ML 2
Weathermood: StarGAN

Software Studio DataLab, CS, NTHU

Outline

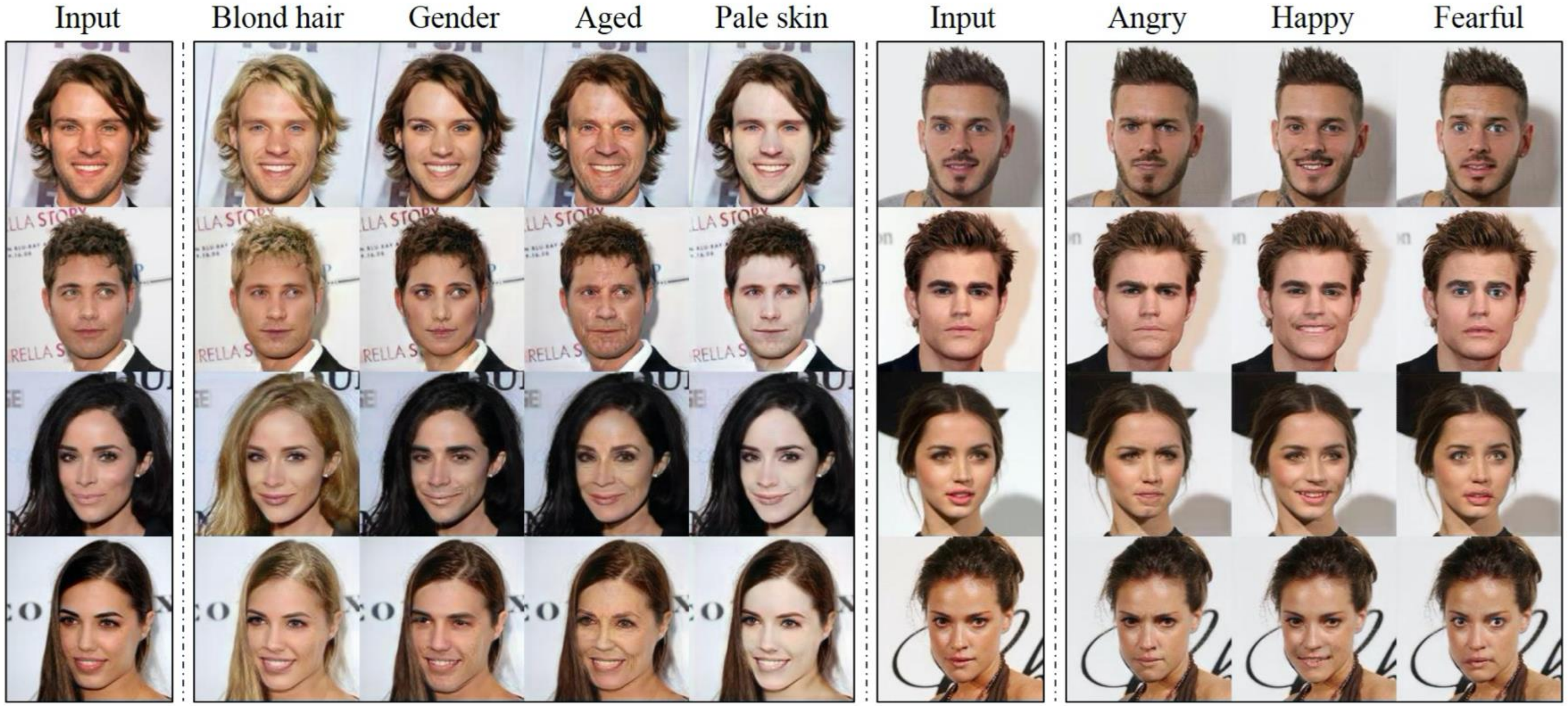
- Introduction to StarGAN
- Weathermood-StarGAN
  - Frontend
  - Backend

