# Hackathons: From Novel Ideas to Novel Deployable Solutions

Eslam A. Hussein[1,2,3], Christopher Thron[4], Ezra Fielding[5], Robert Purdy*[3], Emma Cochrane[3], Bakang Kgopolo[6], Mpho Carol Jan[7], Isaac Opio[6], Karabo K. Ndebele[6], Dihariniaina RABEARIMANANA[8], Nancy TOVOLAHY[9], Tahinasoa Misandratra RAZAFINTSALAMA[10], Rojoniaina RASOAFARIHY[9], Toky Sandratra Miharimamy RASAMOELINA[11], Hery T. Rakotondramiarana[9], Rova Isaia Andriamamy[9], Harena Ranaivoson[12], Valdo Tsiaro Hasina Ndimbisoa [12], Toavina LEONG POCK-TSY[9], Tohavina RAOELINASIMBOLAMALALA [12], RAHAJAHARIVONY ROJOTIANA FANDRESENA JESSICA[12], LIANTSOA RANDRIANASIMBOLARIVELO[9], RAFANOMEZANTSOA PIERROT[13], Marcia Jennifer RAKOTOSON [14], Iriana Anthony Durandal Rasoanaivo[15], i Ilo Mahomby RANDRIANATOANDRO [11], Mirindra Randriamanantena[17], Jeanny Fidelica TSARAMANGA[12], Fanomezana Sarobidy RAZAKAHASINA[12], Tsantaniony RAZAIARIMIHAJASOA[11], Rina Ralijaona[9, 12], Shain Mukungu[18], Gregorio Marychen[18], Landuleni Kashipulwa[18], Max Haikali[19], Sigrid Shilunga[18], Geno Gurirab [18], Eunice dos anjos[20], Gilson G. Vicente[21], Rogério Langa[21], Ezequias Miocha[22], Francisco Ernesto Uqueio [23], Milton Paulo Mbalate[22], UWAMAHORO Leopord[24], DUSENGE Mediatrice[24], BAVUGAMENSHI Valens[24], Silamlak Desye[25], Jemila Muhidin[26], Melat Kebede[27], Addisu Mengistu[26], Chalie Lijalem[28], Turemo Bedaso[26], Josephin Chamoun-Contreras[29], Paula Silva[30, 31], Maria Laura Oyarce[32], Rubén José Carcamo[29], C. E. Perez-Ramirez[33], Nousaba Nasrin Protiti[29], Gabriel Christino[34], Marcela Gouvêa[35], and Matheus Poltronieri[35]

[1]BRICS Astronomy, Observatory, Cape Town, South Africa
[2]Inter-University Institute for Data Intensive Astronomy, Department of Physics and Astronomy, University of the Western Cape, Bellville 7535, South Africa
[3]School of Physics and Astronomy, University of Leeds, Leeds, UK
[4]Department of Science and Mathematics, Texas A&M University-Central Texas, Killeen, USA
[5]Centre National d'Études Spatiales, Toulouse, France
[6]Botswana International University of Science and Technology, Palapye, Botswana
[7]Botswana Accountancy College, Gaborone, Botswana
[8]Grande École de l'Innovation Technologique, Antananarivo, Madagascar
[9]University of Antananarivo, Antananarivo, Madagascar
[10]Laboratoire d'Intelligence Artificielle de Madagascar, Antananarivo, Madagascar
[11]Institut Supérieur Polytechnique de Madagascar, Antananarivo, Madagascar
[12]Mathématique, Informatique et Statistiques Appliquées , Antananarivo, Madagascar
[13]Ecole Supérieure Polytechnique d'Antananarivo, Antananarivo, Madagascar
[14]GEIT Andranomena, Antananarivo, Madagascar
[15]IT University, Antananarivo, Madagascar
[16]École Nationale d'Informatique, Fianarantsoa, Madagascar
[17]Institute of High Energy Physics, Antananarivo, Madagascar
[18]University of Namibia, Windhoek, Namibia
[19]Namibian University of Science and Technology, Windhoek, Namibia
[20]Universidade Metodista Unida de Moçambique, Inhambane, Mozambique
[21]Instituto Superior de Transportes e Comunicações, Maputo, Mozambique
[22]Eduardo Mondlane University, Maputo, Mozambique
[23]WUTIVI University, Maputo, Mozambique
[24]UNIVERSITY OF RWANDA, Kigali, Rwanda
[25]BrainBite, Addis Ababa, Ethiopia
[26]Addis Ababa University, Addis Ababa, Ethiopia

[27]Holland Dairy PLC, Addis Ababa, Ethiopia
[28]Adama Science and Technology University , Addis Ababa, Ethiopia
[29]Centro de Astronomía (CITEVA), Universidad de Antofagasta, Av. Angamos 601, Antofagasta, Chile
[30]Millennium Institute of Astrophysics (MAS), Nuncio Monseñor Sótero Sanz 100, Providencia, Santiago, Chile
[31]Instituto de Alta Investigación, Universidad de Tarapacá, Casilla 7D, Arica, Chile
[32]Pontificia Universidad Católica de Chile, Santiago, Chile
[33]Universidad Católica del Maule, Talca, Chile
[34]Faculdade De Tecnologia SENAC Rio, Rio de janeiro, Brasil
[35]Centro Brasileiro de Pesquisas Físicas, Rio de Janeiro, Brasil

December 8, 2025

**Abstract**

Data science hackathons, like most others, often end with novel ideas or early prototypes, with little effort to explore what more they could achieve. The literature suggests that the main reason for this limitation is their short timeframe, which is often seen as a barrier to achieving more mature outcomes. In this paper, we introduce a new hackathon prototyping workflow that moves beyond idea generation toward producing deployable solutions. We demonstrate it through the Hack4dev Data Science Cascade Hackathon Programme, which tackled a cubesat data multi-classification problem with the goal of developing both accurate and efficient machine-learning models. By examining 20 top solutions, comparing their performance, and exploring how key hackathon process elements relate to solution quality, our study shows how this mechanism turned novel ideas into deployable solutions in three days. The mechanism allowed participants to produce a machine-learning model that first matched and then improved a published baseline by 166 times, in terms of model speed, by the end of a three-day hackathon period. This is the same period in which many expect to generate, at best, early prototypes. This supports a new concept we call time-change compression (TCC), where short time intervals are designed to deliberately produce the most mature possible outcome, opening the possibility for hackathons to go beyond deployable solutions.

keywords: time, machine learning, hackathons, R&D, research, design, prototyping, flipped classroom, cascade model, data science, compression

# 1 Introduction

"***Changing the world***" is a goal that many people seek to achieve in order to make the world a better place. This change can either be forced, for example, due to the disruptive shifts brought by the Fourth Industrial Revolution [1], or driven by a sense of duty to fight existing inequalities [2]. Both of which require fast and adaptive action for those challenges to be addressed [3, 4].

Many believe that hackathons offer a way to bring about such change in a short time [5]. While hackathons have many definitions, a common one is that they are collaborative events aimed at producing a solution to a problem within a short period, typically lasting from one to three days [5,6]. Based on this idea, hackathon organisers generally believe that something concrete can be achieved by the end of the event [7]. As for the agents of change themselves, *"the participants"*, 38% say that their primary motivation is to "change the world" through hackathons [8].

Today, hackathons are witnessing growing popularity across many disciplines and fields, all driven by the motive of accelerating change. These fields range from software engineering and medicine to education and data science [9–12].

However, because hackathons are usually carried out over a short time-scale/ a few days, they rarely produce fully developed innovative solutions [13–15]. Instead, they are most useful for generating new ideas and early functional prototypes [16,17]. Some researchers have even described hackathons as creative sessions, where people from different backgrounds come together to address challenges within a short time frame [7]. As a result, several studies have suggested newly designed frameworks that integrate hackathons into broader innovation cycles, positioning them as open innovation platforms, where they act as starting points for ideas that can be further developed after the event [16, 18–23].

This creates a contradiction. People are drawn to hackathons because of their unique characteristics: the short time frame (often scheduled over a weekend) that makes them easy to commit to [14, 20, 24, 25], and their intensity that pushes them to learn and access new tools, solve new problems and win prizes [8]. Yet, most of these features do

** Corresponding author: R.Purdy@leeds.ac.uk

not continue after the hackathon ends. For instance, further development usually requires time, effort, and resources that teams may no longer have access to [19]. Further development requires resources that teams may no longer have access to as well as continued time and effort. As a result, participants generally lose interest, resulting in many promising projects fading [19, 20, 26]. In fact, a recent study shows that only about 5% of hackathon projects continue for more than five months [27].

Therefore, in their current form, hackathons do not seem to accelerate the innovation process [20], especially when it takes many months for only a few tangible solutions to emerge after hackathons. This raises the questions: Do hackathons have the potential to accelerate change by producing innovative and impactful solutions that go beyond ideas and early prototypes? And if so, how can this be achieved within a short time scale when prior work argues that it is unrealistic to expect teams to deliver mature solutions in just a few days [13, 28–30].

In this paper, we address these questions in data science and show how hackathons can deliver solutions that go beyond novel ideas by reaching deployable models within days. We did this through the development of a novel prototyping workflow, which was implemented across 13 regional hackathons. The workflow produced 20 novel machine-learning solutions, and the best solution matched the accuracy of a published baseline while running 166× faster.

## 2 Literature Review

The Fourth Industrial Revolution has brought a rapid increase in both data variety and volume, creating many new challenges [31]. In response, attention has fallen on the field of data science, a multidisciplinary area that applies scientific methods and algorithms to extract knowledge and insights from data [32]. However, data science is still an emerging field where the size and complexity of data are growing rapidly. In domains like astronomy, data rates can reach $\sim 10$ terabits per second (Tb/s) [33]. As a result, there is a pressing need to explore tools and approaches that can respond to these challenges in an agile and scalable way.

Hackathons, as a tool for accelerating development [30], have found increasing use in the field of data science for several reasons. With many new platforms emerging, hackathons act as fast feedback environments that accelerate their improvement [34–37]. Additionally, as data science is still an emerging field, educators have recognised the value of hackathons as a training tool to accelerate the teaching of data science skills [38–41]. And finally, there is the use of hackathons themselves as a tool for innovation.

Given their short duration and ease of participation, researchers have viewed hackathons as opportunities to foster multidisciplinary collaboration. For example, hackathons have been used to address challenges in solid-state materials chemistry by bringing together chemists and data scientists [42]. Similarly, several studies have documented hackathons that paired clinicians with data scientists to tackle problems in healthcare [43, 44]. The goals of these events vary, from promoting reproducible research to generating new research ideas, with some resulting in the production of paper abstracts and setting the stage for future collaboration and work [43–47].

Other hackathons have managed to produce early working data science solutions; however, further refinement is often needed due to the limited time frame of these events [48–54]. An example of such post-hackathon refinement was carried out by Navas-Olive et al [55] where they took a model developed during a hackathon and improved it through hyperparameter tuning to produce a more mature solution for detecting hippocampal sharp-wave ripples. In contrast, one datathon/hackathon that successfully achieved a fully working mature solution extended over 6 weeks [56].

While hackathons in data science have shown potential for innovation, the literature suggests that they rarely result in fully mature, deployable novel solutions right after the event. In most cases, further development and refinement are required [48–54]. In addition, post-hackathon collaboration also tends to be limited. One study found that in data science–focused hackathons, only about 10% of projects led to publications within a few years [52]. These results align with the broader hackathon literature, which highlights the difficulty of sustaining work after the event, both in continuing collaboration and in producing mature models.

However, to unlock the true potential of hackathons, attention should be directed to the design of the hackathon and how it affects the event and potential outcomes. In the literature, hackathons are seen as speed-design events [57] that are typically carried out through prototyping [58]. The design of prototypes can take many forms, such as paper-based or software prototypes, serving different functions [58], and can follow various design processes, such as iterative or parallel development [59].

Within hackathon environments, participants often follow a design process similar to the Double Diamond model [6]. This model has four phases: Discover: understand the problem; Define: frame the problem and its requirements; Develop: ideate and prototype; and Deliver: refine and converge to a final artefact.

This inherent design process within hackathons has been a recent topic of research, as changing the structure of a hackathon directly influences how participants approach the design of their own prototypes. For instance, Byrne et al [60] introduced the Bridge21 model into a 4-day hackathon to emphasise teamwork, hands-on activities, and technology-driven projects. The model follows a linear/cyclic process consisting of seven stages and integrates

elements of design thinking, namely inspiration, ideation, and implementation. Although the model appears linear, its design thinking foundation allows for flexibility, enabling teams to revisit and revise earlier stages when needed.

Another study introduced debriefings into a typical hackathon structure, with the aim of fostering collaboration and helping students perform better within teams [61]. The results showed that while debriefings improved collaborative processes and supported team reflection, they had no measurable impact on overall team performance.

A further example is described by Ghost and Wu [62], who introduced iterative coordination into a one-day hackathon. Here, teams had to check-ins every two hours to assess the progress of their software development. The results showed that, while iterative coordination led to more functional early-stage prototypes, it came at the expense of novelty. This was because teams had to continuously assess the level of novelty they could maintain throughout the hackathon as a results of these coordination intervals. It is important to note that iterative coordination does not necessarily result in more complete or functional prototypes overall; rather, it promotes the development of more viable software solutions that prioritise value over novelty.

Another study introduced the Software Development Life Cycle (SDLC) framework, commonly used in industry, into a hackathon environment in order to help participants design, test, discuss, and refine software. The authors found that all stages of the SDLC were well-suited to engage participants, except for the final two stages; deployment and maintenance. This was largely because the hackathon targeted youth participants, aiming to make the experience fun while also demonstrating the potential of computing to impact communities [24].

# 3 Methodology

## 3.1 Hackathon process overview

On 18 June 2024, Hack4dev along with several partners launched the first cycle of the Data Science Hackathon Cascade Programme (DSH-CP). The main objective of the programme is to "***Use and Develop Data Science Skills to Address Global Data Science Challenges.***" It is designed to recruit participants to work on pressing global issues that require data-driven solutions, giving them the opportunity to apply and strengthen their data science capabilities.

The implementation of the DSH-CP follows two pivotal stages:

- **Organisers' Hackathon (OH):** This stage equips regional organising teams with the knowledge and skills to run hackathons independently. Hackathon organisers gain hands-on experience in both the technical and logistical aspects of hackathon delivery, supported and guided by experienced Hack4dev team members.
- **Regional Hackathons (RH):** In this stage, the trained regional teams host hackathons at their own institutions, coordinated by Hack4dev. It is through these events that the core objective of the DSH-CP is realised.

The Organisers' Hackathon took place at the University of the Witwatersrand (WITS) in South Africa from 23 to 25 October 2024. Following this, 13 Regional Hackathons were organised between February and April 2025 across 12 different regions: Kenya, Ghana, Ethiopia, Botswana, South Africa, Madagascar, Rwanda, Namibia, Mozambique, India, Brazil, and Chile. RH participants were selected based on their motivation, Python programming language experience, and ability to work independently. An additional requirement was to be based near the event location and able to commute daily to and from the venue.

The OH consisted of a 3-day programme. The first day featured introductory talks, including an overview of the DSH-CP, an introduction to the cloud computing facility, team lead introductions, and a briefing on the challenge itself. The second day was dedicated to hands-on work on the challenge, while the final day focused on team presentations and general feedback, with a discussion on how to approach solving the challenge more effectively.

The RH followed a 3.5-day structure. The first half-day was dedicated to introductory talks and orientation, followed by 2.5 days of focused work on the challenge. The final half-day was reserved for team presentations to a panel of reviewers.

To support the computational needs of the hackathons, servers are provided by the Inter-University Institute for Data Intensive Astronomy (IDIA). Each participant's hackathon team is given access to its own server, with 16 CPUs and 123 GiB of RAM.

## 3.2 The CubeSat Hackathon Challenge

### 3.2.1 CubeSat Classification: Scientific Context and Motivation

A CubeSat is a class of nanosatellite providing small space experiment platforms which have low construction and launch costs. [63]. They offer a compelling alternative to more traditional space missions with their increased accessibility for researchers and institutions, as seen with the successful deployments of CubeSat-based space telescopes

in various astronomical studies [64]. Thanks to the miniaturization of enabling technologies and improvements to mission payloads, CubeSat mission complexity has increased over time, resulting in dramatic increases to the quantity of data produced by nanosatellites in orbit [65]. In addition, due to the small form-factor imposed by the CubeSat standard, major constraints and limitations are placed on the power generation, on-board computing and data downlink capabilities of nanosatellites [66].

