feather: Share and deploy your models in under 15 lines of code

0. Introduction

feather is a Python SDK that lets model developers build shareable user interfaces in under 15 lines of code. Alongside this webpage, feather exposes an API that lets others programmatically request your model. Think Gradio/HuggingFace Spaces, but with API access.

feather was originally planned to be a 'marketplace' of AI models -- enabling transparency and reproducibility in the ML community. The incentive structures would encourage researchers to maintain their models and planned research based features could enable leaderboards and RLHF style feedback.

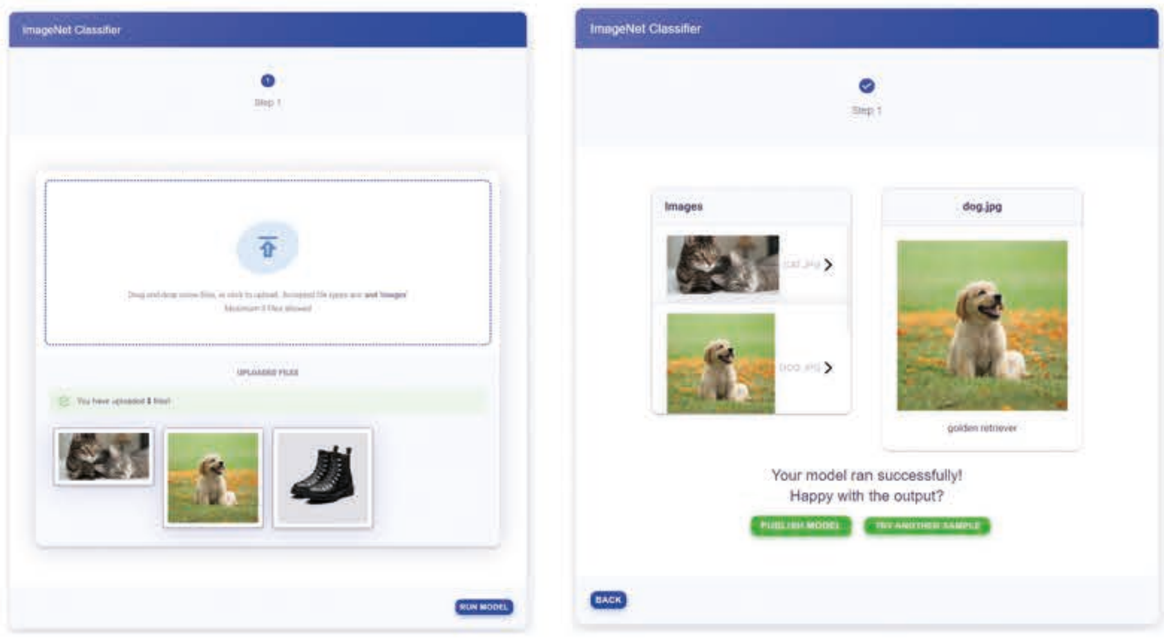
feather is now a dormant project which we are open-sourcing. In this poster, we discuss motivations, technical, and implementation details.

```python
import feather as ftr
from main import my_classifier

def init():
    uploader = ftr.File.Upload(["images"])
    return [uploader]

def run_model(uploader):
    images = uploader.get("images")
    # images = 3 images (List)
    classes = my_classifier(images)
    # classes = [9,113,48]
    return [ftr.Image.View(images, classes)]

if __name__ == "__main__":
    bundle = ftr.bundle("...")
    # bundle is a list of code & model files to upload
    ftr.build(name="ImageNet Classifier",
              init=init, steps=[run_model],
              file_bundle=bundle)
```



An example feather script that deploys an image classifier. Right: The UI rendered by the feather script. 'my_classifier' (line 11) is an inference function that runs a classification model. It is regular Python code that can call any library (e.g. PyTorch, Tensorflow etc.)

1. Motivations

feather was motivated by 1) analysis of existing solutions to deploy/share models; 2) user research.

Analysis of existing solutions: Roughly, there are 3 categories of tools that enable model developers (MDs) to share their models: Code sharing (e.g. GitHub), Self-deployment, and exisiting model sharing tools (e.g. Gradio, Replicate).

Code Sharing	Self-deployment	Model sharing tools
Open source	Full flexibility	Easily shareable
Minimal effort from MD		
Requires environment setup	Different tech stack	No API access
Lack of documentation	Complexity in maintance	Charges the MD
Inaccessible to non-technical users	Discoverability of models	Lack of research-oriented features

Pain points exist for both Model Consumers (MCs) and Model Developers (MDs):

