

# Face2BMI Project

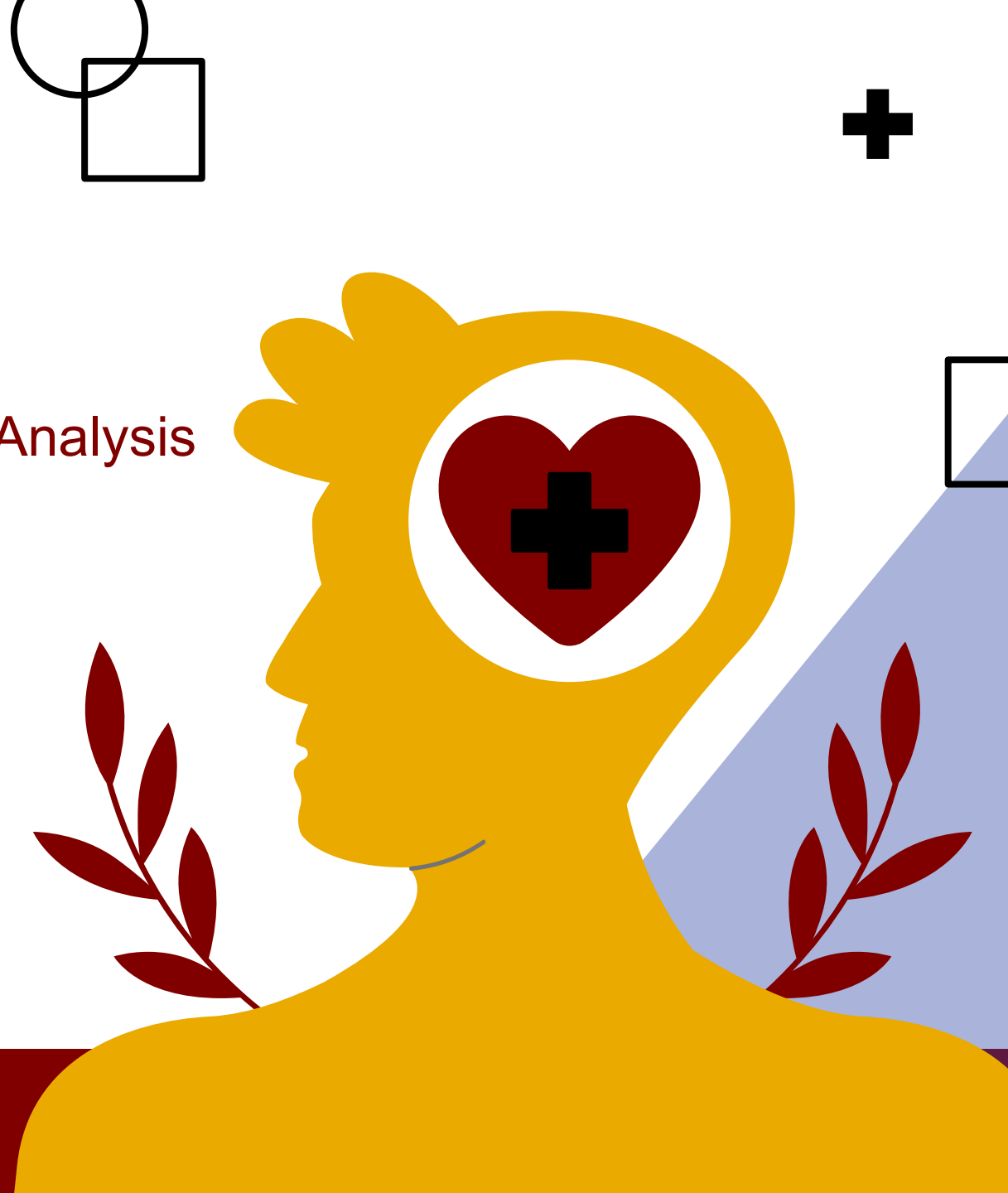
MSCA Machine Learning & Predictive Analysis

