

# From Data to Web Application: Anime Character Image Recognition with Transfer Learning

PyCon HK 2018

Iskandar Setiadi  
Software Engineer at HDE, Inc.





## Github

<https://github.com/freedomofkeima>

## Blog

<https://freedomofkeima.com/blog/>

Speaker in:

- 2017: PyCon Japan, PyCon Indonesia
- 2018: PyCon Malaysia, PyCon Japan, and today, **PyCon Hong Kong!**



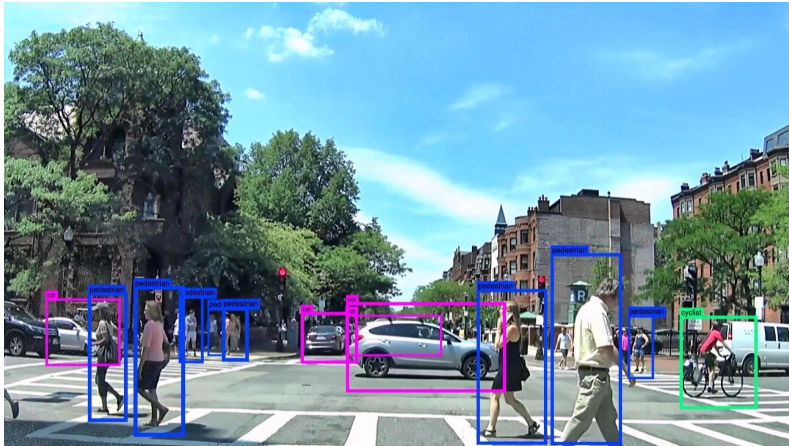
## Iskandar Setiadi

From Jakarta, Indonesia  
Software Engineer in Japan

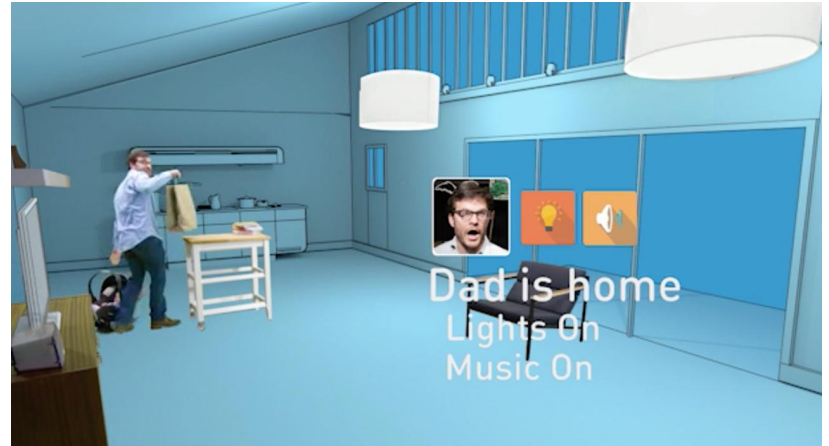
HDE, Inc. (<https://hde.co.jp/en/>)



# Image Recognition in Real Life

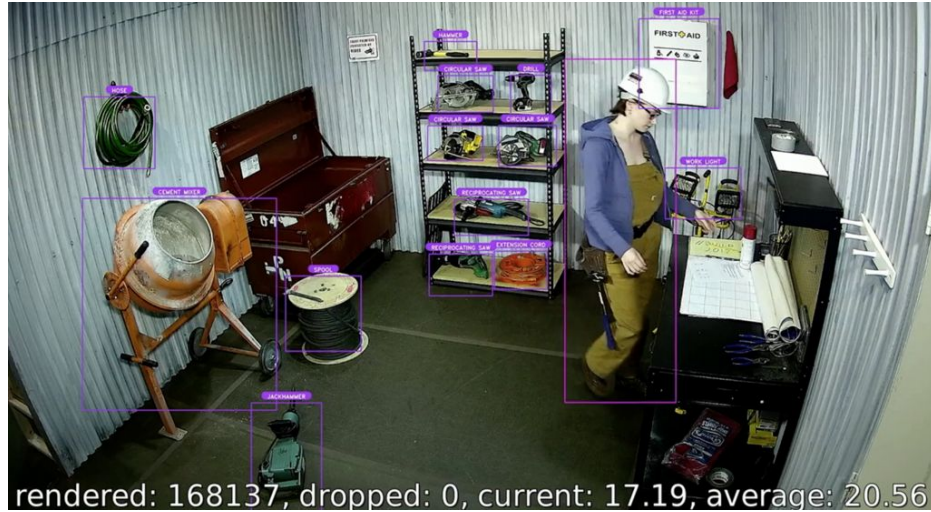


Self-driving car



Smart home

# Image Recognition in Daily Life



Smart workplace



Face ID

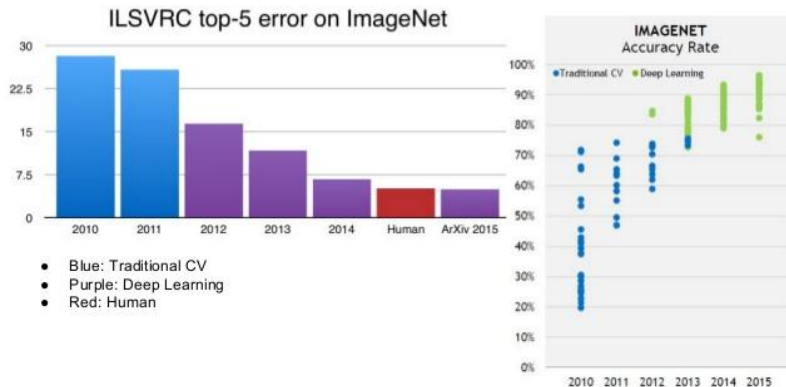


# ILSVRC

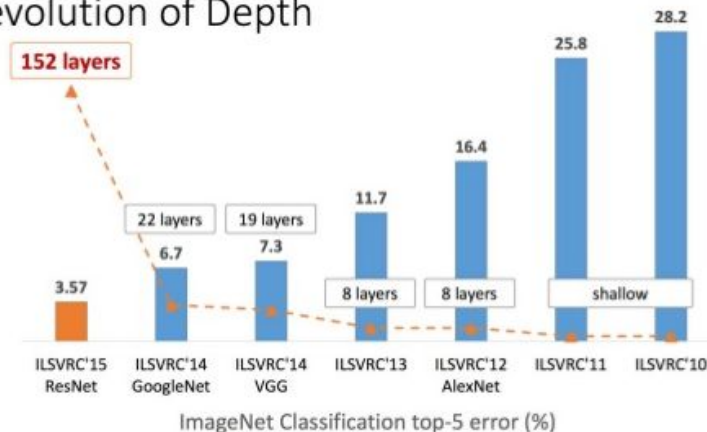
Largest Computer Vision Competition

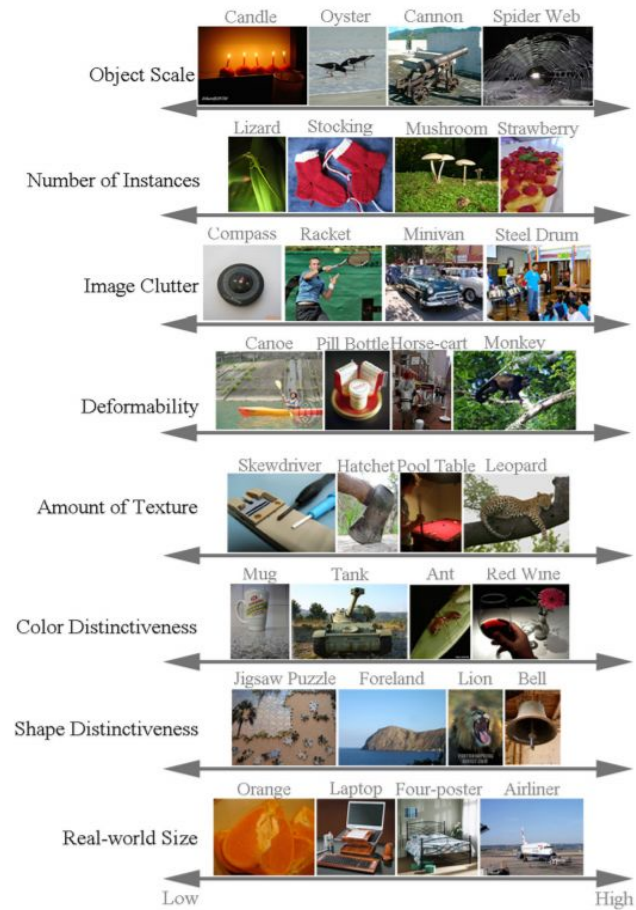
Starting from 2015, deep learning has better top-5 error score compared to human (1000 categories)!