Of particular concern is the constraint on data downlink, which would limit the rate at which data from the CubeSat can be retrieved. This impedes the generation of scientific value and delays the success of the full mission [65]. Therefore, a means to filter data so that the most useful data is sent to the ground first will prove beneficial for scientific value generation. By autonomously classifying data onboard the nanosatellite, the satellite itself can determine which data should be downlinked and which should be discarded or recaptured. To be truly valuable, autonomous classification should be done efficiently, respecting the power and compute constraints imposed on CubeSats.

#### 3.2.2 Hackathon Challenge Description

Therefore, the goal of the CubeSat Challenge was to use machine learning to accurately and efficiently classify the types of astronomy images captured by CubeSats. Participants were provided with an astronomical dataset and a baseline solution from a published study [64] and were tasked with creating a lightweight solution that improves on the accuracy and computational efficiency of the existing baseline.

The dataset used contained a total of 20,229 images, each with a size of 512×512×3, and labeled across five classes: Blurry, Corrupt, Missing Data, Noisy, and Priority. The imagery was designed to mimic data captured by VERTECS [67], a 6U astronomical nanosatellite, during operations. The dataset was initially divided using an 80:20 training-to-test split, resulting in 16,185 images for training and 4,044 for testing.

The test set from the original study was used as a completely blind ground truth dataset during the hackathon and was not available to participants. As for the training data from the original study (16,185 images), it was further split into 9,711 for training, 3,237 for validation, and 3,237 for internal testing.

In this paper, we only report model performance on the blind test set, which is identical to the one used in the original study.

#### 3.2.3 Baseline Model

The baseline method is an implementation of the CubeSatNet ultra-light CNN model [68] based on the approach used in [64]. It is less than 100 KB in size and can be easily deployed on ARM Cortex-based MCUs such as the Raspberry Pi CM4 included on the VERTECS Camera Control Board (CCB) [69]. From an accuracy perspective, CubeSatNet achieved a higher F1 score compared to other machine learning methods such as SVM, DBN, and AE trained to classify CubeSat images [68]. The ultimate goal of the model was to create a lightweight architecture without compromising classification accuracy.

This baseline implementation of CubeSatNet uses an 11-layer architecture, including four convolutional layers for feature extraction. The input layer is designed to handle standard 512×512 images. Max pooling layers are used to reduce the number of trainable parameters, thus lowering the computational load. The final layers include global average pooling and a dense layer with a SoftMax activation function for classification to one of the five classes.

### 3.3 Hackathon Structure Design

The nature of hackathons is that they are short in duration, which naturally leads to the development of early-stage prototypes at best. However, if these outcomes are to shift from simple prototypes to mature, innovative solutions, then it is necessary to rethink how hackathons themselves are designed and structured. However, this should not involve changing their core structure, such as by drastically extending their duration, as was done by Cohen et el. [56].

Figure 1 summarises our novel prototyping workflow. The journey begins two weeks before the hackathon, when participants are introduced to a well-defined challenge and its associated dataset, along with a research tips video that allows them to start exploring possible approaches in advance. During this stage independent parallel prototyping also takes place across participants from different regions. Once the hackathon begins, the process shifts into rapid iterative prototyping, driven by the event's time pressure and collaborative environment. At this point, both parallel prototyping within teams and competitive prototyping across regions also take place. Additionally, mentors in each regional hackathon play a key role in guiding participants through problem-solving and model refinement.

This novel prototyping workflow is introduced through three main criteria to increase solution maturity; First, as well-defined, open-ended ML challenge that teams could work on throughout the event. Second, a flipped-classroom approach that encouraged participants to engage with the hackathon materials, and even begin tackling the challenge, before the event. Third, a cascade model tailored to hackathons that enabled parallel development across regions and yielded more outputs.

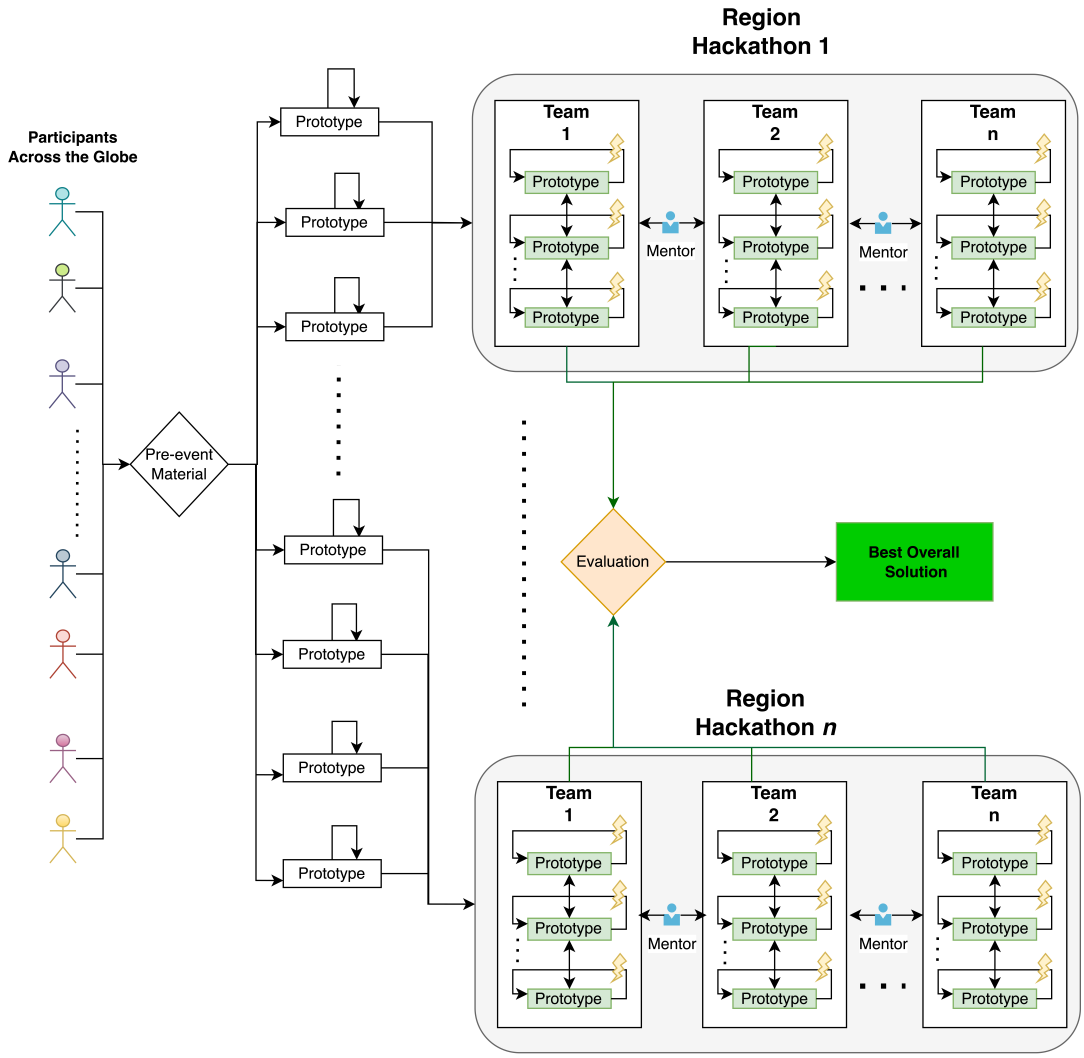


Figure 1: Schematic Representation of the Hackathon Prototyping Workflow

### 3.3.1 A Well Defined Challenge

In many hackathons, teams follow a process of steps which usually starts with teams identifying a specific problem or pain point within a general theme announced on the opening day [44,52,70]. These early activities consume time that could otherwise be used to develop and refine solutions. In our case, we fully defined the CubeSat challenge before the event. This allowed participants to focus on building more mature prototypes that exceeded the CubeSatNet baseline model.

### 3.3.2 The Flipped Classroom Model

To increase participants' skills and extend the time they spent tackling the problem, we drew inspiration from the flipped-classroom model. The flipped classroom is a student-centered instructional approach that encourages active learning [71, 72]. It is a methodology in which traditional teaching and learning activities, such as watching lectures or reading texts, are completed by students outside the classroom, while in-class time is reserved for interactive and collaborative tasks [72, 73].

In the context of our hackathons, tasks that could be automated or completed independently, such as reviewing the dataset, understanding the challenge, and exploring baseline models or alternative solutions, were moved to the pre-hackathon phase, allowing participants to complete them at their own pace. In contrast, activities that benefit from human interaction, such as sharing insights, comparing solutions, and refining models collaboratively, were reserved for the in-person regional hackathon.

In this cycle of the programme, the two most important shared materials were the introductory notebooks [74], which presented the hackathon challenge, datasets, and baseline, and introductory video which outlined different strategies [75]. A summary of these strategies is shown in Table 3 in Appendix A. All materials were intended to be shared two weeks before the regional hackathons, so participants across regional events could engage with:

1. **Challenge Familiarisation:** Participants were introduced to the hackathon challenge, becoming familiar with the dataset, baseline model, and evaluation metrics.
2. **Early Iterative Design:** Participants began refining their model design through individual experimentation, leading to early improvements in both accuracy and efficiency.
3. **Parallel Individual Prototyping:** Participants explored multiple solution paths independently before team formation. This broadened the design space and increased the potential for innovative approaches.

### 3.3.3 The Hackathon Cascade model

Increasing the number of independent teams competing to solve a challenge leads to an increase in the chances of finding a viable solution. This process aligns with what is referred to as broadcast search, a method commonly used by corporations to crowd-source solutions to R&D challenges that internal teams have been unable to resolve [76, 77].

To adapt this approach for hackathons, we propose the development of a new mechanism aimed at increasing the number of independent, competing models. We refer to this mechanism as the Hackathon Cascade Model.

The cascade model is a top-down, pyramidal approach used for knowledge transfer and human capacity development [78]. The term "cascade" reflects the downward flow of information, much like a waterfall [79, 80]. Typically, a small group of experts trains an initial cohort, who then go on to train others, multiplying the reach of the intervention [79].

The cascade model is known for its ability to enable large-scale change in a short amount of time, while also being highly cost-effective [81, 82]. As such, it has been widely adopted, especially in the education sector, where it is used to train teachers in novel techniques.

Using the cascade, multiple design dynamics take place during the implementation of the regional hackathon:

1. **Iterative prototyping** of individuals within teams as they refine their solutions;
2. Parallel prototyping among individuals in the same team, each exploring different design options independently;
3. **knowledge exchange** between participants within the same team;
4. **Competitive prototyping** across teams from all regional hackathons, each aiming to design the most accurate and efficient solution.

## 3.4 Evaluating Model Performance

For the hackathon, a Jupyter notebook was used to evaluate the performance of each machine learning solution. The notebook was designed to assess the full workflow, from data preprocessing to final predictions, ensuring that each solution was efficient and accurate.

While the original study evaluated its model using a Raspberry Pi CM4 [64], which is the MCU included on the VERTECS CCB [69], it was infeasible to provide each hackathon event with such hardware. Therefore, to emulate the constrained compute environment on a CubeSat as well as to ensure fair comparisons, each evaluation was run on a single CPU core. The notebook assessed the pipeline's overall performance, not just the final model's predictions.

The evaluation measured the following:

1. **Accuracy**: The percentage of correct predictions.
2. **F1 Score**: A balanced metric combining precision and recall, useful especially for imbalanced datasets.
3. **Confusion Matrix**: A detailed breakdown of predictions across the five classes.
4. **Evaluation Time**: The total time taken to preprocess the data and make predictions.
5. **Memory Usage**: The peak memory used during the pipeline execution.
6. **Algorithm Code Size**: The total size of the solution, including the serialized model and preprocessing functions, measured in megabytes (MB).

In this study, we focus on accuracy, F1 score, runtime, memory usage, CPU usage, and model size, all measured on the ground truth (blind test) data only.

## 3.5 Submission and Solution Evaluation Criteria

Alongside other submission materials, such as presentation slides, each team was required to submit one clear and comprehensive notebook that included all the steps taken to reach their final results. The notebook had to be written in a way that allowed anyone to run the code successfully. In addition, the trained model had to be included as part of the submission. Submitting more than one notebook or pipeline would lead to disqualification. All required materials (the notebook and the trained model) had to be submitted before the team's presentation on the final day of the hackathon.

The following list outlines the minimal requirements used to evaluate submitted models:

1. **Complete Submissions:** Only teams that submitted a working solution were considered.
2. **Alignment with the Challenge:** Submissions needed to directly address the original problem statement (e.g., working with 5 classes as defined in the challenge).
3. **Consistency Between Notebook and Slides:** The methods described in the slides (e.g., data augmentation) needed to match what was implemented in the notebook.
4. **Executable and Error-Free Code:** The code should run without errors that affect reproducibility. Example: a model expecting 3-channel images should not be paired with 1-channel data.
5. **Clean Data Handling:** Submissions needed to follow good data science practices, such as keeping training and testing data separate during preprocessing.
6. **High-Performing Models:** Due to the high number of submissions, only models with over 95% accuracy and those that outperformed the baseline in total run time were considered for analysis.

Based on the above criteria, 20 out of the 48 submitted solutions passed the minimum selection threshold and are presented in this paper. Additionally, many teams attempted to optimise their models through quantisation after developing an initial working version. However, since our evaluation process did not account for quantised models, only the original (pre-quantisation) versions were considered in the final analysis.

Among solutions that achieved at least 99% accuracy and used less than 13,000MB of memory, the winner was selected based on the lowest total run time, as this corresponded to the amount of power that would be used. The winning solution received a prize of 10,000 ZAR, funded by Development in Africa with Radio Astronomy (DARA).

# 4 Results

## 4.1 Regional Hackathons top Solutions

In this section, we present the 20 successful solutions that emerged from the regional hackathons and met the selection criteria outlined in Section 3.5. These criteria included the requirement that only methods achieving over 95% accuracy or outperforming the baseline in total run time were considered. We begin by presenting the teams behind the submitted models, along with their preprocessing methods, which are summarised in table 1. This is followed by a brief description of the models used, including architecture details for deep learning models where applicable. Finally, we provide an overview of the performance and efficiency metrics for all qualifying submissions, summarised in table 2 .

### 4.1.1 Pre-processing Techniques and Model Descriptions

Table 1 shows a variety of trained models accompanied by different pre-processing methods. Among the 20 submitted solutions, seven used non-deep learning models, specifically teams 01, 03, 05, 13, 16, 18, and 19. Three of these models were shallow neural networks, while the others used classical machine learning methods, including Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), and Stochastic Gradient Descent (SGD).

The remaining 13 submissions used deep learning approaches, all based on convolutional neural networks (CNNs) for image classification. Several teams (02, 10, 14, and 20) utilised MobileNetV2 as a feature extractor while frozen.

The rest used custom CNN architectures, often incorporating lightweight components such as separable convolutions or depthwise convolutions to reduce computational load and model size. These include teams 04, 06, 07, 08, 09, 11, 12, 15, and 17. These models typically followed a standard CNN architecture, multiple convolutional layers followed by pooling, global average pooling, and a dense output layer with a SoftMax activation function.

As for preprocessing, all solutions, except for the baseline method and team15, incorporated some form of preprocessing in their pipelines. Notably, both of these exceptions used deep learning methods. In contrast, The table clearly shows that the complexity of preprocessing increases in non-deep learning models. Among the remaining 18 solutions, the most common preprocessing steps included resizing , greyscaling and normalisation.

Non-deep learning methods often rely on feature extraction techniques. These included extracting features such as mean, entropy, quantiles, histograms, edge descriptors, and other statistical or texture-based characteristics, which were then used as direct inputs to the models. This design choice is reflected in the number of features, where non-deep learning models consistently used far fewer features compared to deep learning models.

### 4.1.2 Performance and Efficiency Metrics

Table 2 shows the performance and efficiency of the 20 solutions, all of which have been evaluated on the blind test set as described earlier.