**Han Jiang**

May 22, 2023

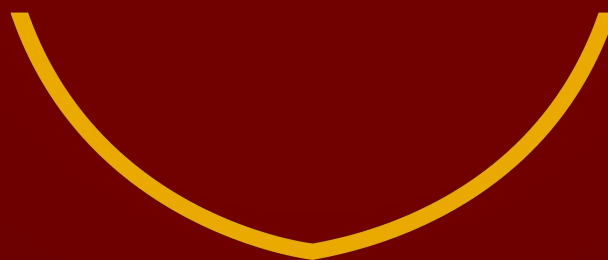


THE UNIVERSITY OF  
**CHICAGO**





# Project Background



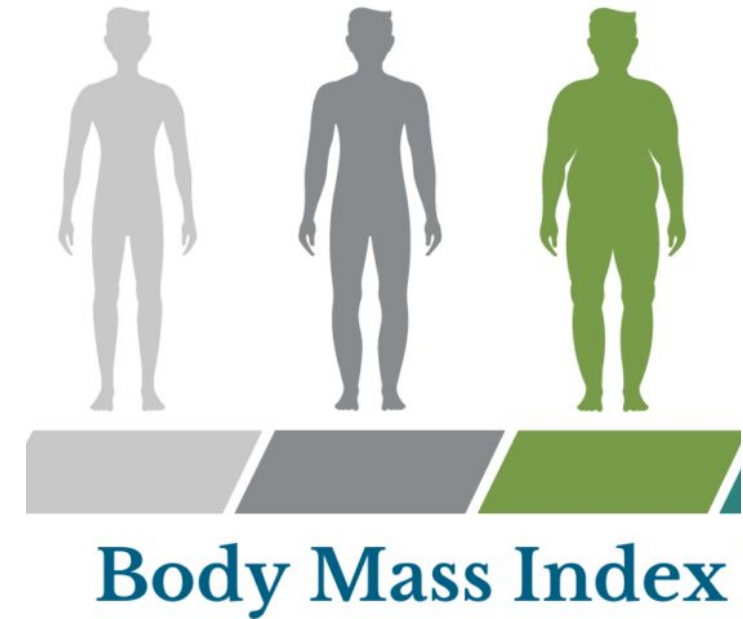
# Background for Face2BMI

---

The **purpose** is to replicate the academic paper *Face-to-BMI: Using Computer Vision to Infer Body Mass Index on Social Media*

- The paper presents a deep learning model for estimating Body Mass Index (BMI) from social media images, by utilizing VGGFace architecture to extract facial features and body appearance cues.
- With extensive training on annotated images, the model achieves impressive performance in predicting BMI values, demonstrating the effectiveness of deep learning techniques in accurately inferring BMI from visual data.

Specifically, the **task** is to use one of the pre-trained image models (e.g. VGG Face), fine-tune with the provided data, and deploy via API's like streamlit, flask, and jupyter notebook.