## Case #2: ILSVRC 2010-2015



## Revolution of Depth





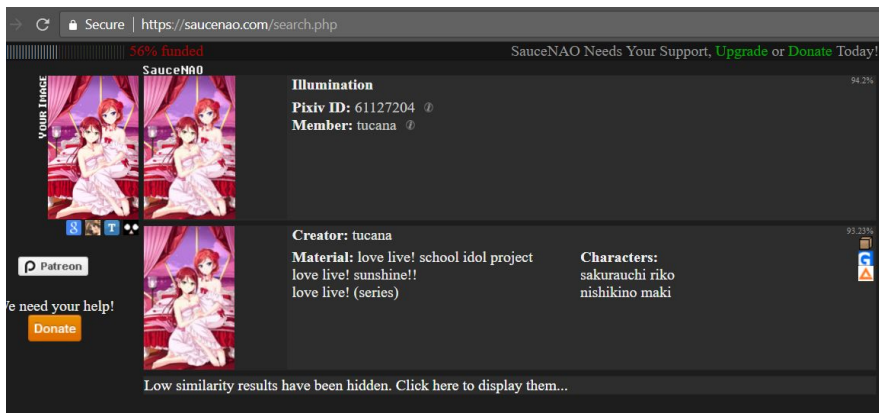


# Background



# Problem

- Cropped images
- Edited images
- Unindexed images





---

# What is Machine Learning?

INTRODUCTION?



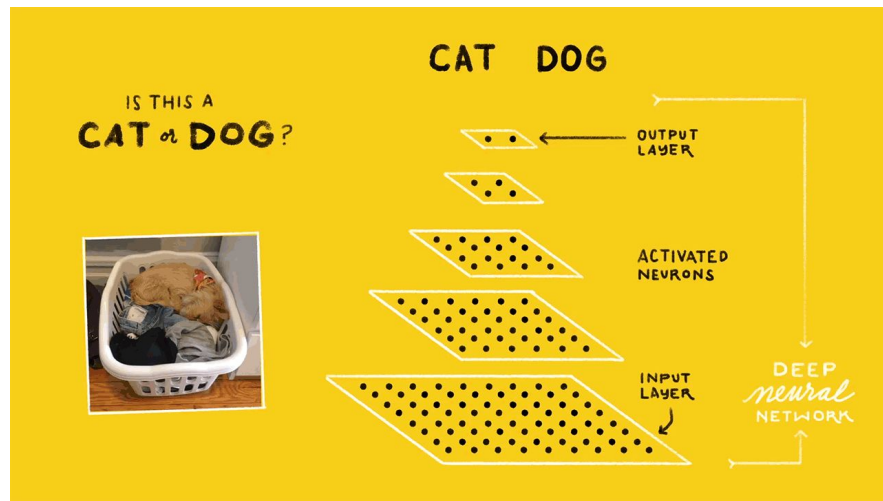
# Deep Learning

## Traditional ML:

Increasing number of training iterations will eventually get stagnated at certain point.

## Alternative proposal:

More layers! But it is slow :(



# Deep Learning: Convolution

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

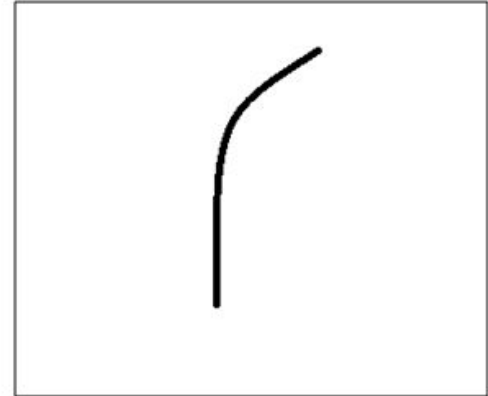
Image

4		

Convolved  
Feature

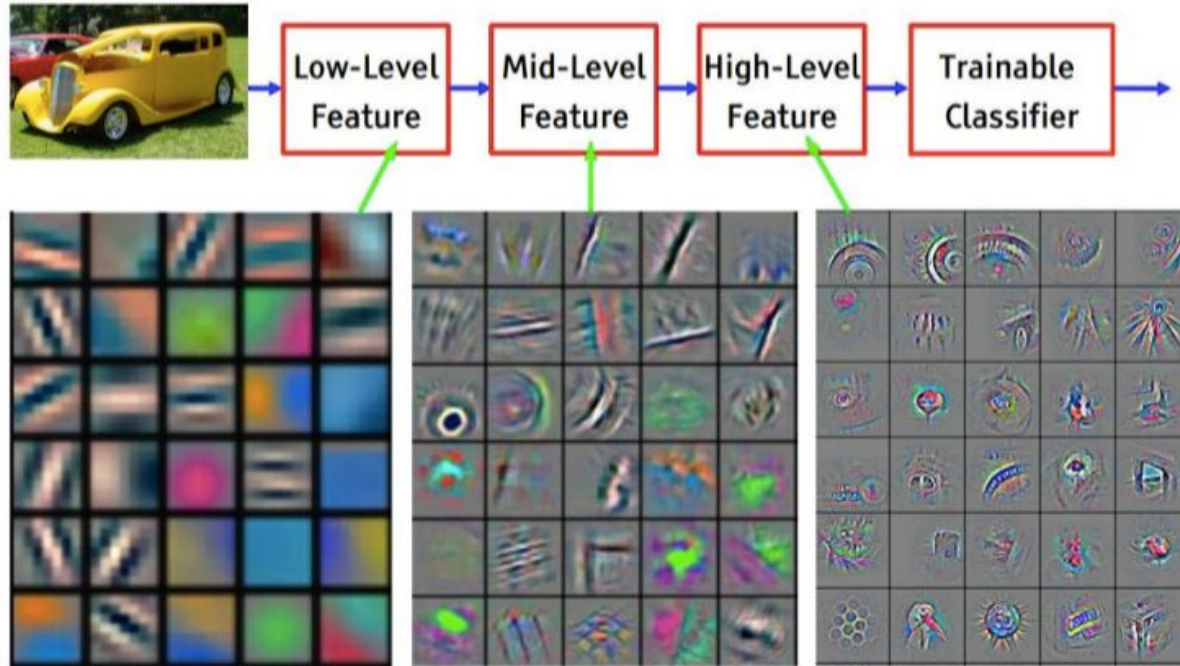
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

# Deep Learning



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

---

# Step 1. Preparation



# TensorFlow Installation

```
$ pip3 install --upgrade tensorflow
```

or

```
$ pip3 install --upgrade tensorflow-gpu
```

URL: <https://www.tensorflow.org/install/>



# Why should we use Transfer Learning?

From certain Top-5 characters indexing website:

- 35000 registered characters
- Top 1000 characters: 70+ images
- Top 2000 characters: 40+ images

Dataset size is small! Google Inception-v3 uses > 1000 images per category.

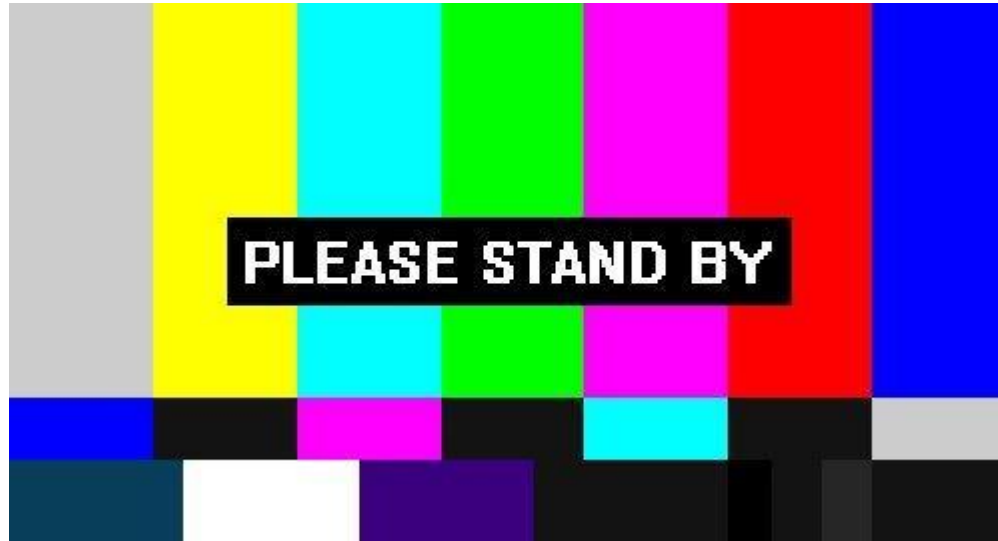
With transfer learning, we don't need to retrain low-level features extraction model.

Result: **100 categories**, ~ 60 images per category.

