ML 2
Weathermood: StarGAN

Software Studio DataLab, CS, NTHU

Outline

- GAN and StarGAN
- Weathermood-StarGAN
- Backend

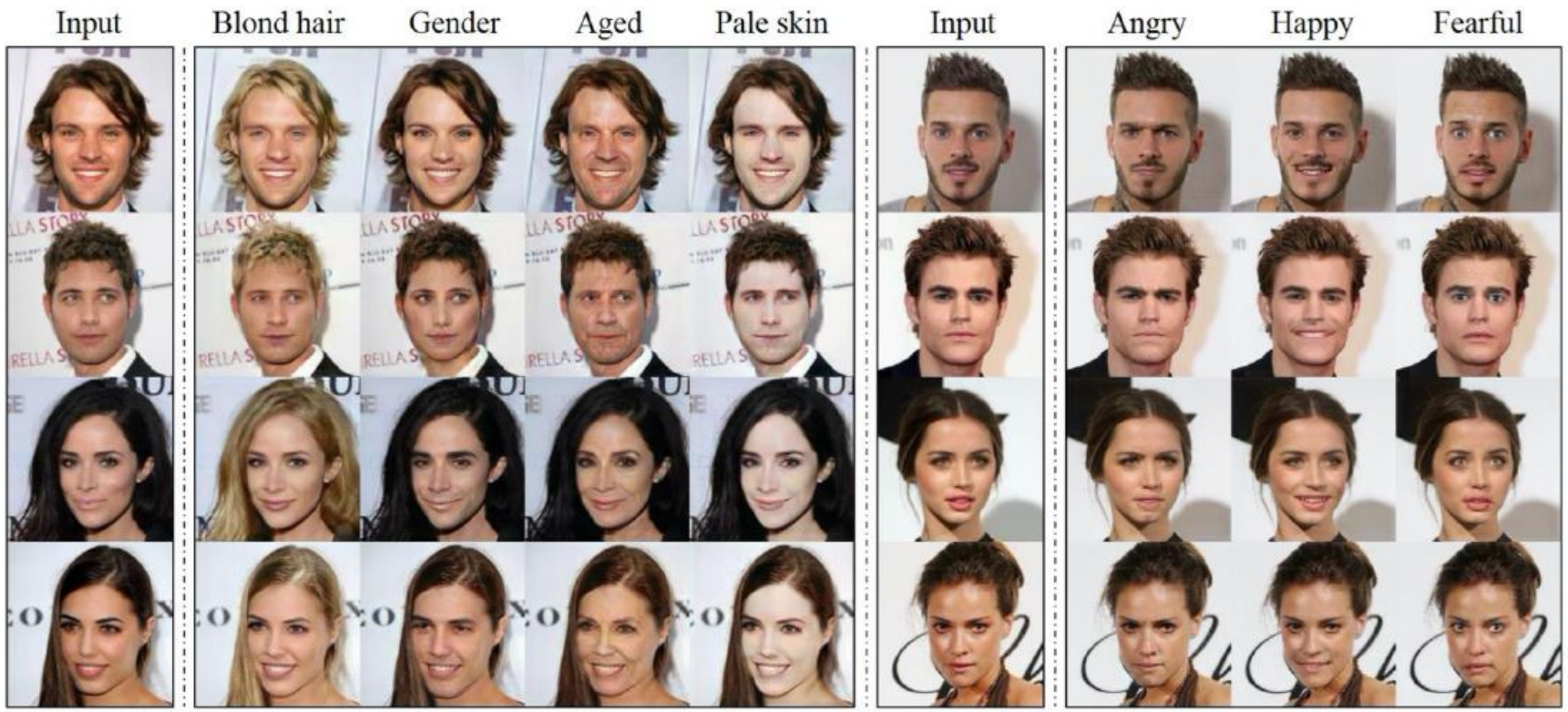
GAN

- Generative and Adversarial Network.
  - Generator: Generate data from random noise.
  - Discriminator: Separate generated data from real.

