





Allsky Camera Network for Detecting Bolides



Tyler Turner
Vincent Quintero
Jean-Pierre Derbes
Charles Derbes
Dr. Csaba Palotai

Task Matrix (Milestone 3)

| Task | Completion | Tyler | Vincent | Jean-Pierre | Charles |
|---------------------------------|------------|-------|---------|-------------|---------|
| Replace current C++ camera code | 100% | 10% | 35% | 45% | 10% |
| Implement Server API | 99% | 50% | 0% | 50% | 0% |
| Implement Client API | 99% | 20% | 0% | 20% | 60% |
| Begin writing CLI | 30% | 30% | 10% | 50% | 10% |
| IoT Style Setup | 99% | 20% | 0% | 10% | 70% |
| Classification | 99% | 0% | 33% | 33% | 34% |
| Start Writing UI | 0% | 20% | 70% | 0% | 10% |
| Create setup process for node | 75% | 75% | 0% | 25% | 0% |

Task Discussion

Replace Current C++ Camera Code -> Node records and sends videos in 10 minute chunks

Implement Server API -> Consumer queue that workers pull from to process video

Implement Client API -> Configuration moved to .env, made gps integration easier

Begin writing CLI -> Classification pipeline written

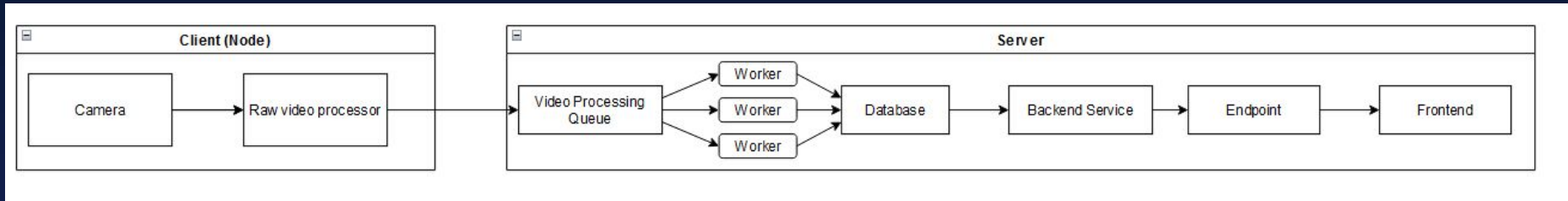
IoT Style Setup -> Still need to add captive portal

Classification -> Complete, may need to look into a transformer for even better accuracy. Still needs to be tested "in the wild".

Start Writing UI -> No physical work was done on this, only "work" done was throwing ideas around

Create Setup Process for Node -> Ansible playbook that dictates all of the software and configuration a node needs to operate

The life of a video



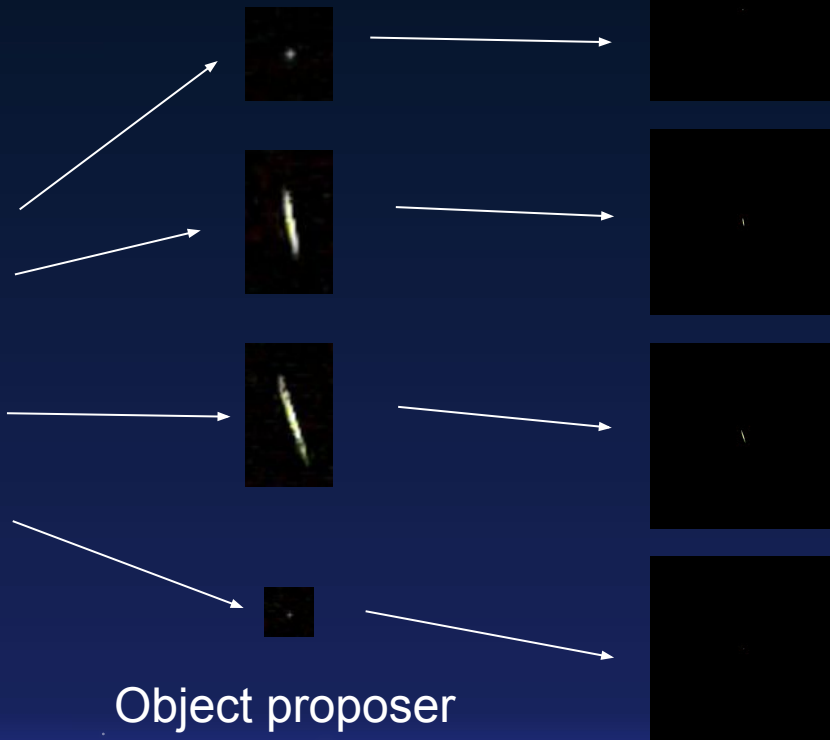
Classification Pipeline



Classification Pipeline



Composite



Object proposer

Molding to 512x512

Bolide Classification Model

- Dataset size: 4000
- 70-15-15 train, validation, test split
- Label encodings: {'bolides': 0, 'notbolides': 1}
- 4000 samples, 2000 of them are notbolide, and 2000 of them are bolide
- Hyperparams:
 - lr = 0.001
 - epochs = 20
 - loss = BCELoss (Binary Cross Entropy Loss)
 - Adam optimizer
- Inputs are transformed to 128x128, making training much faster

a. Decreasing kernel sizes 7 -> 5 -> 3 -> 3

```
self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=7, padding=3)
self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=5, padding=2)
self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
self.conv4 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1)
```