URL: [https://www.tensorflow.org/tutorials/image\\_retraining](https://www.tensorflow.org/tutorials/image_retraining)



## Scrap & clean up the data



Note: Please contact me if you want to know the details



# Google Dataset Search (Beta)

Secure | <https://toolbox.google.com/datasetsearch>



[About](#)

## Google Dataset Search Beta

Search for Datasets



Try [boston education data](#) or [weather site:noaa.gov](#)

**kaggle** Tagged Anime Illustrations

www.kaggle.com

Updated Jul 30, 2018

**kaggle** Anime Recommendations Database

www.kaggle.com

Updated Dec 21, 2016

**kaggle** The Simpsons Characters Data

www.kaggle.com

Updated Apr 13, 2018

 ArtiosCAD Workspace File

vistablog.com.ua

## Danbooru2017: A large-scale crowdsourced and tagged anime illustration dataset

The first set of data comes from the imageboard Danbooru. The entire corpus of Danbooru images was scraped from the site with permission and was collected into a dataset. The zip files included here have the full metadata for these images as well as a subset of 300,000 of the images in normalized 512px x 512px form. Full information about this dataset is available here:

<https://www.gwern.net/Danbooru2017>

From the article:

*Deep learning for computer vision relies on large annotated datasets. Classification/categorization has benefited from the creation of ImageNet, which classifies 1m photos into 1000 categories. But classification/categorization is a coarse description of an image which limits application of classifiers, and there is no comparably large dataset of images with many tags or labels which would allow learning and detecting much richer information about images. Such a dataset would ideally be >1m images with at least 10 descriptive tags each which can be publicly distributed to all interested researchers, hobbyists, and organizations. There are currently no such public datasets, as ImageNet, Birds, Flowers, and MS COCO fall short either on image or tag count or restricted distribution. I suggest that the image-boorus be used. The image boorus are longstanding web databases which host large numbers of images which can be tagged or labeled with an arbitrary number of textual descriptions; they were developed for and are most popular among fans of anime, who provide detailed annotations.*

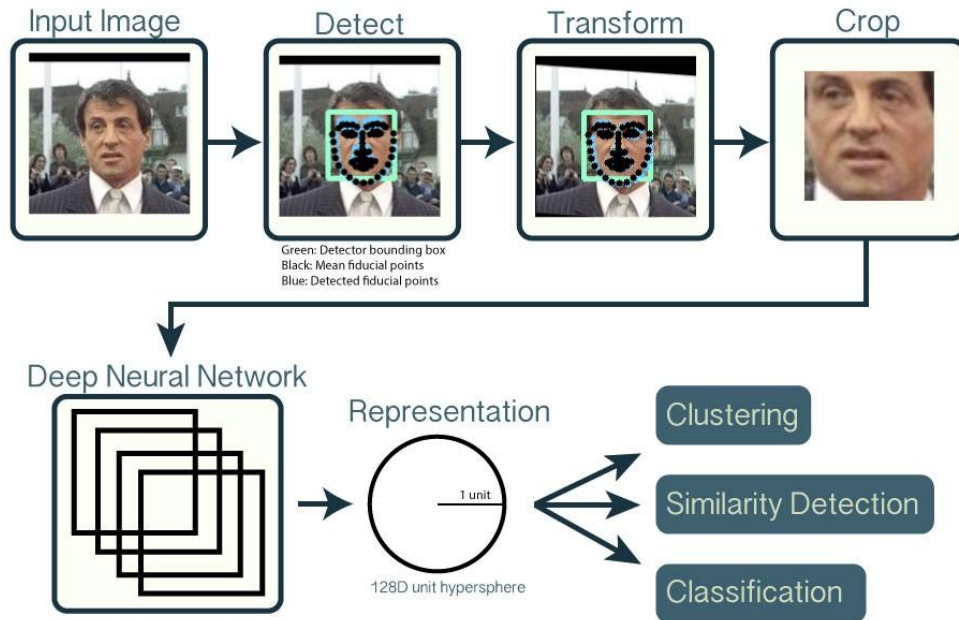
*The best known booru, with a focus on quality, is Danbooru. We create & provide a torrent which contains ~1.9tb of 2.94m images with 77.5m tag instances (of 333k defined tags, ~26.3/image) covering Danbooru from 24 May 2005 through 31 December 2017 (final ID: #2,973,532), providing the image files & a JSON export of the metadata. We also provide a smaller torrent of SFW images downscaled to 512x512px JPG (241GB; 2,232,462 images) for convenience.*

*Our hope is that a Danbooru2017 dataset can be used for rich large-scale classification/tagging & learned embeddings, test out the transferability of existing computer vision techniques (primarily developed using photographs) to illustration/anime-style images, provide an archival backup for the Danbooru community, feed back metadata improvements & corrections, and serve as a testbed for advanced techniques such as conditional image generation or style transfer.*

---

## Step 2. Analyze the Problem

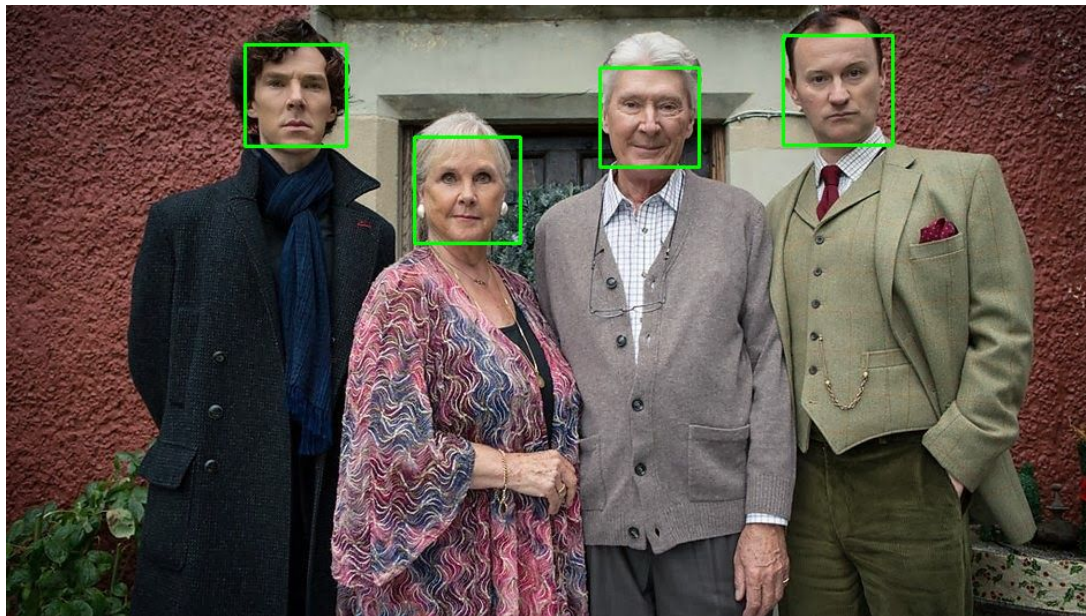
# Face Detection: Introduction





---

## Face Detection (Human Face)



---

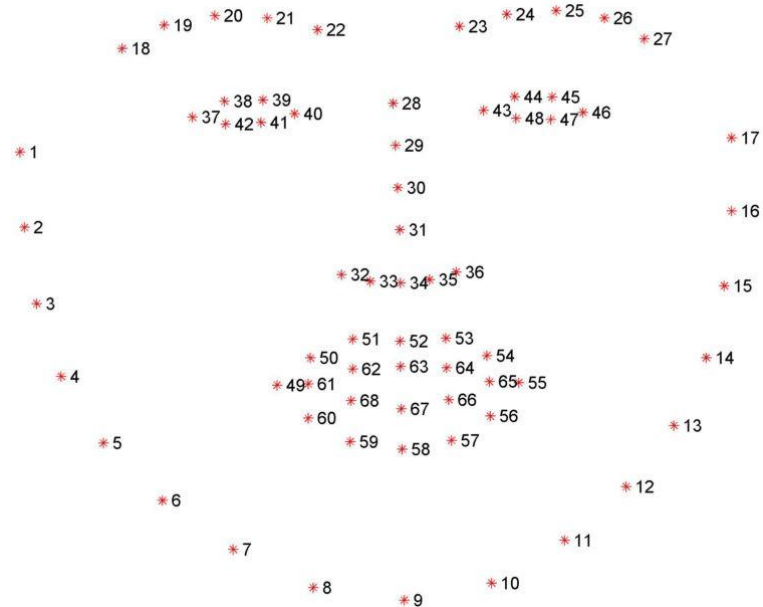
## Face Detection (Same Model - Anime Face)



# 2D is ~~Better~~ not equal to 3D face!

Facial features are different!

e.g.: 2D has no nose







---

## Face Detection (Anime Face)



---

## Face Detection (Same Model - Human Face)





# Face Recognition

Face Detection → “Accomplished”

