



# Deep Learning Toolkit *(Gradio & HuggingFace)*

Rowel Atienza, PhD

University of the Philippines

[github.com/roatienza](https://github.com/roatienza)

2023

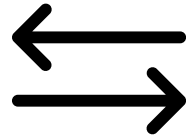
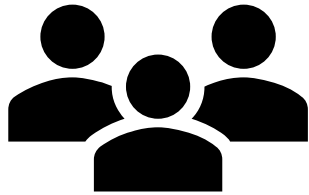


# Gradio & Hugging Face

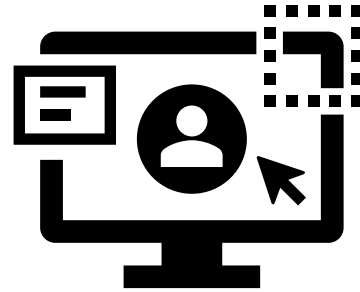
<https://gradio.app/>

<https://huggingface.co/>

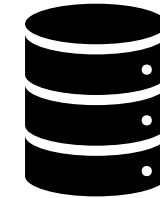
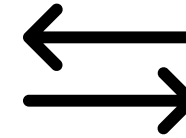
Users



Front-End



Back-End



# Why Gradio?

Easy to use APIs to **build** model app UI

Easy to **demonstrate** model functionalities

**Deploy** model app through Hugging Face Spaces

# Why Hugging Face?

A collection of pre-trained models in **Model Hub**

APIs to access and utilize pre-trained models using **Pipeline**

Hosting of deep learning apps using **Spaces**

# Install

```
pip3 install transformers gradio --upgrade
```

# Introducing Gradio

```
import gradio as gr

def greet(name):
    return "Hello " + name + "!!"

gr.Interface(fn=greet,
             inputs="text",
             outputs="text").launch()
```

NAME

Clear

Submit

Flag

view the api  • built with gradio 



# Using Gradio as ResNet18 Front End

# 1k Object Recognition

Demonstrates a pre-trained model from torchvision for image classification.

IMG

Drop Image Here  
- or -  
Click to Upload

Clear

Submit

## Examples



# Building Web App for ResNet18

```
gr.Interface(fn=classify,  
            inputs="image",  
            outputs=gr.Label(num_top_classes=5),  
            title="1k Object Recognition",  
            examples=['assets/wonder_cat.jpg',  
                    'assets/aki_dog.jpg', 'assets/birdie1.jpg'],  
            description="Demonstrates Resnet18.",  
            allow_flagging="never").launch(inbrowser=False)
```

```
def classify(img):  
    # By default, gradio image is numpy  
    img = torch.from_numpy(img)  
    # Numpy image is channel last. PyTorch is channel 1st.  
    img = img.permute(2, 0, 1)  
  
    # The transforms before prediction  
    img = torchvision.transforms.Resize(256, antialias=True)(img)  
    img = torchvision.transforms.CenterCrop(224)(img).float()/255.  
    img = normalize(img)  
  
    # We insert batch size of 1  
    img = img.unsqueeze(0)
```

```
# The actual prediction
with torch.no_grad():
    pred = resnet(img)

# Convert the prediction to probabilities
pred = torch.nn.functional.softmax(pred, dim=1)
# Remove the batch dim.
pred = pred.squeeze()

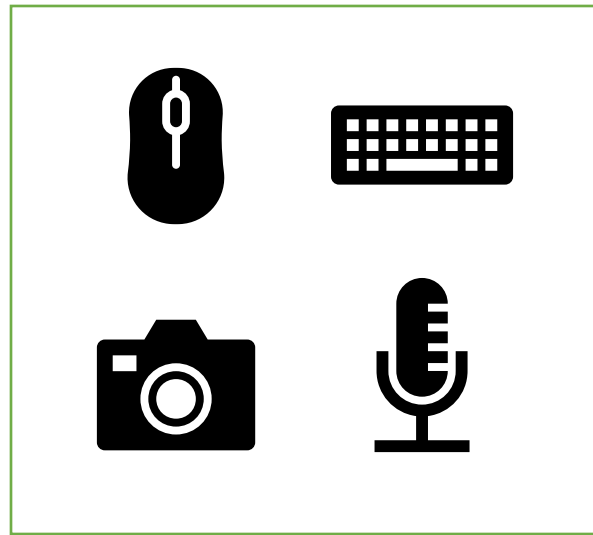
# torch to numpy space
pred = pred.cpu().numpy()

return {labels[i]: float(pred[i]) for i in range(1000)}
```

