ma,u,me	me,u,ga	me,u,ga
7	me,s,me	bi,s,me	bi,s,me	me,s,me	me,s,me	me,u,me	me,s,me	me,s,ga	me,u,me	ma,s,me	me,s,me	me,s,me
8	me,s,ga	me,s,me	me,s,me	bi,u,ga	bi,s,ga	bi,u,me	ma,s,ga	me,s,me	me,s,me	me,u,me	bi,s,me	bi,s,me
9	me,u,me	me,s,ga	bi,u,ga	bi,s,me	me,s,ga	me,u,ga	bi,u,me	bi,u,me	bi,s,me	bi,s,me	me,s,ga	me,s,ga
10	ma,s,ga	ma,s,ga	ma,s,ga	ma,s,ga	ma,s,ga	bi,s,me	bi,s,me	bi,s,me	ma,s,ga	bi,s,ga	ma,s,ga	ma,s,ga
11	bi,u,ga	me,u,ga	me,u,ga	me,s,ga	me,u,ga	bi,s,ga	me,u,ga	me,u,ga	me,u,ga	bi,u,me	bi,u,me	bi,u,me
12	me,u,ga	bi,u,ga	me,s,ga	me,u,ga	me,u,me	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga

Table 6: Ranking of task models from Section 2.1 trained under different train pipelines (rows) for each individual test pipeline (columns 2 - 13).

	Microscopy-CC										
Rank	identity	gauss noise	shot	impulse	speckle	gauss blur	zoom	contrast	brightness	saturate	elastic
1	ma,u,me	ma,u,me	bi,u,me	bi,u,me	ma,s,ga	bi,s,ga	bi,s,ga	bi,s,ga	me,s,me	ma,s,me	bi,s,ga
2	ma,u,ga	ma,s,ga	ma,s,ga	me,u,me	bi,u,me	ma,u,me	ma,u,ga	bi,u,ga	ma,s,me	me,u,me	ma,u,ga
3	bi,s,ga	me,u,me	me,s,me	bi,u,ga	me,s,me	ma,u,ga	ma,s,me	me,u,ga	bi,u,ga	me,s,me	ma,u,me
4	me,s,me	me,s,ga	ma,u,me	me,s,me	me,u,me	bi,u,me	ma,u,me	ma,s,me	ma,s,ga	bi,u,ga	ma,s,me
5	ma,s,me	bi,u,me	me,s,ga	ma,s,me	bi,u,ga	me,u,me	bi,u,me	ma,u,me	bi,s,me	bi,s,ga	me,u,me
6	me,u,me	ma,u,ga	me,u,me	ma,u,me	ma,s,me	ma,s,me	me,s,me	bi,s,me	bi,u,me	bi,u,me	me,s,ga
7	me,s,ga	me,s,me	bi,s,me	ma,u,ga	ma,u,me	me,s,ga	bi,u,ga	bi,u,me	me,s,ga	ma,u,ga	me,s,me
8	bi,u,me	bi,s,me	bi,u,ga	me,s,ga	me,s,ga	ma,s,ga	me,u,ga	me,s,me	ma,u,ga	ma,s,ga	bi,u,ga
9	bi,u,ga	ma,s,me	ma,s,me	me,u,ga	bi,s,me	me,s,me	me,u,me	ma,s,ga	me,u,ga	bi,s,me	bi,u,me
10	ma,s,ga	bi,u,ga	ma,u,ga	ma,s,ga	ma,u,ga	bi,u,ga	me,s,ga	ma,u,ga	bi,s,ga	me,s,ga	ma,s,ga
11	bi,s,me	bi,s,ga	bi,s,ga	bi,s,me	me,u,ga	bi,s,me	ma,s,ga	me,u,me	me,u,me	me,u,ga	me,u,ga
12	me,u,ga	me,u,ga	me,u,ga	bi,s,ga	bi,s,ga	me,u,ga	bi,s,me	me,s,ga	ma,u,me	ma,u,me	bi,s,me

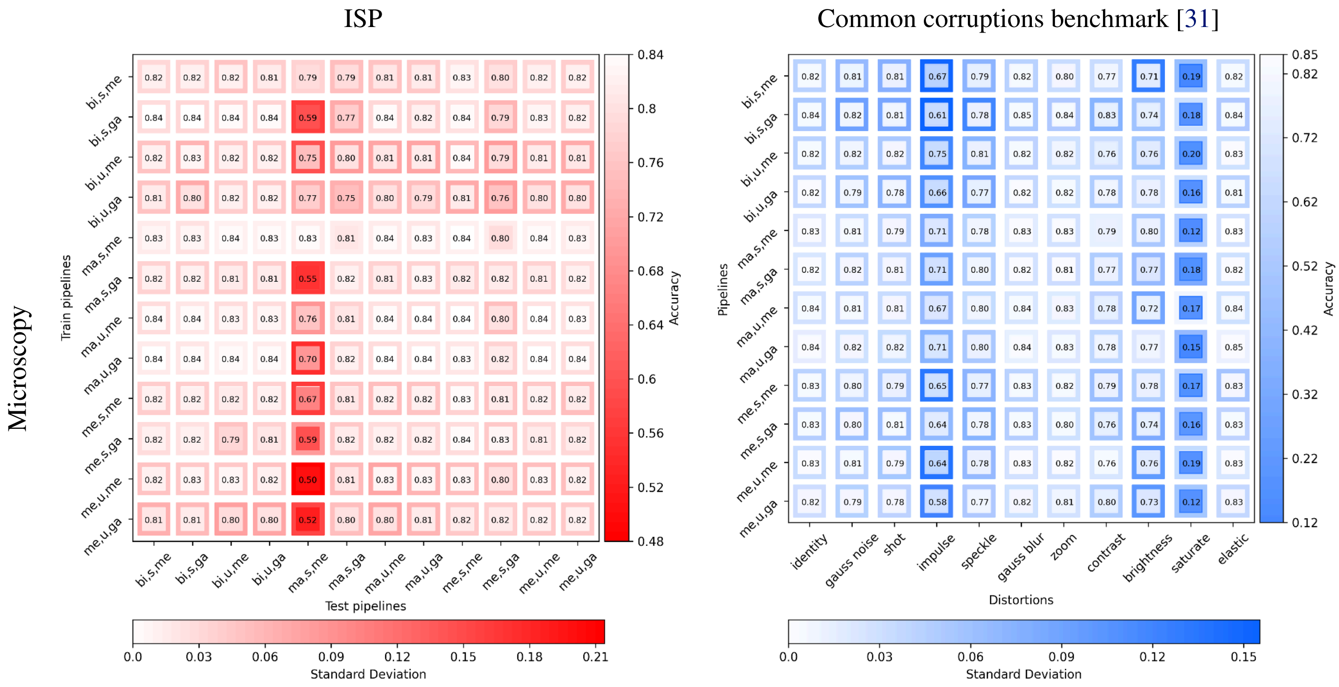
Table 7: Ranking of task models from Section 2.1 trained under different train pipelines (rows) for each individual test corruptions (columns 2 - 12).

