Deep Learning in Python: Image Recognition for Anime Characters with Transfer Learning

1st PyCon in Indonesia - 2017

Iskandar Setiadi

PYCON

HDE
Liberation of Technology

Github

https://github.com/freedomofkeima

Website

https://freedomofkeima.com/

From Jakarta, Indonesia
Graduated from ITB - 2015
Speaker in PyCon JP - 2017

Iskandar Setiadi

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

Software Engineer at Japan

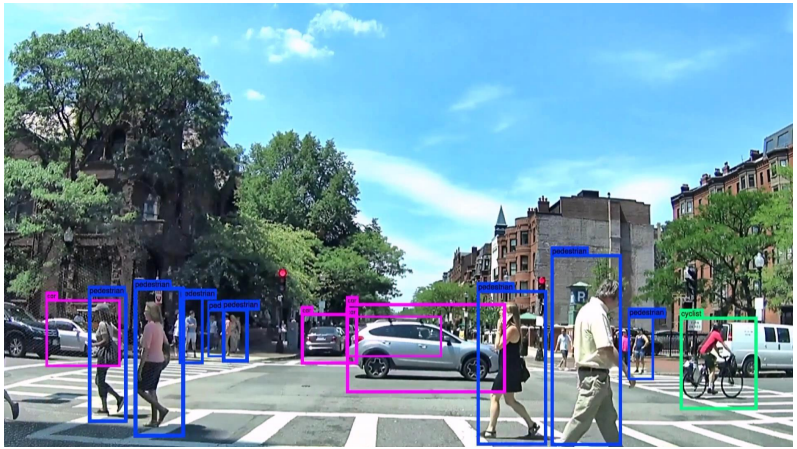
Why Python?

