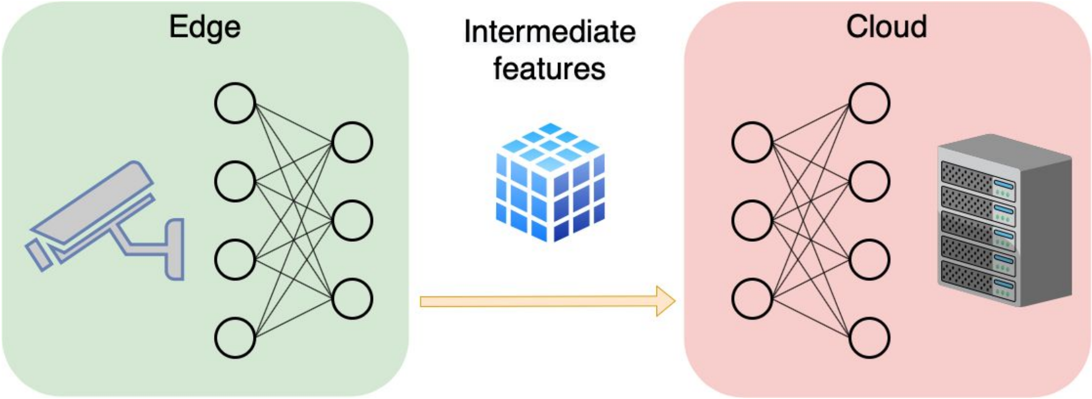
ColliFlow: A Library for Executing Collaborative Intelligence Graphs

Mateen Ulhaq
Ivan V. Bajić



Outline

1. Background
2. Library usage example
3. Demo: Android
4. Q&A

Shared inference

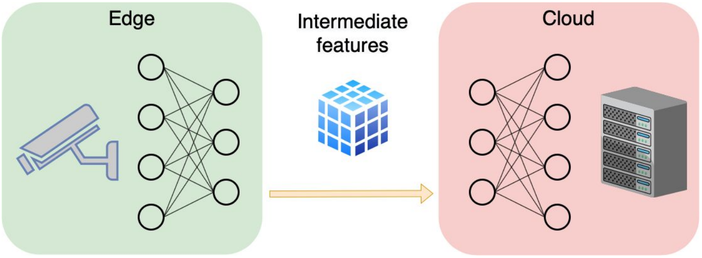
Key idea: less data sent over network

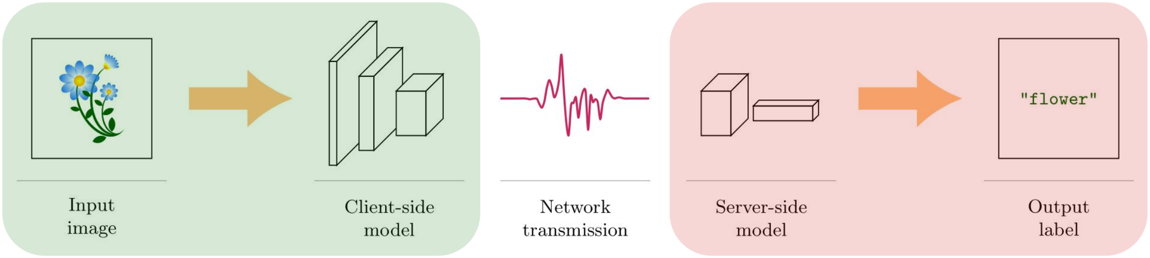
Versus cloud-only inference:

- Save bandwidth
- Save device energy
- Reduce inference times

Versus edge-only inference:

- Bigger models
- Reduce resource usage
- Reduce inference times



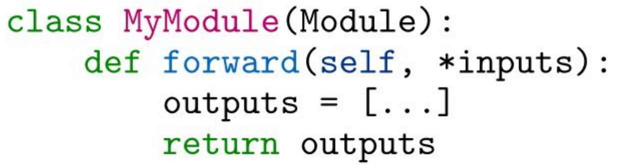


Library goals

- Collaborative intelligence graphs
- Easy implementation for developers
- Fast experimentation for researchers
- Common API for:
  - edge devices (Android, Kotlin)
  - servers (Python)