	Drone-ISP											
Rank	bi,s,me	bi,s,ga	bi,u,me	bi,u,ga	ma,s,me	ma,s,ga	ma,u,me	ma,u,ga	me,s,me	me,s,ga	me,u,me	me,u,ga
1	bi,s,me	bi,s,ga	bi,u,me	bi,u,me	ma,u,ga	ma,s,ga	ma,u,ga	ma,u,ga	ma,s,me	ma,s,ga	ma,u,ga	ma,u,ga
2	bi,u,me	bi,s,me	bi,s,me	bi,s,me	ma,s,me	me,s,ga	me,u,me	me,u,me	ma,s,ga	me,s,ga	me,u,me	me,u,ga
3	ma,u,ga	ma,u,ga	bi,u,ga	bi,u,ga	bi,s,ga	ma,s,me	ma,u,me	ma,u,me	ma,u,ga	ma,s,me	ma,s,me	me,u,me
4	bi,s,ga	ma,s,me	ma,u,ga	ma,u,ga	me,u,ga	me,s,me	bi,s,me	bi,s,me	bi,s,ga	me,s,me	me,u,ga	ma,s,me
5	me,u,me	me,u,ga	me,u,me	me,u,me	ma,s,ga	bi,s,ga	ma,s,me	ma,s,me	me,u,ga	bi,s,ga	ma,u,me	ma,u,me
6	bi,u,ga	ma,s,ga	bi,s,ga	bi,s,ga	ma,u,me	ma,u,ga	bi,s,ga	bi,s,ga	me,s,me	ma,u,ga	bi,s,me	bi,s,me
7	ma,s,me	ma,u,me	ma,u,me	ma,u,me	me,u,me	me,u,ga	me,u,ga	me,u,ga	me,s,ga	me,u,ga	bi,u,me	bi,s,ga
8	me,u,ga	me,s,ga	ma,s,me	ma,s,me	me,s,me	me,u,me	bi,u,me	bi,u,me	me,u,me	me,u,me	bi,s,ga	bi,u,me
9	ma,u,me	me,u,me	me,u,ga	me,u,ga	bi,s,me	ma,u,me	bi,u,ga	ma,s,ga	ma,u,me	ma,u,me	me,s,me	ma,s,ga
10	me,s,me	me,s,me	me,s,me	me,s,me	me,s,ga	bi,s,me	ma,s,ga	me,s,me	bi,s,me	bi,s,me	bi,u,ga	me,s,me
11	ma,s,ga	bi,u,me	me,s,ga	ma,s,ga	bi,u,me	bi,u,me	me,s,me	bi,u,ga	bi,u,me	bi,u,me	ma,s,ga	bi,u,ga
12	me,s,ga	bi,u,ga	ma,s,ga	me,s,ga	bi,u,ga	bi,u,ga	me,s,ga	me,s,ga	bi,u,ga	bi,u,ga	me,s,ga	me,s,ga

Table 8: Ranking of task models from Section 2.1 trained under different train pipelines (rows) for each individual test pipeline (columns 2 - 13).

	Drone-CC										
Rank	identity	gauss noise	shot	impulse	speckle	gauss blur	zoom	contrast	brightness	saturate	elastic
1	ma,s,ga	ma,s,ga	ma,s,ga	ma,s,ga	ma,s,ga	ma,s,ga	bi,s,me	bi,s,ga	bi,s,ga	ma,s,ga	ma,s,ga
2	bi,s,ga	me,s,ga	me,s,ga	me,s,ga	me,s,ga	bi,s,ga	ma,s,ga	ma,s,ga	ma,s,ga	ma,s,me	ma,u,ga
3	me,s,ga	bi,s,ga	bi,s,ga	me,s,me	bi,s,ga	ma,s,me	bi,s,ga	me,s,me	ma,s,me	ma,u,ga	ma,s,me
4	ma,s,me	me,s,me	ma,s,me	bi,s,ga	ma,s,me	ma,u,ga	me,s,ga	ma,s,me	me,s,me	me,u,ga	bi,s,ga
5	ma,u,ga	ma,u,ga	me,s,me	ma,u,ga	me,s,me	bi,u,me	ma,u,me	bi,s,me	ma,u,me	me,s,ga	bi,s,me
6	bi,s,me	ma,u,me	ma,u,ga	ma,u,me	ma,u,ga	bi,s,me	me,s,me	ma,u,me	ma,u,ga	bi,s,ga	bi,u,me
7	me,u,ga	me,u,me	ma,u,me	me,u,me	bi,s,me	me,s,ga	ma,s,me	ma,u,ga	me,u,me	bi,s,me	me,s,ga
8	bi,u,me	ma,s,me	bi,s,me	ma,s,me	ma,u,me	ma,u,me	bi,u,me	me,s,ga	bi,s,me	me,s,me	me,u,me
9	ma,u,me	bi,s,me	me,u,me	bi,s,me	me,u,me	me,u,me	me,u,me	bi,u,me	me,u,ga	me,u,me	me,u,ga
10	me,u,me	me,u,ga	me,u,ga	me,u,ga	me,u,ga	me,s,me	bi,u,ga	bi,u,ga	me,s,ga	bi,u,me	me,s,me
11	me,s,me	bi,u,me	bi,u,me	bi,u,me	bi,u,me	me,u,ga	ma,u,ga	me,u,ga	bi,u,me	ma,u,me	ma,u,me
12	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga	bi,u,ga	me,u,ga	me,u,me	bi,u,ga	bi,u,ga	bi,u,ga

Table 9: Ranking of task models from Section 2.1 trained under different train pipelines (rows) for each individual test corruptions (columns 2 - 12).


ISP

Common corruptions benchmark [31]

Figure 8: Experiment from Section 2.1 with weak severity (level 1) for the Common corruptions benchmark.