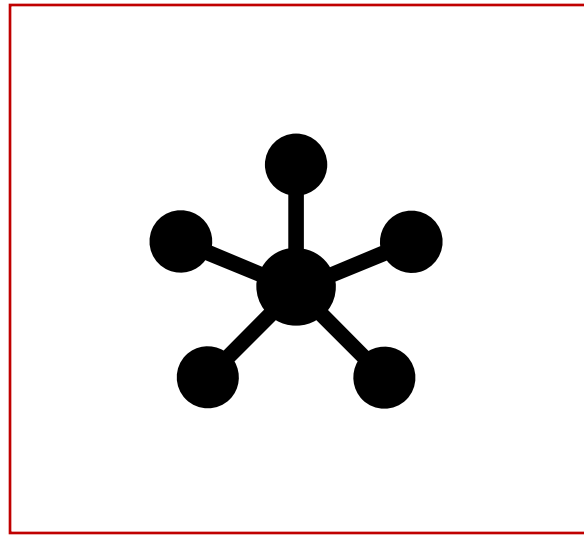
# Pipeline

[https://huggingface.co/docs/transformers/main\\_classes/pipelines](https://huggingface.co/docs/transformers/main_classes/pipelines)

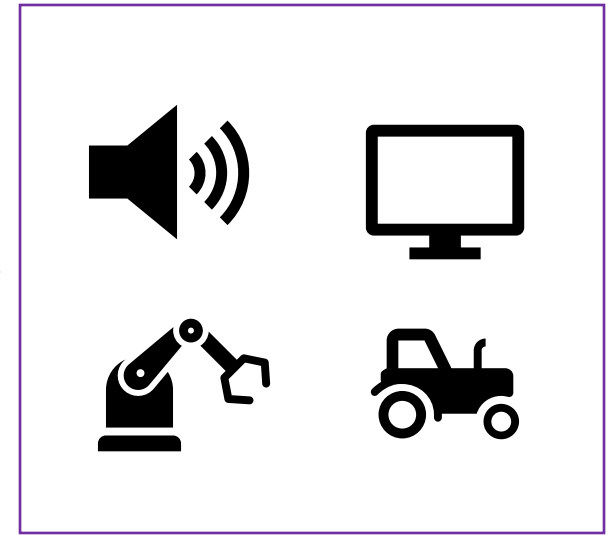
# Pipeline



Input Pre-Processing



Model Inference



Output Post-Processing

# Pipeline + Gradio

```
import gradio as gr
from transformers import pipeline

pipe = pipeline(task="image-classification",
                model="microsoft/beit-base-patch16-224-pt22k-ft22k")

gr.Interface.from_pipeline(pipe,
    title="22k Image Classification",
    description="Object Recognition using Microsoft BEIT",
    examples = ['wonder_cat.jpg', 'aki_dog.jpg'],
    allow_flagging="never").launch(inbrowser=False)
```