Ref: https://easyai.tech/ai-definition/gan/

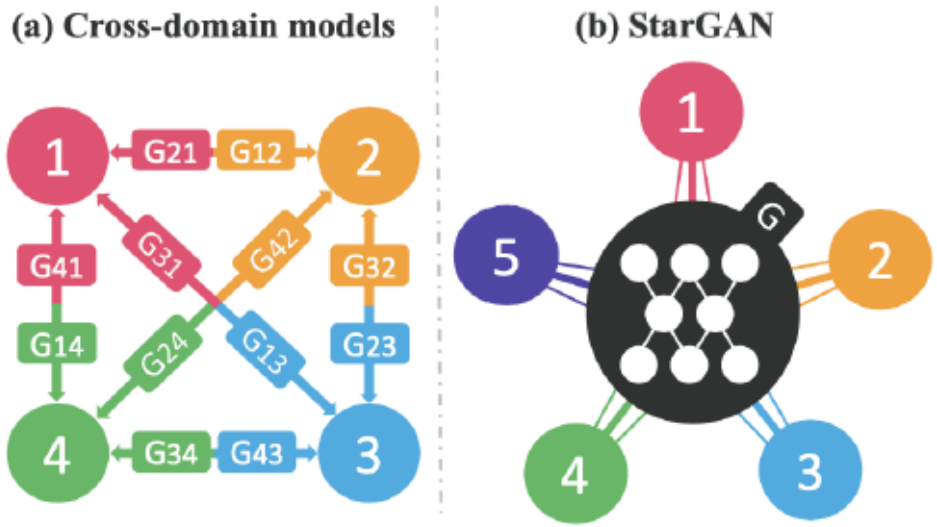
StarGAN

- Image-to-image cross domain translation.
  - Domain: A set of images sharing the same attribute value.