The results in Table 2 show that five models achieved exceptional accuracy ($\geq 0.999$), with teams 20, 05, 04, 10, and 02 leading the results and slightly outperforming the baseline model (0.9985). Teams 20 and 05 reached perfect accuracy (1.0000), corresponding to a +0.15% improvement over the baseline. We also note that, despite the imbalance among the five data classes, the F1-score for all models closely matched their accuracy matrix, indicating balanced performance across classes.

In terms of execution time, team07 stood out as the fastest model, running 166× faster than the baseline (2.68 sec vs. 445.48 sec), with only a 0.74% drop in accuracy (0.9911 vs. 0.9985). Team09 also achieved a 46× speed-up while maintaining a high accuracy, just 0.25% below the baseline. Team03 performed nearly 33× faster, with a 0.12% reduction in accuracy.

The results also show that all teams that outperformed the baseline in accuracy also ran faster than the baseline model (445.48,sec). For instance, the two teams with perfect accuracy (teams 05 and 20), were substantially faster. Team05 ran over 15× faster (28.06,sec vs. 445.48,sec), while team20 was nearly 5× faster (92.46,sec vs. 445.48,sec).

As for comparing the execution time between models, figure 2 shows a histogram of $\log_{10}$ execution times, highlighting the difference between deep learning and non-deep learning models. The histogram reveals a wider spread of execution times among deep learning models, reflecting the variation between shallow, lightweight CNNs and more sophisticated architectures. In contrast, non-deep learning models exhibit a narrower distribution, indicating more consistent behavior and shorter, more efficient execution times overall.

Further comparison of deep learning models versus non-deep learning models can be seen in Figure 3, which shows a scatter plot of both approaches. A clear distinction can be observed between them.

Non-deep learning models, coloured in green, generally used far fewer features, often fewer than 2,000, and achieved relatively low execution times, highlighting their efficiency. These models also exhibited a linear trend on the y-axis, reflecting an efficient and predictable execution pattern, which corresponds to our earlier observations in Figure 2.

In contrast, deep learning models, coloured in blue, mostly operated on fixed-size image inputs (e.g., $128 \times 128$ or $224 \times 224$), resulting in feature counts clustering around 50,000 and higher. Their execution times varied more widely, suggesting that model architecture has a greater impact on performance than feature count alone. Notably, team07 managed to achieve the lowest execution time among all submissions despite using over 5,000 features, indicating strong architectural optimization.

The baseline model had the highest number of features and was among the slowest in execution time, reinforcing the challenge of balancing complexity and speed.

Figure 4 explores the relationship between execution time and the number of model parameters, with point colour representing accuracy. As expected, models with a larger number of parameters tend to have longer execution times. However, this is not always the case. For instance, team10 achieves high accuracy with over 1 million parameters but remains relatively efficient, while team15 shows high runtime despite having fewer parameters than the baseline.

A cluster of efficient models can be seen in the lower-left region, including team04, team08, team09, and team07, which maintain relatively low parameter counts and execution times while still achieving strong accuracy (darker yellow). Team07, in particular, stands out for combining a low parameter count and a very short runtime, while achieving accuracy very close to the baseline, highlighting it as a well-optimized solution.

Overall, the figure suggests that while more parameters often lead to increased runtime, accuracy is not strictly tied to model size, and careful architecture choices can yield compact, fast, and accurate models.

In terms of memory usage, several teams delivered solutions that consumed significantly less memory than the baseline model (12976.80,MB). We have Team07 again stood out, using just 6245.40,MB, over 2× smaller than the baseline. Other memory-efficient solutions included team01 (about 2× smaller), and team16 (about 1.6× smaller). On the other hand, a few models, such as team02, team04, team06, team08, and team12, used more memory than the baseline, all of which are deep learning methods.

When looking at code size, team01 achieved the smallest submission at just 0.0029,MB, which is over 400× smaller than the baseline model (1.1632,MB). Other compact solutions included team05 (about 110× smaller) and team16 (about 30× smaller). However, some models had substantially larger footprints, particularly team02, team10, and team14, whose models exceeded 7,MB, all of which used the frozen pre-trained layer MobileNetV2 in their model architecture.

Table 1: Overview of Teams, Preprocessing Steps, and Models Used

| **Team#** | **Contributors** | **Preprocessing** | **Model Name** |
|---|---|---|---|
| Baseline | Keenan A. A. Chatar, Ezra Fielding, et al. | No preprocessing | Conv2D(16→128, 3×3) + MaxPool ×4 → GlobalAvgPool → Dense(5, softmax) |
| team01 | Bakang Kgopolo, Mpho Carol Jan, Isaac Opio, Karabo K Ndebele | Greyscale → Crop → Resize → Extract 23 features (stats, texture) → Standardizatoin | SGDClassifier |
| team02 | Dihariniaina RABEARIMANANA, Nancy TOVOLAHY, Tahinasoa Misandratra RAZAFINTSALAMA | Resize to (224×224) → Normalize (/255) → Augment → Class balance | Dual-branch: MobileNetV2 (frozen) + custom CNN → GlobalAvgPool → Concat → Dense(128) → Dense(5, softmax) |
| team03 | Rojoniaina RASOAFARIHY, Toky Sandratra Miharimamy RASAMOELINA, i Hery Tiana RAKOTONDRAMIARANA | Greyscale → Resize (128×128) → Extract 8 features (stats, edges) | Random Forest |
| team04 | Rova Isaia Andriamamy, Harena Ranaivoson, Valdo Tsiaro Hasina Ndimbisoa | Resize to (128×128) → Normalize (/255) | SeparableConv2D(16→64, 3×3) + MaxPool ×2 → GlobalAvgPool → Dense(5, softmax) |
| team05 | Toavina LEONG POCKTSY, Tohavina RAOELINASIMBOLAMALALA | Greyscale → FeatExt (LBP histogram, 256 bins) | Linear SVM |
| team06 | RAHAJAHARIVONY ROJOTIANA FANDRESENA JESSICA, LIANTSOA RANDRIANASIMBOLARIVELO, RAFANOMEZANTSOA PIERROT | Normalize (/255) → Resize to (128×128) | SeparableConv2D(16→128, 3×3) + BN + ReLU + MaxPool ×4 → GlobalAvgPool → Dropout → Dense(5, softmax) |

| Team# | Contributors | Preprocessing | Model Name |
|---|---|---|---|
| team07 | Marcia Jennifer RAKOTOSON , Iriana Anthony Durandal Rasoanaivo, i Ilo Mahomby RANDRIANATOANDRO | Downsample using slicing (::12) → (42×42) | Conv2D(20) → DepthwiseConv2D(3×3, s=3) → Conv2D(28, 1×1) → DepthwiseConv2D(3×3, s=2) → Conv2D(36) → GAP → Dense(5, softmax) |
| team08 | Mirindra Randriamanantena, Jeanny Fidelica TSARAMANGA, Fanomezana Sarobidy RAZAKAHASINA | Resize to (128×128) → Normalize (/255) | Conv2D(8, s=2) → DWConv → Conv2D(16) → MaxPool → DWConv → Conv2D(32) → MaxPool → DWConv → Conv2D(64) → MaxPool → GAP → Dense(5, softmax) |
| team09 | Tsantaniony RAZAIARIMIHAJASOA, Rina Ralijaona | Resize to (128×128) | Conv2D(8, s=2) → 3× MobileNet-style bottlenecks → GlobalAvgPool → Dense(5, softmax) |
| team10 | Shain Mukungu, Gregorio Marychen, Landuleni Kashipulwa | Resize to (128×128) → Normalize (/255) | MobileNetV2 (frozen, $\alpha$=0.75) → GlobalAvgPool → Dense(128, ReLU) → Dropout → Dense(5, softmax) |
| team11 | Max Haikali, Sigrid Shilunga, Geno Gurirab | Resize (256×256) → CLAHE (L-channel) → Edge overlay → Normalize (/255) → Data Aug | Conv2D(16→128, 3×3) + MaxPool ×4 → GlobalAvgPool → Dense(64) → Dense(5, softmax) |
| team12 | Eunice dos anjos, Gilson G. Vicente, Rogério Langa | Resize to (128×128) | Conv2D(16, s=2) → 3× Bottleneck (expand + DWConv + project) → GlobalAvgPool → Dense(5, softmax) |
| team13 | Ezequias Miocha, Francisco Ernesto Uqueio, Milton Paulo Mbalate | Resize (128×128) → FeatExt: HOG + Color Hist (896 features) | Dense(64, ReLU) → Dense(5, softmax) |
| team14 | UWAMAHORO Leopord, DUSENGE Mediatrice, BAVUGAMENSHI Valens | Resize (224×224) → Normalize (/255) | MobileNetV2 (frozen) → GlobalAvgPool → Dense(5, softmax) |
| team15 | Silamlak Desye, Jemila Muhidin, Melat Kebede | No preprocessing | Conv2D(8→32, 3×3) + MaxPool ×3 → SeparableConv2D(64) + BN + MaxPool → GlobalAvgPool → Dense(5) |
| team16 | Addisu Mengistu, Chalie Lijalem, Turemo Bedaso | Greyscale → Resize (128×128) → Normalize (/255) → FeatExt (Quantiles, Sobel, Laplacian, HOG, LBP) → SMOTE → FeatRed | Logistic Regression |
| team17 | Paula Silva, Josephin Chamoun-Contreras | Adjust contrast/brightness → Normalize (/255) → Resize (95×95) → Greyscale → RGB | Conv2D(16→128, 3×3) + MaxPool ×4 → GlobalAvgPool → Dense(5, softmax) |
| team18 | Rubén José Carcamo, María Laura Oyarce | Greyscale → FeatExt (Fourier: Energy, Stats, Missing Ratio, Intensity STD) | Random Forest |
| team19 | C. E. Perez-Ramirez, Nousaba Nasrin Protiti | Greyscale → FeatExt (Laplacian mean/std, entropy, %zeros, mean) | ELM (5 input features → hidden layer → 5-class output) |
| team20 | Gabriel Christino, Marcela Gouvêa, Matheus Poltronieri | Resize (224×224) → Normalize to [-1, 1] | MobileNetV2 (frozen, $\alpha$=0.35) → GlobalAvgPool → Dense(5, softmax) |

Table 2: Performance and resource metrics of the top 20 solutions

| Team# | Total feat | v | time (sec) | memory (MB) | CPU | accuracy | f1_score | Code Size |
|---|---|---|---|---|---|---|---|---|
| Baseline | 786432 | 98085 | 445.48 | 12976.80 | 87.478 | 0.9985 | 0.9985 | 1.1632 |
| team01 | 23 | N/A | 36.73 | 6728.65 | 99.670 | 0.9842 | 0.9841 | 0.0029 |
| team02 | 150528 | 2450565 | 212.50 | 16426.17 | 95.750 | 0.9993 | 0.9993 | 11.3868 |
| team03 | 8 | N/A | 13.43 | 8595.25 | 99.148 | 0.9973 | 0.9973 | 1.8158 |
| team04 | 49152 | 3504 | 25.42 | 14687.11 | 71.740 | 0.9998 | 0.9998 | 0.0411 |
| team05 | 256 | N/A | 28.06 | 8649.85 | 99.588 | 1.0000 | 1.0000 | 0.0105 |
| team06 | 49152 | 13440 | 44.56 | 14859.14 | 76.853 | 0.9983 | 0.9983 | 0.2280 |
| team07 | 5547 | 2857 | 2.68 | 6245.40 | 52.030 | 0.9911 | 0.9911 | 0.0695 |
| team08 | 49152 | 4613 | 15.22 | 13718.76 | 76.288 | 0.9951 | 0.9951 | 0.1553 |
| team09 | 49152 | 2797 | 9.61 | 12940.55 | 88.653 | 0.9960 | 0.9960 | 0.1362 |
| team10 | 49152 | 1546677 | 46.30 | 9306.31 | 92.607 | 0.9995 | 0.9995 | 7.6959 |
| team11 | 196608 | 106021 | 167.89 | 21357.07 | 66.470 | 0.9993 | 0.9993 | 1.2586 |
| team12 | 49152 | 106981 | 24.81 | 14822.80 | 58.481 | 0.9751 | 0.9750 | 0.1528 |
| team13 | 896 | 57733 | 18.95 | 8772.23 | 99.381 | 0.9948 | 0.9948 | 0.2303 |
| team14 | 150528 | 2264389 | 214.10 | 14691.22 | 82.108 | 0.9985 | 0.9985 | 9.1702 |
| team15 | 786432 | 9013 | 265.45 | 12596.30 | 88.569 | 0.9993 | 0.9993 | 0.1495 |
| team16 | 1782 | N/A | 31.59 | 8289.06 | 99.721 | 0.9993 | 0.9993 | 0.0395 |
| team17 | 27075 | 98085 | 26.95 | 8323.79 | 94.607 | 0.9943 | 0.9943 | 1.1632 |
| team18 | 7 | N/A | 71.41 | 8325.00 | 99.834 | 0.9946 | 0.9946 | 2.2433 |
| team19 | 5 | N/A | 44.75 | 8341.16 | 100.427 | 0.9990 | 0.9990 | 0.1263 |
| team20 | 150528 | 416613 | 92.46 | 14575.18 | 79.206 | 1.0000 | 1.0000 | 2.1688 |

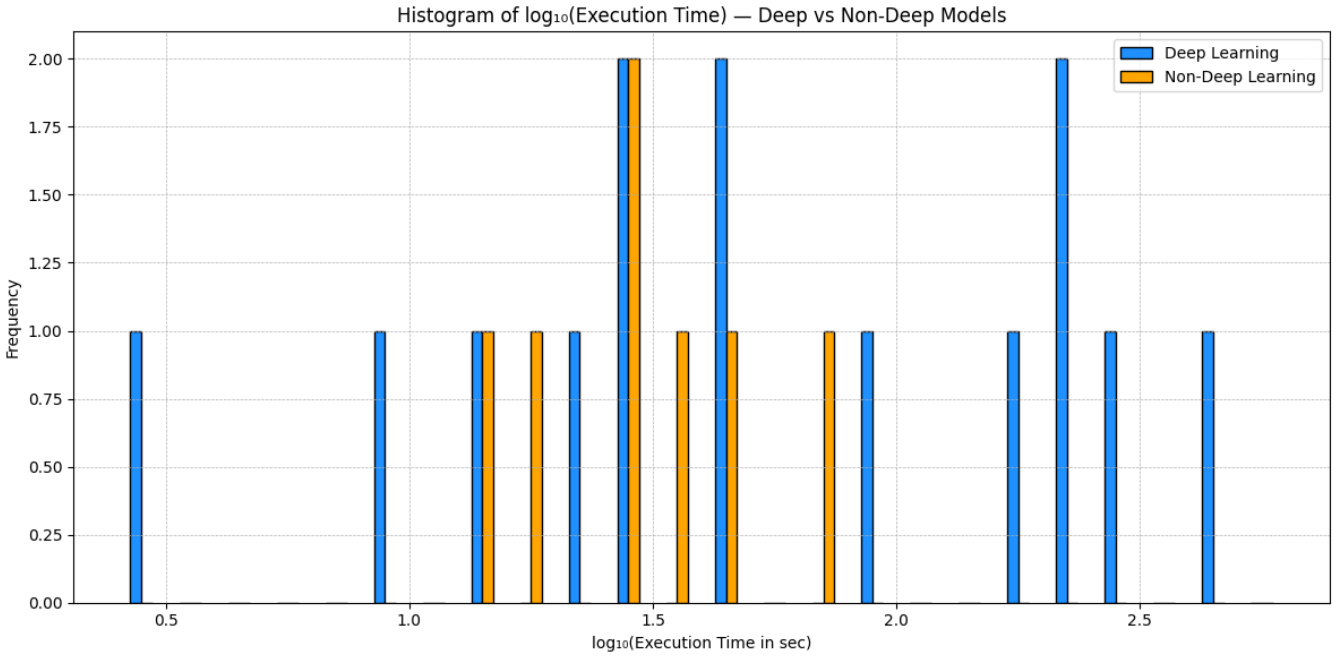


Figure 2: Histogram comparing the log execution time of deep learning and non-deep learning models.