Full-layered Deep Learning → Requires a huge dataset, weeks to train

**Google Inception-v3:** 1.2 million training data, 1000 classes, 1 week to train

---

## Step 3. Train & Validate the Model

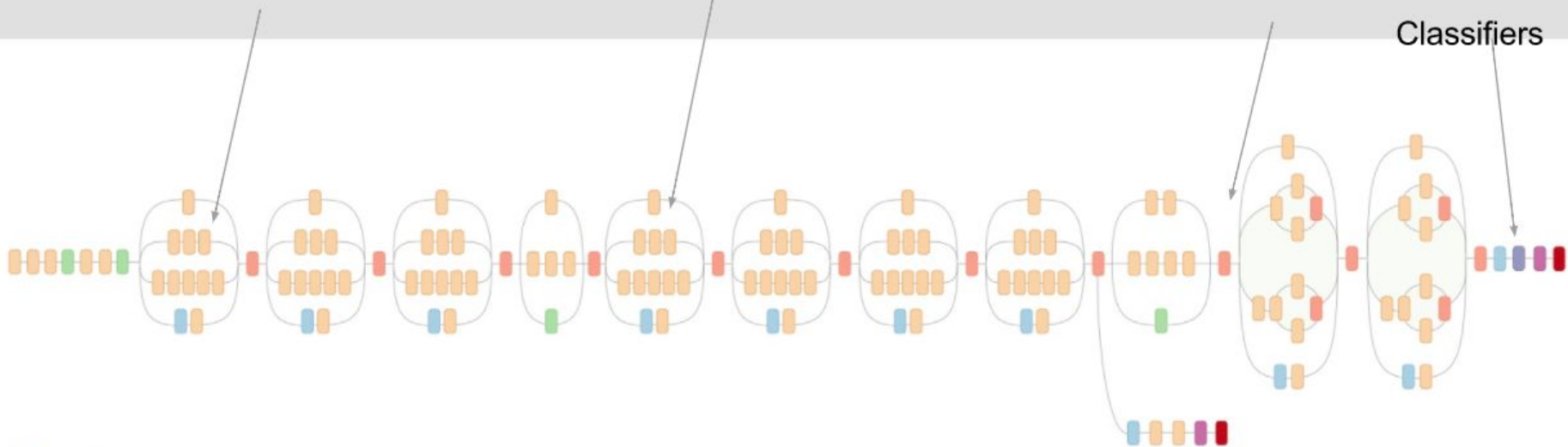
# What does the layers learn?

Edges

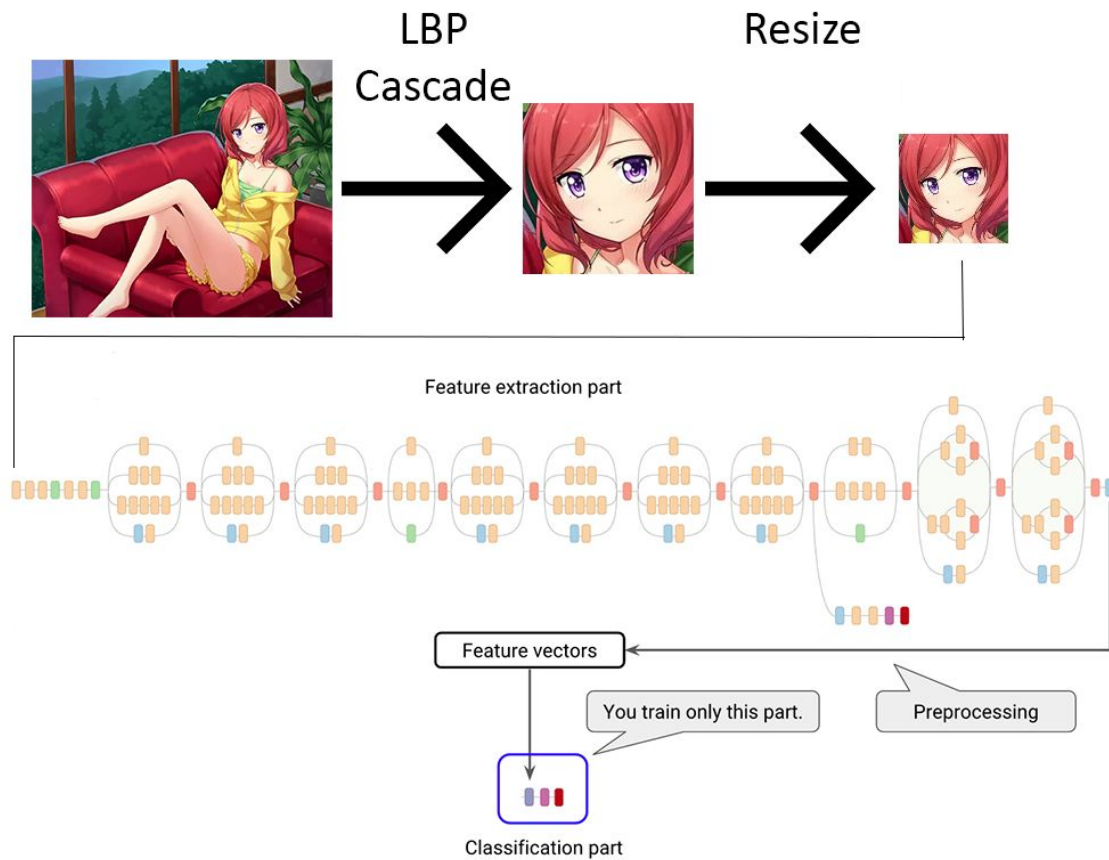
Shapes

High level features

Classifiers



- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax





# Transfer Learning: Retrained Layers

Dropout: Dropping out units to prevent overfitting

Fully Connected: Extracting global features, every node in the layer is connected to the preceding layer

Softmax: Squashing final layer to make a prediction, which sums up to 1. For example, if we have 2 classes and class A has the value of 0.95, then class B will have the value of 0.05.





# Transfer Learning: Retrain Final Layer

Build the retrainer:

```
$ bazel build tensorflow/examples/image_retraining:retrain
```

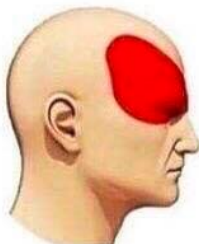
Execute the retrainer:

```
$ bazel-bin/tensorflow/examples/image_retraining/retrain --image_dir ~/images
```

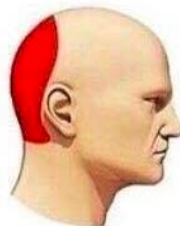
Hyperparameters: learning rate, number of iterations, distortions factor, ...

Types of headache

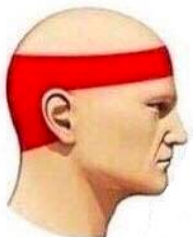
**MIGRAINE**



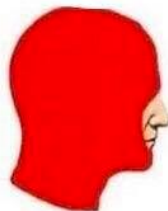
**HYPERTENSION**



**STRESS**



**Working with classifiers**



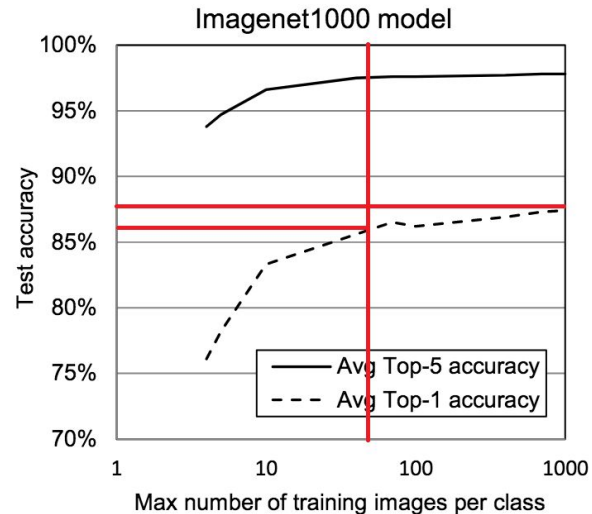
# Original ImageNet: Accuracy vs Dataset size

From 10 to 100 training images per class, you can improve your Top-1 accuracy by 5%.

From 100 to 1000 training images per class, you can only improve it by less than 2%.