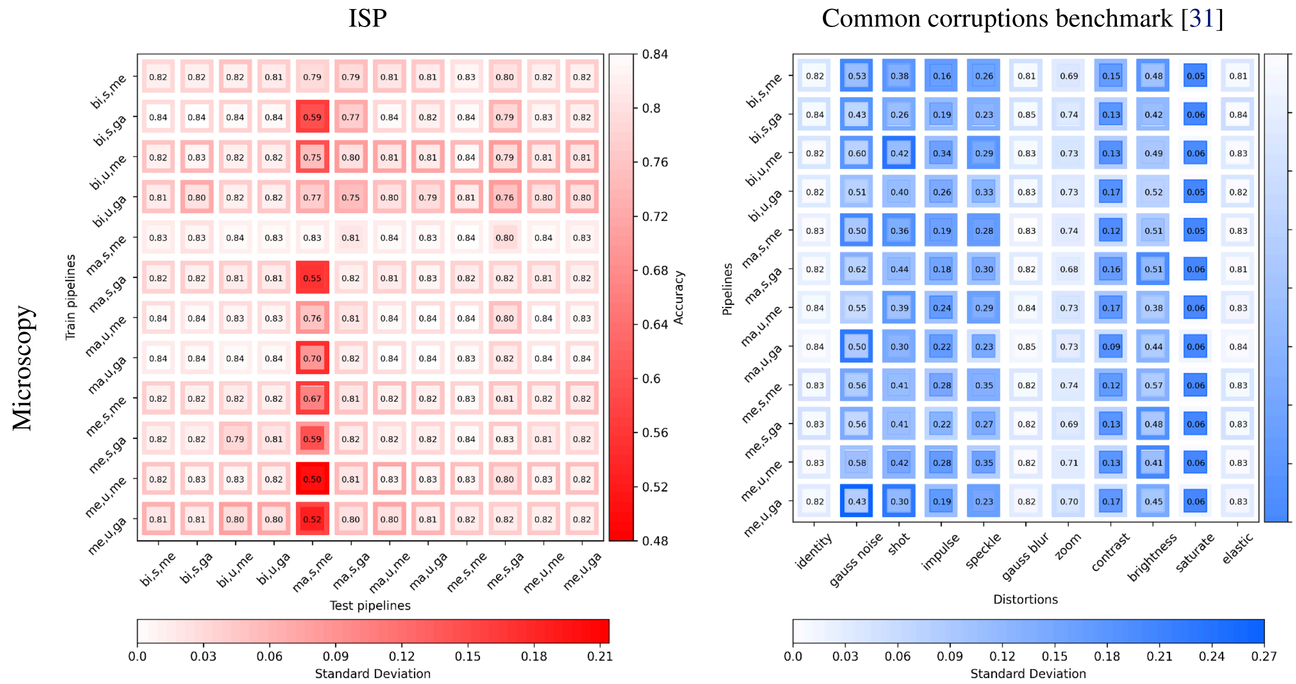


Figure 9: Experiment from Section 2.1 with strong severity (level 5) for the Common corruptions benchmark.

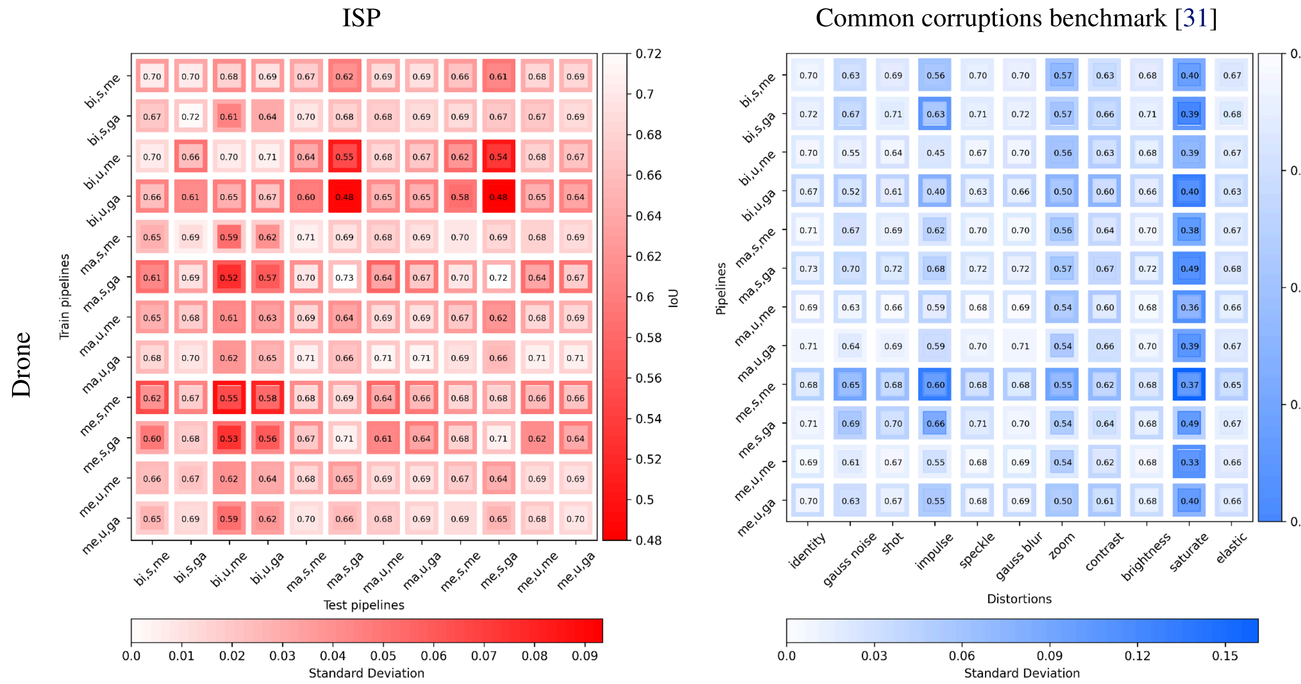


Figure 10: Experiment from Section 2.1 with weak severity (level 1) for the Common corruptions benchmark.

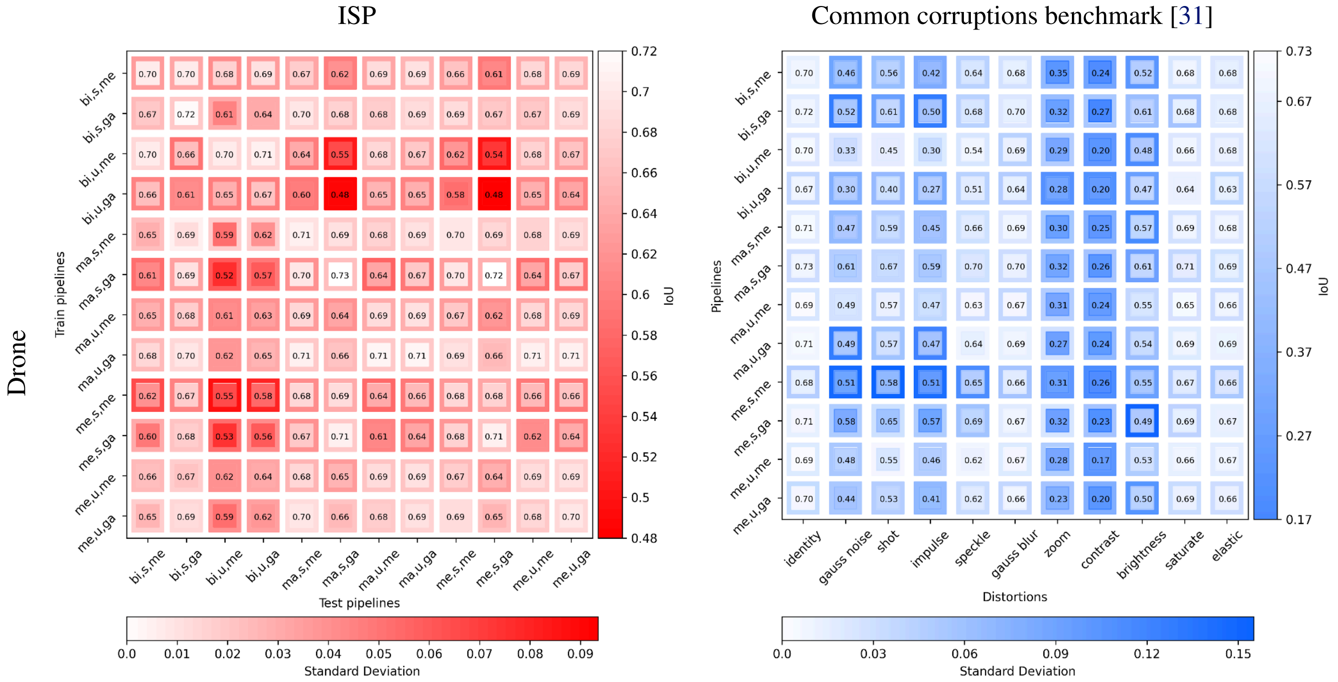


Figure 11: Experiment from Section 2.1 with strong severity (level 5) for the Common corruptions benchmark.