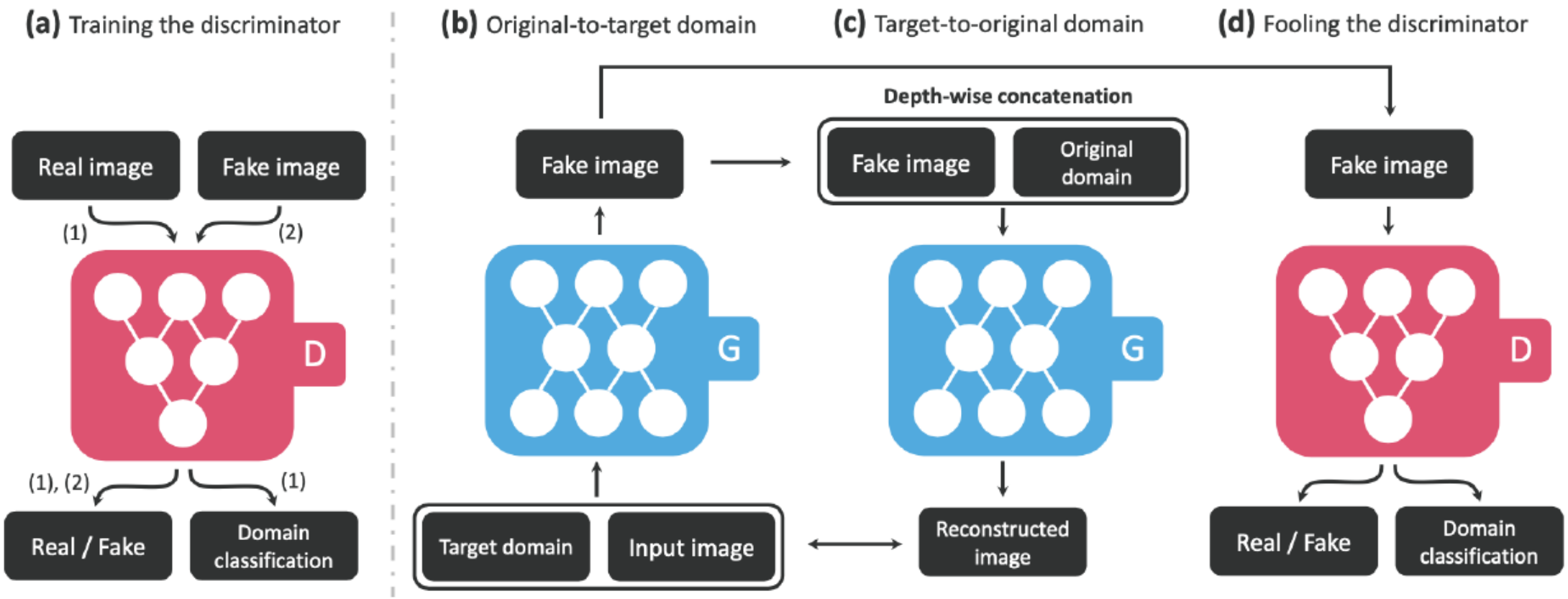


StarGAN

- Existing approaches have limited scalability and robustness in handling more than two domains.
- StarGAN is a novel and scalable approach that can perform image-to-image translations for multiple domains ***using only a single model***.



StarGAN


(a) Training the discriminator
(b) Original-to-target domain
(c) Target-to-original domain
(d) Fooling the discriminator
