Computer Vision Competition Project Report

Oulaya BENANI DAKHAMA, Kerrian LE BARS, Ulysse MACÉ, Arthur MORVAN
supervised by Louise DAVY (emails: surname.name@essec.edu)

January 18, 2026

Link to GitHub : GitHub Project

This report aims to address the key questions related to the Computer Vision Kaggle Competition. It presents and compares the different models explored during the project, explains how the work was distributed among group members, and discusses the main challenges encountered throughout the development process, along with the strategies implemented to overcome them.

1. **On which platform/software did you implement your model? Did all team members use the same setup? If not, specify what each person used. What motivated these choices? /1**

   The development of our model was carried out using multiple coding environments, reflecting both individual preferences and task-specific needs. While the team did not use a single unified setup, all tools were fully compatible with our shared codebase and workflow.

   Arthur used Cursor. Cursor was chosen for its strong integration with AI models, enabling rapid prototyping, quick testing, and efficient code refactoring. Its plan mode was particularly useful for handling more complex tasks by structuring multi-step implementations and exploring alternative solutions.

   Ulysse and Oulaya used Visual Studio Code (VS Code). VS Code was selected for its flexibility, stability, and seamless integration with GitHub, which facilitated version control and collaboration. In addition, it supports GitHub Copilot, and as it is the environment the team is most familiar with, it reduced development friction and improved efficiency.

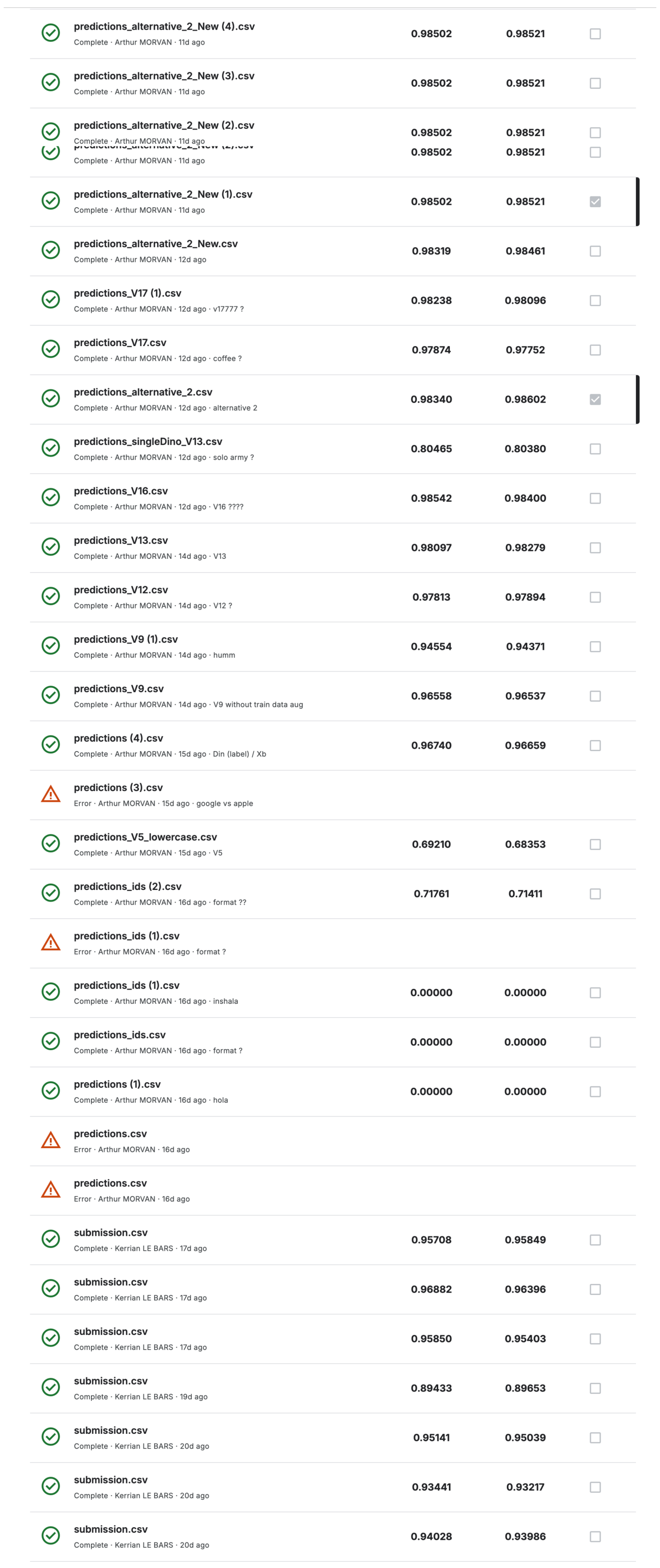
   Kerrian used Antigravity Editor. Antigravity was used for its integrated agent-based workflow, which supports structured code development through execution plans, implementation roadmaps, and checklists. It also provided effective assistance for debugging, as well as for identifying and correcting errors and warnings throughout the development process.