→ Easy to use

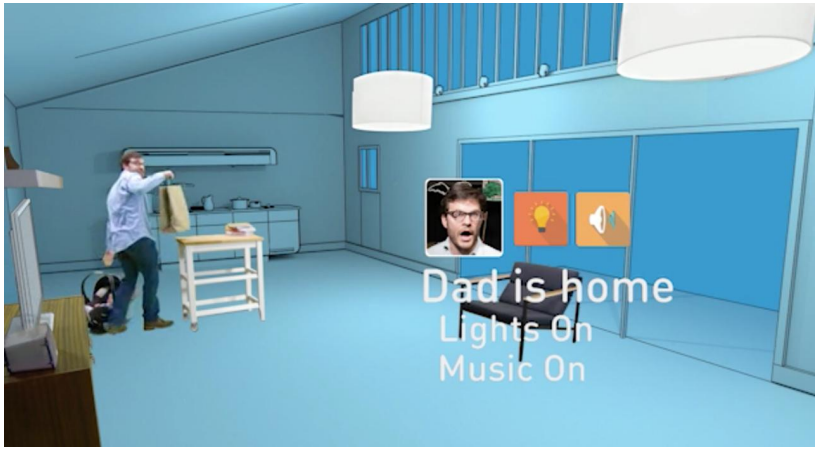
→ Great community

→ Swiss army knife: website development, data science, etc

Image Recognition in Daily Life

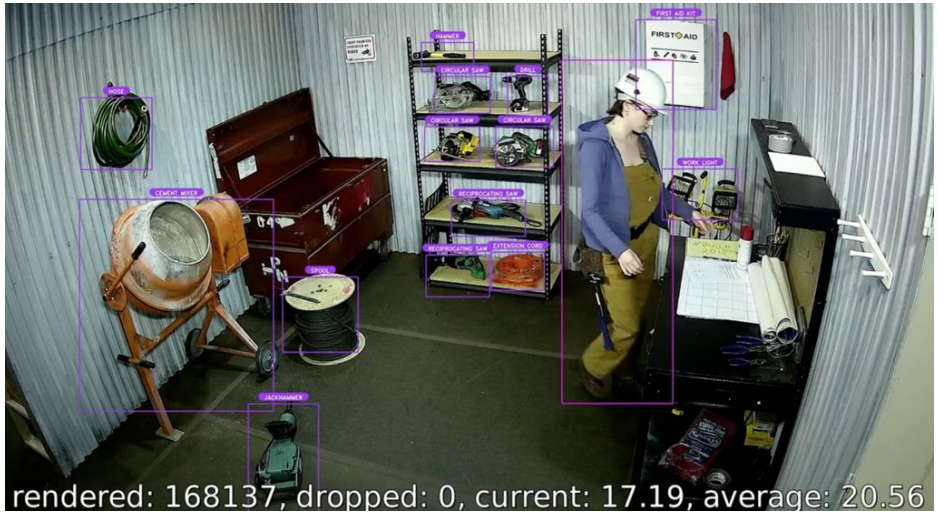


Self-driving car

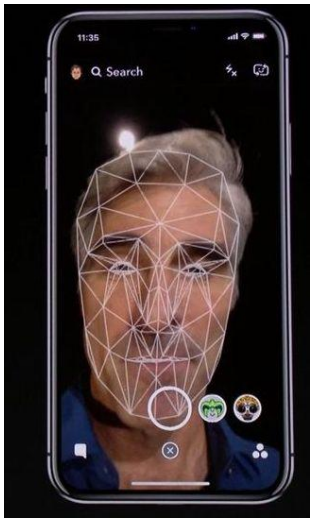


Smart home

Image Recognition in Daily Life



Smart workplace

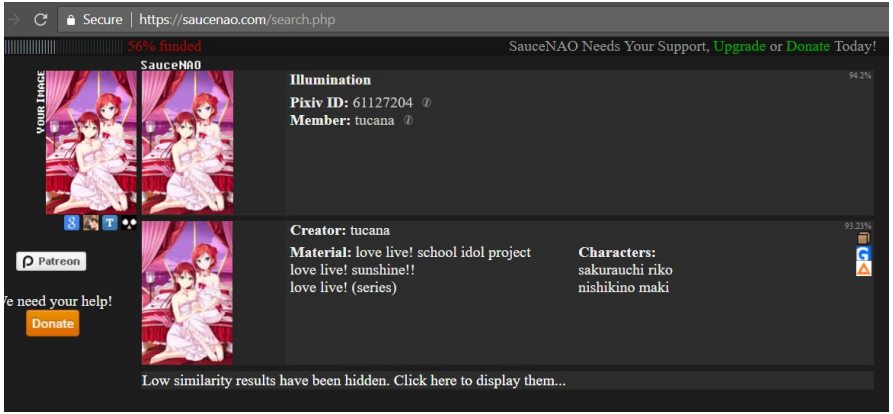


Face ID

Background

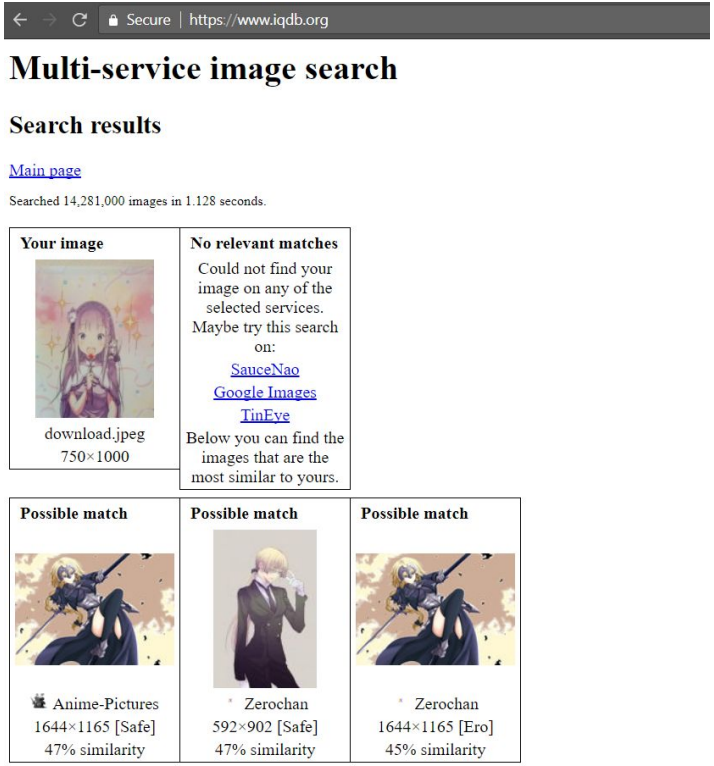