Low number of images → bad accuracy

Huge number of images → stagnated accuracy



---

## Step 4. Serve as a Web App



# MoeFlow: Specification

→ Build with Sanic (Flask-like Python 3.5+ web server); **Caveat:** Good for prototyping, have some memory handling issues

→ While training model requires huge GPU resources (g2.2xlarge), using retrained model can be hosted in server with small resources (t2.small)

## What it does:

- Run face detection with OpenCV
- Resize image to a fixed proportion
- Run classification with TensorFlow

# MoeFlow: Web App

---

```
@app.listener('before_server_start')
async def initialize(app, loop):
    moeflow_path = os.environ.get('MOEFLOW_MODEL_PATH')
    label_path = os.path.join(os.sep, moeflow_path, "output_labels_2.txt")
    model_path = os.path.join(os.sep, moeflow_path, "output_graph_2.pb")
    app.label_lines = [
        line.strip() for line in tf.gfile.GFile(label_path)
    ]
    graph = tf.Graph()
    graph_def = tf.GraphDef()
    with tf.gfile.FastGFile(model_path, 'rb') as f:
        graph_def.ParseFromString(f.read())
    with graph.as_default():
        tf.import_graph_def(graph_def, name='')
    app.graph = graph
    logging.info("MoeFlow model is now initialized!")
```

URL: <https://github.com/freedomofkeima/MoeFlow>

```
def classify_resized_face(file_name, label_lines, graph):
    results = []
    logging.info('Processing classification')
    with tf.Session(graph=graph) as sess:
        # Feed the image data as input to the graph and get first prediction
        softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')
        input_operation = sess.graph.get_operation_by_name("Mul")
        t = read_tensor_from_image_file(file_name)
        predictions = sess.run(
            softmax_tensor,
            {input_operation.outputs[0]: t}
        )
        # Sort to show labels of first prediction in order of confidence
        top_k = predictions[0].argsort()[-3:][::-1]

        for node_id in top_k:
            human_string = label_lines[node_id]
            score = predictions[0][node_id]
            results.append((human_string, score))
    return results
```



```
2017-12-06 01:51:53 - (network)[INFO][127.0.0.1:49154]: GET http://127.0.0.1:8888/static/images/83a97094d7634333b5c9d7e8e0901da4.jpg 200 119582
INFO:network:
2017-12-06 01:51:53 - (network)[INFO][127.0.0.1:49156]: GET http://127.0.0.1:8888/static/images/e96410fcf8114fbb88a4f33ede40205e.jpg 200 5902
INFO:network:
2017-12-06 05:48:33 - (network)[INFO][127.0.0.1:59920]: GET http://127.0.0.1:8888/ 200 1180
INFO:network:
INFO:root:Height: 1416, Width: 1280
INFO:root:Input file is created at /tmp/tmpgm8odxy2.jpg
INFO:root:Processing classification
INFO:root:[('tedeza rize', 0.8941825), ('yui', 0.015284972), ('takanashi rikka', 0.014930859)]
2017-12-06 05:49:38 - (network)[INFO][127.0.0.1:59970]: POST http://127.0.0.1:8888/ 200 1565
INFO:network:
2017-12-06 05:49:39 - (network)[INFO][127.0.0.1:59972]: GET http://127.0.0.1:8888/static/images/fda2e4333b194f30a845ed8d320bf449.jpg 200 294605
INFO:network:
2017-12-06 05:49:39 - (network)[INFO][127.0.0.1:59974]: GET http://127.0.0.1:8888/static/images/d9a530d76c17404fbdbde7a2e5bf9c55.jpg 200 5807
INFO:network:
```





Demo

Input:



Output:

Character



Prediction (Top-3)

- kafuu chino
- miki sayaka
- kashima

## Input:



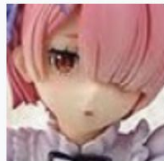
## Output:

### Character

### Prediction (Top-3)



- rem
- tobiichi origami
- tachibana kanade





- ram
- rem
- tomoe mami

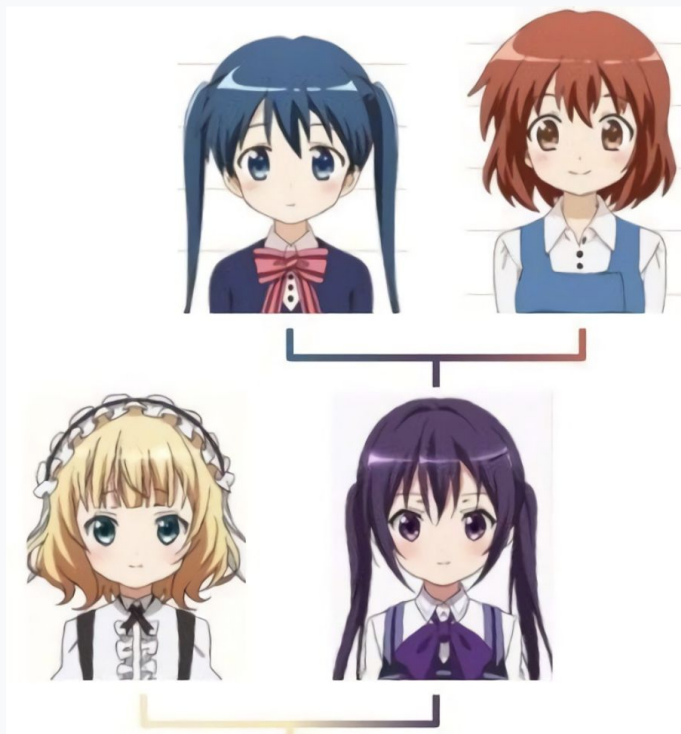
Input:



Output:

Character	Prediction (Top-3)
	<ul style="list-style-type: none"><li>• kafuu chino</li><li>• miki sayaka</li><li>• rem</li></ul>
	<ul style="list-style-type: none"><li>• natsu megumi</li><li>• minami kotori</li><li>• suzukaze aoba</li></ul>

Input:



Output:

Character	Prediction (Top-3)
	<ul style="list-style-type: none"><li>• kirima sharo</li><li>• kirisaki chitoge</li><li>• inokuma youko</li></ul>
	<ul style="list-style-type: none"><li>• tedeza rize</li><li>• aragaki ayase</li><li>• yatogami tohka</li></ul>
	<ul style="list-style-type: none"><li>• komichi aya</li><li>• miki sayaka</li><li>• tedeza rize</li></ul>
	<ul style="list-style-type: none"><li>• inokuma youko</li><li>• yuigahama yui</li><li>• nibutani shinka</li></ul>

Input:



rotation / axis problem

Output:

Character

Prediction (Top-3)



- suzukaze aoba
- kafuu chino
- yamada elf



- yagami kou
- ayase eli
- tomoe mami

---

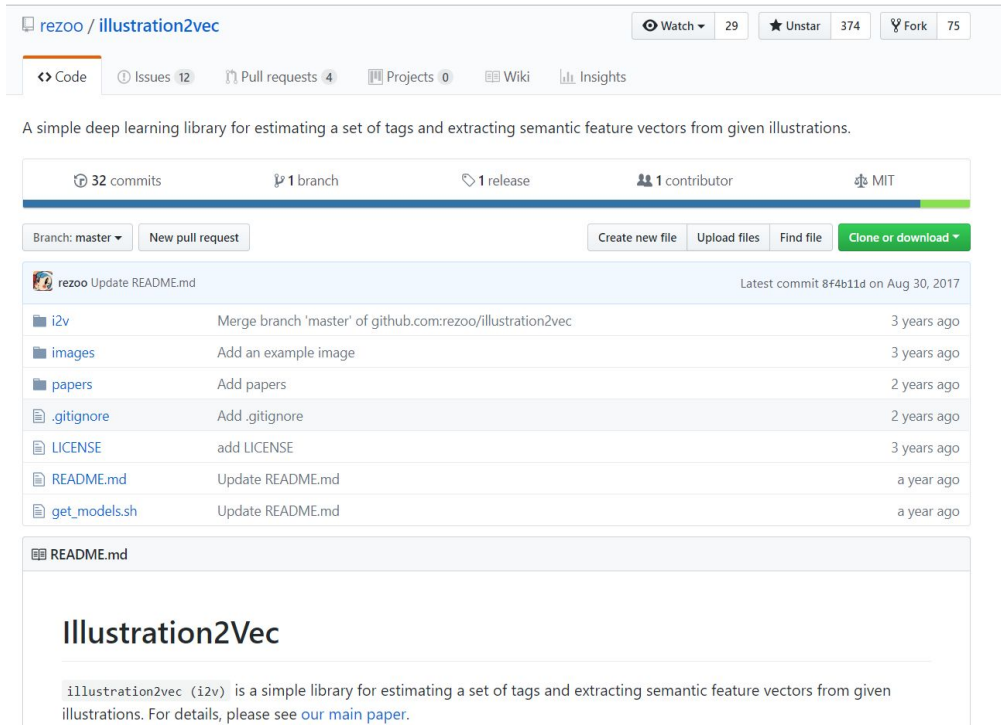
# Other Projects / Alternatives

# rezoo/illustration2vec

Semantic feature vectors.