Outline

- Introduction to StarGAN
- Weathermood-StarGAN
  - Frontend
  - Backend

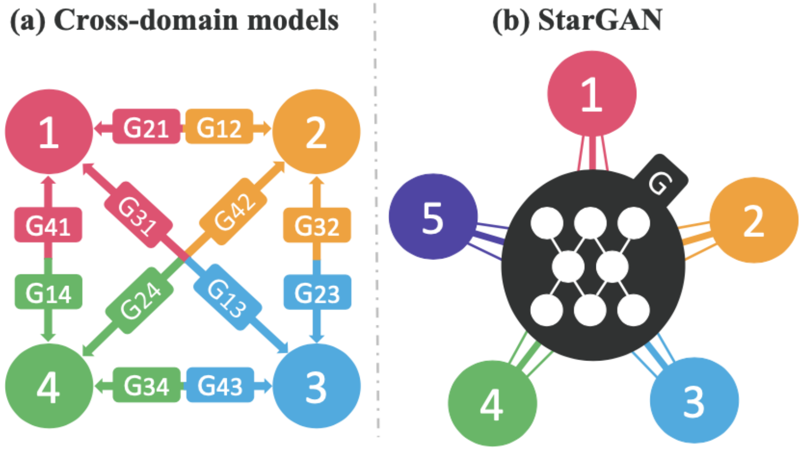
StarGAN

- Image-to-image cross domain translation



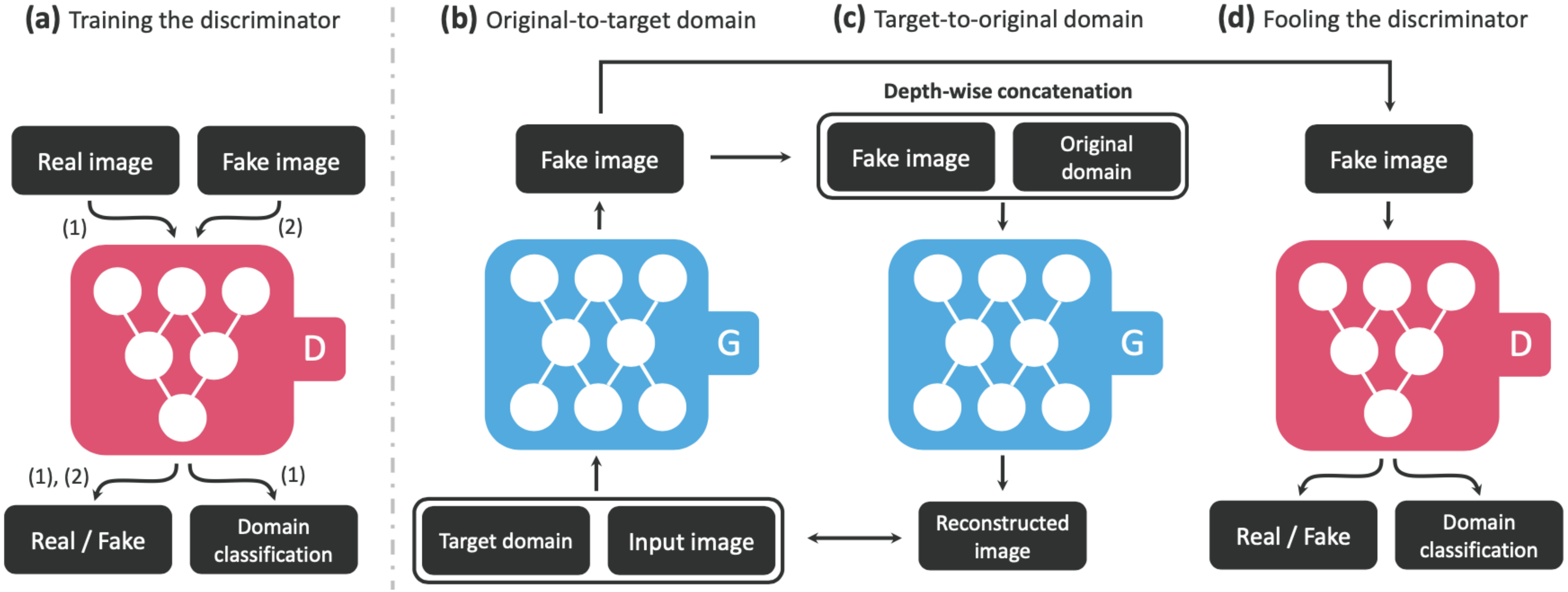
StarGAN

- Normal cross-domain models require multiple model weights to handle
- starGAN is capable of learning the mappings using a single generator