Depth-wise concatenation
Real image
Fake image
(1)
(2)
D
(1), (2)
(1)
Real / Fake
Domain classification
Fake image
G
Target domain
Input image
Fake image
Original domain
G
Reconstructed image
Fake image
D
Real / Fake
Domain classification

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.

Dataset

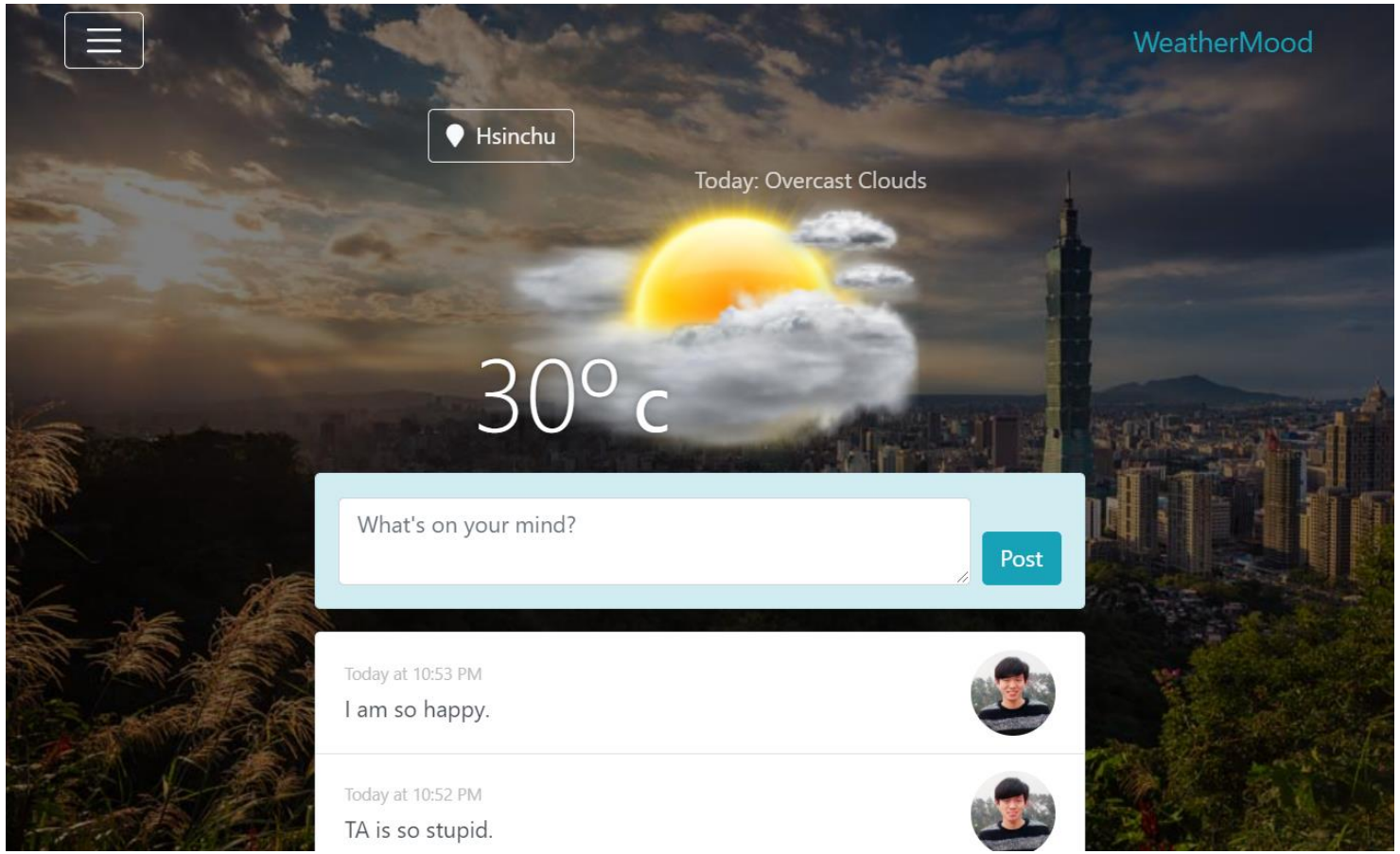
- RaFD dataset.
  - http://www.socsci.ru.nl:8180/RaFD2/RaFD