2. **How did you divide the work? How did you collaborate while coding? /1**

   The work was divided by initially exploring different modeling approaches, including classical machine learning methods, CNNs, transfer learning, and ensemble modeling. Each team member independently investigated and experimented with approaches they considered promising or that were inspired by strong ideas from other groups.

   Collaboration was mainly sequential and iterative: we regularly shared insights, explained code choices, and exchanged suggestions for improvement. Code and results were discussed collectively, allowing successful ideas to be refined, merged, or discarded, ensuring a coherent final model despite parallel exploration.

3. **Include a full screenshot of the Kaggle Submission page. /1**



✓	predictions_alternative_2_New (4).csv Complete · Arthur MORVAN · 11d ago	0.98502	0.98521	☐
✓	predictions_alternative_2_New (3).csv Complete · Arthur MORVAN · 11d ago	0.98502	0.98521	☐
✓	predictions_alternative_2_New (2).csv Complete · Arthur MORVAN · 11d ago	0.98502	0.98521	☐
✓	predictions_alternative_2_New (2).csv Complete · Arthur MORVAN · 11d ago	0.98502	0.98521	☐
✓	predictions_alternative_2_New (1).csv Complete · Arthur MORVAN · 11d ago	0.98502	0.98521	✓
✓	predictions_alternative_2_New.csv Complete · Arthur MORVAN · 12d ago	0.98319	0.98461	☐
✓	predictions_V17 (1).csv Complete · Arthur MORVAN · 12d ago · v17777 ?	0.98238	0.98096	☐
✓	predictions_V17.csv Complete · Arthur MORVAN · 12d ago · coffee ?	0.97874	0.97752	☐
✓	predictions_alternative_2.csv Complete · Arthur MORVAN · 12d ago · alternative 2	0.98340	0.98602	✓
✓	predictions_singleDino_V13.csv Complete · Arthur MORVAN · 12d ago · solo army ?	0.80465	0.80380	☐
✓	predictions_V16.csv Complete · Arthur MORVAN · 12d ago · V16 ????	0.98542	0.98400	☐
✓	predictions_V13.csv Complete · Arthur MORVAN · 14d ago · V13	0.98097	0.98279	☐
✓	predictions_V12.csv Complete · Arthur MORVAN · 14d ago · V12 ?	0.97813	0.97894	☐
✓	predictions_V9 (1).csv Complete · Arthur MORVAN · 14d ago · humm	0.94554	0.94371	☐
✓	predictions_V9.csv Complete · Arthur MORVAN · 14d ago · V9 without train data aug	0.96558	0.96537	☐
✓	predictions (4).csv Complete · Arthur MORVAN · 15d ago · Din (label) / Xb	0.96740	0.96659	☐
⚠	predictions (3).csv Error · Arthur MORVAN · 15d ago · google vs apple			
✓	predictions_V5_lowercase.csv Complete · Arthur MORVAN · 15d ago · V5	0.69210	0.68353	☐
✓	predictions_ids (2).csv Complete · Arthur MORVAN · 16d ago · format ??	0.71761	0.71411	☐
⚠	predictions_ids (1).csv Error · Arthur MORVAN · 16d ago · format ?			
✓	predictions_ids (1).csv Complete · Arthur MORVAN · 16d ago · inshala	0.00000	0.00000	☐
✓	predictions_ids.csv Complete · Arthur MORVAN · 16d ago · format ?	0.00000	0.00000	☐
✓	predictions (1).csv Complete · Arthur MORVAN · 16d ago · hola	0.00000	0.00000	☐
⚠	predictions.csv Error · Arthur MORVAN · 16d ago			
⚠	predictions.csv Error · Arthur MORVAN · 16d ago			
✓	submission.csv Complete · Kerrian LE BARS · 17d ago	0.95708	0.95849	☐
✓	submission.csv Complete · Kerrian LE BARS · 17d ago	0.96882	0.96396	☐
✓	submission.csv Complete · Kerrian LE BARS · 17d ago	0.95850	0.95403	☐
✓	submission.csv Complete · Kerrian LE BARS · 19d ago	0.89433	0.89653	☐
✓	submission.csv Complete · Kerrian LE BARS · 20d ago	0.95141	0.95039	☐
✓	submission.csv Complete · Kerrian LE BARS · 20d ago	0.93441	0.93217	☐
✓	submission.csv Complete · Kerrian LE BARS · 20d ago	0.94028	0.93986	☐