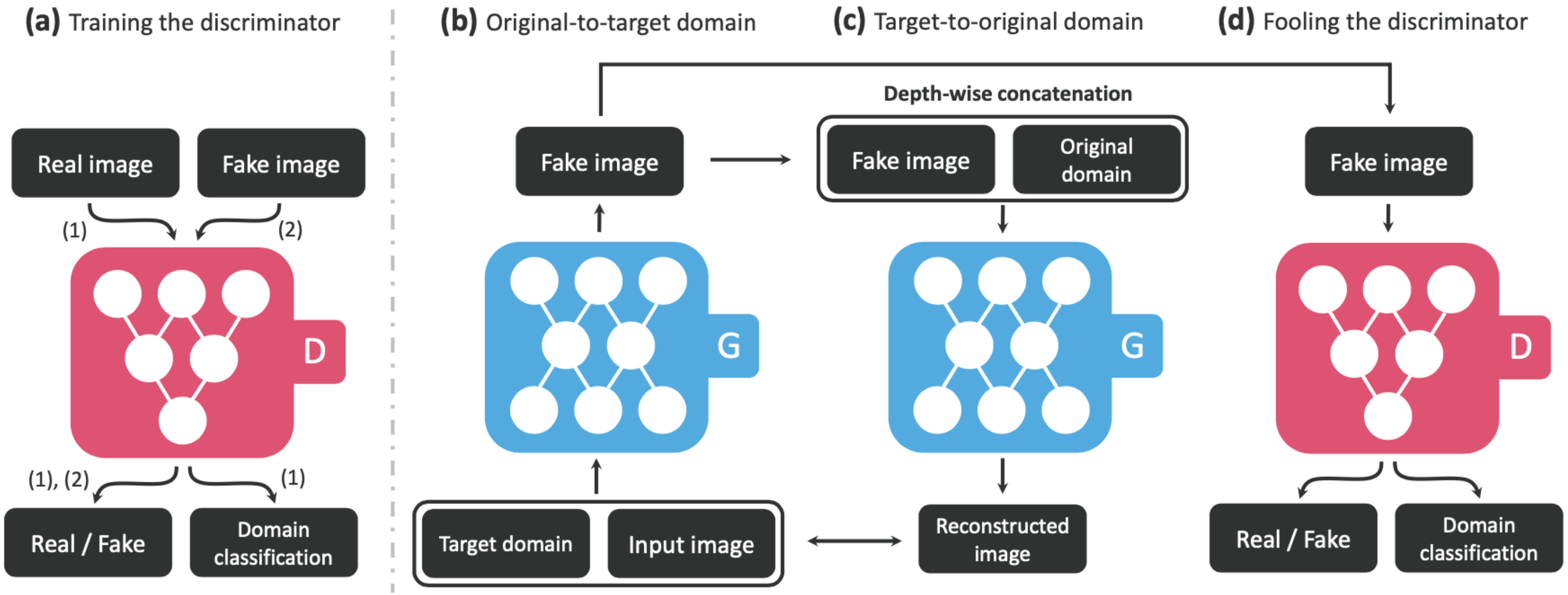
StarGAN

- Training process



StarGAN

- Training process



Reconstruction loss

Adversarial loss

Domain classification loss

StarGAN

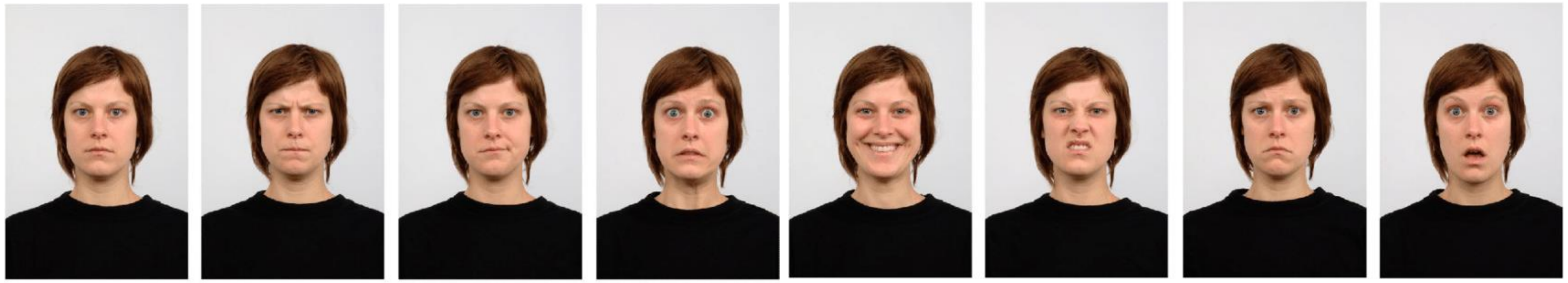
- Adversarial Loss
  - Distinguish real/fake images

- Domain Classification Loss
  - Classify the domain correctly

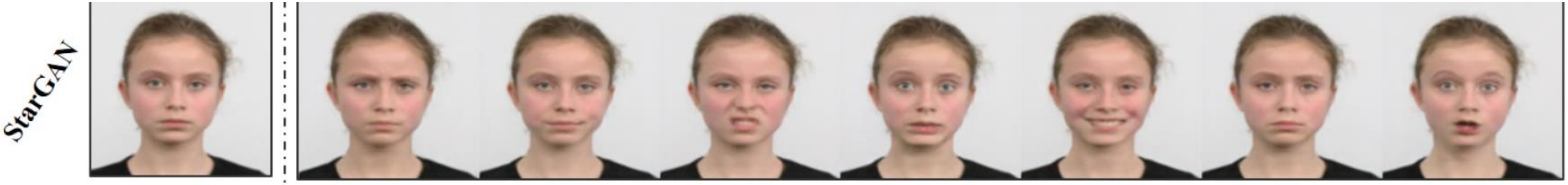
- Reconstruction Loss
  - Generator should be able to reconstruct the image using a same domain input

RaFD dataset

- Face dataset with eight emotion labels
  - angry, contemptuous, disgusted, fearful, happy, neutral, sad, surprised