For example:

- Character
- General (hair color, eye color, etc)



The screenshot shows the GitHub repository page for `rezoo/illustration2vec`. At the top, it displays the repository name, a watch button (29), an unstar button (374), and a fork button (75). Below this, there are navigation links for Code, Issues (12), Pull requests (4), Projects (0), Wiki, and Insights. A description of the repository is provided: "A simple deep learning library for estimating a set of tags and extracting semantic feature vectors from given illustrations." Below the description, statistics are shown: 32 commits, 1 branch, 1 release, 1 contributor, and MIT license. There are buttons for "Branch: master", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". A list of recent commits is shown, including updates to README.md, i2v, images, papers, .gitignore, LICENSE, and get\_models.sh. The README.md file is selected, showing the title "Illustration2Vec" and a description: "illustration2vec (i2v) is a simple library for estimating a set of tags and extracting semantic feature vectors from given illustrations. For details, please see our main paper."

rezoo / illustration2vec

Watch 29 Unstar 374 Fork 75

Code Issues 12 Pull requests 4 Projects 0 Wiki Insights

A simple deep learning library for estimating a set of tags and extracting semantic feature vectors from given illustrations.

32 commits 1 branch 1 release 1 contributor MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

rezoo Update README.md Latest commit 8f4b11d on Aug 30, 2017

i2v	Merge branch 'master' of github.com:rezoo/illustration2vec	3 years ago
images	Add an example image	3 years ago
papers	Add papers	2 years ago
.gitignore	Add .gitignore	2 years ago
LICENSE	add LICENSE	3 years ago
README.md	Update README.md	a year ago
get_models.sh	Update README.md	a year ago

README.md

## Illustration2Vec

illustration2vec (i2v) is a simple library for estimating a set of tags and extracting semantic feature vectors from given illustrations. For details, please see [our main paper](#).



# Chainer + illustration2vec

← → 🔒 Secure | [https://chainer-colab-notebook.readthedocs.io/ja/latest/notebook/hands\\_on\\_ja/chainer/classify\\_anime\\_characters.html#](https://chainer-colab-notebook.readthedocs.io/ja/latest/notebook/hands_on_ja/chainer/classify_anime_characters.html#)

🏠 Chainer Colab Notebook  
latest

Search docs

👤 Show on Colaboratory

学習ループを書いてみよう  
Trainerを使ってみよう  
新しいネットワークを書いてみよう  
データセットクラスを書いてみよう  
Chainerハンズオン: Pythonによるディープラーニング入門  
Chainer Tutorial Bookへようこそ!

Classify Anime Characters with Fine-tuning Model

ChainerRL で atari のゲームを DQN で解かしてみる  
ChainerRL クイックスタートガイド  
Sentiment Analysis with Recursive Neural Network  
Synthesize Human Speech with WaveNet  
Word2Vec: Obtain word embeddings  
1. Introduction

```
In [ ]: %matplotlib inline
import matplotlib.pyplot as plt

from PIL import Image
from chainer import cuda

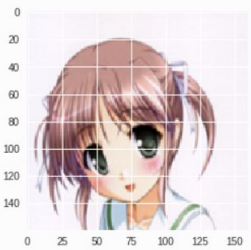
chainer.config.train = False
for _ in range(10):
    x, t = valid[np.random.randint(len(valid))]
    x = cuda.to_gpu(x)
    y = F.softmax(model.predictor(x[None, ...]))

    pred = os.path.basename(dnames[int(y.data.argmax())])
    label = os.path.basename(dnames[t])

    print('pred:', pred, 'label:', label, pred == label)

x = cuda.to_cpu(x)
x += mean[:, None, None]
x = x / 256
x = np.clip(x, 0, 1)
plt.imshow(x.transpose(1, 2, 0))
plt.show()
```

pred: 168\_asagiri\_mai label: 168\_asagiri\_mai True





## PaaS - Platform as a Service

- Clarifai → since 2016
- Azure Custom Vision → Beta release as 2018
- Google Cloud AutoML → Beta release as 2018



Cloud AutoML Vision

Amazon Rekognition

Metrics

Demos

Object and scene detection

Image moderation

Facial analysis

Celebrity recognition

**Face comparison**

Text in image

Video Demos

Video analysis

Additional Resources

Getting started guide

Download SDKs

Developer resources

Pricing

FAQ

Forum

## Face comparison

Compare faces to see how closely they match based on a similarity percentage.

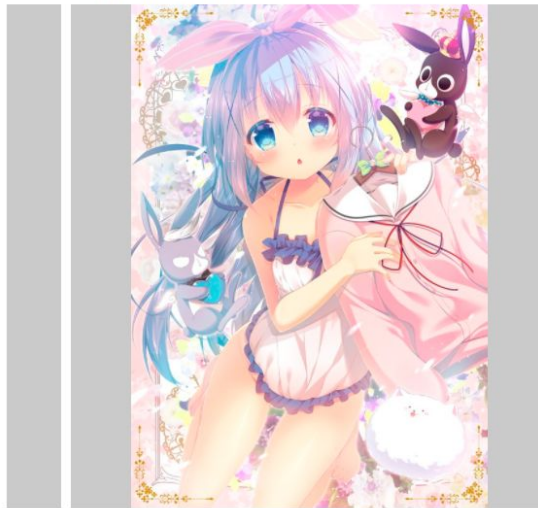
**InvalidParameterException (400)**  
Request has Invalid Parameters (Image must contain detectable faces)

Reference face



Choose a sample image

Comparison faces



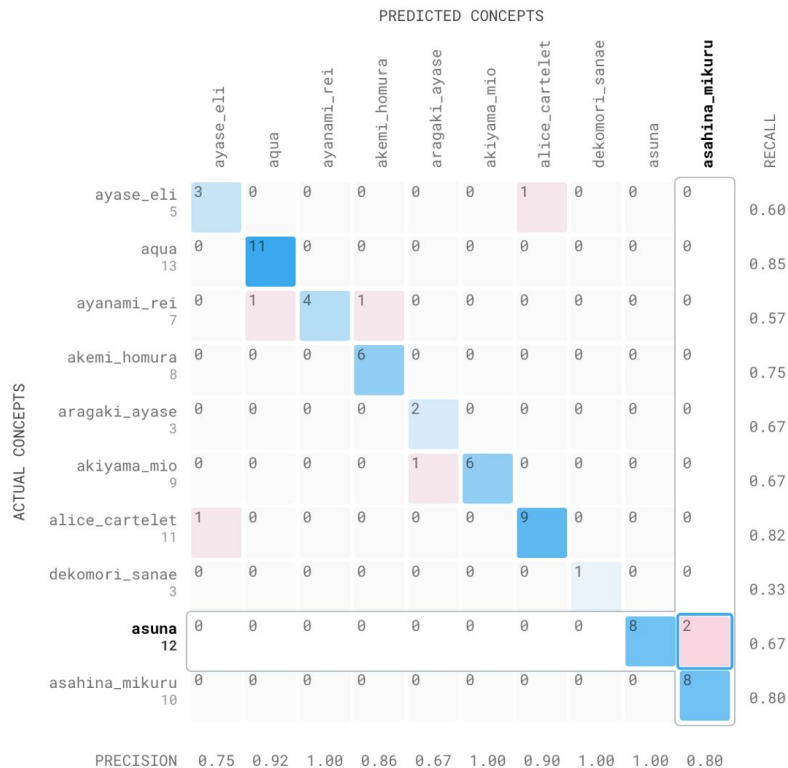
Choose a sample image

Done with the demo?

[Learn more](#)

- ▶ Results
- ▶ Request
- ▶ Response




# Clarifai UI



## Version Details

ID	bdd8aa6d39aa4c8ab1289ce0af910...
CREATED AT	Jun 4, 2018 5:00 pm
UNIQUE CONCEPTS	10
CONCEPTS MUTUALLY EXCLUSIVE	true
CLOSED ENVIRONMENT	false