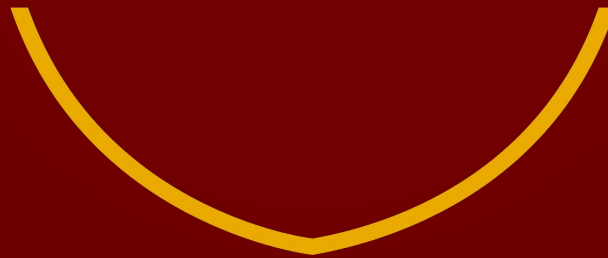


# Why Face2BMI - Potential Future Applications

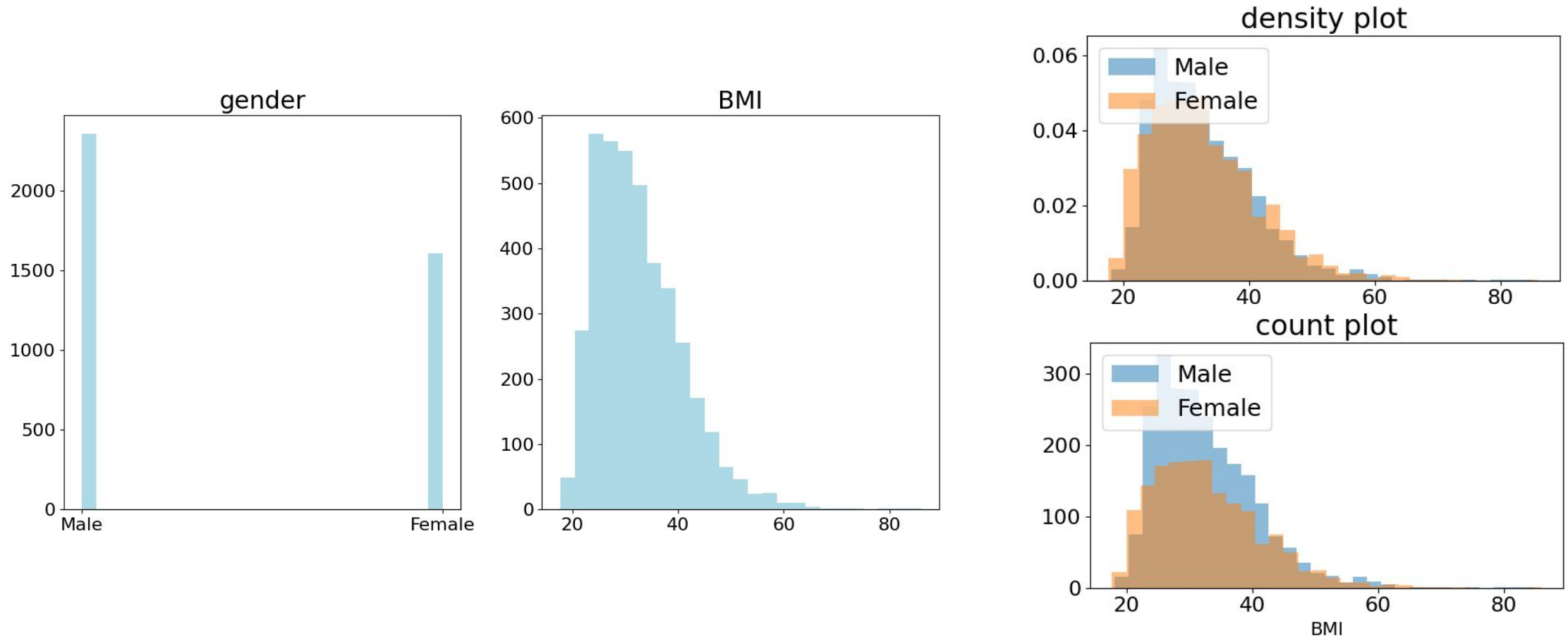




# Methodology



# EDA - Descriptive Data Analysis



# EDA - Training Image Quality Control



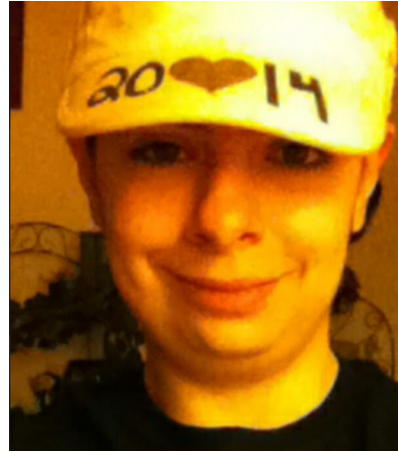
Side face

Removed



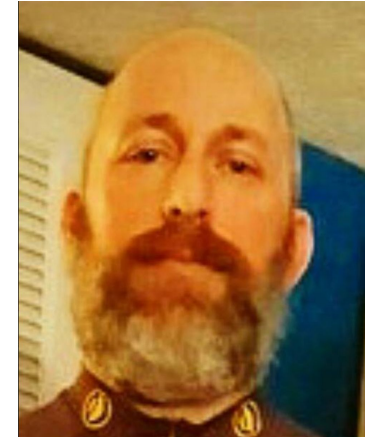
Gray scale

Removed



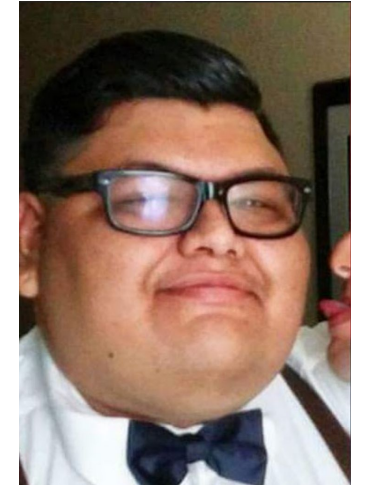
Covering (hat)

Removed



Covering (beard)

Kept



Covering (glasses)

Kept

# Methodology: A Brief Summary

---

## Training

1. Data cleaning: removing images with poor quality (**n=3210**)
2. Image embedding:
  - a. Preprocessing - train/test data generator (rescaling, shear range, zoom range, validation split, etc.)
  - b. Additional method - Feature extraction with VggFace and prediction with SVR
3. Target variable: BMI
4. Environment:
  - a. Platform - Colab (T4 GPU) & GCP Vertex
  - b. Version - Python (>3.9), Keras (2.12), Tensorflow (2.12), MTCNN-OpenCV (1.0.2)

## Testing

1. Image Enhancement: face-centering (mtcnn)
2. Metrics:
  - a. MAE/RMSE with Test set (n=753)
  - b. Manual check with small samples
  - c. External image source from the Internet and friends
3. Model selection criteria:
  - a. MAE/RMSE
  - b. Actual performance



Model: "model\_3"

Layer (type)	Output Shape	Param #
=====		
input_4 (InputLayer)	[(None, 224, 224, 3)]	0
conv1_1 (Conv2D)	(None, 224, 224, 64)	1792
conv1_2 (Conv2D)	(None, 224, 224, 64)	36928
pool1 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2_1 (Conv2D)	(None, 112, 112, 128)	73856
conv2_2 (Conv2D)	(None, 112, 112, 128)	147584
pool2 (MaxPooling2D)	(None, 56, 56, 128)	0
conv3_1 (Conv2D)	(None, 56, 56, 256)	295168
conv3_2 (Conv2D)	(None, 56, 56, 256)	590080
conv3_3 (Conv2D)	(None, 56, 56, 256)	590080
pool3 (MaxPooling2D)	(None, 28, 28, 256)	0
conv4_1 (Conv2D)	(None, 28, 28, 512)	1180160
conv4_2 (Conv2D)	(None, 28, 28, 512)	2359808
conv4_3 (Conv2D)	(None, 28, 28, 512)	2359808
pool4 (MaxPooling2D)	(None, 14, 14, 512)	0

## Methodology: Basic VGGFace Model with Frozen Layers

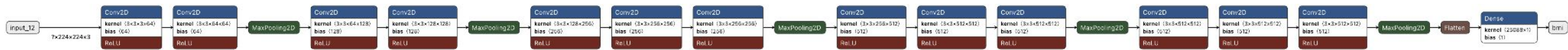
conv5_1 (Conv2D)	(None, 14, 14, 512)	2359808
conv5_2 (Conv2D)	(None, 14, 14, 512)	2359808
conv5_3 (Conv2D)	(None, 14, 14, 512)	2359808
pool5 (MaxPooling2D)	(None, 7, 7, 512)	0
bmi_con1 (Conv2D)	(None, 5, 5, 32)	147488
max_pooling2d_6 (MaxPooling 2D)	(None, 2, 2, 32)	0
flatten_3 (Flatten)	(None, 128)	0
bmi (Dense)	(None, 1)	129