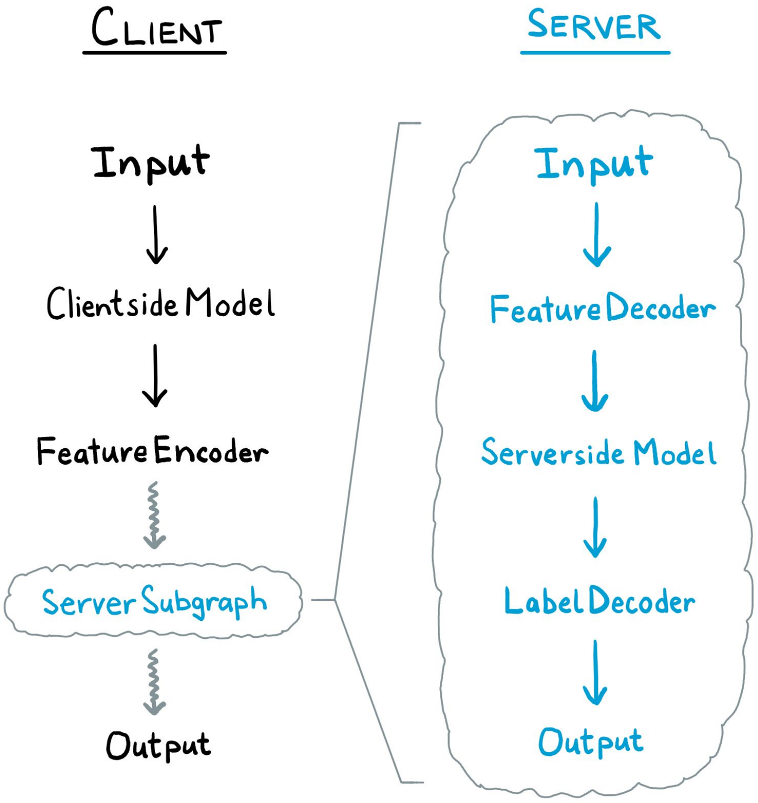
Module graph

Module definition:



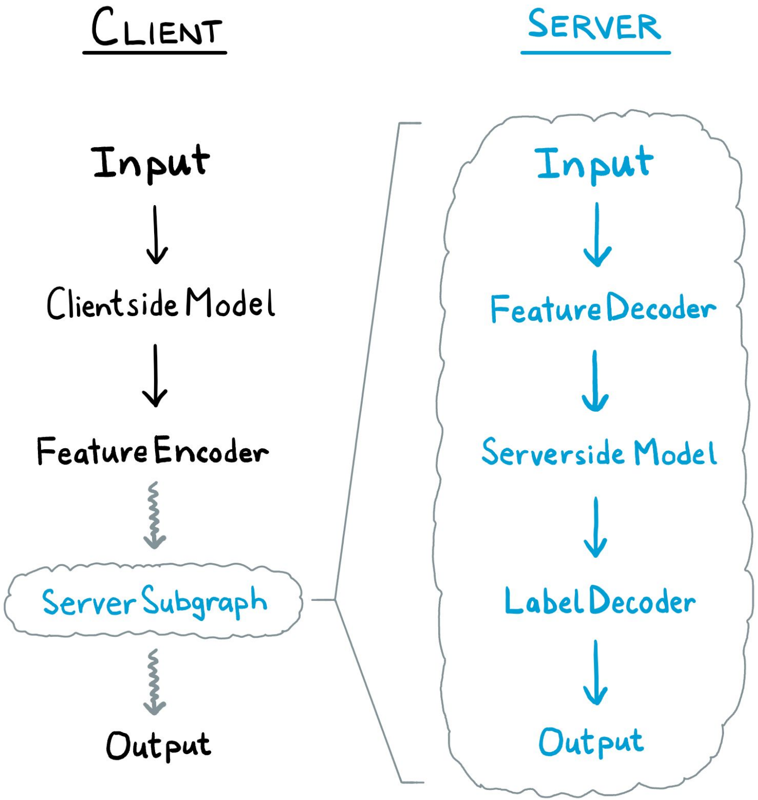
```python
class MyModule(Module):
    def forward(self, *inputs):
        outputs = [...]
        return outputs
```