- StarGAN result on RaFD dataset



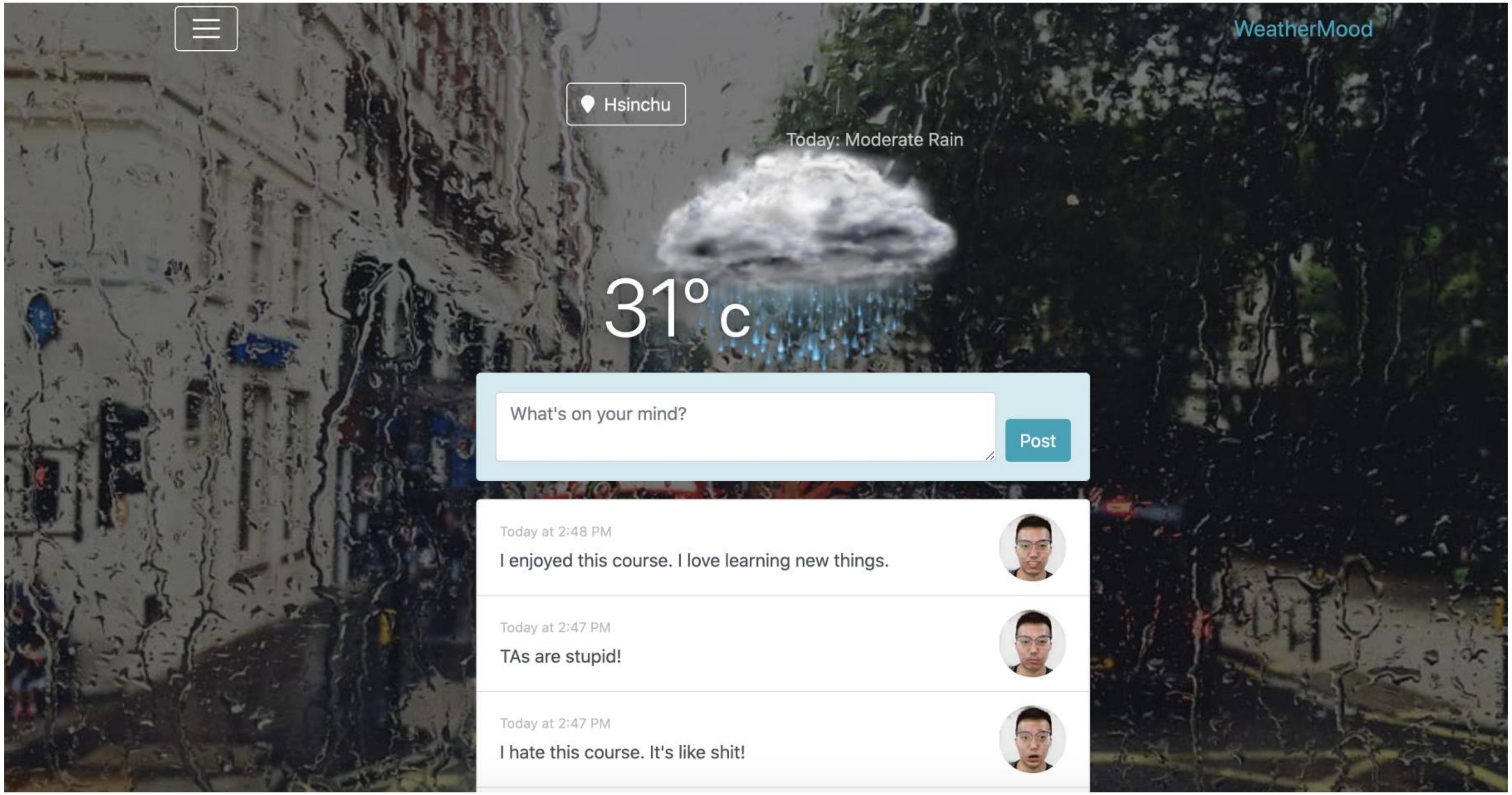
Outline

- Introduction to StarGAN

- Weathermood-StarGAN
  - Frontend
  - Backend

Weathermood-StarGAN

- The face will change based on the toxicity detection result



Today at 2:48 PM

I enjoyed this course. I love learning new things.

Today at 2:47 PM

TAs are stupid!

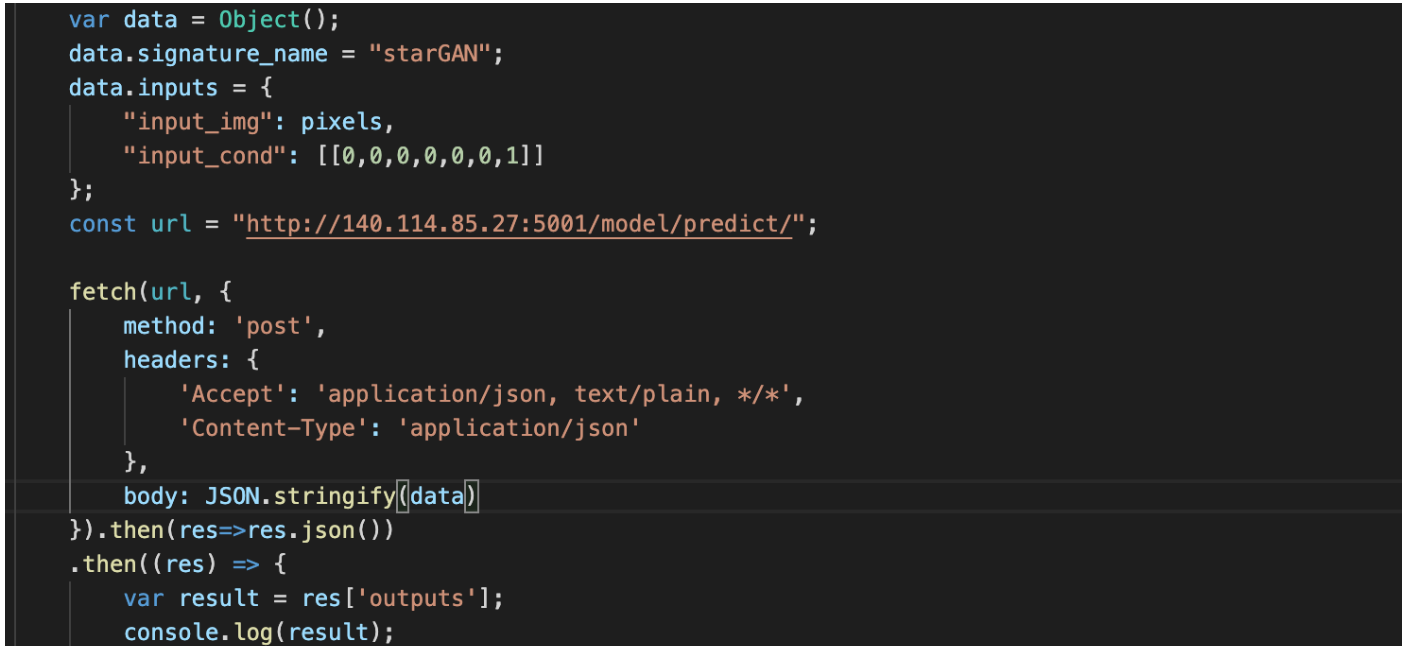
Today at 2:47 PM

I hate this course. It's like shit!