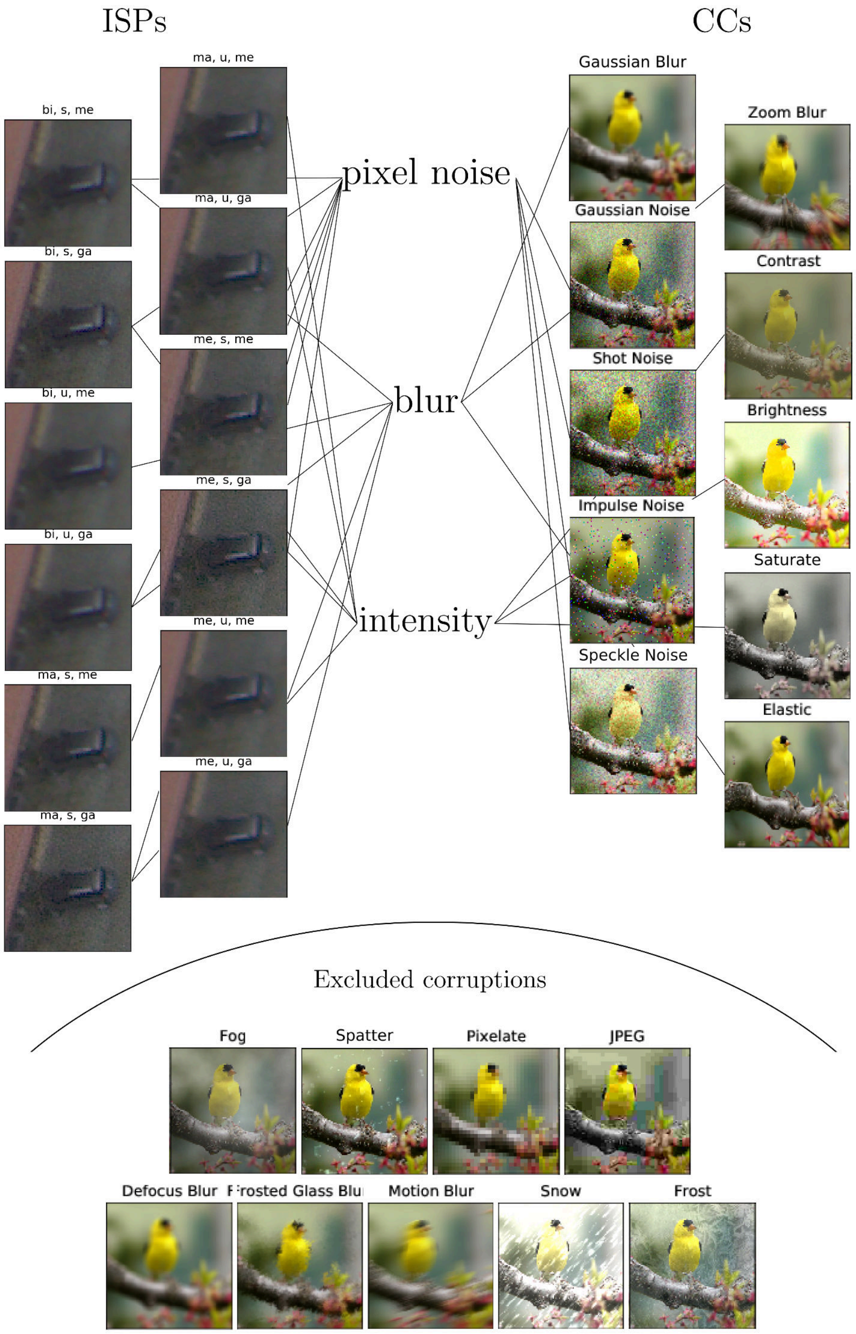


Figure 12: A comparative overview of the physically faithful data models (ISPs, top-left) and the Common Corruptions (CC, top-right) used in the the drift synthesis experiments of Section 2.1. A matching heuristic based on possible visual perception of the drift artifacts (top-middle) is provided for readers who would like to relate specific data models to specific corruptions. However, we emphasize that this is a *purely qualitative heuristic* and has no metrological basis. Since CCs are not physically faithful it is not clear how to relate them to actual variations in the optical data generating process. Finally, corruptions that were excluded from the experiments in Section 2.1 are displayed (bottom). The CC examples where stitched from the original paper [32] for authenticity.

	Augmentation testing	Catalogue testing	Data models
Simulation of test samples	✓	✗	✓
Physically faithful test samples	✗	✓	✓
Differentiable data model	✗	✗	✓

Table 10: Comparison of empirical dataset drift validation methods. Augmentation testing enables ad-hoc test case synthesis but lacks physical faithfulness, unlike catalogue testing. Pairing raw data with data models allows for synthesis of physically faithful test cases, and differentiable data models offer novel drift controls like drift forensics and adjustments.

B.4. Related work

While physically sound data models of images have to the best of our knowledge not yet found their way into the machine learning and dataset drift literature, they have been studied in other disciplines, in particular physical optics and metrology. Our ideas on data models and dataset drift controls we present in this manuscript are particularly indebted to the following works.

Data models for images [87; 69] employ deep convolutional neural networks for modelling a raw image data processing which is optimized jointly with the task model. In contrast, we employ a parametric data model with tunable parameters that enables the modular drift forensics and synthesis presented later. [94] propose a differentiable image processing pipeline for the purpose of camera lens manufacturing. Their goal, however, is to optimize a physical component (lens) in the image acquisition process and no code or data is publicly available. Existing software packages that provide low level image processing operations include Halide [68], Kornia [70] and the rawpy package [81] which can be integrated with our Python and PyTorch code. We should also take note that outside optical imaging there are areas in machine learning and applied mathematics, in particular inverse problems such as magnetic resonance imaging (MRI) or computed tomography, that make use of known operator learning [51; 49] to incorporate the forward model in the optimization [14] or, as in the case of MRI, learn directly in the k-space [102].

Drift synthesis As detailed in the introduction, the synthesis of realistic drift test cases for a task model in computer vision is often done by applying augmentations directly to the input view $\boldsymbol{v}_{\text{GC}}$, e.g. a processed `.jpeg` or `.png` image. Hendrycks et al. [31] have done foundational work in this direction developing a practical, standardized benchmark. However, there is no guarantee that noise $\boldsymbol{\xi}$ added to a processed image $\boldsymbol{v}$ will be physically faithful, i.e. that $\boldsymbol{v} + \boldsymbol{\xi} \in \tilde{\boldsymbol{\Phi}}_{\text{Proc}}[\mathcal{X}_{\text{RAW}}]$. This is problematic, as nuances matter [21] for assessing the cascading effects data models have on the task model $\boldsymbol{\Phi}_{\text{Task}}$ downstream [5; 77]. For the same reason, the use of generative models [25] like GANs has been limited for test data generation as they are known to hallucinate visible and less visible artifacts [17; 80]. Other approaches, like the WILDS data catalogue [9; 41], build on manual curation of so called natural distribution shifts, or, like [84], on artificial worst case constructions. These are important tools for the study of dataset drifts, especially those that are created outside the camera image signal processing. Absent explicit, differentiable data models and raw sensor data, the shared limitation of catalogue approaches is that metrologically faithful drift *synthesis* is not possible and the data generating process cannot be granularly studied and manipulated.