## 4.2 Impact of Hackathon Design on Hackathon Solutions

In this section, we present results that show how the hackathon structure affected the solutions. The results came from the participant survey administered on the final day of each regional hackathon. In total, we received 120 survey

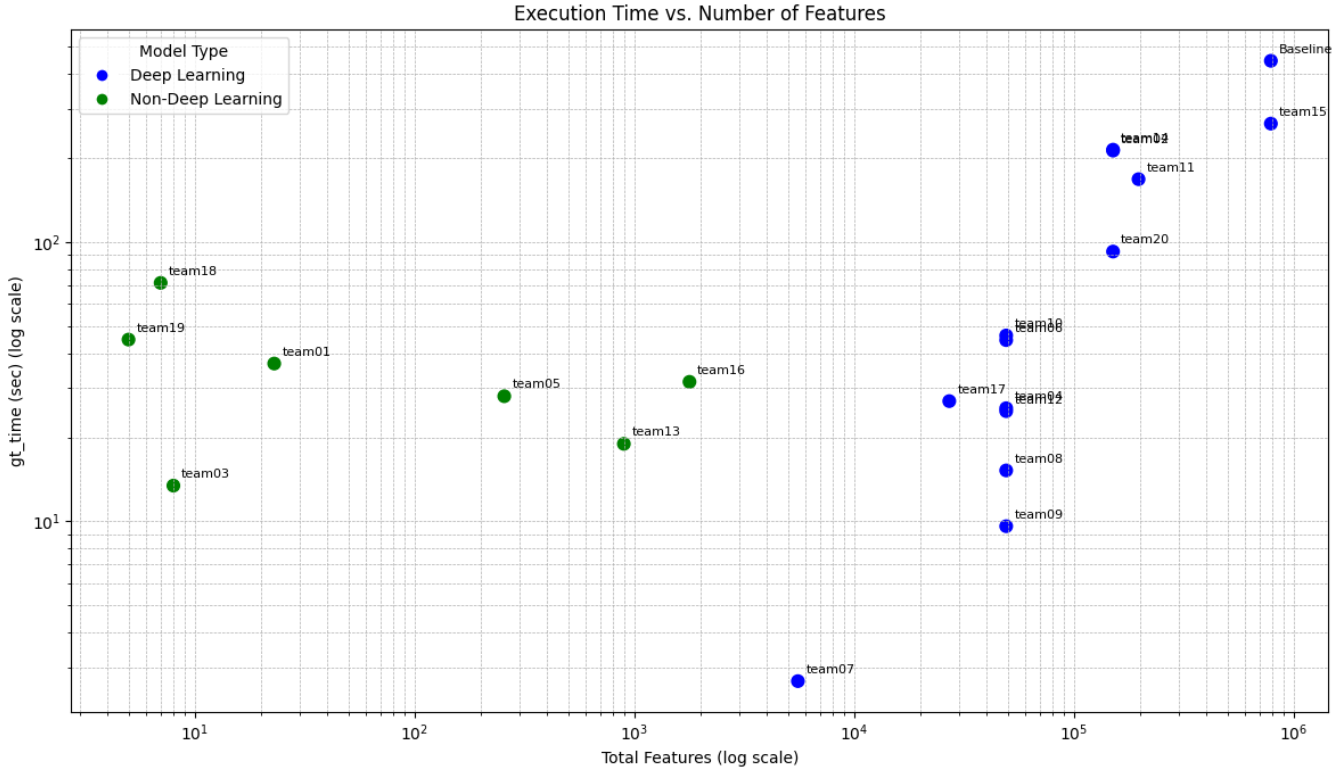


Figure 3: Scatter plot showing the relationship between execution time and the total number of features for deep learning and non-deep learning models.

responses out of 189 participants (120/189).

For the pre-event material (the building blocks of the flipped classroom), Figure 5 shows that about 65% of participants spent more than 4 hours on the materials, with 21%(25 participants) spending more than 9 hours. Not all participants received the materials a full two weeks in advance: only about 50% reported receiving them at the two-week mark, while others received them later due to late registrations or local logistics. Regarding perceived usefulness, Figure 6 shows that more than 30 participants rated both the challenge notebook and the video as "very" or "extremely" helpful.

For the hackathon environment overall, 80% of respondents reported that it encouraged them to think creatively and analyse problems in new ways. To gain greater insight, we then asked which hackathon aspect helped the most in solving the challenge. Three prominent answers emerged: mentor assistance, teamwork, and the pre-event materials. Out of 122 respondents, 27 mentioned mentor guidance (e.g., "The mentor guidance provided invaluable insights, clarified complex concepts, and helped refine our ideas."). Teamwork was cited by 47 respondents, highlighting diverse skills within teams and openness to each other's ideas (e.g., "Dividing tasks and sharing ideas helped us test approaches efficiently and find solutions faster."). The highest category was the pre-event materials, mentioned by 57 respondents, often for clarity and time savings (e.g., "The detailed problem statement and suggested resources helped us focus early and saved valuable time during the hackathon.").

Another key factor behind achieving solutions that exceeded the baseline was the iterative trial-and-error process that allowed teams to move quickly between approaches, as described by several teams in Appendix B. Many teams followed this adaptive workflow. For instance, teams 02, 04, 06, 07, 08, and 10 reported iterative testing cycles where models were continuously refined and compared based on early results. Team 07 (fastest running solution), in particular, held review sessions every two hours to evaluate progress and adjust architectures, while team 09 alternated between CNNs and classical ML methods before ultimately selecting a CNN due to its more promising performance

Several teams also combined background research with experimentation, using open-source notebooks, online tutorials, and even the pre-hackathon videos to guide their design choices (e.g., teams 03, 04, 11, 15, and 16). Some drew direct inspiration from the introductory video strategies, where images were represnted by a histogram. Team15, for example, optimized the baseline by reducing filter sizes and applying SeparableConv2D, while team16 discovered that logistic regression could perform competitively with minimal features.

This iterative process of rapid testing, switching between methods, and refining architectures enabled participants to explore a wide solution space in a short time. Further documentation of these processes can be found in Appendix B, which includes the short summary reports submitted by the top 20 regional teams.

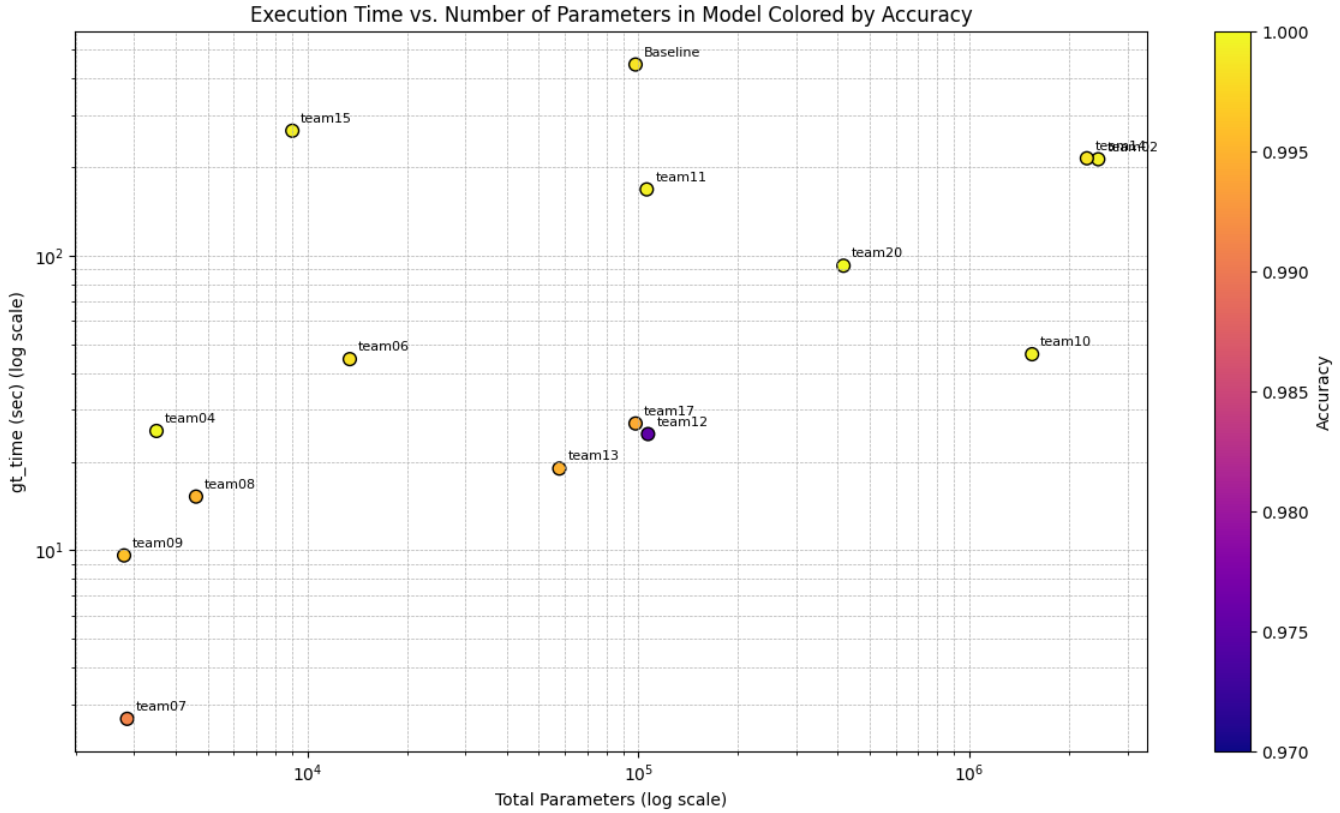


Figure 4: Execution time vs total parameters for the deep learning models, colored by accuracy.

# 5 Discussion

## 5.1 Data Science Solutions

There is a long debate in the literature arguing that deep learning models adopted in research are often unnecessarily complex, when in fact simpler models could perform just as well, and with greater efficiency [83]. The baseline method used in this study reflects this issue where an overly complicated model was used for the task at hand. By contrast, solutions from regional participants produced models that were simpler much more efficient while achieving comparable (or better) accuracy.

Unlike tasks such as face recognition in humans or animals, which require sophisticated deep architectures, our challenge involved identifying simple objects in the sky that fall into five categories: Priority, Blurry, Corrupt, Missing Data, and Noisy. This relatively simple classification problem does not necessarily benefit from deep or large-scale architectures. Furthermore, the nature of the CubeSat challenge requires models that are not only accurate but also efficient [64]. This was the main motivation behind the baseline model, which the authors described as an ultra-light CNN, featuring an 11-layer architecture, just 100 KB in size, and uses no preprocessing.

While two models from the regional hackathons achieved perfect accuracy (100%) with shorter runtimes than the baseline, we chose to highlight a different model for its practical trade-off: a much faster runtime while maintaining nearly the same accuracy. This was the solution submitted by team07, our winning model, which ran 166× faster than the baseline. Compared to the baseline's 11-layer architecture with 98,085 parameters, team07's model used only 2,857 parameters, representing a drastic reduction in complexity. It also consumed less than half the memory of the baseline model, making it highly suitable for deployment in resource-constrained environments, such as CubeSats.

In just three days, regional participants submitted 48 models. Of these, 20 met our evaluation criteria, achieving over 95% accuracy and running faster than the baseline. This shows that well-designed hackathons are not only for novel ideas; they can deliver novel, deployable solutions in very short timeframes.

Among the 20 qualifying models, both deep and non-deep learning strategies were represented, with some teams using classical methods like Random Forest and Logistic Regression, while others used CNNs, including pre-trained architectures such as MobileNetV2.

The fact that the task involved five different classes did not seem to negatively affect model performance. Accuracy and F1-scores were nearly identical, even though the dataset was imbalanced. For instance, the Priority class had about 7,460 images ( 36%), while the Corrupt class had only 1,338 ( 7%).

Several of the strategies outlined in Table 3 were clearly reflected in the submitted solutions. For example, many

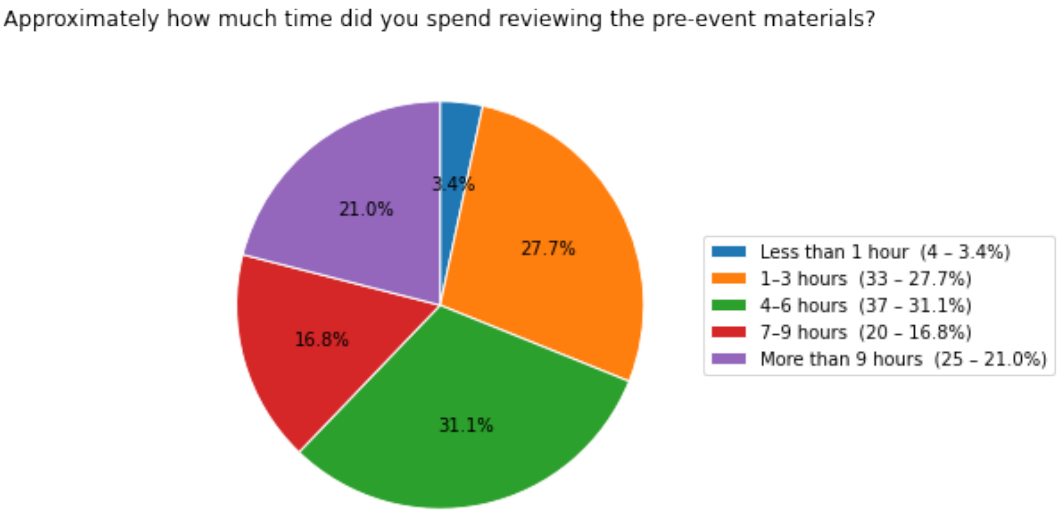


Figure 5: Time spent on pre-event materials.

teams began with pre-trained or shallower CNNs, aligning with the recommendation that a simpler method might be sufficient. As for the preprocessing teams used greyscaling, resizing, and intensity normalization, often avoiding excessive downscaling, as was advised in the video. However, it's worth noting that the winning solution actually did apply excessive downscaling, a strategy that worked well in their case but was not recommended in the video.

Non-deep learning teams frequently applied feature extraction methods such as quantiles, histograms, or edge descriptors, showing a clear link to the traditional ML strategies mentioned. That said, a few teams out of the 20 analysed solutions went in the opposite direction, using large frozen models that often led to much longer runtimes, contradicting the advice for compact design. As for ensemble methods were also suggested in the video, very few teams appeared to try them.

The video received close to 900 views on YouTube, and based on available insights such as survey responses, most participants did watch it. However, hackathons tend to encourage total creativity and freedom, so suggesting specific strategies might be perceived as limiting innovation [84]. Still, expert advice, often grounded in intuition built through years of experience which can save lots of time and effort. That said, the connection between the video and the participants was quite loose. Ultimately, it is the final testing on data that judged whether a strategy yields good results or not.

So the success of a solution depends heavily on the testing procedure each team adopted. Fast, iterative testing is what allows teams to converge toward better solutions in a short amount of time, a suggestion also emphasized in the video. All of this aligns with principles from design science: building effective prototypes depends not only on creative ideas but on the iterative refinement process behind them.

## 5.2 Time-Change Compression

Time is the most important defining feature of hackathons. Their short timeframe creates intensity and pressure, which can result in accelerating innovation. However, most hackathon literature ends up producing novel ideas or early prototypes. Our work shows that hackathons can go beyond this by generating deployable solutions, resulting in the compression of change through the compression of time. This shifts the focus from just acceleration to the introduction of the Time-Change Compression (TCC) concept.

In time–space compression, new technologies reduce the time it takes to overcome distance, allowing information, goods, and people to move faster. However, In TCC, well-designed short temporal frames enable more mature outcomes in less time, shortening the path to a specific desired change. In other words, it is the annihilation of time in attaining the highest order of change.