Weathermood-StarGAN

- Self-hosted starGAN model using tensorflow-serving
  - https://www.tensorflow.org/tfx/serving/serving_basic

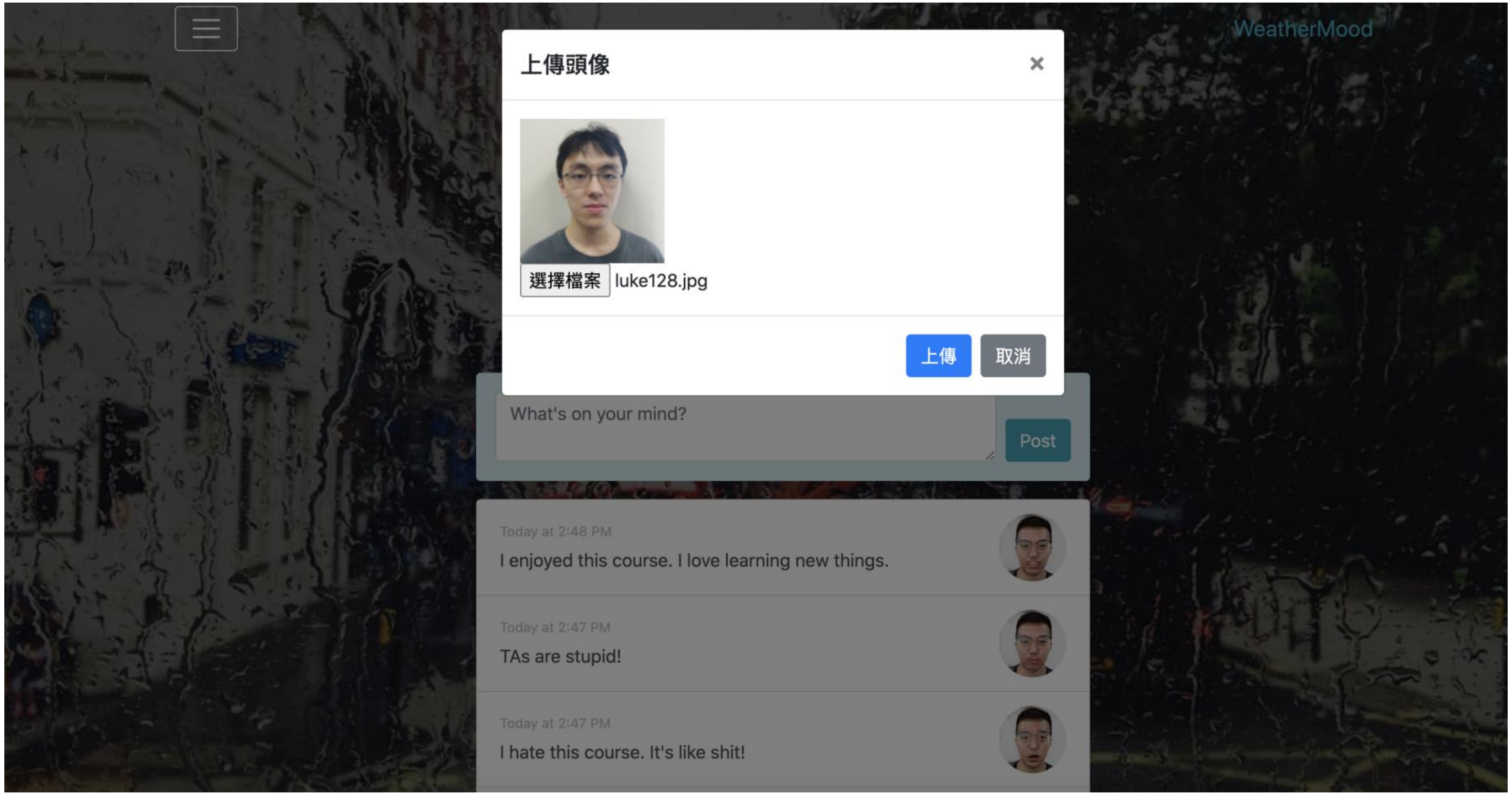
- Fetch the predicted result from a remote server



```
var data = Object();
data.signature_name = "starGAN";
data.inputs = {
    "input_img": pixels,
    "input_cond": [[0,0,0,0,0,0,1]]
};
const url = "http://140.114.85.27:5001/model/predict/";

fetch(url, {
    method: 'post',
    headers: {
        'Accept': 'application/json, text/plain, */*',
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(data)
}).then(res=>res.json())
.then((res) => {
    var result = res['outputs'];
    console.log(result);
```

Weathermood-StarGAN

- Upload custom images



luke128.jpg

What's on your mind?