Drift forensics Phan et al. [65] use a differentiable raw processing pipeline to propagate the gradient information back to the raw image. Similar to this work, the signal is used for adversarial search. However, Phan et al. optimize adversarial noise on a per-image basis in the raw space $\boldsymbol{x}_{\text{RAW}}$, whereas our work modifies the parameters of the data model $\boldsymbol{\Phi}_{\text{Proc}}$ itself in pursuit of harmful parameter configurations. The goal in this work is not simply to fool a classifier, but to discover failure modes and susceptible parameters in the data model $\boldsymbol{\Phi}_{\text{Proc}}$ that will have the most influence on the task model's performance.

Drift optimization An explicit and differentiable image processing data model allows joint optimization together with the task model $\boldsymbol{\Phi}_{\text{Proc}}$. This has been done for radiology image data [71; 86; 50] though the measurement process there is different and the focus lies on finding good sampling patterns. For optical data, a related strand of work is modelling inductive biases in the image acquisition process which is explained and contrasted to this work above [94; 35].

Raw image data Camera raw files contain the data captured by the camera sensors [7]. In contrast to processed formats such as `.jpeg` or `.png`, raw files contain the sensor data with minimal processing [96; 58; 46]. The processing of the raw data usually differs by camera manufacturer thus contributing to dataset drift. Existing raw data sets from the machine learning, computer vision and optics literature can be organized into two categories. First, datasets that are sometimes treated - usually not by the creators but by users of the data - as raw data but which are in fact not raw. Examples for this category can be found for both modalities considered here [11; 18; 3; 33; 12; 95; 22; 48; 53; 82; 103]. All of the preceding examples are processed and stored in formats including `.jpeg`, `.tiff`, `.svs`, `.png`, `.mp4` and `.mov`. Second, datasets that are labelled raw data which are raw. In contrast to the labelled and precisely calibrated raw data presented here, existing raw datasets [19; 15; 1; 29] are collected from various sources for image enhancement tasks without full specification of the measurement conditions or labels for classification or segmentation tasks.

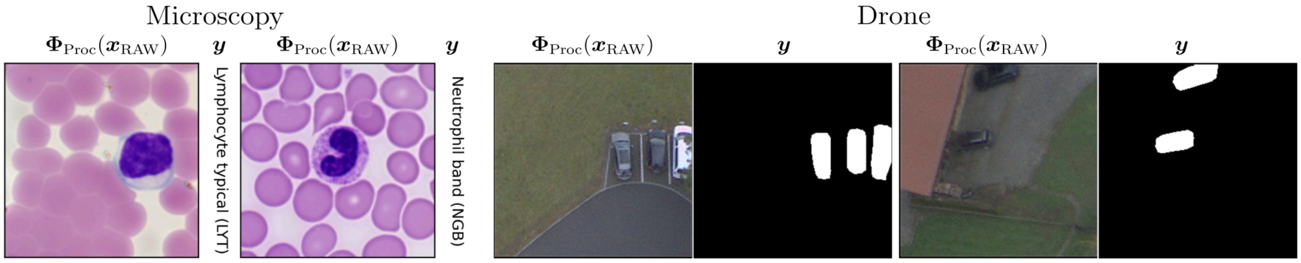


Figure 13: Processed samples and labels of the two datasets, Raw-Microscopy (columns one to four) and Raw-Drone (columns five and eight), that were acquired for the dataset drift study presented here.

B.5. Datasets details

B.5.1. Data acquisition

In the following, core information on the two acquired datasets is provided. In Appendix B.5.2 you can also find detailed datasheets for both datasets, following the documentation good practices introduced by [24].

Raw-Microscopy Assessment of blood smears under a light microscope is a key diagnostic technique for many healthcare services such as cancer treatment and kidney failure as well as blood disorder detection [6]. The creation of image datasets and machine learning models on them has received wide interest in recent years [43; 56; 4]. Variations in the image processing can affect the downstream task model performance [91]. Dataset drift controls can thus help to specify the perimeter of safe application for a task model. A raw dataset was collected for that purpose. A bright-field microscope was used to image blood smear cytopathology samples. The light source is a halogen lamp equipped with a 0.55 NA condenser, and a pre-centred field diaphragm unit. Filters at 450 nm, 525 nm and 620 nm were used to acquire the blue, green and red channels respectively. The condenser is followed by a 40× objective with 0.95 NA (Olympus UPLXAPO40X). Slides can be moved via a piezo with 1 nm spatial resolution, in three directions. Focus was achieved by maximizing the variance of the pixel values[6]. Images are acquired at 16 bit, with a 2560 × 2160 pixels CMOS sensor (PCO edge 5.5). The point-spread function (PSF) was measured to be 450 nm with 100 nm nanospheres. Mechanical drift was measured at 0.4 pixels per hour. Imaging was performed on de-identified human blood smear slides (Ma190c Lieder, J. Lieder GmbH & Co. KG, Ludwigsburg/Germany). All slides were taken from healthy humans without known hematologic pathology. Imaging regions were selected to contain single leukoytes in order to allow unique labelling of image patches, and regions were cropped to 256 × 256 pixels. All images were annotated by a trained hematological cytologist using the standard scheme of normal leukocytes comprising band and segmented neutrophils, typical and atypical lymphocytes, monocytes, eosinophils and basophils [47]. To soften class imbalance, candidates for rare normal leukocyte types were preferentially imaged, and enrich rare classes. Additionally, two classes for debris and smudge cells, as well as cells of unclear morphology were included. Labelling took place for all imaged cells from a particular smear at a time, with single-cell patches shown in random order. Raw images were extracted using JetRaw Data Suite features. Blue, red and green channels are metrologically rescaled independently in intensity to simulate a standard RGB camera condition. Some pixels are discarded complementary on each channel in order to obtain a Bayer filter pattern.

Raw-Microscopy for segmentation comes with 940 raw images, twelve differently processed variants totaling 11280 images and six additional raw intensity levels totaling 5640 samples.

Raw-Drone Automated processing of drone data has useful applications including precision agriculture [42] or environmental protection [36]. Variation in image processing has been shown to affect task model performance [52; 96], underlining the need for drift controls. For the purposes of this study, a raw car segmentation dataset was created for the drone image modality. A DJI Mavic 2 Pro Drone was used, equipped with a Hasselblad L1D-20c camera (Sony IMX183 sensor) having $2.4\,\mu\mathrm{m}$ pixels in Bayer filter array. The lens has a focal length of $10.3\,\mathrm{mm}$. The f-number was set to $N = 8$, to emulate the PSF circle diameter relative to the pixel pitch and ground sampling distance (GSD) as would be found on images from high-resolution satellites. The PSF was measured to have a circle diameter of $12.5\,\mu\mathrm{m}$. This corresponds to a diffraction-limited system, within the uncertainty dominated by the wavelength spread of the image. Images were taken at $200\,\mathrm{ISO}$, a gain of $0.528\,\mathrm{DN}/e^{-}$. The 12-bit pixel values are however left-justified to 16-bits, so that the gain on the 16-bit

[6] Figure 14 in Appendix B.5.1 provides an illustration of the imaging setup.