Modules are linked together in a graph



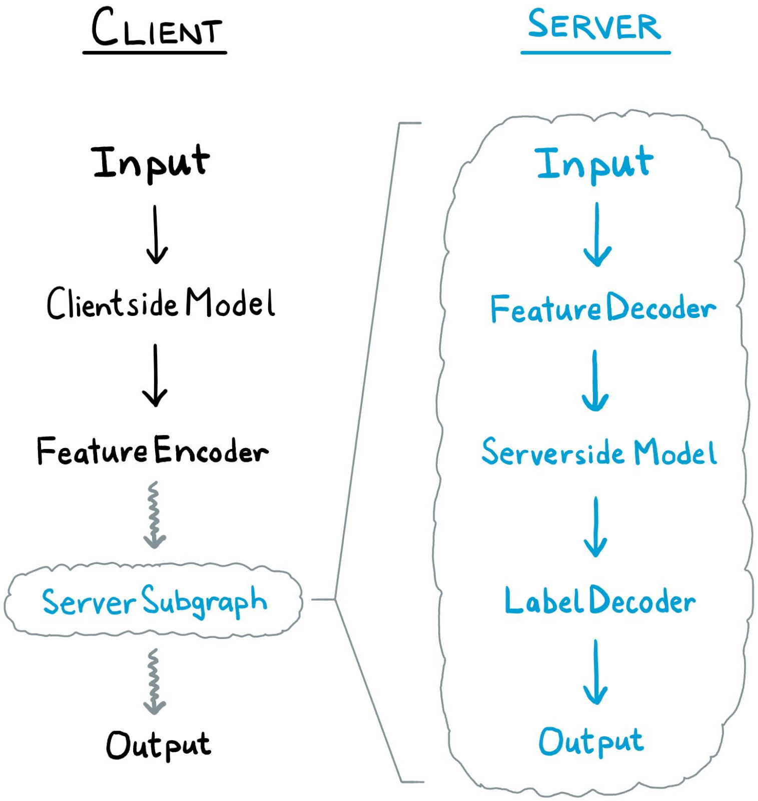
client.py

```
from colliflow import *

def create_client_graph():
    inputs = [Input(shape=(224, 224, 3), dtype="uint8")]
    x = inputs[0]
    x = H5Model(filename="client-model.h5")(x)
    x = JpegEncoder()(x)
    x = TcpServerModule(
        graph=create_server_graph(), addr="example.com:5678"
    )(x)
    outputs = [x]
    return Model(inputs=inputs, outputs=outputs)
```



client.py

```python
def create_server_graph():
    inputs = [Input(shape=(None,), dtype="bytes")]
    x = inputs[0]
    x = JpegDecoder()(x)
    x = H5Model(filename="server-model.h5")(x)
    x = DecodeTopImagenetLabels(top_n=3)(x)
    outputs = [x]
    return Model(inputs=inputs, outputs=outputs)
```

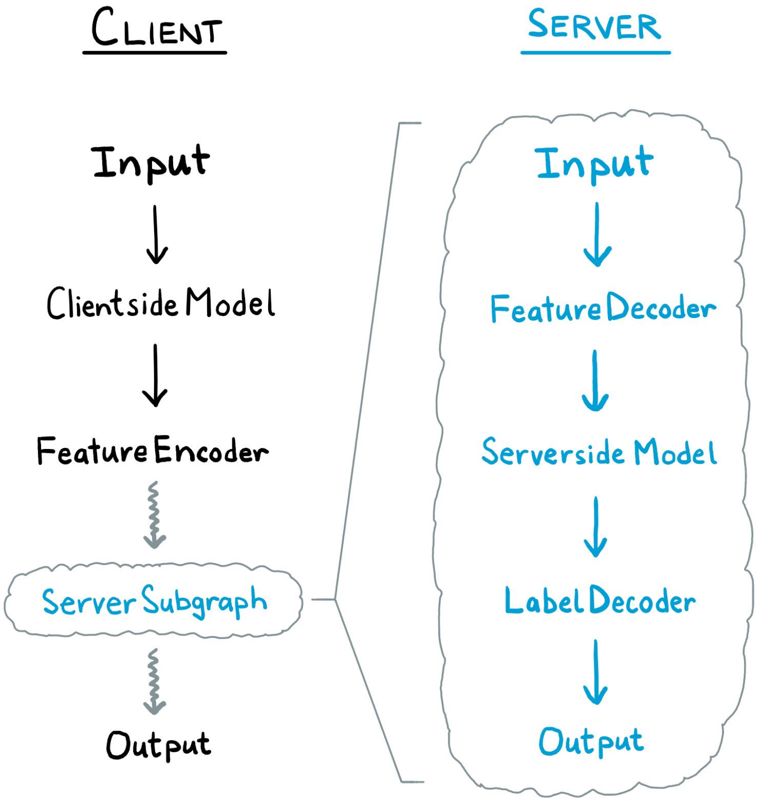


client.py

```
frames = video_source("example.mp4")
client_graph = create_client_graph()
outputs = client_graph.start(inputs=[frames])
outputs[0].subscribe(print)
```