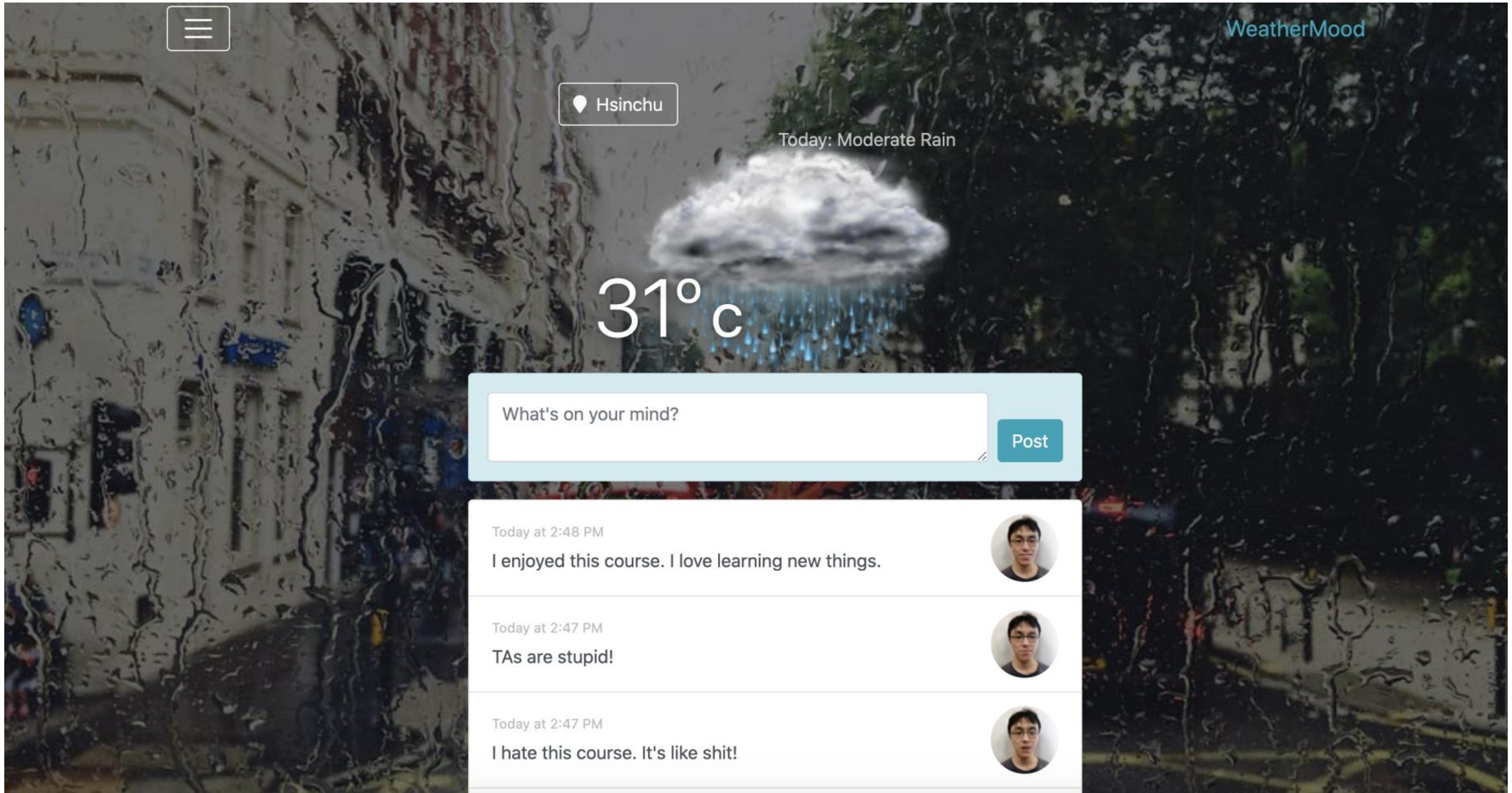
Today at 2:48 PM
I enjoyed this course. I love learning new things.

Today at 2:47 PM
TAs are stupid!

Today at 2:47 PM
I hate this course. It's like shit!

Weathermood-StarGAN

- Result



Today at 2:48 PM

I enjoyed this course. I love learning new things.

Today at 2:47 PM

TAs are stupid!

Today at 2:47 PM

I hate this course. It's like shit!