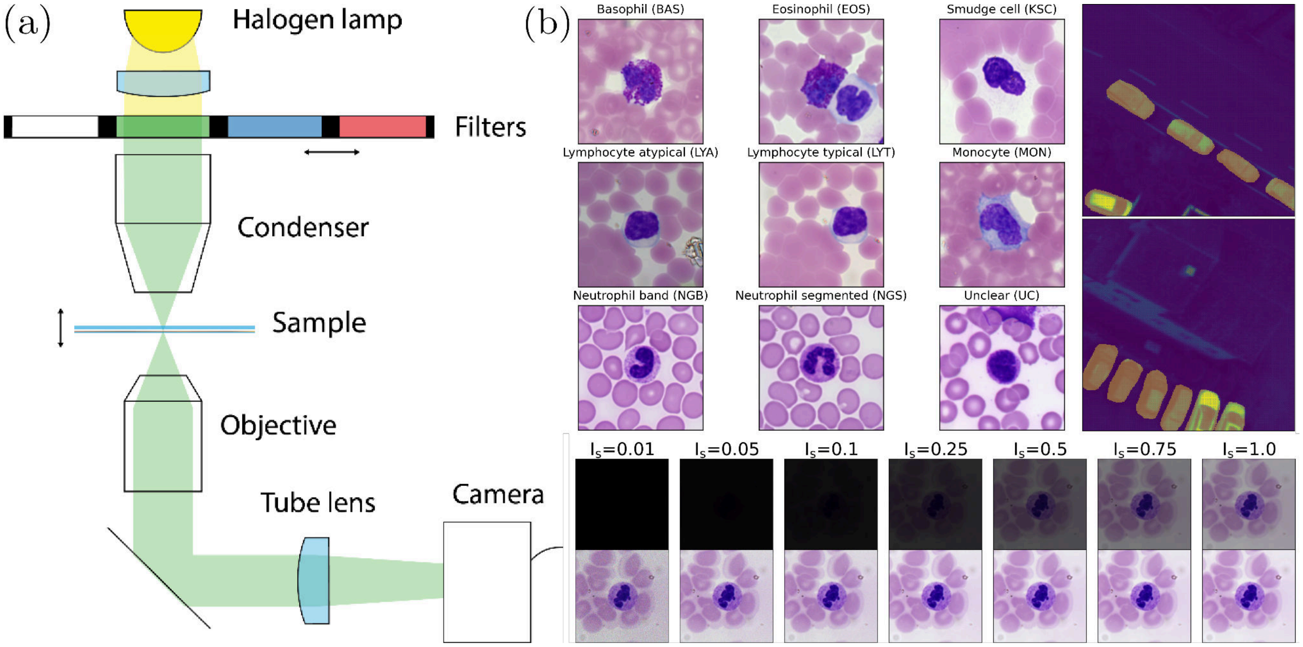


Figure 14: (a) An illustration of the imaging setup. (b) Datasets visualization. (Top-left) RGB raw microscopy classes are shown. (Top-right) Drone raw images are shown with the segmentation mask applied over it. (Bottom) Different intensity realizations are shown for the microscopy case. Images on the top are directly print out in the same scale of the original image. Images in the bottom row are normalized on their own min and max values to highlight the role of noise levels on low intensity images.

Composition of Raw-Microscopy	
Type of instances	Image and label
Objects on images	White blood cells
Type of classes	Morphological classes
Number of instances	940
Number of classes	9
Image size	256 by 256 pixels
Image format	`.tif`
Raw image format	Please see Section 1.1

Class	Proportion in %
Basophil (BAS)	1.91
Eosinophil (EOS)	5.74
Smudge cell / debris (KSC)	17.34
atypical Lymphocyte (LYA)	3.19
typical Lymphocyte (LYT)	24.47
Monocyte (MON)	20.32
Neutrophil (band) (NGB)	0.85
Neutrophil (segmented) (NGS)	22.98
Image that could not be assigned a class (UNC)	3.19

Composition of Raw-Drone	
Type of instances	Image and mask
Objects on images	Landscape shots from above
Number of instances	548
Number of original images	12
Image size	256 by 256 pixels
Mask size	256 by 256 pixels
Original image size	3648 by 5472
Image format	`.tif`
Mask format	`.png`
Raw image format	`.DNG`

Table 11: Summaries of the compositions of Raw-Microscopy and Raw-Drone

numbers is $8.448\,\mathrm{DN}/e^{-}$. The images were taken at a height of $250\,\mathrm{m}$, so that the GSD is $6\,\mathrm{cm}$. All images were tiled in 256 × 256 patches. Segmentation masks were created to identify cars for each patch. From this mask, classification labels were generated to detect if there is a car in the image. The dataset is constituted by 548 images for the segmentation task.

Raw-Drone for segmentation comes with 548 raw images, twelve differently processed variants totaling 6576 images and six additional raw intensity levels totaling 3288 samples.

B.5.2. Datasheets

We follow the datasheets documentation framework proposed in [24], using the template https://de.overleaf.com/latex/templates/datasheet-for-dataset-template/jgqyyzyprxth from Christian Garbin.

Datasheet for Raw-Microscopy

Motivation

For what purpose was the dataset created?

With Raw-Microscopy we provide a publicly available raw image dataset in order to examine the effect of the image signal processing on the performance and the robustness of machine learning models. This dataset enables to study these effects for a supervised multiclass classification task: the classification of white blood cells (WBCs).

Who created this dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?

This dataset has been created by the Laboratory of Applied Optics of the Micro-Nanotechnology group at HEPIA/HES-SO, University of Applied Sciences of Western Switzerland. Single-cell images were annotated by a trained cytologist.

Who funded the creation of the dataset?

The creation of the dataset has been funded by HEPIA/HES-SO.

Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)?

An instance is a tuple of an image and a label. The image shows a human WBCs and the label indicates the morphological class of this cell. The following eight morphological classes appear in the dataset: Basophil (BAS), Eosinophil (EOS), Smudge cell / debris (KSC), atypical Lymphocyte (LYA), typical Lymphocyte (LYT), Monocyte (MON), Neutrophil (band) (NGB), Neutrophil (segmented) (NGS). The nith class consists of images that could not be assigned a class (UNC) during the labeling process.

How many instances are there in total (of each type, if appropriate)?

The data set consists of 940 instances. For the proportion of each class in the dataset see table 12.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?

The dataset does not contain all possible instances. It is limited to WBC classes normally present in the peripheral blood of healthy humans. In order to cope with intrinsic class imbalance in cell distribution, rare cell class candidates such as Basophils were preferentially imaged.

What data does each instance consist of? "Raw" data (e.g., unprocessed text or images) or features?

Each instance consists of an image of 256 by 256 pixels. The image is a raw image in .tiff format.

Is there a label or target associated with each instance?

Each instance is associated to a label, that indicates the morphological class of the image.

Is any information missing from individual instances?

No information is missing.

Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)?

No, relationships between individuals are not made explicit.