Humanity faces many challenges that demand faster problem-solving, creaeing natural pressure and a need to respond in an agile way. Technology is usually the default option because it can reduce time, for example by enabling quicker experimentation [85]. But technology's ability to compress time is only one part of the picture. In many cases, current incentives push technology toward financial returns rather than the most urgent social needs. Therefore, besides using or developing technology itself, we should explore other forms of time compression that deliver immediate change. In fact, a fuller picture is where technology is one piece of the puzzle, which can only make change more immediate and meaningful when guided by people. We can design compressed-time environments

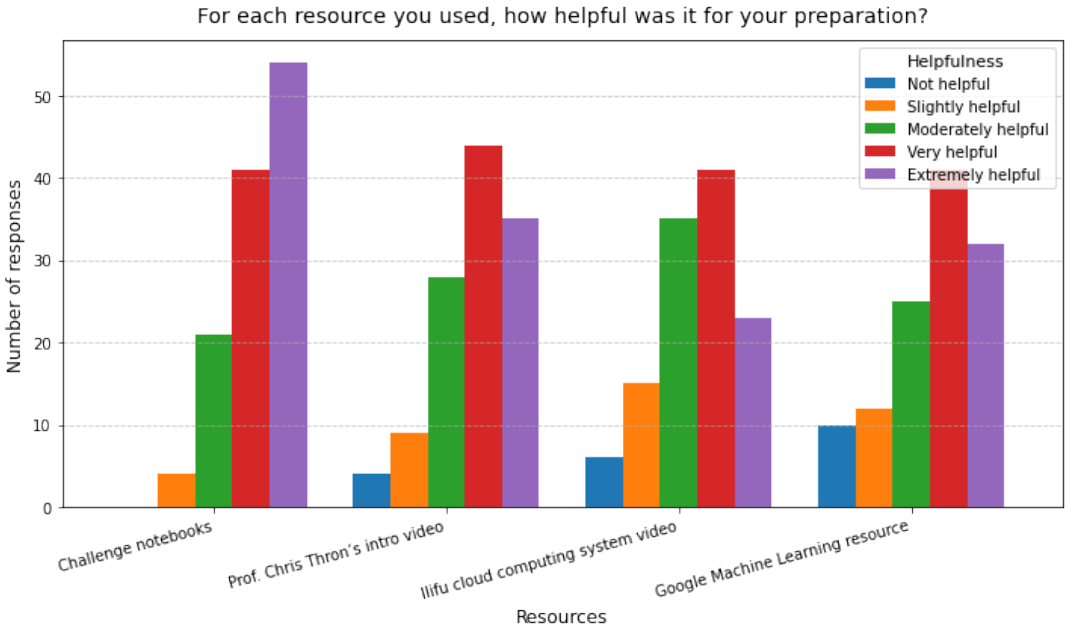


Figure 6: Perceived usefulness of pre-event materials.

where people with the right will, motivation, and skills work with the right advanced techniques and tools to compress work that would take years into days or even hours, to improve human well-being.

In this study, we showed that outcomes more mature than those commonly reported in hackathon literature can emerge from short, well-designed temporal structures. We achieved this by increasing the rate of iterative, parallel, and competitive prototyping, all guided by shared resources such as the research video, the notebook challenges, and other materials, and supported by computing power, and by selecting skilled young individuals.

It is hoped that further improvement to this new design structure will produce more mature outcomes in less time. One practical step is to run all regional hackathons in the same week, allowing maximum parallel experimentation, while also letting participants test prototypes in real or realistic contexts. For example, in this hackathon event, each team could have tested their prototypes on a Raspberry Pi 4, which mimics the constrained environment of CubeSats. Another step is to create a public board where team scores are posted so all teams can see them and try to improve [86]. These examples push solutions toward higher maturity by increasing the rate of iterative and parallel prototyping, moving from deployable to field-tested solutions.

Compared with existing literature, our approach follows a more structured design with the explicit aim of producing a more mature solutions that outperform a published baseline. Many hackathons described in the literature allow for a looser structure. For example, some hackathons pre-define multiple challenges and assign different teams to work on different problems [46, 47, 87]. Others adopt an even more flexible format, allowing participants to define their own challenges during the hackathon and then ideate possible solutions [25, 42–44, 50, 52, 60, 88–91]. Some also accept participants based only on interest, with no minimum skill requirement [86]. While these events are easier to implement and scale, their impact is typically limited to novel ideas or early prototypes.

# 6 Conclusion

Time is the defining characteristic of hackathons; it is the art of compressing activities into a short period to achieve ambitious outcomes for humanity. In this study, we show how hackathons can be advanced as a tool to deliver more mature solutions, moving from novel ideas to deployable solutions. Our novel prototyping mechanism through hackathons produced 20 deployable solutions, with the best running 166× faster than the baseline while maintaining the same accuracy. The solutions also reveal current trends and trajectories in machine learning and data science, as well as demonstrating how participants from different parts of the world apply these methods to solve challenges. For future studies, we suggest testing different short-time structures, together with techniques and tools that compress time, to achieve even higher-quality solutions in less time, resulting in even greater impact.

# References

[1] Philip Ross and Kasia Maynard. Towards a 4th industrial revolution, 2021.

[2] Joyful E. Mdhluli, Charles Takalana, Ramasamy Venugopal, Kevin Govender, Eslam Hussein, and Dominic Vertue. Astronomy as a strategic driver for sustainable development. *Nature Astronomy*, June 2025.

[3] Igor Chernov, Melchior Elsler, Thomas Maillart, Caterina Cacciatori, Simona Tavazzi, Bernd Manfred Gawlik, Yuliya Vystavna, Afroditi Anastasaki, Carolyn DuBois, Stuart Warner, et al. Innovative solutions for global water quality challenges: insights from a collaborative hackathon event. *Frontiers in Water*, 6:1363116, 2024.

[4] Anna Sigridur Islind and Livia Norström. Learning sustainable work through critical design: a case study of a hackathon to prepare the future workforce. *Journal of Workplace Learning*, 32(8):641–651, 2020.

[5] Tomoko Yokoi, Nikolaus Obwegeser, and Michela Beretta. Crisis innovation: Leveraging virtual hackathons for rapid ideation. *IEEE Transactions on Engineering Management*, 2021.

[6] Kiev Gama et al. The developers' design thinking toolbox in hackathons: a study on the recurring design methods in software development marathons. *International Journal of Human–Computer Interaction*, 39(12):2269–2291, 2023.

[7] Issam Attalah, Petra A Nylund, and Alexander Brem. Who captures value from hackathons? Innovation contests with collective intelligence tools bridging creativity and coupled open innovation. *Creativity and Innovation Management*, 32(2):266–280, 2023.

[8] Gerard Briscoe and Catherine Mulligan. Digital innovation: The hackathon phenomenon.(2014). *Google Scholar There is no corresponding record for this reference*, 2014.

[9] Connie W. Chau and Elizabeth M. Gerber. On hackathons: A multidisciplinary literature review. In *CHI 2023 - Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Conference on Human Factors in Computing Systems - Proceedings. Association for Computing Machinery, April 2023. Publisher Copyright: © 2023 ACM.; 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023 ; Conference date: 23-04-2023 Through 28-04-2023.

[10] Ben Heller, Atar Amir, Roy Waxman, and Yossi Maaravi. Hack your organizational innovation: literature review and integrative model for running hackathons. *Journal of Innovation and Entrepreneurship*, 12(1):6, 2023.

[11] Jerôme Aboab, Leo Anthony Celi, Peter Charlton, Mengling Feng, Mohammad Ghassemi, Dominic C Marshall, Louis Mayaud, Tristan Naumann, Ned McCague, Kenneth E Paik, et al. A "datathon" model to support cross-disciplinary collaboration. *Science Translational Medicine*, 8(333):333ps8–333ps8, 2016.

[12] Yejin Kim, Xiaoqian Jiang, Samden D Lhatoo, Guo-Qiang Zhang, Shiqiang Tao, Licong Cui, Xiaojin Li, Robert D Jolly, Luyao Chen, Michael Phan, et al. A community effort for automatic detection of postictal generalized EEG suppression in epilepsy, 2020.

[13] Nada Endrissat. Hackathons: A field of dreams for 'collaborative innovation'?: A review of recent studies. *Entreprendre & innover*, 38(3):69–75, 2018.

[14] Frank J Frey and Michael Luks. The innovation-driven hackathon: one means for accelerating innovation. In *Proceedings of the 21st European Conference on Pattern Languages of Programs*, pages 1–11, 2016.

[15] Kristian R Olson, Madeline Walsh, Priya Garg, Alexis Steel, Sahil Mehta, Santorino Data, Rebecca Petersen, Anthony J Guarino, Elizabeth Bailey, and David R Bangsberg. Health hackathons: theatre or substance? A survey assessment of outcomes from healthcare-focused hackathons in three countries. *BMJ innovations*, 3(1), 2017.

[16] Workneh Ayele, Gustaf T Juell-Skielse, Anders Hjalmarsson, and Paul Johanneson. Evaluating open data innovation: a measurement model for digital innovation contests. 2015.

[17] Stefano Franco, Angelo Presenza, and Antonio Messeni Petruzzelli. Boosting innovative business ideas through hackathons. The "Hack for Travel" case study. *European Journal of Innovation Management*, 25(6):413–431, 2022.

[18] Cristian Granados and Montserrat Pareja-Eastaway. How do collaborative practices contribute to innovation in large organisations? The case of hackathons. *Innovation*, 21(4):487–505, 2019.

[19] Fotis Kitsios and Maria Kamariotou. Digital innovation and open data hackathons success: A comparative descriptive study. In *Proceedings of the 2019 European Triple Helix Congress on Responsible Innovation & Entrepreneurship (EHTAC2019), Thessaloniki, Greece*, pages 20–25, 2019.

[20] Fotis Kitsios and Maria Kamariotou. Digital innovation and entrepreneurship through open data-based platforms: Critical success factors for hackathons. *Heliyon*, 9(4), 2023.

[21] Waqaas A Butt, Amir Shariff, Sadaf Khan, and Asad I Mian. Global surgery hackathons: A case study from pakistan. *Surgical innovation*, 28(4):496–501, 2021.

[22] Julie K Silver, David S Binder, Nevena Zubcevik, and Ross D Zafonte. Healthcare hackathons provide educational and innovation opportunities: a case study and best practice recommendations. *Journal of medical systems*, 40(7):177, 2016.

[23] Arlin Stoltzfus, Hilmar Lapp, Naim Matasci, Helena Deus, Brian Sidlauskas, Christian M Zmasek, Gaurav Vaidya, Enrico Pontelli, Karen Cranston, Rutger Vos, et al. Phylotastic! Making tree-of-life knowledge accessible, reusable and convenient. *BMC bioinformatics*, 14(1):158, 2013.

[24] Anja Remshagen and Kim C. Huett. Youth hackathons in computing for the community: A design case. *TechTrends*, 67(3):508–520, 2023.

[25] Karen Day, Gayl Humphrey, Sophie Cockcroft, et al. How do the design features of health hackathons contribute to participatory medicine? *Australasian Journal of Information Systems*, 21, 2017.

[26] Anders Hjalmarsson, Paul Johannesson, Gustaf Juell-Skielse, and Daniel Rudmark. Beyond innovation contests: A framework of barriers to open innovation of digital services. In *ECIS 14 22nd European Conference on Information Systems, Tel Aviv, 5-13 June 2014*, 2014.

[27] Alexander Nolte, Irene-Angelica Chounta, and James D. Herbsleb. What happens to all these hackathon projects?: Identifying factors to promote hackathon project continuation. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2):1–26, 2020.

[28] Ari Happonen, Daria Minashkina, Alexander Nolte, and Maria Angelica Medina Angarita. Hackathons as a company–university collaboration tool to boost circularity innovations and digitalization enhanced sustainability. In *AIP Conference Proceedings*, volume 2233. AIP Publishing, 2020.

[29] Alar Leemet, Fredrik Milani, and Alexander Nolte. Utilizing hackathons to foster sustainable product innovation–the case of a corporate hackathon series. in 2021 ieee/acm 13th international workshop on cooperative and human aspects of software engineering (chase)(pp. 51–60), 2021.

[30] Jeanette Falk, Alexander Nolte, Daniela Huppenkothen, Marion Weinzierl, Kiev Gama, Daniel Spikol, Erik Tollerud, Neil Chue Hong, Ines Knäpper, and Linda Bailey Hayden. The future of hackathon research and practice. *arXiv preprint arXiv:2211.08963*, 2022.

[31] CM Ibegbulam, JA Olowonubi, SA Fatounde, and OA Oyegunwa. Artificial intelligence in the era of 4IR: drivers, challenges and opportunities. *Engineering Science & Technology Journal*, 4(6):473–488, 2023.

[32] Deepti Goyal, Richa Goyal, G Rekha, Shaveta Malik, and Amit Kumar Tyagi. Emerging trends and challenges in data science and big data analytics. In *2020 International conference on emerging trends in information technology and engineering (ic-ETITE)*, pages 1–8. IEEE, 2020.

[33] Tao An. Science opportunities and challenges associated with SKA big data. *Science China Physics, Mechanics & Astronomy*, 62:1–6, 2019.

[34] Mukund Deshpande, Dhruva Ray, Sameer Dixit, and Avadhoot Agasti. Shareinsights: An unified approach to full-stack data processing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1925–1940, 2015.

[35] Francisco Bernardo, Mick Grierson, and Rebecca Fiebrink. User-centred design actions for lightweight evaluation of an interactive machine learning toolkit. *Journal of Science and Technology of the Arts*, 10(2):25–38, 2018.

[36] Mattia Zeni, Ivano Bison, Fernando Reis, Britta Gauckler, and Fausto Giunchiglia. Improving time use measurement with personal big data collection–the experience of the european big data hackathon 2019. *Journal of Official Statistics*, 37(2):341–365, 2021.

[37] Nicola Plant, Clarice Hilton, Marco Gillies, Rebecca Fiebrink, Phoenix Perry, Carlos González Díaz, Ruth Gibson, Bruno Martelli, and Michael Zbyszynski. Interactive machine learning for embodied interaction design: A tool and methodology. In *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 1–5, 2021.

[38] Jane Wyngaard, Heather Lynch, Jaroslaw Nabrzyski, Allen Pope, and Shantenu Jha. Hacking at the divide between polar science and hpc: using hackathons as training tools. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 352–359. IEEE, 2017.

[39] Mataroria P Lyndon, Michael P Cassidy, Leo Anthony Celi, Luk Hendrik, Yoon Jeon Kim, Nicholas Gomez, Nathaniel Baum, Lucas Bulgarelli, Kenneth E Paik, and Alon Dagan. Hacking hackathons: preparing the next generation for the multidisciplinary world of healthcare technology. *International Journal of Medical Informatics*, 112:1–5, 2018.

[40] Daniela Huppenkothen, Anthony Arendt, David W Hogg, Karthik Ram, Jacob T VanderPlas, and Ariel Rokem. Hack weeks as a model for data science education and collaboration. *Proceedings of the National Academy of Sciences*, 115(36):8872–8877, 2018.

[41] Antonella Longo, Marco Zappatore, Angelo Martella, and Chiara Rucco. Enhancing data education with datathons: an experience with open data on renewable energy systems. In *Proceedings of the 1st International Workshop on Data Systems Education*, pages 26–31, 2022.

[42] Taylor D Sparks, Frank E Curtis, Daniel C Fredrickson, and Nicole A Benedek. Insights and Innovations from the SSMCDAT 2023: Bridging Solid-State Materials Chemistry and Data Science, 2024.

[43] J Aboab, LA Celi, P Charlton, M Feng, M Ghassemi, DC Marshall, L Mayaud, T Naumann, N McCague, KE Paik, et al. A" datathon" model to support cross-disciplinary collaboration. sci transl med. 2016 apr 06; 8 (333): 333ps8. 10.1126/scitranslmed. aad9072.

[44] Leo Anthony Celi, Sharukh Lokhandwala, Robert Montgomery, Christopher Moses, Tristan Naumann, Tom Pollard, Daniel Spitz, and Robert Stretch. Datathons and software to promote reproducible research. *Journal of Medical Internet Research*, 18(8):e6365, 2016.

[45] Marianne A Messelink, Nadia MT Roodenrijs, Bram van Es, Cornelia AR Hulsbergen-Veelken, Sebastiaan Jong, L Malin Overmars, Leon C Reteig, Sander C Tan, Tjebbe Tauber, Jacob M van Laar, et al. Identification and prediction of difficult-to-treat rheumatoid arthritis patients in structured and unstructured routine care data: results from a hackathon. *Arthritis Research & Therapy*, 23:1–10, 2021.

[46] Andrew L Ferguson, Tim Mueller, Sanguthevar Rajasekaran, and Brian J Reich. Conference report: 2018 materials and data science hackathon (matdat18). *Molecular Systems Design & Engineering*, 4(3):462–468, 2019.