Figure 1: Kaggle submission page screenshot

4. **How did you handle loading the data into RAM? /1**

   To manage RAM and GPU memory efficiently, we used a rented GPU for computation and loaded the data in mini-batches (typically of size 8 to 16, depending on the model). This batching strategy prevented the full dataset from being loaded into memory at once and allowed training to remain stable. The main bottleneck was the model size rather than the data itself, which required careful code optimization and, in some cases, longer execution times to obtain outputs.

5. **How did you preprocess your data before training? What motivated these choices? /1**

   Before training, we leveraged pretraining on an external dataset to improve performance and reduce overfitting, which was particularly important given the limited size of the competition data. We also applied data augmentation, primarily using deterministic transformations, to ensure consistency across samples while enhancing model robustness. Consistency between training and inference was further reinforced through Test-Time Augmentation (TTA). These preprocessing choices were motivated by the need to improve leaderboard performance, compensate for data scarcity, enhance generalization, and ensure a reliable inference pipeline compatible with TTA.

6. **Were there any preprocessing steps or data-handling methods you tried that produced worse results than expected? Do you have an explanation for this? /1**

   Yes, some data-handling strategies led to worse performance than expected. In particular, we experimented with incorporating Test-to-Train pseudo-labeling based on model confidence. However, because the model was already highly confident in its predictions, this approach introduced a large amount of low-diversity or noisy data into the training set. This resulted in overfitting and reduced generalization performance, explaining the observed performance degradation.

7. **Which models and hyperparameters did you test? What motivated these choices (aside from validation score) ? (I am not expecting a description of how each model works, unless it is a model we did not cover in class.) /2**

   We explored a wide range of models in order to capture different inductive biases and complementary strengths. We initially implemented simpler computer vision models to establish strong baselines and understand the data. We then introduced classical machine learning approaches, including tree-based models such as LightGboost, which were later used in an ensemble voting scheme due to their strong performance on structured features.

   As the project progressed, we experimented with more complex architectures, including small-scale Vision Transformers with fixed embedding sizes, followed by several CNN architectures of increasing depth and capacity. In addition, we tested transfer learning approaches using pretrained models such as ResNet50 and EfficientNetB0. These models were first used as frozen feature extractors, with an additional trainable convolutional layer and a classification head. Subsequently, we partially fine-tuned the pretrained networks by selectively unfreezing upper layers to adapt higher-level representations to the task.

   We also performed extensive hyperparameter tuning, particularly for tree-based classifiers, including the number of trees, feature subsampling, and minimum samples per split.

   These choices were motivated by the desire to explore a diverse set of modeling paradigms and leverage their complementary properties. Simple CNNs proved lightweight and efficient, transformers offered strong generalization but were less effective when used alone in this setting, and heavier CNNs were well adapted to the task but benefited from ensembling. Tree-based models provided more stable contributions than coefficient-based voting methods. Overall, the strategy was driven by robustness, generalization, computational constraints, and an interest in innovative model combinations rather than solely by validation performance.

   All our models can be viewed using the GitHub link provided with the report.

8. **How did you evaluate your models outside of Kaggle? Did you use a validation scheme, and if so, which one? /1**

   We did not use a traditional validation scheme. Instead, we applied a pseudo-labeling strategy: after running inference, we added only the predictions for which the model was extremely confident (over 98–99%) to the training set. We then retrained the model for a few additional epochs. This method is more aligned with competition optimization than standard evaluation, but it improved our leaderboard score by a few points.

9. **Did you set random seeds to ensure reproducibility? Were the results stable? /0,5**

   Yes, we set a fixed random seed (42) to ensure reproducibility. However, due to the long runtime of the full pipeline (over 20 hours on GPU), we could not repeat the complete training multiple times to fully verify stability. We therefore expect the results to be consistent, with only minor variations caused by non-deterministic components such as Test-Time Augmentation (TTA), which should not significantly affect the final score.

10. **Which model achieved your best results? Describe its functioning, architecture, and the hyperparameters used during training. /2**

    Our best results were obtained from an ensemble model combining multiple architectures and learning paradigms. The final pipeline combined CNN-based models and transformer-based models trained across multiple folds, and aggregated their predictions (including through TTA). These outputs were then leveraged in a stacked approach together with additional handcrafted/statistical features to improve robustness and final performance.