Are there recommended data splits (e.g., training, development/validation, testing)?

There are no recommended data splits. All the data splits that we used for our experiments were randomly picked.

Are there any errors, sources of noise, or redundancies in the dataset?

To the best of our knowledge, there are no errors in the dataset. However, a key source of variability between slides from different laboratories and processing times is stain intensity. The samples used in this work all come from the same source, hence we assume the preanalytic treatment and staining protocol to be similar. As all images were obtained on the same microscopy equipment, focus handling and illumination are identical for all samples. Image labelling was performed by one trained morphologist with experience in hematological routine diagnostics. It is known that morphology annotations are subject to inter- and intra-rater variability. However, as we limit ourselves to normal WBCs the labeling is expected to be stable.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?

The dataset is self-contained.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals non-public communications)?

The dataset consist of medical data, disclosing the morphological classes of single human WBCs. In principle, the distribution of cell types conveys information on the health state of a patient.
However, the subjects in this dataset are fully de-identified, so that the image data cannot be linked back to the healthy donors of the scanned blood smears. Furthermore, it is not disclosed which cell image was taken from which blood smear, so that no frequencies of individual cell types can be determined. Additionally, we only consider cell types present in normal blood, so that no specific hematologic pathology can be deduced from cell morphologies.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No. The dataset does not contain data with any of the above properties.

Does the dataset relate to people?

Yes. The dataset consist of images of human WBCs.

Does the dataset identify any subpopulations (e.g., by age, gender)?

The donors of the blood smears used in this dataset are fully deidentified, and no information on subpipulation composition is provided.

Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset?

No. It is not possible to identify individuals from an image of their white blood cells or visa versa.

Does the dataset contain data that might be considered sensitive in any way (e.g., data that reveals racial or ethnic origins, sexual orientations, religious beliefs, political opinions or union memberships, or locations; financial or health data; biometric or genetic data; forms of government identification, such as social security numbers; criminal history)?

No. While the distribution of cell types for a specific patient could reveal information about that patient's health status, isolated single-cell images of normal leukocytes do not allow for this inference.

Any other comments?

See table 13 for a summary of the composition of Raw-Microscopy.

Class	**Proportion in %**
Basophil (BAS)	1.91
Eosinophil (EOS)	5.74
Smudge cell / debris (KSC)	17.34
atypical Lymphocyte (LYA)	3.19
typical Lymphocyte (LYT)	24.47
Monocyte (MON)	20.32
Neutrophil (band) (NGB)	0.85
Neutrophil (segmented) (NGS)	22.98
Image that could not be assigned a class (UNC)	3.19

Table 12: The proportion of the classes in Raw-Microscopy.

Collection Process

How was the data associated with each instance acquired?

Images of the dataset have been acquired directly from a CMOS imaging sensor. They are in a raw unprocessed format.

What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)?

Imaging data have been obtained via a custom brightfield microscope.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?

Images have 256 × 256 pixel size and have been cropped from larger images. The dataset corresponds to a selection of white blood cells in the acquired large images. A sampling strategy aimed at increasing the proportion of rare classes of white blood cells has been used.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?

A research assistant has been involved in the data collection process and has been compensated with a monthly salary.

Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances (e.g., recent crawl of old news articles)?

Data have been collected on a timeframe of two months, corresponding to the availability of the physical samples to image. Data have been collected on purpose for this work.

Were any ethical review processes conducted (e.g., by an institutional review board)?

The microscopy data was purchased from a commercial lab vendor (J. Lieder GmbH & Co. KG, Ludwigsburg/Germany) who attained consent from the subjects included.

Does the dataset relate to people?

Yes. The dataset consists of microscopic images of human white blood cells.

Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources (e.g., websites)?

Data have not been obtained via third parties.

Were the individuals in question notified about the data collection?

As the blood smear slides were bought from a company, notification to individuals of the data collection has been performed by the company.

Did the individuals in question consent to the collection and use of their data?

Yes, they did.

If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses?

We do not know the conditions of consent adopted by the selling company. However, we believe the company provided the individuals a complete freedom in revoking their consent in the future, if desired.

Has an analysis of the potential impact of the dataset and its use on data subjects (e.g., a data protection impact analysis) been conducted?

No, this kind of analysis has not been conducted.

Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)?

Intensity scaled images are generated with Jetraw Data Suite for both datasets, which applies a physical model based on sensor calibration to accurately simulate intensity reduction. Microscopy Raw images are extracted from RGB Microscopy data through a pixel selection from images taken with three filters, in order to have a Bayer Pattern. Pixels intensities are rescaled with Jetraw Data Suite to match the measured transmissivities of a Bayer colour filters array.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)?

Raw images are available in the dataset.

Is the software used to preprocess/clean/label the instances available?

All code used in the experiments of this manuscript is publicly available. Jetraw products that were used for acquiring the data are commercially available.

Uses

Has the dataset been used for any tasks already?

The dataset has not yet been used.

Is there a repository that links to any or all papers or systems that use the dataset?

The repository at `https://github.com/aiaudit-org/raw2logit` associated to this work, maintained by Luis Oala.

What (other) tasks could the dataset be used for?

The dataset can be used to study the effect of image signal processing on the performance and robustness of any other machine learing model implemented in PyTorch, designed for a supervised multiclass classification task.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?

To the best of our knowledge, we do not recognize such impacts.

Are there tasks for which the dataset should not be used?

To the best of our knowledge, there are no such tasks.

Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?

Yes. The dataset will be publicly available.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)

A guide to access the dataset is available at https://github.com/aiaudit-org/raw2logit. Moreover, the dataset can be downloaded anonymously and directly at https://zenodo.org/record/5235536 under the doi: 10.5281/zenodo.5235536.

When will the dataset be distributed?

The dataset is already publicly available.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?

The dataset will be distributed under the Creative Commons Attribution 4.0 International.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

There are no such restrictions.

Maintenance

Who will be supporting/hosting/maintaining the dataset?

Luis Oala on behalf of Dotphoton AG.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

By email address via
luis.oala@dotphoton.com.

Is there an erratum?

At the time of submission, there is no such erratum. If an erratum is needed in the future it will be accessible at https://github.com/aiaudit-org/raw2logit.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?

Yes. The dataset will be enlarged wrt. the number of instances.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)?

To the best of our knowledge, there are no such limits.

Will older versions of the dataset continue to be supported/hosted/maintained?

Older versions will be supported and maintained in the future. The dataset will continue to be hosted as long as https://zenodo.org/ exists.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

For any of these requests contact either Luis Oala (luis.oala@dotphoton) or Bruno Sanguinetti (bruno.sanguinetti@dotphoton.com). For now, we do not have an established mechanism to handle these requests.

Composition of Raw-Microscopy	
Type of instances	Image and label
Objects on images	White blood cells
Type of classes	Morphological classes
Number of instances	940
Number of classes	9
Image size	256 by 256 pixels
Image format	`.tif`
Raw image format	Please see Section 1.1

Table 13: A summary of the composition of Raw-Microscopy.

Datasheet for Raw-Drone

Motivation

For what purpose was the dataset created?