[47] Organizing Committee of the Madrid, Antonio Núñez Reiz, Fernando Martínez Sagasti, Manuel Álvarez González, Antonio Blesa Malpica, Juan Carlos Martín Benítez, Mercedes Nieto Cabrera, Ángela del Pino Ramírez, José Miguel Gil Perdomo, Jesús Prada Alonso, et al. Big data and machine learning in critical care: Opportunities for collaborative research. *Medicina intensiva*, 43(1):52–57, 2019.

[48] Dmitrii O Shkil, Alina A Muhamedzhanova, Philipp I Petrov, Ekaterina V Skorb, Timur A Aliev, Ilya S Steshin, Alexander V Tumanov, Alexander S Kislinskiy, and Maxim V Fedorov. Expanding predictive capacities in toxicology: Insights from hackathon-enhanced data and model aggregation. *Molecules*, 29(8):1826, 2024.

[49] Joachim Schaeffer, Paul Gasper, Esteban Garcia-Tamayo, Raymond Gasper, Masaki Adachi, Juan Pablo Gaviria-Cardona, Simon Montoya-Bedoya, Anoushka Bhutani, Andrew Schiek, Rhys Goodall, et al. Machine learning benchmarks for the classification of equivalent circuit models from electrochemical impedance spectra. *Journal of The Electrochemical Society*, 170(6):060512, 2023.

[50] Hampton L Leonard, Ruqaya Murtadha, Alejandro Martinez-Carrasco, Alina Jama, Amica Corda Müller-Nedebock, Ana-Luisa Gil-Martinez, Anastasia Illarionova, Anni Moore, Bernabe I Bustos, Bharati Jadhav, et al. The IPDGC/GP2 Hackathon—an open science event for training in data science, genomics, and collaboration using Parkinson's disease data. *npj Parkinson's Disease*, 9(1):33, 2023.

[51] Justina Mandravickaitė, Milita Songailaitė, Eglė Kankevičiūtė, Anton Volčok, and Tomas Krilavičius. Disinformation analysis and tracking dashboard. In *2023 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–6. IEEE, 2023.

[52] Siqi Liu, Qianwen Stephanie Ko, Kun Qiang Amos Heng, Kee Yuan Ngiam, and Mengling Feng. Healthcare transformation in Singapore with artificial intelligence. *Frontiers in Digital Health*, 2:592121, 2020.

[53] William K Jones, Redouane Lguensat, Anastase Charantonis, and Duncan Watson-Parris. The 2020 climate informatics hackathon: Generating nighttime satellite imagery from infrared observations. In *Proceedings of the 10th International Conference on Climate Informatics*, pages 134–138, 2020.

[54] James J Morrison, Jason M Hostetter, Abhi Aggarwal, and Ross W Filice. Constructing a computer-aided differential diagnosis engine from open-source APIs. *Journal of digital imaging*, 29(6):654–657, 2016.

[55] Andrea Navas-Olive, Adrian Rubio, Saman Abbaspoor, Kari L Hoffman, and Liset M de la Prida. A machine learning toolbox for the analysis of sharp-wave ripples reveals common waveform features across species. *Communications Biology*, 7(1):211, 2024.

[56] Seffi Cohen, Noa Dagan, Nurit Cohen-Inger, Dan Ofer, and Lior Rokach. ICU survival prediction incorporating test-time augmentation to improve the accuracy of ensemble-based models. *IEEE Access*, 9:91584–91592, 2021.

[57] Meagan Flus and Ada Hurst. Design at hackathons: new opportunities for design research. *Design Science*, 7:e4, 2021.

[58] TJ Jaśkiewicz, Ingrid Mulder, Nicola Morelli, and Janice S Pedersen. Hacking the hackathon format to empower citizens in outsmarting "smart" cities. 2019.

[59] B Camburn, V Viswanathan, J Linsey, D Anderson, D Jensen, R Crawford, K Otto, and K Wood. Design prototyping methods: state of the art in strategies, techniques, and guidelines. des sci 3: e13, 2017.

[60] Jake Rowan Byrne, Katriona O'Sullivan, and Kevin Sullivan. An IoT and wearable technology hackathon for promoting careers in computer science. *IEEE Transactions on Education*, 60(1):50–58, 2016.

[61] Verena Schürmann, Daniel Bodemer, and Nicki Marquardt. Applying debriefings in the context of higher education: How joint reflection fosters students' collaborative problem solving. *Social Psychology of Education*, 28(1):62, 2025.

[62] Sourobh Ghosh and Andy Wu. Iterative coordination and innovation: Prioritizing value over novelty. *Organization Science*, 34(6):2182–2206, 2023.

[63] Hank Heidt, Jordi Puig-Suari, Augustus Moore, Shinichi Nakasuka, and Robert Twiggs. CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation. In *Proceedings of the AIAA/USU Conference on Small Satellites*, Technical Session V: Lessons Learned - In Success and Failure, pages SSC00–V–5, 2000. `https://digitalcommons.usu.edu/smallsat/2000/All2000/32/`.

[64] Keenan AA Chatar, Ezra Fielding, Kei Sano, and Kentaro Kitamura. Data downlink prioritization using image classification on-board a 6U cubesat. In *Sensors, Systems, and Next-Generation Satellites XXVII*, volume 12729, pages 129–142. SPIE, 2023.

[65] Ezra Fielding and Akitoshi Hanazawa. Flexible natural language-based image data downlink prioritization for nanosatellites. *Aerospace*, 11(11), 2024.

[66] Keenan Chatar, Kentaro Kitamura, and Mengu Cho. Onboard data prioritization using multi-class image segmentation for nanosatellites. *Remote Sensing*, 16(10), 2024.

[67] Kei Sano, Takao Nakagawa, Shuji Matsuura, Koji Takimoto, Aoi Takahashi, Tetsuhito Fuse, Rodrigo Cordova, Victor Schulz, Pooja Lepcha, Necmi Cihan Örger, Daisuke Nakayama, Joseph Ofosu, Reynel Josue Galindo Rosales, Eyoas Areda, Pema Zangmo, Ezra Fielding, Keenan Chatar, Yukihisa Otani, Hisataka Kawasaki, Bastien Morelle, John Paul Almonte, Shunsuke Nakagawa, Yuto Tome, Shohei Karaki, Chinathip Narongphun, Hari Ram Shrestha, Marco Rosa, David Dai, Wenceslao Bejarano, Akihiro Ikeda, Rin Sato, Kentaro Hayashida, Hiroki Miyagawa, Masahiro Nishioka, Kana Kurosaki, Isami Kato, Satoshi Ikari, Kohji Tsumura, Ichiro Jikuya, Hideo Matsuhara, Umi Enokidani, Hayato Tanaka, Yuki Hirose, Akimasa Ojika, Akane Tsumoto, Taiko Iwaki, Yuki Ohara, Mengu Cho, Kentaro Kitamura, Hirokazu Masui, Mariko Teramoto, Takashi Yamauchi, Ryo Hashimoto, Emino Fukumoto, Zamba Leonel, Arisa Oho, Shoki Yabumoto, Hayato Masuno, Chisato Arakawa, Kouta Miyamoto, Takehiko Wada, Naoki Isobe, Yasuyuki Miyazaki, Ryu Funase, Hajime Kawahara, Keiichi Hirako, Yoichi Yatsu, and Yoshihide Aoyanagi. Astronomical 6U CubeSat mission VERTECS: scientific objective and project status. In Laura E. Coyle, Shuji Matsuura, and Marshall D. Perrin, editors, *Space Telescopes and Instrumentation 2024: Optical, Infrared, and Millimeter Wave*, volume 13092, page 130920W. International Society for Optics and Photonics, SPIE, 2024.

[68] Abhas Maskey and Mengu Cho. Cubesatnet: Ultralight convolutional neural network designed for on-orbit binary image classification on a 1U cubesat. *Engineering Applications of Artificial Intelligence*, 96:103952, 2020.

[69] Ezra Fielding, Victor H. Schulz, Keenan A. A. Chatar, Kei Sano, and Akitoshi Hanazawa. VERTECS: A COTS-based payload interface board to enable next generation astronomical imaging payloads. In Jorge Ibsen and Gianluca Chiozzi, editors, *Software and Cyberinfrastructure for Astronomy VIII*, volume 13101, page 131010J. International Society for Optics and Photonics, SPIE, 2024.

[70] JS Bell, FE Murray, and EL Davies. An investigation of the features facilitating effective collaboration between public health experts and data scientists at a hackathon. *Public Health*, 173:120–125, 2019.

[71] Gökçe Akçayır and Murat Akçayır. The flipped classroom: A review of its advantages and challenges. *Computers & Education*, 126:334–345, 2018.

[72] Héctor Galindo-Dominguez. Flipped classroom in the educational system. *Educational Technology & Society*, 24(3):44–60, 2021.

[73] Jacob Bishop and Matthew A Verleger. The flipped classroom: A survey of the research. In *2013 ASEE annual conference & exposition*, pages 23–1200, 2013.

[74] Hack4dev. Cubesat_imageclassify: Building an efficient ML pipeline for CubeSats using minimal resources, 2025.

[75] Hack4dev. Cubesat challenge. `https://www.youtube.com/watch?v=3N9-eGfQiS0`, 2025.

[76] Thomas Gegenhuber and Marko Hrelja. Broadcast search in innovation contests: Case for hybrid models. *arXiv preprint arXiv:1204.3343*, 2012.

[77] Karim R Lakhani. Broadcast search in problem solving: Attracting solutions from the periphery. In *2006 Technology Management for the Global Future-PICMET 2006 Conference*, volume 6, pages 2450–2468. IEEE, 2006.

[78] André Moulakdi and Yamina Bouchamma. Professional development for primary school teachers in Cameroon: Is the cascade PD model effective? *Creative Education*, 11(7):1129–1144, 2020.

[79] Thanassis Karalis. Cascade approach to training: Theoretical issues and practical applications in non-formal education. *Journal of Education & Social Policy*, 3(2):104–108, 2016.

[80] Linda Gask, Nia Coupe, and Gillian Green. An evaluation of the implementation of cascade training for suicide prevention during the 'Choose Life' initiative in Scotland—utilizing Normalization Process Theory. *BMC health services research*, 19(1):588, 2019.

[81] Fay Turner, Simon Brownhill, and Elaine Wilson. The transfer of content knowledge in a cascade model of professional development. *Teacher development*, 21(2):175–191, 2017.

[82] Harry Kipkemoi Bett. The cascade model of teachers' continuing professional development in Kenya: A time for change? *Cogent Education*, 3(1):1139439, 2016.

[83] Eslam A Hussein, Mehrdad Ghaziasgar, Christopher Thron, Mattia Vaccari, and Antoine Bagula. Basic statistical estimation outperforms machine learning in monthly prediction of seasonal climatic parameters. *Atmosphere*, 12(5):539, 2021.

[84] Elena Romeo-Arroyo, Davide Giacalone, Christina J Birke Rune, Agnieszka Kita, Anna Michalska-Ciechanowska, Malgorzata Korzeniowska, Ángel A Carbonell-Barrachina, Luis Noguera-Artiaga, María Mora, and Laura Vázquez-Araújo. Open innovation with sensory and consumer science: Hackathon as a tool for academic industry-cooperation. *Journal of Sensory Studies*, 40(1):e70025, 2025.

[85] Hila Lifshitz-Assaf, Sarah Lebovitz, and Lior Zalmanson. Minimal and adaptive coordination: How hackathons' projects accelerate innovation without killing it. *Academy of Management Journal*, 64(3):684–715, 2021.

[86] Mate Kisantal, Sumant Sharma, Tae Ha Park, Dario Izzo, Marcus Märtens, and Simone D'Amico. Satellite pose estimation challenge: Dataset, competition design, and results. *IEEE Transactions on Aerospace and Electronic Systems*, 56(5):4083–4098, 2020.

[87] Stephanie Soliz and Sean D Young. Feasibility of a citizen-driven hackathon to increase public engagement and solutions to address the opioid crisis. *Journal of substance use*, 25(6):615–620, 2020.

[88] Jabu Mtsweni and Hanifa Abdullah. Stimulating and maintaining students' interest in computer science using the hackathon model. *The Independent Journal of Teaching and Learning*, 10(1):85–97, 2015.

[89] Jason K Wang, Ravinder D Pamnani, Robson Capasso, and Robert T Chang. An extended hackathon model for collaborative education in medical innovation. *Journal of medical systems*, 42(12):239, 2018.

[90] Nomita Divi and Mark Smolinski. Epihacks, a process for technologists and health experts to cocreate optimal solutions for disease prevention and control: user-centered design approach. *Journal of Medical Internet Research*, 23(12):e34286, 2021.

[91] Justine Lobbe, Florence Bazzaro, and Jean-Claude Sagot. Innovation in collaborative design: an exploratory study in hackathon. *International Journal of Design Creativity and Innovation*, 9(2):119–137, 2021.

# A Appendix: Table

| Category | Strategy |
|---|---|
| **General** | • Start with simple (underperforming) or complex (overdesigned) models and adjust accordingly.<br>• Always make **choices based on reasoning**, not randomness.<br>• Use **preliminary calculations** to guide design decisions (e.g., complexity, MACs).<br>• **Streamline test procedures** for fast iteration and comparison.<br>  – Use **ChatGPT** with specific queries to generate and adapt useful exploratory/test code.<br>• **Look at data** (visuals, class imbalance, labeling issues) before designing solutions.<br>• Evaluate **trade-offs** (speed vs. accuracy, complexity vs. interpretability). |
| **Pre-processing** | • **Downscale** images moderately (e.g., 512→256 instead of 512→64).<br>• Try **cropping image sections** (e.g., 4 corners) to preserve resolution.<br>• Consider using **only 1 channel** instead of RGB or grayscale.<br>• Explore **intensity histograms / quantiles** as compressed representations.<br>• Use **edge detection** (Sobel, Canny) or **OpenCV** to extract features like star boundaries. |
| **Machine Learning** | • Use **flattened inputs** like quantiles or histograms for non-spatial models.<br>• Include zero intensities to help detect missing data.<br>• Try **faster models** like Logistic Regression or SGDClassifier.<br>• Explore combining models: **ensembles, sequential, or voting-based methods**.<br>• Balance classes in the training set to reduce bias (especially for weak models). |
| **Deep Learning** | • Use **CNNs** (not RNNs/LSTMs) for image tasks.<br>• Reduce number of **convolutional layers** if too deep.<br>• Reduce number of **feature maps** when using small kernels (e.g., 3×3).<br>• Consider larger **pooling** (e.g., 3×3), or adding/replacing with **stride**.<br>• Use **ReLU** for activation (fast and effective).<br>• Use **Global Average Pooling** to summarize spatial features. |
| **Training and Testing** | • Use **balanced, reduced datasets** for fast experimentation.<br>• Track **training and validation accuracy/loss** trends to spot issues.<br>• **Repeat training** multiple times to account for random variation.<br>• Investigate **misclassifications** using the confusion matrix.<br>• Use final test notebook to evaluate **accuracy, F1 score, and evaluation time**. |

Table 3: Summary of strategies mentioned in the video for designing efficient and accurate models

# B Appendix: Successfully Submitted Models Descriptions

In this appendix, we share the regional teams' solutions, including the motivation behind their chosen model, how they developed it, the preprocessing steps they applied, and a description of the selected machine learning model.

## B.1 Team01(4/4)

Our team evolved from a basic pixel-based approach to an advanced feature engineering solution for astronomical image classification. We initially chose the SGDClassifier for its computational efficiency and scalability with large datasets. However, after observing poor performance, particularly significant confusion between the Priority, Noisy, and Blurry classes, we pivoted to feature engineering while maintaining the same efficient classifier. This preserved speed and interpretability while dramatically improving accuracy.

We began with standard image preprocessing but quickly identified that raw pixel values were insufficient for subtle astronomical distinctions. Through iterative testing, we discovered that domain-specific features extracted from the images provided much better class separation than raw pixels. The t-SNE visualizations confirmed improved feature space clustering after our enhancements.

In the original preprocessing pipeline, we used grayscale conversion, normalization to [0,1], resizing to 64×64, and flattening to 4,096 pixel features. In the improved pipeline, we applied central cropping (70% of the original image to focus on key regions), grayscale conversion, resizing to 64×64, and then comprehensive feature extraction yielding 23 engineered features.