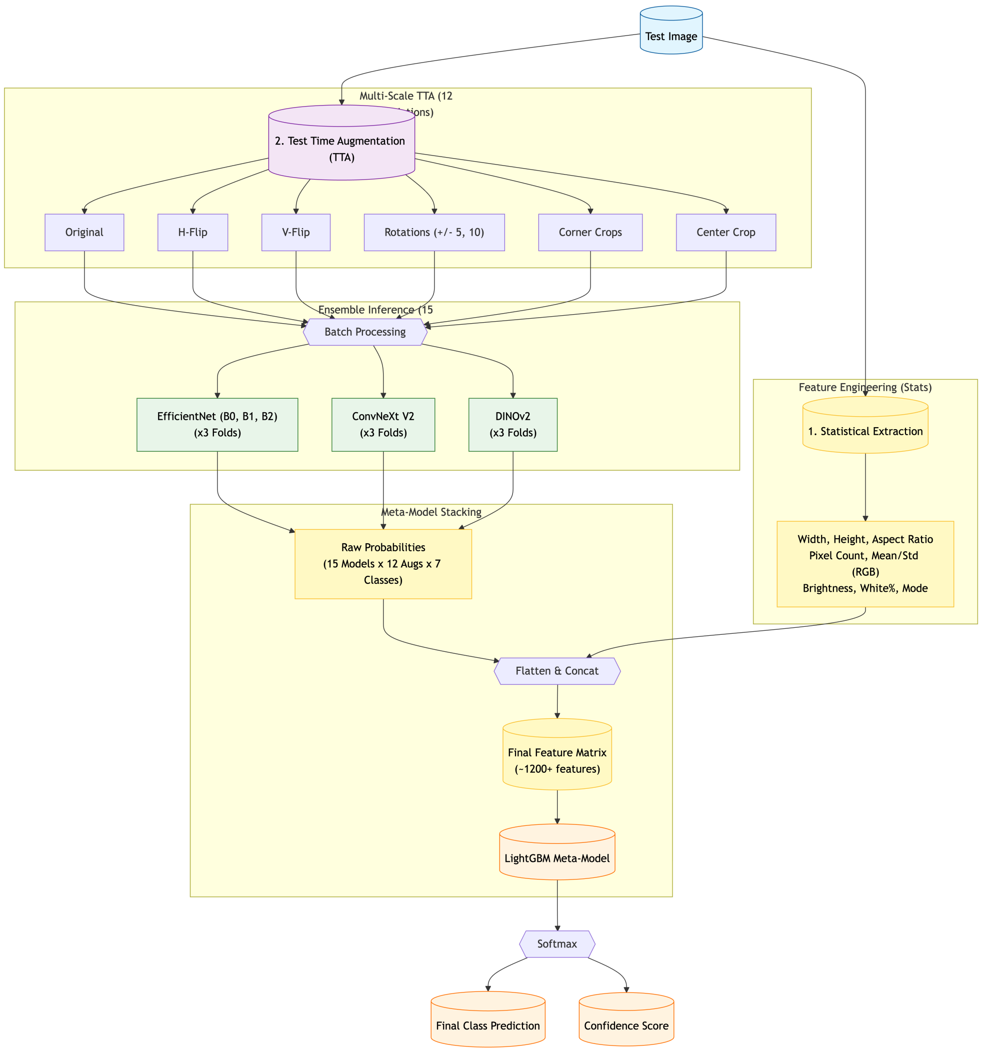


Figure 2: Best model architecture

11. **What were the training times for your models (rougly) ? Did you need to adjust model size or hyperparameters**

for time or GPU budget reasons? /0,5

Although each epoch was relatively fast (only a few minutes), we trained for 15 epochs. When including pretraining, test-time augmentation (TTA), and the full inference pipeline, the total runtime increased to approximately 20 hours.

To manage these constraints, we reduced model sizes by using smaller CNN and transformer architectures. We also loaded models sequentially rather than in parallel to avoid exceeding RAM limits, which resulted in slower overall training and inference. Due to time constraints, we hand-selected the best-performing models from the 15 trained candidates rather than training or evaluating additional variants. Finally, we reduced batch sizes for some models (from 16 to 8) to fit within GPU memory.

12. **Provide the accuracy of your model for each class on your training dataset. Are there classes with significantly different performance levels? How do you explain this? /1**

The model achieved high and relatively consistent performance across most classes on the training dataset. Accuracy (reported via precision, recall, and F1-score) ranged from approximately 93% to 99% depending on the class. For instance, Samsung achieved the highest performance (F1 $\approx$ 0.99), while Mozilla and Messenger showed comparatively lower scores (F1 $\approx$ 0.93).