With Raw-Drone we provide a publicly available raw dataset in order to examine the effect of the image data processing on the performance and the robustness of machine learning models. This dataset enables to study these effects for a segmentation task: the segmentation of cars. The dataset was taken with specified parameters: sensor gain, point-spread function and ground-sampling distance, so that physical models may be used to process the data. It also was taken with a easily accessible and affordable system, so that it may be reproduced.

Who created this dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?

The dataset was created by Bruno Sanguinetti and Marco Aversa on behalf of the company Dotphoton AG.

Who funded the creation of the dataset?

The data collection was funded by Dotphoton AG. The calibration of the image characteristics was jointly funded by Dotphoton AG and the European Space Agency.

Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)?

An instance is a tuple of an image and a segmentation mask. The image shows a landscape shot from above. The segmentation mask is a binary image. A white pixel in this mask corresponds to a pixel within a region in the image where a car is displayed. A black pixel in this mask corresponds to a pixel within a region in the image where no car is displayed.

How many instances are there in total (of each type, if appropriate)?

The dataset consists of 548 instances.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?

The dataset does not contain all possible instances. Only images with at least one white pixel in the associated segmentation mask are considered.

What data does each instance consist of? "Raw" data (e.g., unprocessed text or images) or features?

Both, the image and the segmentation mask consist of 256 by 256 pixels. The image is a raw image in `.tif` format and the the segmentation mask is in `.png` format. The images are cropped sub-images of 12 raw images in `.DNG` format, consisting of 3648 by 5472 pixels.

Is there a label or target associated with each instance?

Each instance is associated to a binary segmentation mask.

Is any information missing from individual instances?

No information is missing.

Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)?

Since every image is a cropped sub-image of an original image, several of these sub-images belong to the same original image. All sub-images are disjoint, i.e. no different images share a pixel from the original image.

Are there recommended data splits (e.g., training, development/validation, testing)?

There are no recommended data splits. All the data splits that we used for our experiments were randomly picked.

Are there any errors, sources of noise, or redundancies in the dataset?

To the best of our knowledge, there are no errors in the dataset. The segmentation mask is created by hand and hence noisy, especially at the boundaries between a region with a car and a region without a car.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?

The dataset is self-contained.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals non-public communications)?

No. The dataset does not contain data of any of the above types.

Does the data set contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No. The dataset does not contain data with any of the above properties.

Does the dataset relate to people?

The dataset does not relate to people. The drone data was screened for PIIs such as faces or license plates on cars and removed by the data collection team.

Any other comments?

See table 14 for a summary of the composition of the Raw-Drone.

Collection Process

How was the data associated with each instance acquired?

The data was collected by flying a drone and saving the raw data. The calibration data for the drone's imager was acquired both under laboratory conditions and using a ground-based calibration target, so that it could be acquired under operating conditions.

What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)?

To acquire the drone images, we used a DJI Mavic 2 Pro Drone, equipped with a Hasselblad L1D-20c camera (Sony IMX183 sensor). This system has $2.4\,\mu\text{m}$ pixels in Bayer filter array. Images were taken with the drone hovering for maximum stability. This stability was verified to be better than a single pixel by calculating the correlation of subsequent images. The objective has a focal length of $10.3\,\text{mm}$. We operated this objective at an f-number of $N = 8$, to emulate the PSF circle diameter relative to the pixel pitch and ground sampling distance (GSD) as would be found on images from high-resolution satellites. Operating at $N = 8$ also minimises vignetting, aberrations, and increases depth of focus. The point-spread function (PSF) was measured to have a circle diameter of $12.5\,\mu\text{m}$ using the edge-spread function technique and a ground calibration target.This corresponds to $\sigma = 2.52\,\text{px}$, which also corresponds to a diffraction-limited system, within the uncertainty dictated by the wavelength spread of the image. Images were taken at $200\,\text{ISO}$, corresponding to a gain of $0.528\,\text{DN}/e^{-}$. The 12-bit pixel values are however left-justified to 16-bits, so that the gain on the 16-bit numbers is $8.448\,\text{DN}/e^{-}$. The images were taken at a height of $250\,\text{m}$, so that the GSD is $6\,\text{cm}$. All images were tiled in 256x256 patches. Segmentation color masks were created to identify cars for each patch. From this mask, classification labels were generated to detect if there is a car in the image. The dataset is constituted by 548 images for the segmentation task, and 930 for classification. Six additional intensity scales were created with Jetraw.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?

The entire dataset is presented.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?

The dataset was taken by a company employee, compensated by his salary. Labeling was performed by both a company employee and a PhD student, who's PhD is funded by the company.

Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances (e.g., recent crawl of old news articles)?

The dataset was taken as the initial step of writing this article.

Were any ethical review processes conducted (e.g., by an institutional review board)?

The dataset does not contain any elements requiring an ethical review process.

Does the dataset relate to people?

The dataset does not relate to people. There are individuals on the images, but it is not possible to identify these individuals.

Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)?

No further processing was applied to the Raw-Drone data.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)?

Raw images are available in the dataset.

Is the software used to preprocess/clean/label the instances available?

All code used in the experiments of this manuscript is publicly available. Jetraw products that were used for acquiring the data are commercially available.

Uses

Has the dataset been used for any tasks already? The dataset has not yet been used.

Is there a repository that links to any or all papers or systems that use the dataset?

The repository at https://github.com/aiaudit-org/raw2logit associated to this work, maintained by Luis Oala.

What (other) tasks could the dataset be used for?

The dataset can be used to study the effect of image signal processing on the performance and robustness of any other machine learing model implemented in PyTorch, designed segmentation task.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?

To the best of our knowledge, we do not recognize such impacts.

Are there tasks for which the dataset should not be used?

To the best of our knowledge, there are no such tasks.

Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?

Yes. The dataset will be publicly available.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)

A guide to access the dataset is available at https://github.com/aiaudit-org/raw2logit. Moreover, the dataset can be downloaded anonymously and directly at https://zenodo.org/record/5235536 under the doi: 10.5281/zenodo.5235536.

When will the dataset be distributed?

The dataset is already publicly available.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?

The dataset will be distributed under the Creative Commons Attribution 4.0 International.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

There are no such restrictions.

Maintenance

Who will be supporting/hosting/maintaining the dataset?

Luis Oala on behalf of Dotphoton AG.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

By email address via
luis.oala@dotphoton.com.

Is there an erratum?

At the time of submisson, there is no such erratum. If an erratum is needed in the future it will be accessible at https://github.com/aiaudit-org/raw2logit.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?

Yes. The dataset will be enlarged wrt. the number of instances.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)?

To the best of our knowledge, there are no such limits.

Will older versions of the dataset continue to be supported/hosted/maintained?

Older versions will be supported and maintained in the future. The dataset will continue to be hosted as long as https://zenodo.org/ exists.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

For any of these requests contact either Luis Oala (luis.oala@dotphoton.com) or Bruno Sanguinetti (bruno.sanguinetti@dotphoton.com). For now, we do not have an established mechanism to handle these requests.

Composition of Raw-Drone	
Type of instances	Image and mask
Objects on images	Landscape shots from above
Number of instances	548
Number of original images	12
Image size	256 by 256 pixels
Mask size	256 by 256 pixels
Original image size	3648 by 5472
Image format	`.tif`
Mask format	`.png`
Raw image format	`.DNG`

Table 14: A summary of the composition of Raw-Drone.