# 22k Image Classification

Object Recognition using Microsoft BEiT

INPUT IMAGE

Drop Image Here  
- or -  
Click to Upload

Clear

Submit

Examples



Gradio & HuggingFace

Deep Learning, University of the Philippines

17

# Model Hub: <https://huggingface.co/models>

## Natural Language Processing



Fill-Mask



Question Answering



Summarization



Table Question Answering



Text Classification



Text Generation



Text2Text Generation



Token Classification



Translation



Zero-Shot Classification



Sentence Similarity



Conversational



Feature Extraction

## Audio



Text-to-Speech



Automatic Speech Recognition



Audio-to-Audio



Audio Classification



Voice Activity Detection

## Computer Vision



Image Classification



Object Detection



Image Segmentation



Text-to-Image



Image-to-Text

# Spaces

<https://huggingface.co/spaces>

<https://huggingface.co/spaces/rowel/22k-image-classification>

# 22k Image Classification

Object Recognition using Microsoft BEiT

Input Image

Drop Image Here  
- or -  
Click to Upload

Clear

Submit

Examples





Models

Datasets

Spaces

Docs



Solutions



Pricing



⌵







## Tasks


 Fill-Mask  Question Answering

 Summarization  Table Question Answering

 Text Classification  Text Generation

 Text2Text Generation  Token Classification




 Translation  Zero-Shot Classification





 Sentence Similarity + 13


## Libraries

 PyTorch  TensorFlow  JAX + 24

## Datasets

 wikipedia  common\_voice  squad

 bookcorpus  c4  glue  conll2003

 dcep europarl jrc-acquis + 810






## Languages

en es fr de zh sv fi ja + 169

## Licenses

Models 31,626

↑↓ Sort: Most Downloads

**distilgpt2** Text Generation • Updated May 21, 2021 • ↓ 24.5M • ♥ 23**bert-base-uncased** Fill-Mask • Updated May 19, 2021 • ↓ 14.4M • ♥ 111 **cross-encoder/ms-marco-MiniLM-L-12-v2** Text Classification • Updated Aug 5, 2021 • ↓ 13.2M • ♥ 3**gpt2** Text Generation • Updated May 20, 2021 • ↓ 12.2M • ♥ 60 **Helsinki-NLP/opus-mt-zh-en** Translation • Updated Feb 27, 2021 • ↓ 7.09M • ♥ 16**xlm-roberta-large-finetuned-conll103-english** Token Classification • Updated Oct 12, 2020 • ↓ 5.67M • ♥ 11**distilbert-base-uncased** Fill-Mask • Updated Aug 30, 2021 • ↓ 5.56M • ♥ 44

Spaces:  rowel/demospace

private

• No application file

App

Files and versions

Settings

### Get started with your Space!

Your new space has been created, follow these steps to get started (or read our full [documentation](#))

Start by cloning this repo by using:

```
$ git clone https://huggingface.co/spaces/rowel/demospace
```

Create your Gradio app.py file:

```
import gradio as gr

def greet(name):
    return "Hello " + name + "!!!"

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

Then commit and push:





Search models, datasets, users...

Spaces: rowel/demospace private See logs Running

App Files and versions Settings

main demospace / requirements.txt

rowel Update requirements.txt 12aea94 less than a minute ago

raw history blame edit delete 18 Bytes

```
1 torch
2 transformers
```

### Repo secrets

No secrets

rowel@eee.upd.edu.ph

.....

Add

### Restart this Space

Click this button to trigger a reboot of your Space

Restart this Space

### Change space visibility

This space is currently **private**. Only you (personal space) or members of your organization (organization space) can see and commit to this space.

Make this space public



# Automatic Speech Recognition (ASR)

<https://huggingface.co/spaces/rowel/asr>

# ASR

```
import gradio as gr
from transformers import pipeline

pipe = pipeline(task="automatic-speech-recognition",
                model="openai/whisper-tiny")

gr.Interface.from_pipeline(pipe,
                           title="Automatic Speech Recognition (ASR)",
                           description="Using pipeline with OpenAI Whisper",
                           examples=['assets/ljspeech.wav'],
                           ).launch(inbrowser=True)
```

# Automatic Speech Recognition (ASR)

Using pipeline with OpenAI Whisper

🎵 Input

• Record from microphone

Clear

Submit

Output

Flag

☰ Examples

ljspeech.wav

# Text to Speech (TTS)

<https://huggingface.co/spaces/rowel/tts>

# TTS

```
import gradio as gr
```

```
gr.Interface.load(  
    "huggingface/facebook/fastspeech2-en-ljspeech",  
    description="TTS using FastSpeech2",  
    title="Text to Speech (TTS)",  
    examples=[["The quick brown fox jumps over the lazy dog."]]  
) .launch()
```

# Text to Speech (TTS)

TTS using FastSpeech2

Input



Clear

Submit

## Examples

The quick brown fox jumps over the lazy dog.

# End