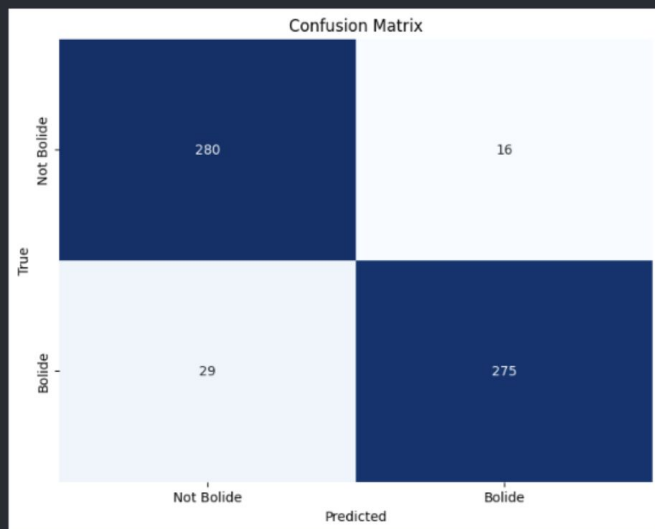
100% | 19/19 [00:01<00:00, 10.69it/s]

Test Loss: 0.2458 Accuracy: 92.50%

Classification Report:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0.0 | 0.91 | 0.95 | 0.93 | 296 |
| 1.0 | 0.95 | 0.90 | 0.92 | 304 |
| accuracy | | | 0.93 | 600 |
| macro avg | 0.93 | 0.93 | 0.92 | 600 |
| weighted avg | 0.93 | 0.93 | 0.92 | 600 |

Testing Confusion Matrix:



Number of misclassified samples: 45

b. Constant kernel sizes 3 -> 3 -> 3 -> 3

```
self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)
self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
self.conv4 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1)
```

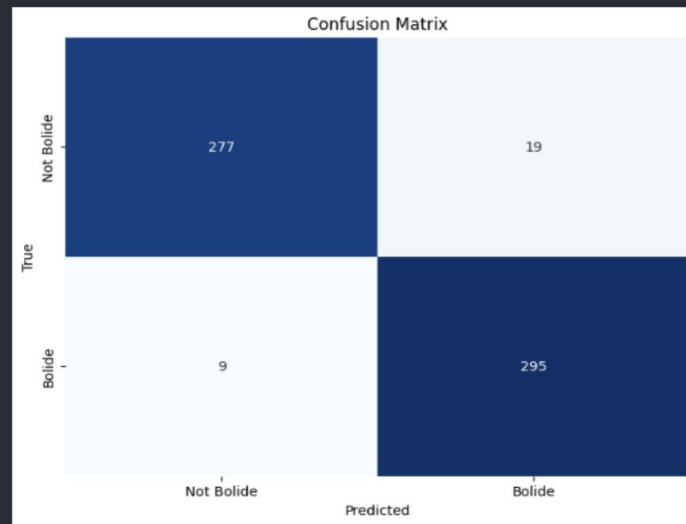
100% | 19/19 [00:01<00:00, 14.73it/s]

Test Loss: 0.1165 Accuracy: 95.33%

Classification Report:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0.0 | 0.97 | 0.94 | 0.95 | 296 |
| 1.0 | 0.94 | 0.97 | 0.95 | 304 |
| accuracy | | | 0.95 | 600 |
| macro avg | 0.95 | 0.95 | 0.95 | 600 |
| weighted avg | 0.95 | 0.95 | 0.95 | 600 |

Testing Confusion Matrix:



Number of misclassified samples: 28

Contribution of Each Member

Tyler Turner

- Looked into captive portal for IoT
- Worked heavily on both APIs and implemented video sending
- Node setup (Ansible playbook)

Vincent Quintero

- Implemented video composites
- Implemented data augmentation for training model

Contribution of Each Member

Jean-Pierre Derbes

- Trained and tuned classification model
- Implemented classification pipeline

Charles Derbes

- Designed classification pipeline
- Implemented object proposer and molder

Task Matrix (Milestone 4)

| Task | Tyler | Vincent | Jean-Pierre | Charles |
|-------------------------------|-------|---------|-------------|---------|
| Implement UI | 10% | 50% | 0% | 40% |
| Polish Server | 50% | 20% | 30% | 0% |
| Polish Client | 30% | 20% | 20% | 30% |
| UI Tests | 0% | 0% | 50% | 50% |
| Server Tests | 50% | 0% | 50% | 0% |
| Client Tests | 0% | 0% | 50% | 50% |
| Create setup process for Node | 75% | 0% | 25% | 0% |

Task Discussion

- Implement UI -> Implement a UI and use researcher feedback to enhance UX.
- Polish Server -> Bugs and performance issues.
- Polish Client -> Video sending needs more testing since it is a core functionality.
- UI Tests -> End-to-end tests using Playwright.
- Server Tests -> Unit tests for each part.
- Create Setup Process for Node -> Hardware testing suite.

Thanks!