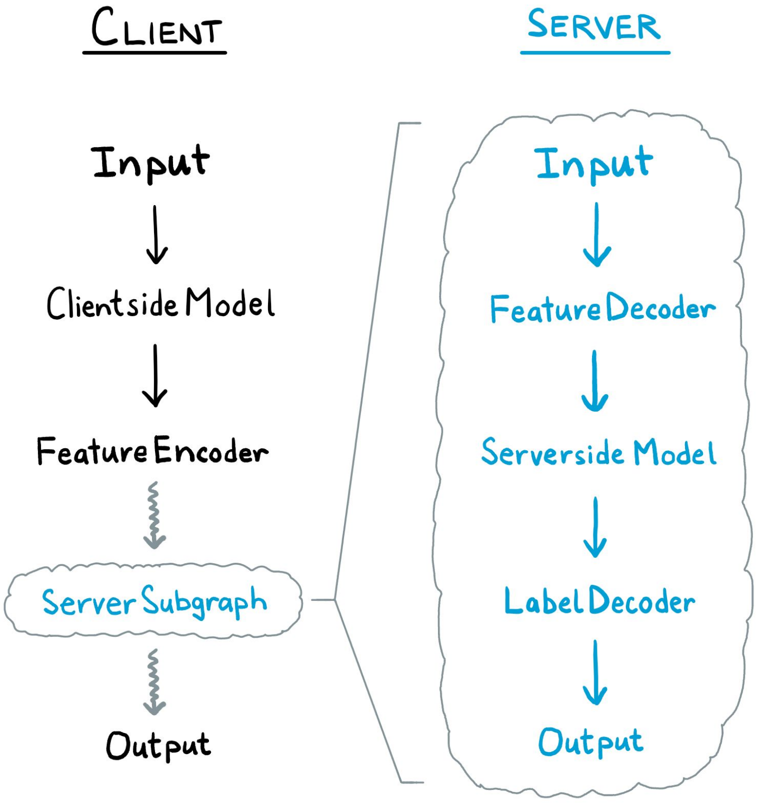
Output:

```
42% cat
21% dog
11% flower
```



server.py

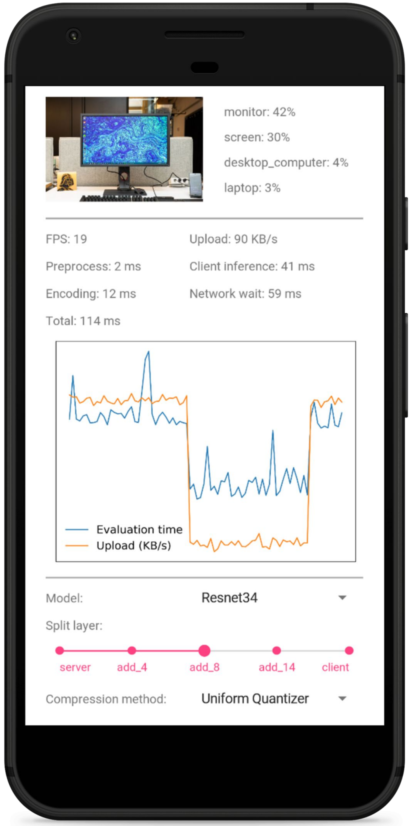
```
from colliflow import *

server = Server()
server.start(port=5678)
```



Demo: Android

Demoed at NeurIPS 2019.

Edge client:	Android;	Kotlin, Tensorflow Lite
Cloud server:	Linux;	Python, Tensorflow



monitor: 42%

screen: 30%

desktop_computer: 4%

laptop: 3%

FPS: 19

Upload: 90 KB/s

Preprocess: 2 ms

Client inference: 41 ms

Encoding: 12 ms

Network wait: 59 ms

Total: 114 ms

Model: Resnet34

Split layer:

Compression method: Uniform Quantizer

ColliFlow

- Define collaborative intelligence graphs via functional API
- Over-the-network graph execution
- Reactive Extensions (Rx) integration
- Built-in modules for feature tensor data compression and transmission

Thank you

`https://github.com/YodaEmbedding/colliflow`