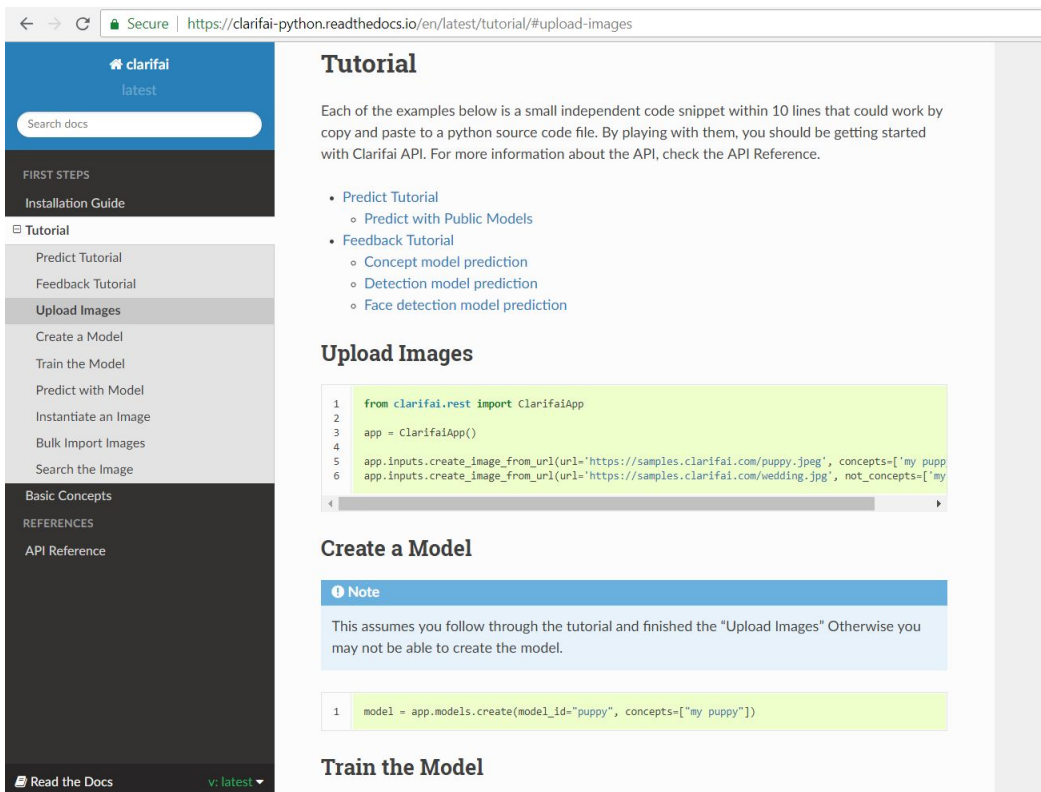
## Selection Details

IMAGES LABELED	PREDICTION PROBABILITY FOR
asuna	asahina_miku...
	0.7432
	0.7236
	0.2910

THRESHOLD



# Clarifai: Python API



The screenshot shows a web browser displaying the Clarifai Python API tutorial. The browser's address bar shows the URL: `https://clarifai-python.readthedocs.io/en/latest/tutorial/#upload-images`. The page has a dark blue header with the Clarifai logo and the text "latest". Below the header is a search bar labeled "Search docs". A sidebar on the left contains a navigation menu with sections: "FIRST STEPS" (Installation Guide), "Tutorial" (Predict Tutorial, Feedback Tutorial, Upload Images, Create a Model, Train the Model, Predict with Model, Instantiate an Image, Bulk Import Images, Search the Image), "Basic Concepts", "REFERENCES" (API Reference), and "Read the Docs" at the bottom with a dropdown menu for "v: latest".

## Tutorial

Each of the examples below is a small independent code snippet within 10 lines that could work by copy and paste to a python source code file. By playing with them, you should be getting started with Clarifai API. For more information about the API, check the API Reference.

- Predict Tutorial
  - Predict with Public Models
- Feedback Tutorial
  - Concept model prediction
  - Detection model prediction
  - Face detection model prediction

## Upload Images

```
1 from clarifai.rest import ClarifaiApp
2
3 app = ClarifaiApp()
4
5 app.inputs.create_image_from_url(url='https://samples.clarifai.com/puppy.jpeg', concepts=['my puppy']
6 app.inputs.create_image_from_url(url='https://samples.clarifai.com/wedding.jpg', not_concepts=['my
```

## Create a Model

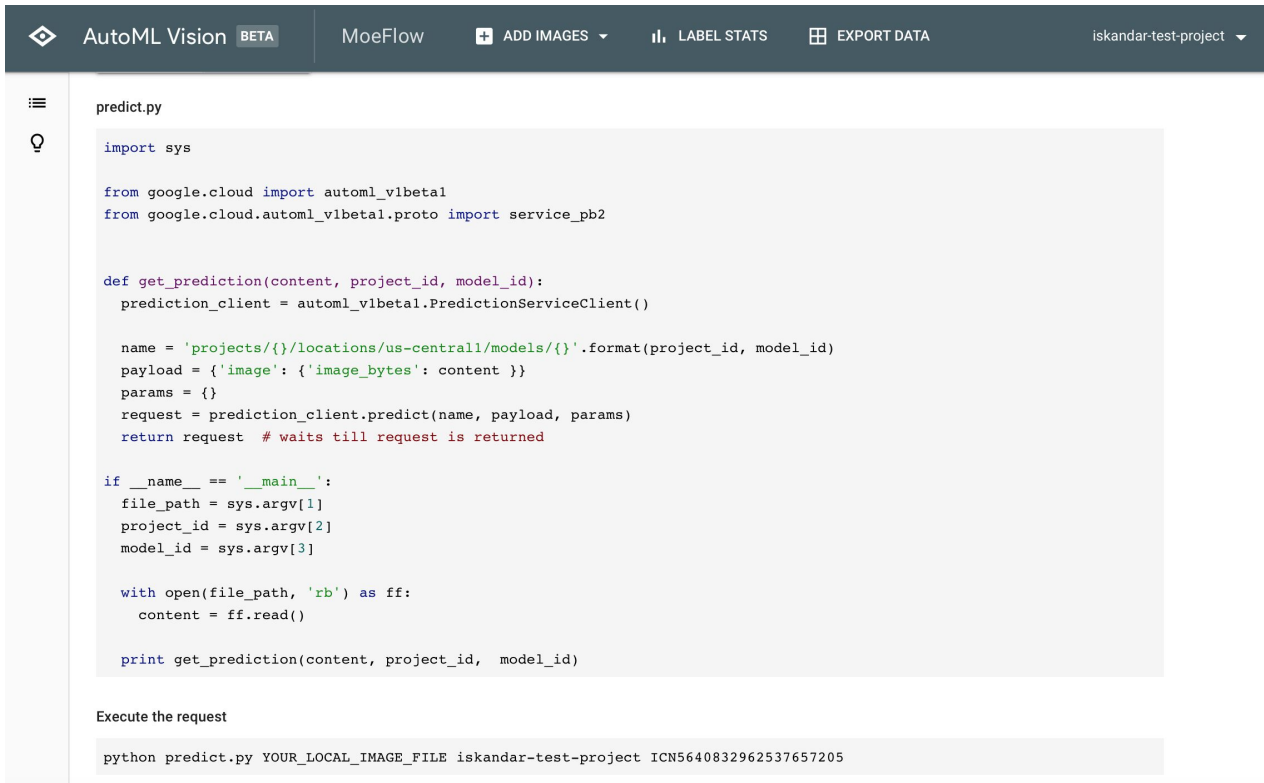
**Note**

This assumes you follow through the tutorial and finished the "Upload Images" Otherwise you may not be able to create the model.

```
1 model = app.models.create(model_id="puppy", concepts=["my puppy"])
```

## Train the Model

# Google Cloud AutoML: Python API



The screenshot displays the Google Cloud AutoML Vision interface. The top navigation bar includes the AutoML Vision logo, a 'BETA' badge, and several menu items: 'MoeFlow', '+ ADD IMAGES', '|| LABEL STATS', and 'EXPORT DATA'. The user's project name, 'iskandar-test-project', is visible on the right.

The main content area shows a Python script named 'predict.py'. The script imports the necessary modules and defines a function to interact with the AutoML Vision API. It uses the 'google.cloud' library to create a 'PredictionServiceClient' and sends a prediction request with an image payload. The script also includes a main block that takes command-line arguments for the file path, project ID, and model ID, and then reads the image file to perform the prediction.

```
predict.py

import sys

from google.cloud import automl_v1beta1
from google.cloud.automl_v1beta1.proto import service_pb2

def get_prediction(content, project_id, model_id):
    prediction_client = automl_v1beta1.PredictionServiceClient()

    name = 'projects/{}/locations/us-centrall/models/{}'.format(project_id, model_id)
    payload = {'image': {'image_bytes': content}}
    params = {}
    request = prediction_client.predict(name, payload, params)
    return request # waits till request is returned

if __name__ == '__main__':
    file_path = sys.argv[1]
    project_id = sys.argv[2]
    model_id = sys.argv[3]

    with open(file_path, 'rb') as ff:
        content = ff.read()

    print get_prediction(content, project_id, model_id)
```

Execute the request

```
python predict.py YOUR_LOCAL_IMAGE_FILE iskanar-test-project ICN5640832962537657205
```



# Quick Comparison (as August 2018)

Top-1 Accuracy with pre-processed model (after face detection, same dataset)

Clarifai → **95.2%** (they have been in this business for years)

Google Cloud AutoML → **93%** (their model is based on improved version of Inception v3, so it fails on almost the same test data)

**MoeFlow (my model)** → **88.6%** (let be fair, I'm not a machine learning expert and my job is unrelated to machine learning 🐱, but at least it's free and I can put it together with my server!)