- CFEE dataset.
  - http://cbcsl.ece.ohio-state.edu/dbform_compound.html

- Data augmentation.

Weathermood-StarGAN

- The face will change based on the toxicity detection result.



Weathermood-StarGAN

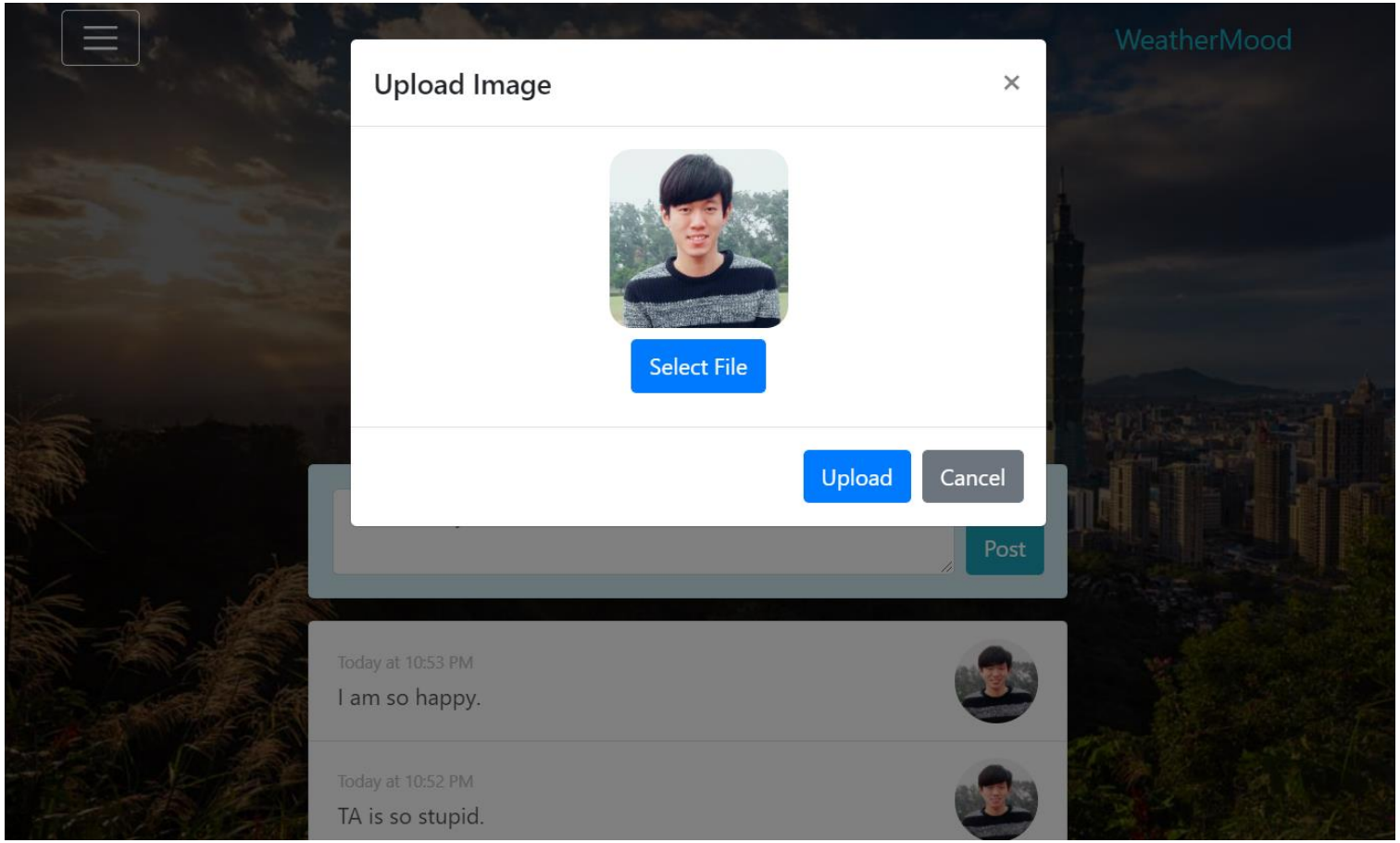
- Self-hosted starGANmodel using tensorflow-serving.
  - https://www.tensorflow.org/tfx/serving/serving_basic