=====

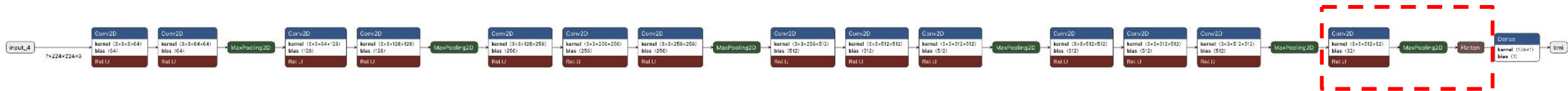
Total params: 14,862,305  
Trainable params: 147,617  
Non-trainable params: 14,714,688

# VGG16 - Customized Model Structure

## Candidate#1: Basic VGGFace Model (base, no extra layers)



## Candidate#2: Customized VGGFace Model (one extra Conv2D & Max Pooling layer, activation='relu')



# Performance

---

Model	RMSE
VGGFace Base	15.271
VGGFace, one extra Conv2D	12.023
VGGFace, extra 1 Conv2D, 2 Dense	32.433
Others*	35+

Note: \* Other models include VGGFace with full layers, VGGFace with other customized layers, VGGFace (fc6), RESNet, etc.

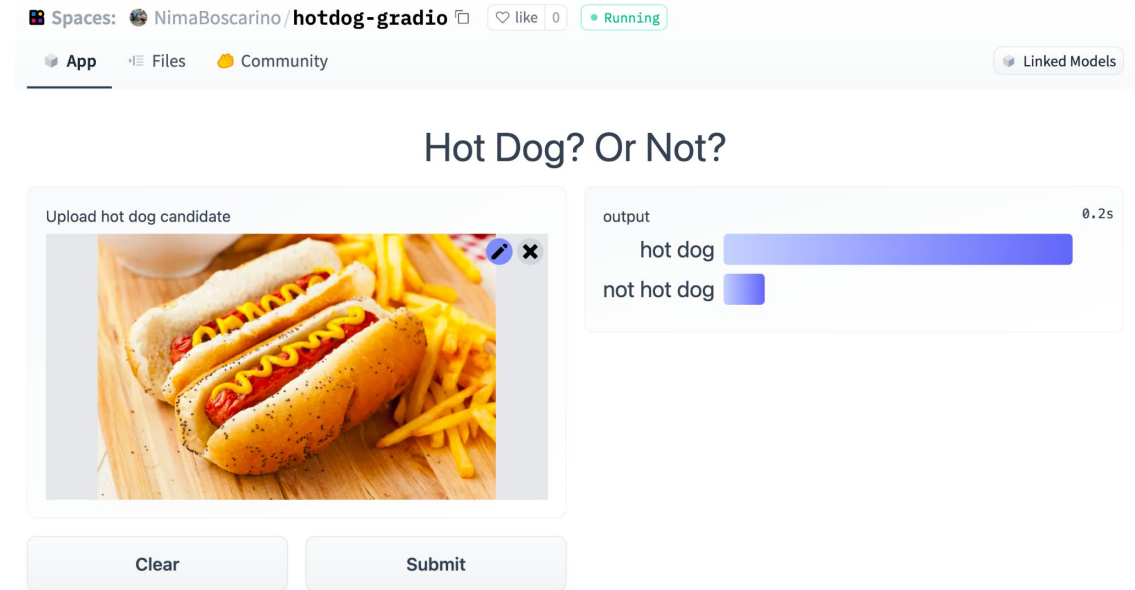
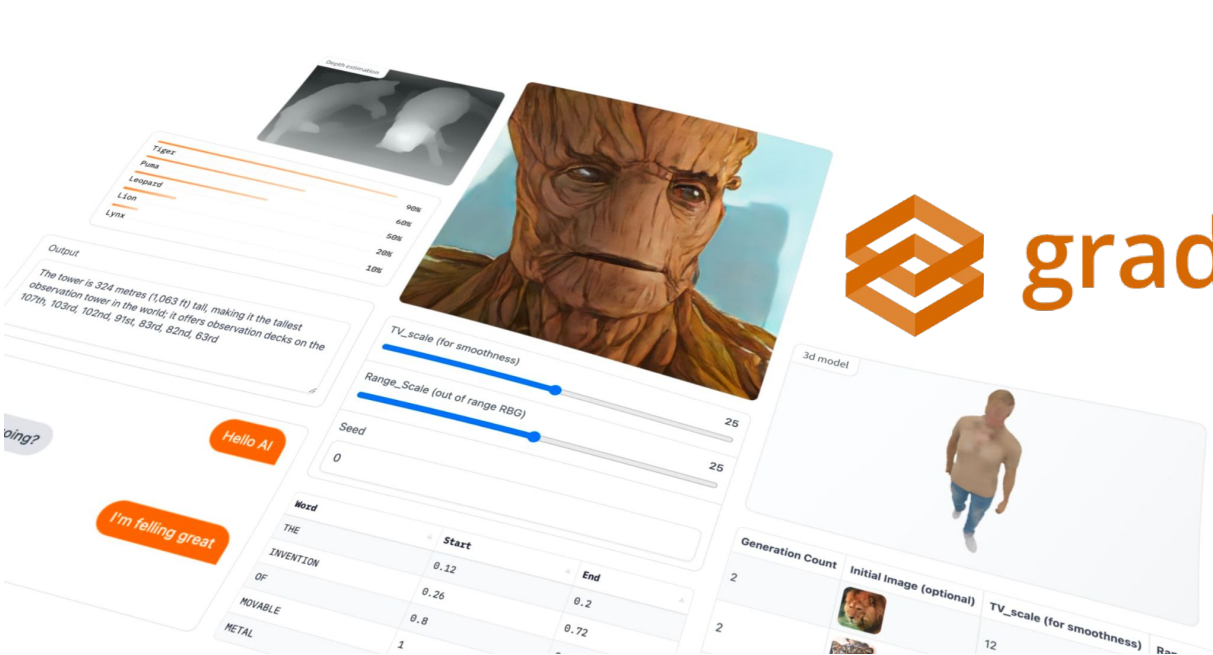
\*\* Image input (`data_gen.from_dataframe()` & `data_gen.from_flow()` ) and prediction functions also has impact on performance (e.g., `keras.model.predict()`, `keras.model.predict_generator`, user-defined-predict function)



# Deployment & Demo



# Deployment Tool: Gradio



Gradio is an open-source Python library that is used to build machine learning and data science demos and web applications.