These differences can largely be explained by data imbalance and visual similarity between classes. Classes with more training samples and more distinctive visual patterns (e.g., Samsung, Apple, Google) achieved higher accuracy. In contrast, classes with fewer examples (e.g., Mozilla) or visually similar logos (e.g., Messenger, WhatsApp, Facebook) were more prone to confusion, leading to slightly reduced performance.

Overall, while performance differences exist, they are consistent with dataset characteristics rather than model instability.

13. **Show three examples of training images that were misclassified by your model. What do you think about these errors? /1**

In the first example, an image from the WhatsApp class was incorrectly predicted as Facebook. Visually, the image is heavily cropped and blurred, with large uniform color regions and limited structural information. The absence of distinctive features likely caused the model to rely on coarse color and shape cues, leading to confusion with another class sharing similar visual characteristics.

In the second example, an image belonging to the Facebook class was misclassified as Google. The image is dominated by a rounded shape with smooth gradients and partially occluded central features. These generic visual patterns reduce class-specific cues and can cause the model to associate the image with a broader set of visual representations learned across multiple classes.

In the third example, an image from the Mozilla class was predicted as Messenger. This image contains mostly uniform tones, soft contours, and very low contrast, providing little discriminative information. Combined with the relatively small number of training samples for this class, such visually ambiguous images are more likely to be misclassified.

14. **If you had one full day to further improve your model, what would you prioritise and why? /1**

If we had one full day to improve our model, we would prioritize specialized sub-models by training multiple small networks, each dedicated to a specific type of image. This approach could enhance performance by allowing each model to learn more targeted features and reduce confusion between visually similar classes.

Additionally, we would explore error-focused training by creating a model trained specifically on the instances that our current pipeline misclassifies. This could help correct systematic weaknesses in the ensemble.

The main risk of these strategies is overfitting, given the limited dataset size. We would therefore apply strong regularization and validate carefully. We already partially implemented this idea through our fold-based training, but further specialization could still yield improvements.

15. **Which ethical considerations arise if this model were to be deployed in a real-world setting? /1**

    If deployed in a real-world setting, the model must be used responsibly and for legitimate purposes. It should not be applied to unethical surveillance, discrimination, or privacy-infringing activities, and its use should align with ethical OSINT (Open Source Intelligence) standards and legal requirements.

    Another key consideration is environmental impact: training and running large GPU models for a relatively simple task like emoji classification is energy-intensive and inefficient. Deploying such a model would therefore need to justify the computational cost versus simpler alternatives, to avoid unnecessary carbon emissions and resource consumption.

16. **Use of generative AI: one reply per team member : /1**

    - **Questions**

      1. **Did you use any generative AI tools such as ChatGPT, Gemini, DeepSeek, Copilot?**
      2. **If yes, which ones?**
      3. **For what purpose (code generation, code debugging, writing, translation, idea generation, explaining results, clarifying course material, etc.)?**
      4. **Do you believe you could have achieved the same results without generative AI tools?**

    - **Team member replies (2-column layout)**

      Ulysse

      1. Yes, but didn't entirely depend on it.
      2. ChatGPT.
      3. Debugging and explanation of errors, as well as helping to understand code that was made by my peers when I didn't fully grasp their explanations for it.
      4. Would have been a much slower and painful process, but ultimately anything is possible.

      Oulaya

      1. Yes, but only as a supportive tool and not as something that will replace my own work.
      2. ChatGPT.
      3. Mainly for debugging and for understanding code written by other team members when their explanations were not fully clear.
      4. Yes, but it will be more difficult and more time-consuming.

      Kerrian

      1. Yes, used generative AI tools during the project, but not as a replacement for personal work.
      2. Claude Sonnet 4.5 and Gemini 3 Pro.
      3. Idea improvement and brainstorming, code debugging, and as a planning assistant to define implementation roadmaps and structured development steps.
      4. Comparable results could still have been achieved, at the cost of additional time, particularly for debugging and GPU configuration.

      Arthur

      1. Yes, used Cursor-provided AIs including Anthropic, OpenAI, and Google models.
      2. Depending on the hardest aspect of the task: Opus 4.5 for code, Google models for thinking capabilities.
      3. Quick testing of ideas and code formatting to ensure a clean and as efficient as possible structure.
      4. Yes, but it would require much more development time (time was a crucial aspect, as training time and development were both important).

(For question 16 : Each team member gets the point if they answer, no matter what their answer is. I do not penalise strong or unreasonable use of gen AI, nor do I penalise if you didn't use any AI tool.)