- Fetch the predicted result from a remote server.

```
var data = Object();
data.signature_name = "starGAN";
data.inputs = {
    "input_img": pixels
};
const url = "http://140.114.88.21:5000/model/predict/";
console.log(`Making request to: ${url}`);
this.props.setPredicting(true);
this.props.changeFaceFile();
fetch(url, {
    method: 'post',
    headers: {
        'Accept': 'application/json, text/plain, */*',
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(data)
```

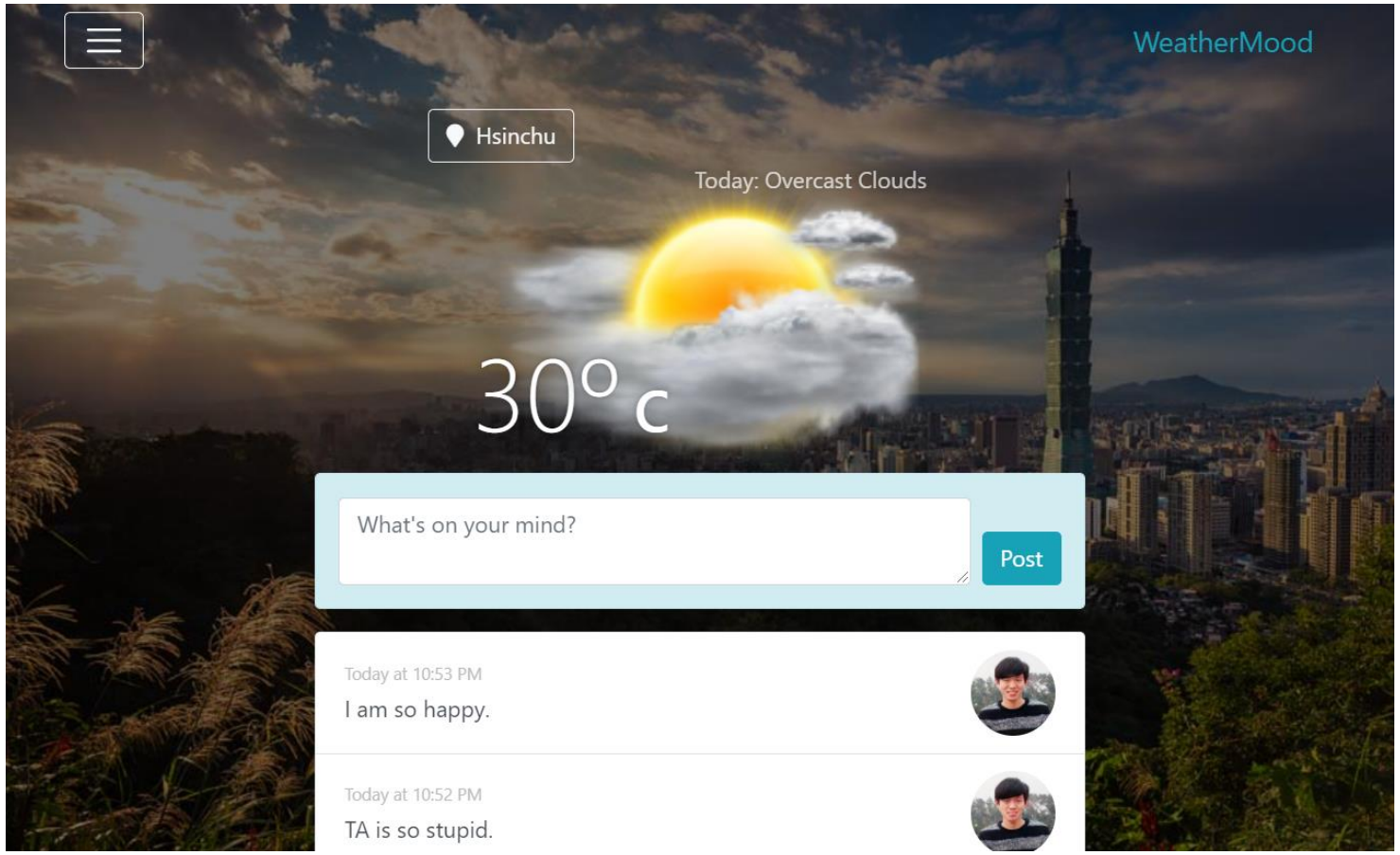
Weathermood-StarGAN

- You can upload your custom face images.



Weathermood-StarGAN

- And you will get the predicted results from the server.



Backend

- You can try to train a model by yourselves or use our pre-trained model.
  - https://github.com/Masao-Taketani/StarGAN-tf2 123
  - StarGan folder in gitlab.

- To run the model, you need a python + tensorflow environment.
  - Download Anaconda
  - Build an environment
  - Install tensorflow==2.3(w/o gpu) or tensorflow-gpu==2.3 in your environment.

Backend

- Run a small server to handle the request and return the predicted results to the client.

```python
def getStarGanOutput(payload):

    img = np.float32(payload['inputs']['input_img'][0])
    cond = tf.constant(payload['inputs']['input_cond'])
    result = generate_html_image(gen, img, cond)

    return result * 255


@app.route('/model/predict/', methods=['POST'])
def predict():
    '''
        Note:
            1. The shape of ['inputs']['input_img'] is (1, 128, 128, 3), so please take the first element for the model input.
    '''
    payload = request.json
    outputs = []
    payload['inputs']['input_cond'] = [[0, 1, 0, 0]]
    outputs.append(getStarGanOutput(payload).tolist())

    payload['inputs']['input_cond'] = [[0, 0, 1, 0]]
    outputs.append(getStarGanOutput(payload).tolist())

    payload['inputs']['input_cond'] = [[0, 0, 0, 1]]
    outputs.append(getStarGanOutput(payload).tolist())


    content = {
        "outputs": outputs
    }
```

Try it !!!