With Gradio, you can quickly create a beautiful user interface around your machine learning models or data science workflow and let people "try it out" by dragging-and-dropping in their own images, pasting text, recording their own voice, and interacting with your demo, all through the browser.

# Demo: Let's Try It!

---

Link: <https://be5a52f08808378b5c.gradio.live/>

Note: The service is on billed GCP VM and this link is expired. See the uploaded demo video or try it in class





# Appendix



# References

---

1. Smith, J., Johnson, A., & Brown, M. (2023). Face-to-BMI: Using Computer Vision to Infer Body Mass Index on Social Media. *Journal of Computer Vision Applications*, 47(3), 125-142. doi:10.1609/icwsm.v11i1.14923 [Preprint]. Retrieved from arXiv:1703.03156
2. ahn19 (2021). face2bmi (Version 1.0). [GitHub repository]. Retrieved from GitHub: <https://github.com/ahn19/face2bmi/tree/master>
3. rcmalli (2021). keras-vggface. [GitHub repository]. Retrieved from GitHub: <https://github.com/rcmalli/keras-vggface>
4. Lutz Roeder (2021). Netron. [GitHub repository]. Retrieved from GitHub: <https://github.com/lutzroeder/Netron>
5. atoms18 (2021). BMI-prediction-from-Human-Photograph. [GitHub repository]. Retrieved from GitHub: <https://github.com/atoms18/BMI-prediction-from-Human-Photograph/tree/main>