We maintained the SGDClassifier but enhanced its effectiveness through feature engineering and optimization. The 23 features included basic statistics (mean, standard deviation, quantiles), edge density via Sobel filters, texture analysis using Laplacian variance, Shannon entropy for complexity measurement, bright pixel detection, Local Binary Pattern histograms, and HOG feature summaries. Model optimization involved balanced class weighting to address imbalanced data, early stopping for convergence efficiency, and parallel processing across four cores. We also applied StandardScaler to normalize the 23 features.

This transformation from 4,096 raw pixel features to 23 engineered features achieved 99.7% accuracy, with perfect classification of critical Priority class images while maintaining computational efficiency suitable for CubeSat deployment.

## B.2 Team02(3/3)

The training competition provided both a challenge and an opportunity to address a real-world problem under realistic constraints. Our participation was driven by two main factors: the immediate prize and, more importantly, the opportunity to contribute as co-authors to the forthcoming CubeSat paper, where the outcomes of the competition would have direct scientific relevance.

We adopted a lightweight and well-documented deep learning model, chosen for its practicality, accessibility, and suitability to the competition's evaluation criteria. While the architecture achieved short evaluation times and required minimal memory, our primary objective was not efficiency alone but reliable classification performance. On the evaluation dataset, the model achieved 100% accuracy and F1 score, a result of particular importance in the astronomical context, where every pixel in an image may contain scientifically valuable information. By eliminating misclassification, we ensured that high-quality images were preserved without discarding data that might later prove scientifically meaningful.

Our methodology combined background research with systematic experimentation, informed by available documentation, tutorials, and shared notebooks. Starting from a base CNN, we refined its performance through careful hyperparameter tuning and architectural modifications. To reduce computational cost while maintaining accuracy, we simplified the design and implemented fine-tuning using MobileNetV2 as a lightweight backbone. This approach offered a strong balance between efficiency and accuracy, aligning with the competition's constraints while retaining scientific reliability.

We also employed an iterative strategy in which alternative model variants were regularly tested and compared, enabling continuous monitoring of improvements. Preprocessing steps were an essential component of this workflow: we balanced the dataset to correct for class representation and applied input size reduction to decrease computational demands. These modifications were systematically validated to confirm their negligible impact on classification performance.

## B.3 Team03(2/3)

We chose Random Forest as ML approach because our task involved image classification. Also, this model relatively has low training time and is easy to implement, which was ideal given the limited computational resources and time constraints of the hackathon. Our priority was maximizing accuracy under time constraints and simplicity of the model architecture understanding.

Our team began by exploring similar problems on Kaggle and GitHub to understand common modeling strategies. We also reviewed tutorials and papers related to image classification. Especially the tutorials given during the few days of pre-hackathon were helpful. Initially, we built a simple baseline model to evaluate feasibility, and then iteratively refined it by tuning hyperparameters and testing different classical model like Support Vector Machine (SVM).

Before feeding the data into our model, we performed several preprocessing steps that provide a big impact into satellite image classification, as known as features extractions. Images were resized to 128x128 pixels to standardize input dimensions and reduce computational load. We also applied grayscale conversion to reduce complexity because color did not provide additional discriminative power in our problems. The most important approaches that we also applied to all images are Laplacian Variance (blur detection), Black Pixel Ratio, Canny (edge feature), Edge Pixel Ratio, Entropy, the image intensity statistics as Mean (brightness), Standard Deviation (contrast), Skewness (brightness asymmetry) and Kurtosis Peakedness (distribution sharpness). Finally, we got feature vector of 8 values per images.

We designed the Random Forest classifier with 50 estimators to limit the number of trees in the forest to 50 and reduce training time and resource usage, -1 jobs allows the model to use all available CPU cores during the training which helps speed up the model fitting process and 42 random state to ensure that the results are reproductible which is crucial for debugging, iterative improvements and comparing results across different experiments. This approach provided a good balance between performance and interpretability.

## B.4 Team04 (3/3)

Our team selected a lightweight Convolutional Neural Network (CNN) architecture to balance performance, speed, and memory efficiency. The use of SeparableConv2D layers was motivated by their computational efficiency, as they reduce parameters while maintaining strong feature extraction capabilities. This design prioritized scalability for resource-constrained environments such as edge devices or cloud platforms with limited memory. Additionally, the model's simplicity enabled faster training and inference times—approximately 12 minutes with 20 epochs and a batch size of 8.

We began by testing various architectures, including pretrained models like MobileNet and EfficientNet, using 128×128 input images. However, these models were slower during training, achieved lower accuracy compared to our SeparableConv2D-based model, and had significantly larger sizes. We also experimented with DepthwiseConv2D but found it less effective in terms of accuracy.

To refine our approach, we conducted extensive research on lightweight CNNs, leveraging online tutorials and open-source notebooks provided by the Hackathon organizers. Through iterative experimentation, we optimized the model by reducing layer complexity, fine-tuning hyperparameters, and adjusting preprocessing steps.

Data preparation included:

- Resizing: Images were resized to 128×128 pixels to standardize dimensions and reduce memory usage.
- Normalization: Pixel values were scaled to [0, 1] to stabilize training and improve convergence.
- One-hot Encoding: Labels were converted into one-hot vectors to align with the categorical cross-entropy loss function.
- These steps ensured consistency across the dataset and enhanced the model's ability to learn meaningful patterns.

The final model architecture included:

- SeparableConv2D Layers: Lightweight convolutional layers that efficiently extracted features while minimizing computational costs.
- Pooling Layers: MaxPooling2D layers progressively reduced spatial dimensions, focusing on high-level features while keeping the parameter count low.
- GlobalAveragePooling2D: Replaced fully connected layers, reducing model size and mitigating overfitting.
- Output Layer: A dense layer with softmax activation to produce probability distributions over the five classes.

The model was compiled with the Adam optimizer and categorical cross-entropy loss, chosen for their robustness in multi-class classification tasks.

## B.5 Team05(2/2)

Our team opted for this model due to its low resource usage and interpretability. The texture-based feature extraction we used, however, had to be implemented from scratch, and although we tried to optimise it using NumPy vectorisation, it still is not fully optimised and compromises slightly in speed compared to most CNN models. However, the model performance and explanability make it reliable for the task, which led to its choice.

We came up with our solution after researching through literature on computer vision ranging from old methods relying on manual feature extraction techniques to modern CNN and transformer-based computer vision models. We evaluated the advantages and disadvantages of each technique and made our decision regarding the solution after conducting tests on several options.

The model relies on an SVM classifier after extracting features from the data using local binary patterns. We selected local binary patterns because, through the visualisation of the transformed images, we can identify similar traits among images belonging to the same class. Additionally, further visualisation of the features through techniques like t-SNE revealed that they were highly discriminative. The selection of the classifier was based on evaluations of various models, including logistic regression, random forest, and gradient boosting, across multiple hyperparameter configurations. For SVM we tuned the kernel by testing linear, poly and RBF kernels. The SVM classifier with a linear kernel turned out to yield the best accuracy and F1 score among the tested classifiers.

## B.6 Team06(3/3)

Our team addressed the CubeSat Image Classification challenge, where the main limitation is the extremely restricted computational power, storage, and downlink capacity of CubeSats. Transmitting all raw images to Earth is impractical, so we designed a lightweight and accurate onboard model to prioritize the most valuable images for transmission. Our goal was to balance efficiency, accuracy, and minimal resource usage, which is essential for embedded space applications.

We chose a compact CNN with SeparableConv2D layers to reduce computation and memory while maintaining high accuracy. Separable convolutions decompose standard convolutions into depthwise and pointwise operations, cutting parameters and operations, and making the model suitable for real-time, resource-constrained deployment. This approach prioritizes speed and deployability over very deep conventional CNNs.

Our approach combined a literature review on mobile-friendly CNNs with iterative testing. Standard convolutions were replaced by separable ones after comparing accuracy and inference speed. Class weights were applied to mitigate data imbalance, and hyperparameters such as batch size and learning rate were tuned for better generalization.

Images were resized from 512×512 to 128×128 and normalized to [0,1] to reduce computation. Light augmentation, including rotations and flips, improved robustness without distorting essential features. Aggressive denoising was avoided to preserve texture details crucial for class separation.

The final model includes four feature extraction blocks consisting of SeparableConv2D, BatchNormalization, ReLU, and MaxPooling2D, followed by GlobalAveragePooling2D, Dropout, and a Dense Softmax output. Training used the Adam optimizer, categorical crossentropy, a batch size of 32, and 15 epochs. Replacing fully connected layers with global pooling reduced parameters while maintaining discriminative power.

This lightweight CNN achieves an effective balance of accuracy, speed, and efficiency, supporting onboard CubeSat image triage under strict hardware constraints.

## B.7 Team07 (3/3)

We chose deep learning over traditional machine learning because CNNs are specifically designed for image processing and avoid the need for manual feature extraction. For this hackathon, we used a custom, lightweight CNN architecture inspired by the base CNN model that was already efficient and easy to interpret. We simplified the architecture to improve computational efficiency while ensuring that priority images could still be accurately classified.

We relied on the provided resources as well as external ones such as videos and research papers. We started with the base CNN architecture and focused on improving its performance by experimenting with different hyperparameters and architectural variations. To reduce computational cost, we simplified the existing architecture as much as possible while maintaining accuracy. We followed an iterative and empirical approach, regularly comparing the results of different models. Every two hours, we held discussions to evaluate which architecture was most likely to produce the best results.

For preprocessing, we downsampled the input images from 512×512 to 42×42 by slicing, which reduced computations by 99% with minimal impact on accuracy. While this reduction is unconventional, since downsampling is usually done by powers of two, it was justified in our case, as the star images are relatively simple and contain limited visual complexity, which helped preserve the essential features despite the drastic size reduction. We used a 7 layers Sequential CNN model with:

- One Convolution layer with ReLU activation and (3,3) kernels and 20 filters

- One Depthwise convolution layer with ReLU activation and (3,3) kernels and strides(3,3)
- One Convolution layer with ReLU activation and (1,1) kernels and 28 filters
- One Depthwise convolution layer with ReLU activation and kernels (3,3) and strides (2,2)
- One Convolution layer with ReLU activation and kernels (1,1) and 36 filters
- Global Average Pooling 2D to minimize parameters
- A dense layer with softmax output layer for 5-class classification

## B.8 Team08(3/3)

Our team's goal was to design a lightweight yet accurate image classification solution suited for resource-constrained environments like CubeSats. We explored several approaches to balance accuracy, model size, and inference speed.

We evaluated different CNN architectures and machine learning pipelines. MobileNetV3-like CNN provided strong accuracy by using Squeeze-and-Excitation and inverted residual blocks. SqueezeNet offered a very compact architecture with accuracy close to AlexNet but with 50× fewer parameters, matching our aim for deployment on limited hardware. Classical pipelines combining CNN features or handcrafted features, such as HOG with SVM, performed reasonably well but required more complex preprocessing and had slower inference.

We carried out extensive background research, consulted tutorials and papers, and iteratively built and tested prototypes. Models were compared by accuracy, size, and computational load. After several experiments, MobileNet-like models emerged as the best trade-off between accuracy and efficiency.

Preprocessing included resizing images to fit model input, normalization to stabilize learning, and using the imblearn framework to address class imbalance. For the handcrafted pipeline, we extracted HOG features and tested CNN-based feature extraction combined with dimensionality reduction.

The final model, based on a MobileNet-like architecture, uses inverted residual blocks and Squeeze-and-Excitation modules to emphasize important features while keeping the model lightweight. It was trained from scratch with tuned hyperparameters to maximize accuracy without adding unnecessary complexity. Alternative pipelines using CNN feature extractors followed by SVM classifiers achieved decent accuracy but were slower and less suited for real-time deployment.

Throughout the project, we focused on creating a model that is both efficient and robust for deployment on small satellites. After comparing different strategies, we concluded that the MobileNet-like architecture best balanced size, speed, and accuracy, aligning with our initial goal of a lightweight yet effective solution.

## B.9 Team09 (2/2)

We aimed to build an image classification pipeline that runs efficiently in the limited environment of a CubeSat. Given limited memory and compute, our priority was to build a lightweight yet accurate model .We adopted a CNN architecture based on MobileNet-style bottleneck blocks, inspired by MobileNetV2 to address strict resource constraints. Our priority was to design a model that offered a trade-off between efficiency and accuracy.

We initially experimented with standard architectures such as MobileNetV2, but found them too large and slow for our use case. So we simplified it by reducing the number of layers and channels, while still retaining its core efficiency mechanisms.

To come up with the solution, we divided the work: one focused on CNN design and optimization, while the other explored traditional machine learning methods to compare. Early testing showed that the CNN approach was more promising, so we shifted our efforts toward refining and compressing it for deployment. We consulted the research paper MobileNetV2: Inverted Residuals and Linear Bottlenecks as well as the Keras documentation on depthwise separable convolutions to guide our design decisions for building an efficient CNN. Through multiple iterations, we removed redundant components and tested various configurations until we found one that was both compact and accurate.

For preprocessing, we kept things minimal. We only resized all input images to 128×128 pixels using a Resizing layer embedded in the model itself. This helped reduce memory load and simplified the pipeline. We didn't normalize pixel values because we found that the model performed well without it.

The final model architecture begins with a 3×3 convolutional stem, followed by several bottleneck blocks with different configurations (including downsampling and channel expansion), and ends with a global average pooling layer, dropout, and a dense softmax output layer. We trained the model for 8 epochs using the Adam optimizer and categorical cross-entropy loss. And this model achieved 99.9% accuracy and a 0.999 F1 score, while being over 10× smaller and more than 200× faster than the baseline. These results confirmed that a carefully optimized CNN can deliver strong performance even under severe hardware constraints.

## B.10 Team10 (3/3)

Our team built an image classification model for the hackathon using the provided dataset, aiming to maximize accuracy within the event's tight time and resource limits. We selected a convolutional neural network (CNN) with transfer learning as our primary approach. This choice balanced speed, accuracy, and generalization: pretrained backbones such as ResNet and MobileNet offered high-quality feature extraction while drastically reducing the time needed to train from scratch.

Our preparation began by reviewing the dataset's characteristics — including image sizes, class distribution, and visual variability — and researching effective architectures for similar problems. Drawing on open-source notebooks, research papers, and prior experience, we established a baseline model quickly, then iteratively improved it through testing and refinement. Each cycle involved modifying hyperparameters, experimenting with different augmentations, and tracking performance on a validation split.

Data preprocessing was central to our success. All images were resized to a consistent dimension suitable for the model's input layer, pixel values were normalized to improve convergence, and augmentation techniques such as random rotations, flips, zoom, and brightness shifts were applied to increase diversity. These steps helped counteract overfitting and improved the model's robustness to variations in the dataset.

The final architecture integrated a pretrained CNN backbone for feature extraction with custom dense layers for classification. ReLU activations provided non-linear modeling power, while a softmax output produced class probability distributions. Dropout layers reduced overfitting, and the Adam optimizer ensured efficient convergence. Categorical cross-entropy was chosen as the loss function for its suitability to multi-class classification.

By combining strong preprocessing, transfer learning, and iterative tuning, we achieved high accuracy on the provided dataset in a short timeframe. Our approach demonstrates that with strategic model selection and disciplined experimentation, competitive results can be delivered rapidly — a key requirement in hackathon settings.

## B.11 Team11 (3/3)

When our team first encountered the CubeSat challenge, we approached it from three distinct academic backgrounds: a master's student in astrophysics, a computer science student specializing in software development, and a physics student. This diversity became one of our key strengths, as it enabled us to view the problem from different perspectives and to continuously challenge one another's ideas.

The short time frame meant we had to move quickly, so we started with a mix of brainstorming,dividing tasks, and deep research. We explored published work on satellite image classification, browsed GitHub repositories, and looked at blog solutions to understand what could be realistically built in the time available. Accuracy was important, but so were speed and efficiency, so we quickly narrowed our approach to a Convolutional Neural Network (CNN) paired with OpenCV for preprocessing.