Azure Custom Vision → **77.4%** (at least it's better than Microsoft, but they are still in Beta)

# Clarifai



## Details

TRANSFER-LEARNING-ANIME PREDICT [TRAIN](#)

<input checked="" type="checkbox"/> aragaki_ayase	0.59	<input type="checkbox"/> dekomori_sanae	0.00
<input type="checkbox"/> akemi_homura	0.25	<input type="checkbox"/> asuna	0.00
<input type="checkbox"/> ayanami_rei	0.06	<input type="checkbox"/> ayase_eli	0.00
<input type="checkbox"/> aqua	0.05	<input type="checkbox"/> asahina_mikuru	0.00
<input type="checkbox"/> akiyama_mio	0.04	<input type="checkbox"/> alice_cartelet	0.00

Create Concept

GENERAL-V1.3

GENERAL-V1.3

illustration	0.97	face	0.90
woman	0.96	man	0.90
fun	0.94	people	0.89
fashion	0.94	vector	0.89
young	0.92	music	0.88




# Google Cloud AutoML

AutoML Vision BETA MoeFlow + ADD IMAGES LABEL STATS EXPORT DATA iskandar-test-project

Model  
MoeFlow\_v20180803063827

### Test your model on new images

If your model will be used to make predictions on people, test your model on images that capture the diversity of your userbase. [Learn more](#)



**Predictions**

Only top 5 labels are shown.

aragaki_ayase	0.680
akiyama_mio	0.277
akemi_homura	0.032
asuna	0.007
asahina_mikuru	0.001

# MoeFlow

Input:



Output:

Character



Prediction  
(Top-3)

- aragaki ayase
- hyoudou michiru
- miki sayaka

```
Downloads — ec2-user@ip-172-31-48-241:~ — ec2-user@ip-172-31-48-241:~/
INFO:root:[('aragaki ayase', 0.54832304), ('hyoudou michiru', 0.21569832), ('miki sayaka', 0.079943478)]
2018-06-04 08:48:44 - (network)[INFO][127.0.0.1:58852]: POST http://127.0.0.1:8888/ 200 4226
INFO:network:
2018-06-04 08:48:44 - (network)[INFO][127.0.0.1:58854]: GET http://127.0.0.1:8888/static/images/1b0201bd4c6941f0a4ec66617c290e5b.jpg 200 32458
INFO:network:
2018-06-04 08:48:45 - (network)[INFO][127.0.0.1:58856]: GET http://127.0.0.1:8888/static/images/65a0f659c5734b69a27665dd56fe25b2.jpg 200 32458
INFO:network:
```

# Azure Custom Vision



## My Tags

aragaki\_ayase

## Predictions

Tag	Probability
aqua	11%
ayanami_rei	9.9%
akemi_homura	8.8%
akiyama_mio	6.3%
aragaki_ayase	5.7%
ayase_eli	2.1%
asuna	2%
asahina_mikuru	0.6%



## When should we use PaaS for Image Recognition?

(+) Image recognition is not directly related to your main business and you don't want to invest too much time on it

(+) You don't need the capability of customizing training model

(+) You have money \$\$\$, PaaS tends to get more expensive in the long run

(+) You are satisfied with the result, of course, you should compare all available PaaS out there, they have their own strong points



## What to compare:

- Single vs multi-objects, e.g.: Car vs [Car, Ball, ...]
- Single class vs multiple tags, e.g.: Car vs Toyota (Brand name), or [Vehicle: 0.99, Car: 0.97, ...]
- Top-1 vs Top-x accuracy
- Cost

---

# Closing Remarks

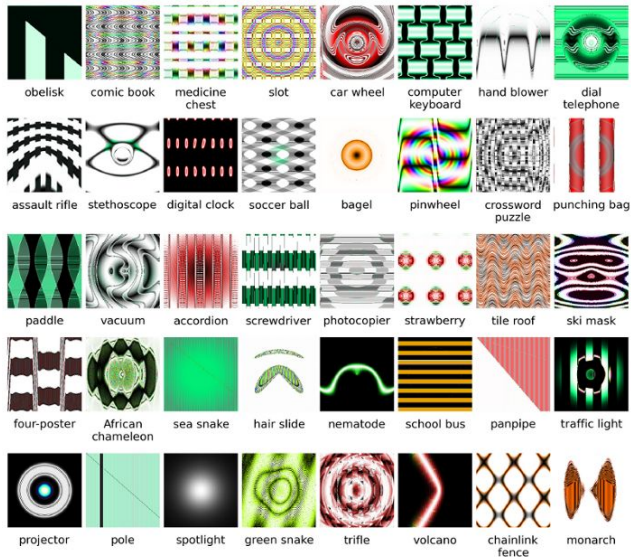


# “Never-ending” Development

As the number of category increases, Top-1 accuracy will also decrease since some characters are too similar (free-style art) and transfer learning effect is diminished.

- Image noise
- Rotation / axis
- Face expressions (closed eyes, etc)
- Characters with “multiple” forms
- Brightness & Contrast

# Fooling Neural Network



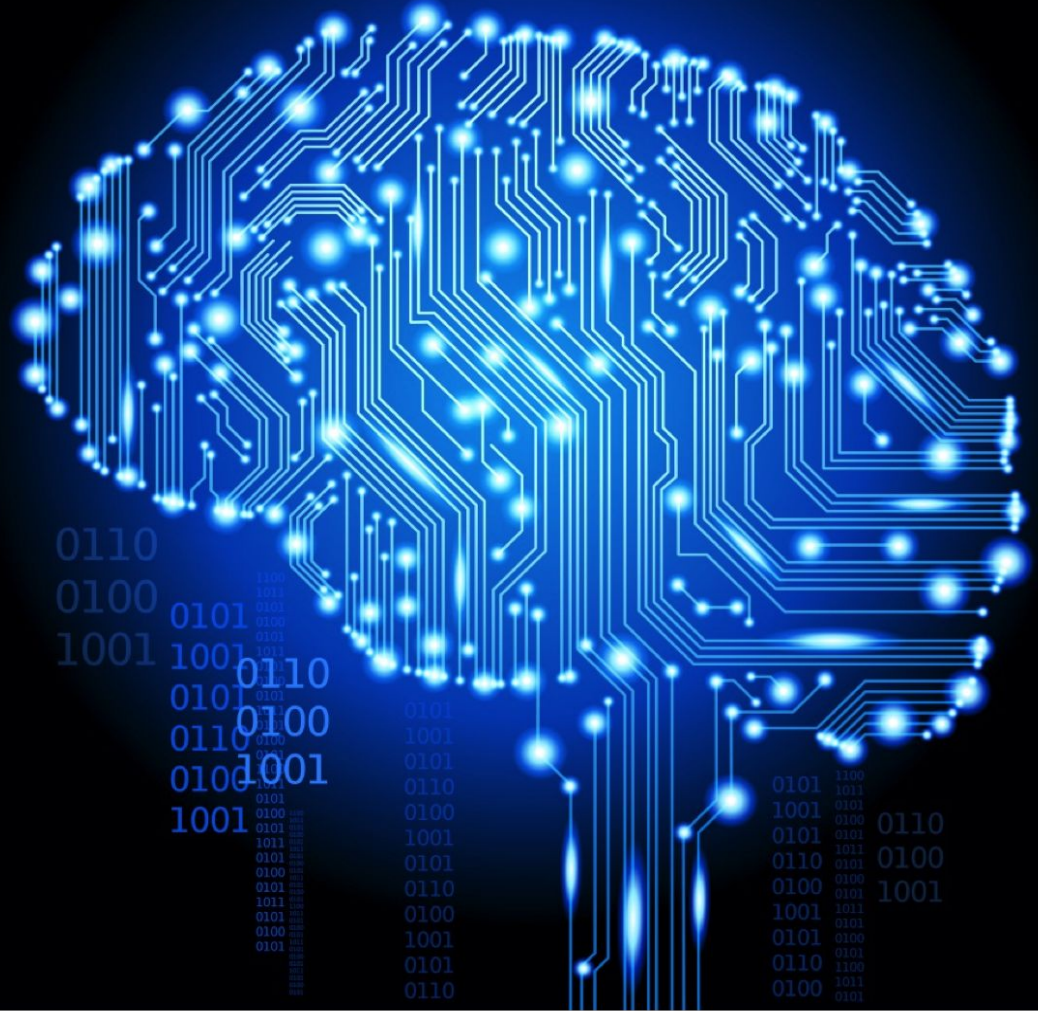
Original image  
Output Label: Teapot



Noisy image (10% impulse noise)  
Output Label: **Biology**



**Generalizing  
Ideas  
-  
Solve Your  
Own Problems!**





## Github Projects

**freedomofkeima/MoeFlow**: Repository for anime characters recognition website (Alpha)

**freedomofkeima/transfer-learning-anime**: Transfer Learning for Anime Characters Recognition

**freedomofkeima/opencv-playground**: Compare 2D and 3D OpenCV Cascade Classifier

## Presentation Slide

<https://freedomofkeima.com/pyconhk2018.pdf>

## Curated List

<https://github.com/kjw0612/awesome-deep-vision>

<http://www.themtank.org/a-year-in-computer-vision>



# HDE, Inc. at Shibuya, Tokyo



Global Internship Program  
<https://www.hde.co.jp/en/gip/>



**Full-timer benefits:** Get your CfP accepted and the company will handle the rest!





Thank you!



\*\*\*