Model Consumers

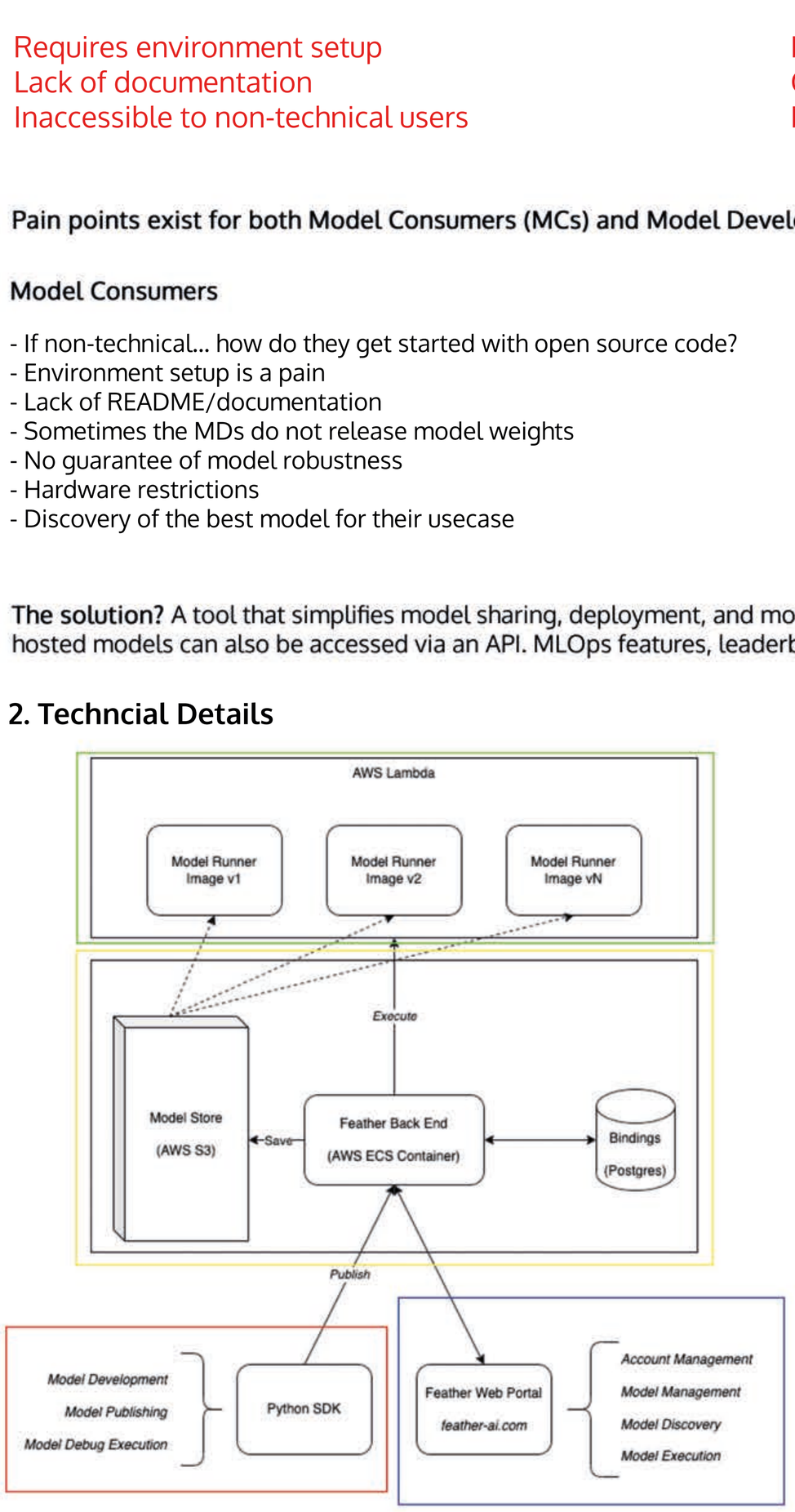
- If non-technical... how do they get started with open source code?
- Environment setup is a pain
- Lack of README/documentation
- Sometimes the MDs do not release model weights
- No guarantee of model robustness
- Hardware restrictions
- Discovery of the best model for their usecase

Model Developers (ML Researchers / Hobbyists / Industry Professionals)

- ML Researchers have good intentions when releasing code, but have many responsibilities and no clear incentives to maintain code. Thus code becomes outdated.
- Hobbyists want their model to be used more by non-technical MCs. The process of sharing their model (e.g. creating a front-end, deploying a model) takes a long time.
- MDs in industry use screen sharing to report progress to stakeholders. Other people are hired to assist with the deployment/accessibility side of models. Internal adoption and user validation could be sped up and improved. Also industry has a high want for MLOps (e.g. viewing data sent to a model).

The solution? A tool that simplifies model sharing, deployment, and model maintanence. Specifically, an SDK that allows MDs to build shareable web interfaces for their model. The hosted models can also be accessed via an API. MLOps features, leaderboards etc. can then be trivially added.

2. Techncial Details



The Python SDK is a client-facing python library. It implements feather's component APIs which allows MCs to quickly create multi-step models. Rich metadata about the model is generated, and this information is later used when executing the model.

The Web Portal is single page application which allows unauthenticated users to browse and execute available models. Authenticated users can use the portal to manage their published models. The execution requires utilising the per model metadata (created by the SDK during publishing) to generate the UI dynamically for each model.

feather's back end associated database, provides server-side functionality via a REST API. Both the SDK and the Web Portal talk directly with this service (for authentication, data validation, etc) to perform all actions on the feather platform.

The model runner is an AWS Lambda image that is dynamically created from a Base Image for each model + dependencies type. The runner is loaded as needed to execute an MDs model. During execution, the image is loaded and executed by the Lambda runtime in a sandbox environment for safe execution of user-provided code.


```json
{"steps": [
  {
    "name": "upload_images",
    "inputs": [
      {
        "component": "File.Upload",
        "props": {
          "max_files": 5,
          "min_files": 1,
          "title": "Upload images",
          "types": [
              ".jpg",
              ".png"
          ]
        },
        "schema": {...}
      }
    ]
  },
  {
    "name": "select_objects",
    "inputs": [
      {
        "component": "Image.WithSelectMulti",
        "props": {...},
        "schema": {
          "images": [
            {
              "attributes": [
                {
                  "item": "string",
                  "selected": "bool"
                }
              ],
              "data": "b64",
              "name": "string"
            }
          ]
        }
      }
    ]
  },
  {...}
]}
```


JSON metadata produced by the SDK. The JSON is sent to the front-end to dynamically render an interactive UI