Early experiments showed us that preprocessing could significantly influence performance. We tested different image enhancement techniques, and the combination of CLAHE (Contrast Limited Adaptive Histogram Equalization) for local contrast improvement and Canny edge detection gave the most consistent results. This step helped emphasise the structural details in the CubeSat images, allowing the CNN to focus on the most informative features.

For the network itself, we chose a streamlined CNN architecture with a progression of convolutional and pooling layers, a global average pooling stage to reduce complexity, and fully connected layers for classification. We introduced data augmentation rotations, flips, and zooms after noticing our first models struggled with minor image variations. This helped generalise performance without greatly increasing training time. Along the way, we monitored class balance to avoid bias, tweaked batch sizes and learning rates, and removed unnecessary complexity. There were plenty of moments where we had to throw out code and try again, but each failure pushed us towards a cleaner, faster, and more accurate pipeline.

By the end, we had a solution that was accurate, efficient, and shaped equally by each member's contributions. The combination of different academic perspectives, rapid iteration, and targeted preprocessing allowed us to deliver a model that met the challenge's requirements and reflected the best of our collaborative effort.

## B.12 Team12(3/3)

We analyzed existing models to identify possible starting points and understood that creating a neural network from scratch would not be feasible within the set timeframe. We concluded that the best approach, in line with the challenge's goal, was to improve an existing model. Our team's solution was built with the aim of achieving the best overall efficiency. The final model, based on Depthwise Separated Convolution paired with Bottleneck technologies, was designed to be as accurate as possible without compromising speed or significantly increasing resource consumption.

To develop the solution, we established standards for speed, accuracy, and complexity. Through research, we identified MobileNet as the best starting point, given its strong balance between performance and efficiency. We

determined that incorporating Depthwise Separated Convolution and Bottleneck layers could improve accuracy without adding substantial spatial or temporal complexity.

For preprocessing, we applied a single step of resizing all images to 128×128 pixels.

The model architecture is a deep learning CNN built on DSC and Bottleneck layer technologies. It begins with an input layer of shape (128, 128, 3), representing a 128×128 pixel image with three color channels. This is followed by a bottleneck block where DSC and bottleneck layers are combined, consisting of stacked blocks with an increasing number of filters. Each block contains:

- Expansion – a 1×1 Conv2D layer to increase channels.
- Depthwise Convolution – a 3×3 DepthwiseConv2D layer for spatial feature extraction.
- Projection – a 1×1 Conv2D layer to reduce channels back to the desired number.
- Residual Connection – adds the input to the output if shapes match without downsampling.

The network then applies Global Average Pooling to reduce feature maps to single values, followed by a Dense output layer with softmax activation for five-class classification.

Depthwise Separated Convolution was chosen to reduce parameters and computation, while the Inverted Bottleneck structure was used to optimize efficiency. This combination delivers faster results without sacrificing accuracy or making the model heavier, fully aligning with the group's original goal.

## B.13 Team13 (3/3) (edited)

When we first started the challenge, we were uncertain about the best steps to take in order to achieve results beyond the baseline. Our approach began by reflecting on what resources we already had. Many teams seemed to overlook an important insight from Chris Thron's introductory video, where he demonstrated how images could be represented as histograms. This became our starting point. We designed a function to extract bins across the three channels, horizontally stacked them, resized the images to 128×128, and extracted HOG features.

Our first model choice was a Random Forest. While it achieved 100% accuracy, it came with major drawbacks in terms of speed and size: training and evaluation took about 30 minutes, and the model size was around 40 MB (using 20 trees with a depth of 15). To improve efficiency, we attempted a Decision Tree model, but its precision was poor. We also experimented with Gradient Boosting, but the results were still unsatisfactory. At that point, we realized it was necessary to start thinking about new solutions.

We then discussed the possibility of using deep learning and transfer learning models. However, we recognized that relying on readily available pretrained models such as TinyVGG, SqueezeNet, EfficientNet, or MobileNet would likely be a common approach among many teams. To distinguish our work, we decided we needed to innovate and pursue a more original solution.

We first passed the images through a CNN to examine their representations. This led us to consider passing the extracted features through a simple dense layer. We then implemented a feedforward network with three layers, using activations across dimensions of $384 \rightarrow 2048 \rightarrow 5$. Training was extremely fast, taking about 1 ms per epoch, so we trained for 2000 epochs. The resulting model achieved an evaluation time of 4.96 seconds, a model size of 3 MB, and a precision of 93%. While the results were promising, we realized that using 256-pixel images could further improve performance, though at the cost of longer training time. Upon analysis, we found that the 2048-dimensional layer was responsible for the large model size. By reducing this layer and incorporating HOG features, we achieved an evaluation time of about 15 seconds, a model size of only 0.25 MB, and a precision of 99.8%. All of this was accomplished within three days.ys.

## B.14 Team14 (3/3)

Our team developed a deep learning model based on transfer learning using MobileNetV2. Our motivation was to build an accurate and fast model that could operate under resource constraints, aligning with the hackathon challenge . The solution was grounded in the team's experience in data science, literature insights, and iterative testing. We initially fine-tuned the model proposed as CubaNet by adding layers, changing the optimizer, and adjusting the batch size and the improvement was not reached. Based on our experience, traditional machine learning models were excluded early because they do not perform well on large-scale image data.

Through literature review and trials, we considered transfer learning. We conducted several tests with MobileNetV2, but the results were not meeting expectations. This led us to the idea that performance could improve if the model focused more on feature extraction. As a result, we froze all layers of MobileNetV2 and used it solely as a feature extractor, adding a custom final 2D output layer.

We preprocessed the data by resizing images to 224×224 pixels and normalizing pixel values to the [0, 1] range to match the input requirements of the network. One-hot encoding was applied to the labels.

The final model consisted of four layers: (1) an input layer with shape 224×224×3, (2) a frozen MobileNetV2 pretrained layer, (3) a GlobalAveragePooling2D layer, and (4) a trainable Dense output layer with softmax activation. The dataset was split 80/20 for training and validation. We used the Adam optimizer and categorical cross-entropy loss for model training and batch size of 32.

### B.15 Team15 (3/3)

When first approaching this problem, we began with exploring a range of different image categorizing models and testing them on the provided dataset. We weren't able to achieve better performance than the baseline model CNN mentioned in the paper. Given the time we had, we needed to change strategy and work on optimizing the baseline model into making it a smaller model while still maintaining the accuracy or even bettering it slightly. We needed to change strategy and work on reducing the size of the images through preprocessing for better accuracy and faster processing.

Regarding preprocessing, we tried different techniques such as normalization, resizing the image into smaller image dimensions, and reducing RGB channels to a single luminance channel, since we thought that brightness would be a strong indicative feature for identifying a quality star image. However, these approaches weren't able to produce accuracy improvements over the baseline CNN model. So, we didn't implement any preprocessing. We needed to change strategy once again and work on optimizing the baseline model into making it a smaller model while still maintaining the accuracy or even bettering it slightly.

We started doing research into CNN-based models that are lightweight to identify techniques we can inherit to make the baseline model light. That's when we came across MobileNet, a family of lightweight convolutional neural network (CNN) architectures designed for mobile and embedded vision applications. We adopted the SeparableConv2D, which separates depth-wise and point-wise computations to reduce parameters and computation, and BatchNormalization to improve the generalizability of the model, while it results in better accuracy.

After going through different experimentation, we have identified the optimal structure when we reduced the initial filter size from 16 to 8 and applied SeparableConv2D in the last convolutional stage (the fourth layer) before BatchNormalization. When we retrained the baseline CNN, it achieved 98% accuracy, while our optimized model resulted in an accuracy of 97%. Where there is a 0.01 decrease in accuracy, it is a trade-off for a significant reduction in model size and computational cost. This optimized model is suited for a deployment in a resource-constrained environment.

### B.16 Team16

Our task was to classify over 16000 CubeSat-taken images into categories such as blurry, corrupt, noisy, missing, and priority, helping a Raspberry Pi onboard the satellite decide which images to transmit to Earth. The model needed not only high accuracy but also to be lightweight and optimized for real-time inference on limited hardware.

We researched similar satellite classification problems and tried to build ideas upon those papers. We initially tested CNN models with various layer configurations and lightweight CNNs like MobileNet and SqueezeNet; they only provided slight performance improvements while significantly increasing model size, making them unsuitable for space deployment. While re-evaluating our approach, we began experimenting with classical machine learning models to explore lighter and more efficient alternatives.

After experimenting with various classical models, we were surprised to find that logistic regression delivered strong results even with the minimal features we already extracted. Since we have seen its potential, we build a more robust preprocessing pipeline by resizing the image to 128x128, applying grayscale conversion, and extracting hand-crafted features such as Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), quantile statistics, and edge detection features (such as Sobel magnitude and Laplacian variance). These features helped the model capture key visual distinctions between the classes without relying on deep learning.

After feature extraction, we selected the top 50 features, applied a standard scaler for normalization, performed PCA for dimensionality reduction, and finally trained a logistic regression model, fine-tuning its hyperparameters for better performance. The final quantized model achieved excellent accuracy, misclassifying only one image, and had a size of just 1 KB after conversion to ONNX format, making it an excellent fit for onboard deployment on the Raspberry Pi. Despite intensive preprocessing, the lightweight model allowed real-time inference with minimal computational load.

Our solution demonstrates that with careful feature design, classical models can outperform complex deep learning approaches in constrained environments. Our team's solution novelty and performance really make it stand out and outperform the complex deep learning approaches in constrained environments.

## B.17 Team17(2/3)

We developed an optimized solution using a Convolutional Neural Network (CNN) enhanced by improved image preprocessing, mainly for its efficiency. The preprocessing phase focused on addressing data quality issues through techniques such as missing value imputation, normalization, and categorical encoding, with the methods selected based on recent literature. During the learning phase, we incorporated decision trees, ensemble methods, and support vector machines to handle different data patterns. The CNN was designed with four convolutional blocks (Conv2D + MaxPooling), followed by a Global Average Pooling layer and a Dense softmax output layer with 5 classes: Blurry, Corrupt, Missing_Data, Noisy, Priority. The model is trained with categorical cross-entropy loss and the Adam optimizer.To optimize performance, hyperparameter tuning paired with cross-validation was used to identify the best configurations, ensuring both robustness and scalability. The adoption of an ensemble approach further improved accuracy and consistency across various data partitions.

The approach achieved an accuracy and F1 score of 0.996 on the test set. Although this represents a 0.002 (0.2%) decrease compared to the baseline accuracy, the computational improvements are substantial. The total pipeline time, including data preprocessing and prediction, is 65.06 seconds, uses just 4.7 GB of RAM and consumes 10% less CPU than the previous solution. The efficiency gains are primarily achieved through batch memory management and enhanced data preprocessing, including increased image brightness and contrast. Initially, the model struggled to recognize priority images when trained with just 100 samples. By increasing the dataset size and applying further preprocessing enhancing brightness and contrast, converting images to grayscale, and leveraging TensorFlow operations for better memory efficiency before converting results to NumPy arrays we were able to improve performance significantly. Additionally, to overcome memory limitations during data loading and preprocessing, we utilized the np.load() function with the mmap mode parameter for memory mapping, enabling selective access to large files, and processed data in batches to further reduce memory usage.

## B.18 Team18(2/2) edited

We developed an efficient workflow for image classification based on frequency-domain analysis using the Fourier Transform. Instead of working in the spatial domain, we converted images to grayscale and applied the FFT, enabling the extraction of compact and informative statistical features without sacrificing the original image size. This approach prioritizes speed and interpretability while avoiding the complexity of deep learning models. According to the literature, analyzing images in the frequency domain allows relevant information to be obtained efficiently, making the process scalable and suitable for environments with limited computational resources.

From each image, we extracted eight features: total spectral energy, mean frequency, variance, skewness, and kurtosis of the frequency spectrum, as well as the ratio of missing data and the standard deviation of intensity in the spatial domain. These metrics allowed us to distinguish between blurry, noisy, incomplete, and priority images. Computing these features for the whole data took only 3 minutes. To define this feature set, we tested various combinations: we started with total spectral energy, mean frequency, and variance, but as results were not satisfactory, we added skewness and kurtosis, which improved classification. However, some confusion between categories persisted, so we included intensity standard deviation (key for distinguishing blurry from noisy) and the fraction of missing data (to separate priority from missing data).

Once the preprocessing was optimized, and using the notebook provided during the hackathon as a starting point, we tested several machine learning models. We found that Random Forest offered the best performance, thanks to its ability to capture nonlinear relationships and complex feature combinations. Hyperparameter tuning was performed using grid search, optimizing for both accuracy and execution time, with the best results found for n _estimators = 100 and max_depth = 10.

Training the model using only the extracted features significantly reduced data size and complexity, requiring just one second to complete. Our final model achieved 99.3% accuracy and an F1 score of 0.993 on the test set, yielding near-perfect classification across all categories.

## B.19 Team19(2/2)

In our solution for the CubeSat Challenge, we focused on the detection of corrupted or missing satellite image data through a lightweight and computationally efficient approach. Instead of processing raw image data directly (which is often resource intensive), we extracted a compact set of five statistical and structural features from each grayscale image, and used it on an Extreme Learning Machine (ELM). A feedforward neural network which maintaining high values of accuracy while is a single-layer, fast and lightweight.

First, we converted each image to grayscale. We then applied a Laplacian filter and computed its mean and standard deviation to capture edge information, which proved helpful in identifying missing or corrupted regions. We calculated the mean pixel value, as higher mean values often indicate image degradation. Additionally, we computed the percentage of zero-valued (black) pixels, which helped identify missing data since such pixels typically

represent loss. Lastly, we applied an entropy filter to quantify texture complexity, assisting in the identification of noise or unexpected patterns.

These five features were combined into a concise vector, allowing us to significantly reduce data dimensionality while retaining important information. This vector served as input to an ELM. We used 1650 hidden neurons, a value determined through manual experimentation. The sigmoid activation function yielded the best results, and we set the regularization parameter to $1 \times 108$. No formal hyperparameter tuning was performed, though such optimization could likely reduce the model's size without compromising performance.

The resulting model is highly compact (approximately 0.13 MB of code) and delivers outstanding classification metrics: 99.8% accuracy and F1-score. One drawback is computational time—the model uses 100% of a single CPU core for approximately 36.5 seconds during preprocessing and prediction.

This approach demonstrates that high-performance image corruption detection is possible even in constrained environments, making it well-suited for edge applications such as CubeSat systems.

## B.20 Team20 (3/3)

The CubeSat challenge aimed to improve the efficiency and accuracy of a machine learning model classifying data captured by CubeSats. The goal was to prioritize transmission of the most scientifically valuable images back to Earth under constraints such as limited onboard resources and low communication bandwidth. This demanded a lightweight model with strong performance and high accuracy.

We began by evaluating and modifying the CNN architecture provided. After testing multiple configurations and exploring transfer learning, we adopted MobileNetV2—known for efficiency on mobile devices—as our model base. Using pretrained ImageNet weights, we added dense layers atop the feature extractor to classify five image quality categories. Initially, we focused on maximizing classification accuracy, testing configurations to find a stable, high-performing setup. After consistently strong results on validation and test sets, we shifted focus to computational efficiency. We reduced network width via the alpha parameter and chose a smaller input resolution, maintaining performance while significantly lowering resource usage. These optimizations enhanced suitability for resource-constrained platforms like CubeSats.

The final model processed input images resized to $224 \times 224$ pixels and normalized to [-1, 1]. It employed MobileNetV2 with a width multiplier (alpha) of 0.35, followed by Global Average Pooling and a Dense Softmax output layer with five classes. The feature extractor was frozen during training. Training and validation used the full dataset—9,711 training and 3,237 validation images—showing stable convergence and near-perfect classification across all categories. Evaluation on the test set (3,237 images) confirmed the model's ability to generalize to unseen data: it produced a perfectly diagonal confusion matrix, with accuracy, precision, recall, and F1-score all equal to 1.0. It also outperformed the original CNN baseline by correctly classifying samples the baseline misclassified. In terms of efficiency, our solution achieved a $9.5\times$ speedup—reducing inference time from 265 to 28 seconds—while using only 28% more memory and approximately four CPU cores (batch size 32). In summary, we developed a compact, high-performance classifier using transfer learning. It delivers fast inference, low memory usage, and strong generalization—key traits for onboard image selection in CubeSat missions.