Outline

- Introduction to StarGAN

- Weathermood-StarGAN
    - Frontend
    - Backend

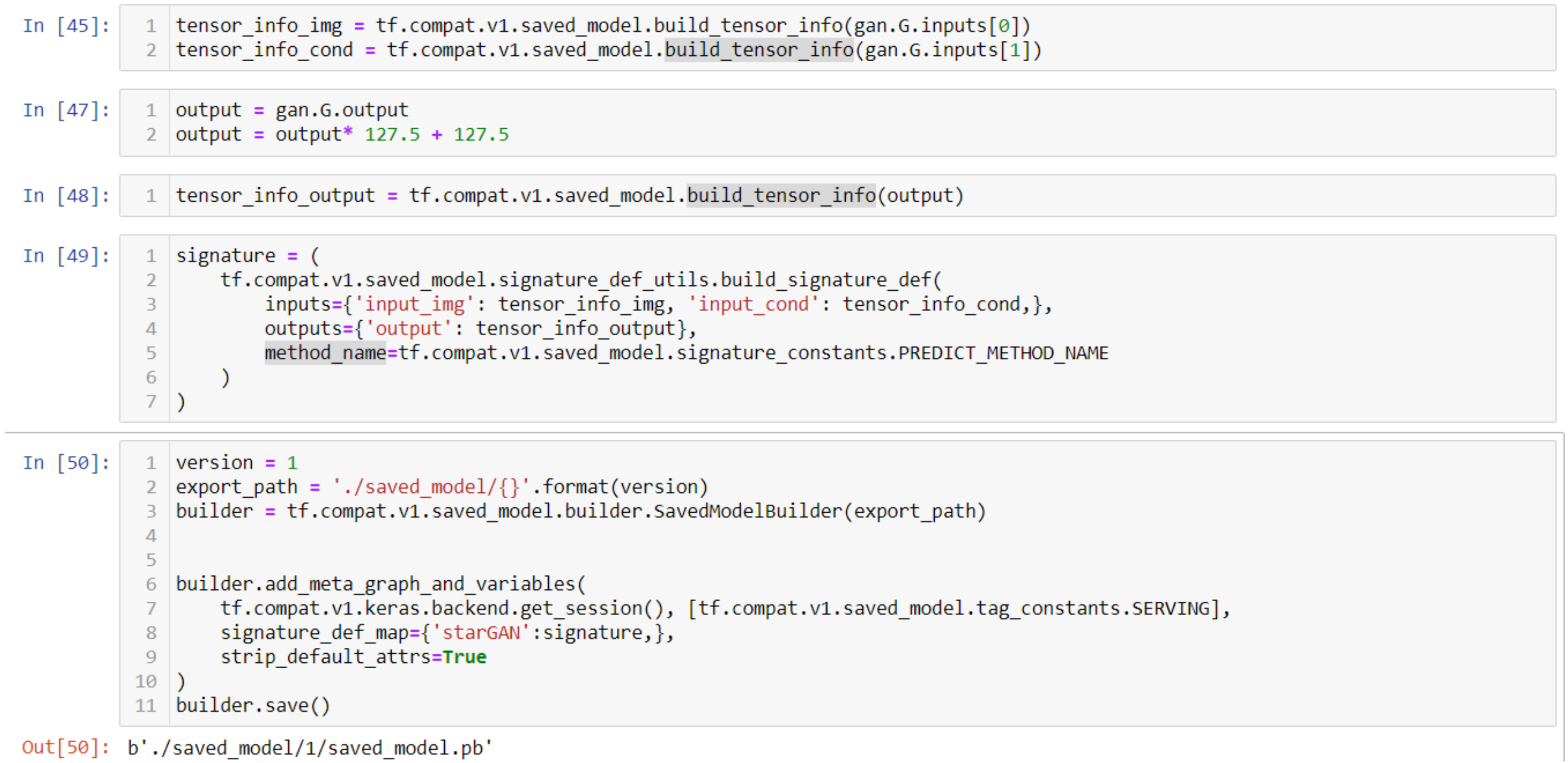
Backend

- To host your model, you need to have a model first
  - Tensorflow-compatible model
  - https://github.com/hoangthang1607/StarGAN-Keras
  - Git clone and follow the readme file

- To run your model, you need a python + tensorflow environment
  - https://www.tensorflow.org/install
  - You can install using Anaconda
  - https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/

Backend

- To use Tensorflow-serving, first save your model with SavedModel format



```
In [45]:
tensor_info_img = tf.compat.v1.saved_model.build_tensor_info(gan.G.inputs[0])
tensor_info_cond = tf.compat.v1.saved_model.build_tensor_info(gan.G.inputs[1])
```

```
In [47]:
output = gan.G.output
output = output* 127.5 + 127.5
```

```
In [48]:
tensor_info_output = tf.compat.v1.saved_model.build_tensor_info(output)
```

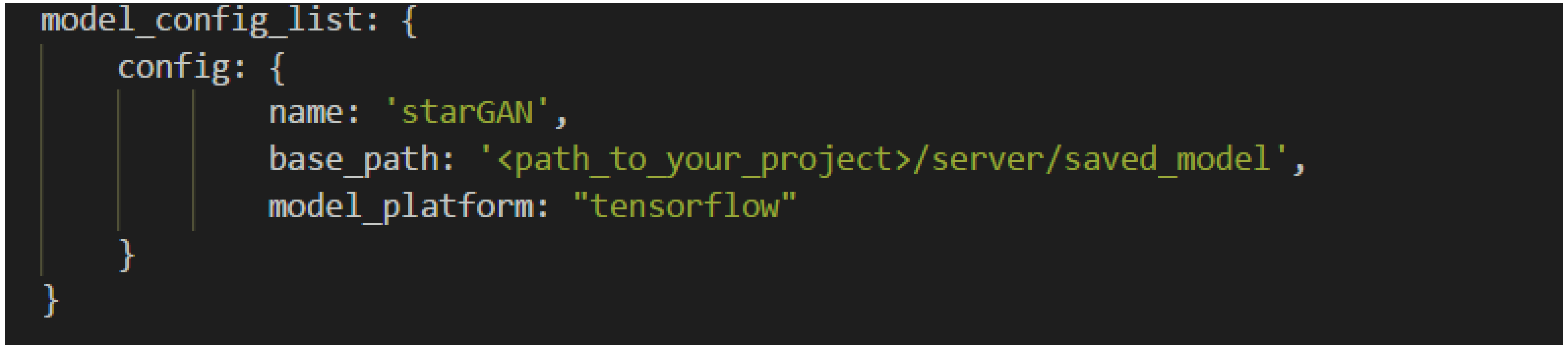
```
In [49]:
signature = (
    tf.compat.v1.saved_model.signature_def_utils.build_signature_def(
        inputs={'input_img': tensor_info_img, 'input_cond': tensor_info_cond,},
        outputs={'output': tensor_info_output},
        method_name=tf.compat.v1.saved_model.signature_constants.PREDICT_METHOD_NAME
    )
)
```

```
In [50]:
version = 1
export_path = './saved_model/{}'.format(version)
builder = tf.compat.v1.saved_model.builder.SavedModelBuilder(export_path)


builder.add_meta_graph_and_variables(
    tf.compat.v1.keras.backend.get_session(), [tf.compat.v1.saved_model.tag_constants.SERVING],
    signature_def_map={'starGAN':signature,},
    strip_default_attrs=True
)
builder.save()

Out[50]: b'./saved_model/1/saved_model.pb'
```

Backend

- Setup Tensorflow-serving
  - https://www.tensorflow.org/tfx/serving/setup

- Config file
  - https://www.tensorflow.org/tfx/serving/serving_config#model_server_configuration