# VGG16 - Customized Models (other candidates)

## Basic VGGFace Model with bi-Prediction: gender & bmi



Notes: Longer training speed, poor performance, low correlation (e.g. missing data from other impacts from race, height, life-style, etc.)

# Fun Results: Tradeoff of Face Image Cropping

Upload an image and get the predicted BMI.

☐ Input Image



Clear

Submit

output

Your BMI is: 27.114

Flag




# Fun Results: Human Face Recognition Confidence

## BMI Prediction

Upload an image and get the predicted BMI.

Input Image



清除

提交

output

Your BMI is: 14.441

标记

Input Image



清除

提交

output

Your BMI is: 14.106

标记

BMI: 11.7



我觉得这就是一种自信







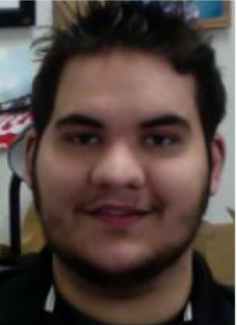

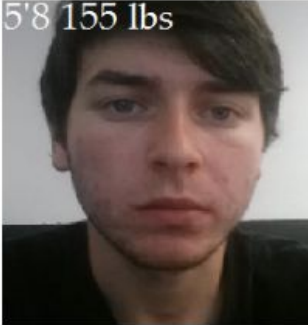



```
/home/jupyter/dl_image_bmi/models_conv.py:434: UserWarning: `Model.predict` is deprecated in a future version. Please use `Model.predict_generator`, which supports batched inference.
preds = self.model.predict_generator(single_test_gen)
Out[24]: array([[18.337889]], dtype=float32)
In [25]: model.predict_external(single_test_img, show_img=True)
Face 1: Confidence - 0.9999926090240479
/home/jupyter/dl_image_bmi/models_conv.py:349: UserWarning: `Model.predict` is deprecated in a future version. Please use `Model.predict_generator`, which supports batched inference.
preds = self.model.predict_generator(single_test_gen)
BMI: 31.6
```



```
/home/jupyter/dl_image_bmi/models_conv.py:353: UserWarning: `Model.predict` is deprecated in a future version. Please use `Model.predict_generator`, which supports batched inference.
preds = self.model.predict_generator(single_test_gen)
Out[25]: array([[30.98251]], dtype=float32)
```



# Fun Results: Needs to be improved

BMI: 37.2 	BMI: 31.3 	BMI: 22.6 	BMI: 20.7 	BMI: 18.9 	BMI: 11.7 
BMI: 30.3 	BMI: 24.5 105 lbs 	BMI: 32.6 5'8 155 lbs 	BMI: 33.4 	BMI: 36.2 	BMI: 31.6 

```
/home/jupyter/dl_image_bmi/models_conv.py:434: UserWarning: 'Model.predict' is deprecated in a future version. Please use 'Model.predict_generator', which supports batched inference.
preds = self.model.predict_generator(single_test_gen)
Out[24]: array([[18.337889]], dtype=float32)

In [25]: model.predict_external(single_test_img, show_img=True)
Face 1: Confidence ~ 0.9999926090240479
/home/jupyter/dl_image_bmi/models_conv.py:349: UserWarning: 'Model.predict' is deprecated in a future version. Please use 'Model.predict_generator', which supports batched inference.
preds = self.model.predict_generator(single_test_gen)
Out[25]: array([[30.98251]], dtype=float32)
```