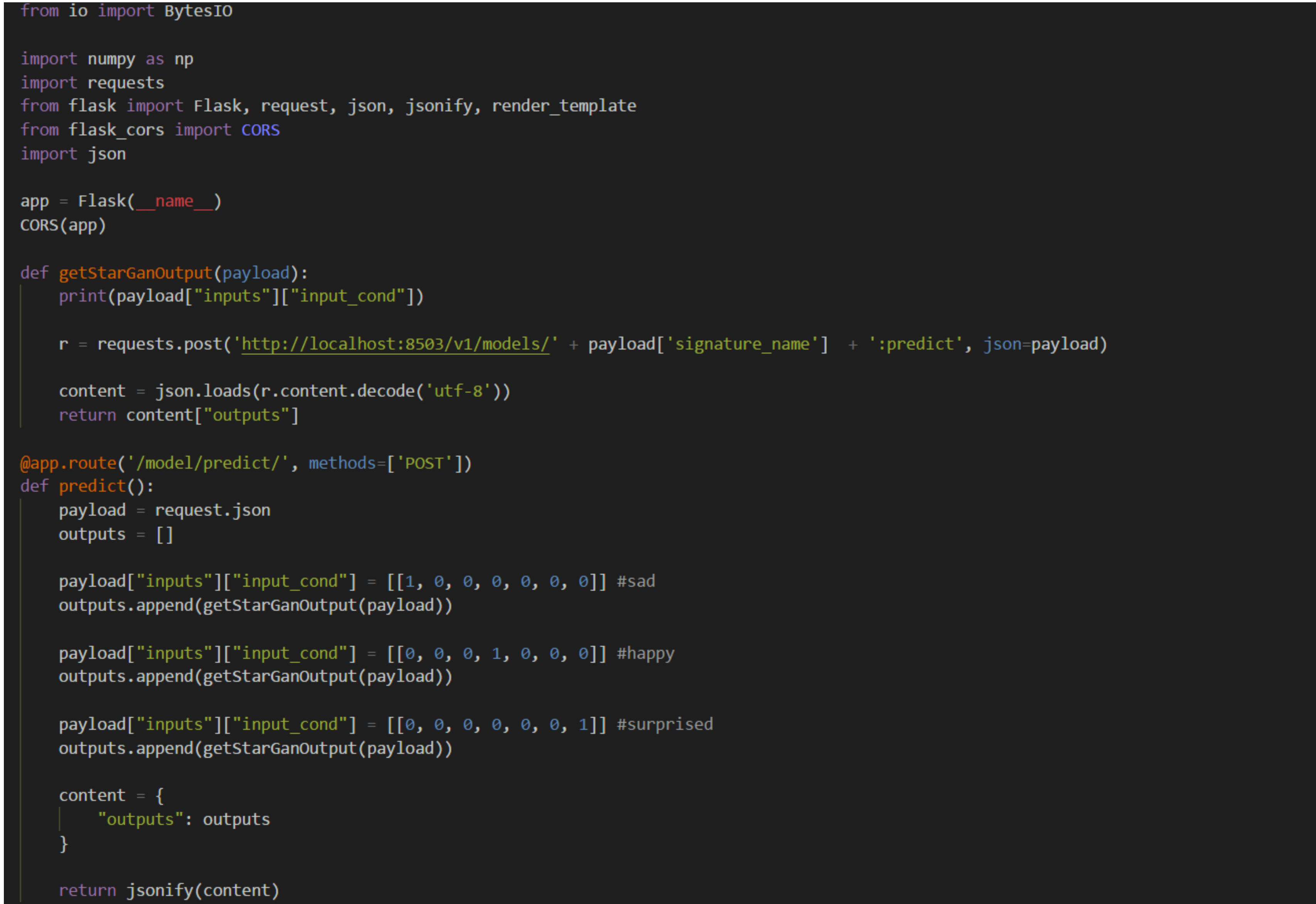
```
model_config_list: {
    config: {
            name: 'starGAN',
            base_path: '<path_to_your_project>/server/saved_model',
            model_platform: "tensorflow"
    }
}
```

Backend

- Host your model with Tensorflow-serving
  - tensorflow_model_server --rest_api_port=<your_port> --model_config_file=<your_path>/models.conf
- Test your API
  - https://www.tensorflow.org/tfx/serving/api_rest

Backend

- Proxy Server



```python
from io import BytesIO

import numpy as np
import requests
from flask import Flask, request, json, jsonify, render_template
from flask_cors import CORS
import json

app = Flask(__name__)
CORS(app)

def getStarGanOutput(payload):
    print(payload["inputs"]["input_cond"])

    r = requests.post('http://localhost:8503/v1/models/' + payload['signature_name']  + ':predict', json=payload)

    content = json.loads(r.content.decode('utf-8'))
    return content["outputs"]

@app.route('/model/predict/', methods=['POST'])
def predict():
    payload = request.json
    outputs = []

    payload["inputs"]["input_cond"] = [[1, 0, 0, 0, 0, 0, 0]] #sad
    outputs.append(getStarGanOutput(payload))

    payload["inputs"]["input_cond"] = [[0, 0, 0, 1, 0, 0, 0]] #happy
    outputs.append(getStarGanOutput(payload))

    payload["inputs"]["input_cond"] = [[0, 0, 0, 0, 0, 0, 1]] #surprised
    outputs.append(getStarGanOutput(payload))

    content = {
        "outputs": outputs
    }

    return jsonify(content)
```

Thank You~