Problem

- Cropped images
- Edited images
- Unindexed images



Problem

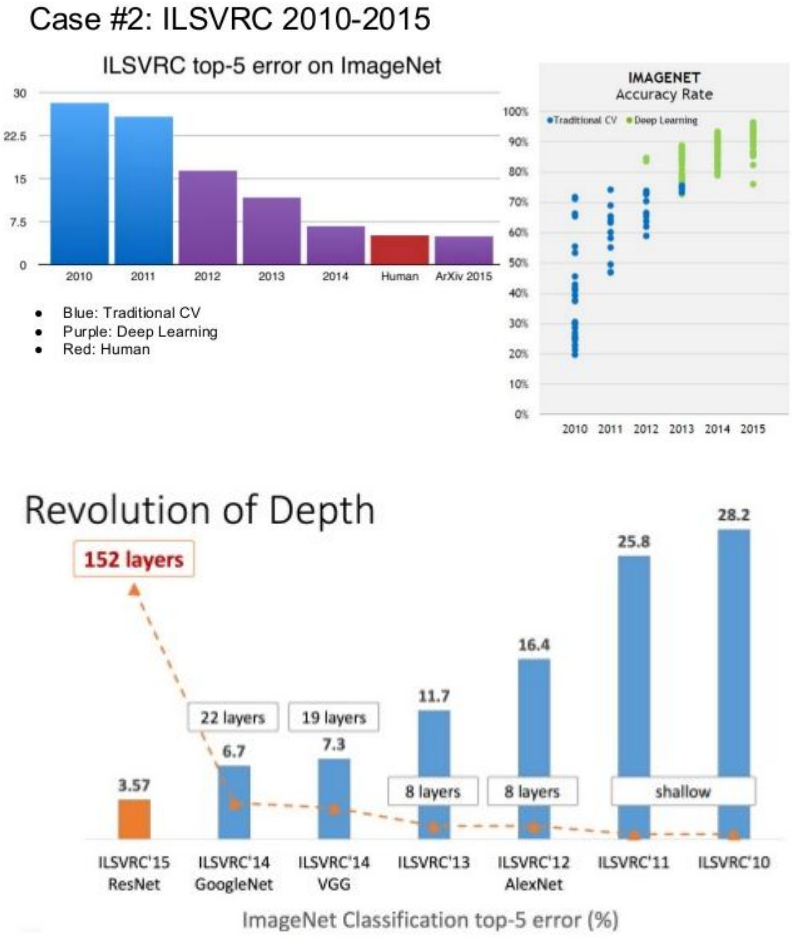
- Photos



ILSVRC

Largest Computer Vision Competition

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



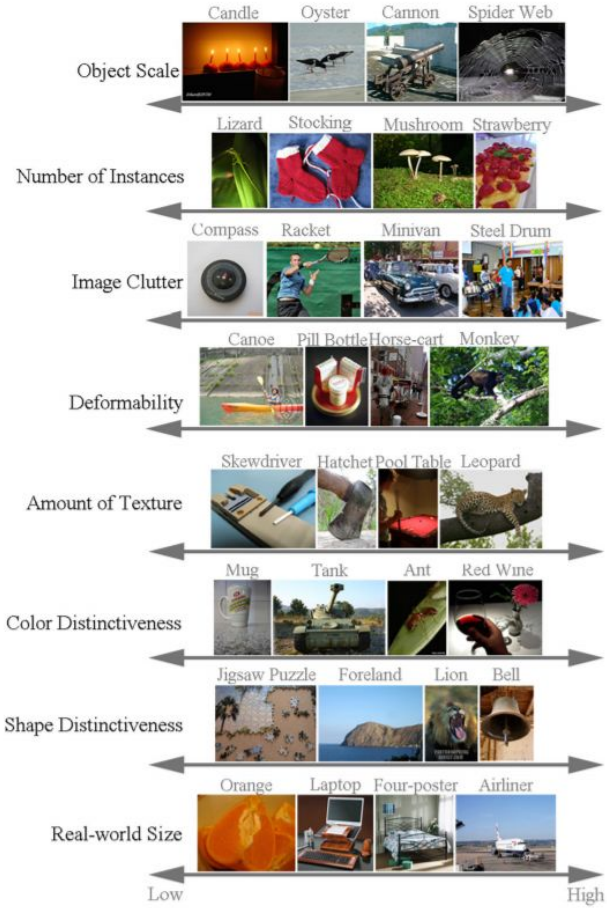


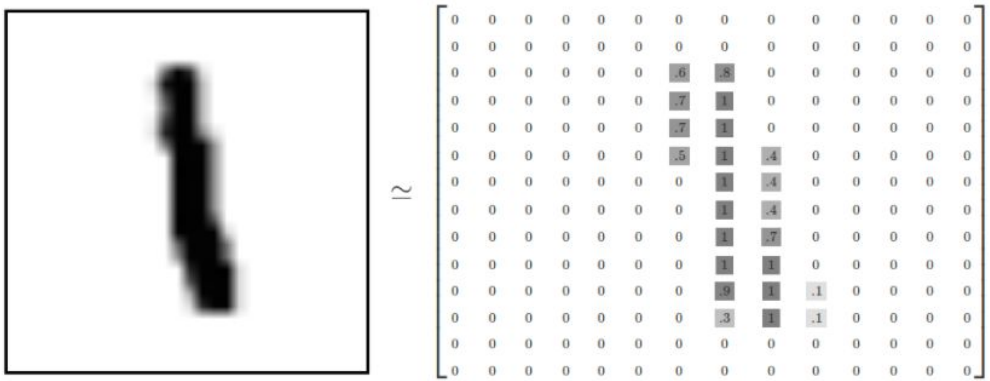
Image Recognition Challenges

Tutorial for ML Beginner: MNIST & TensorFlow

55000 Training data

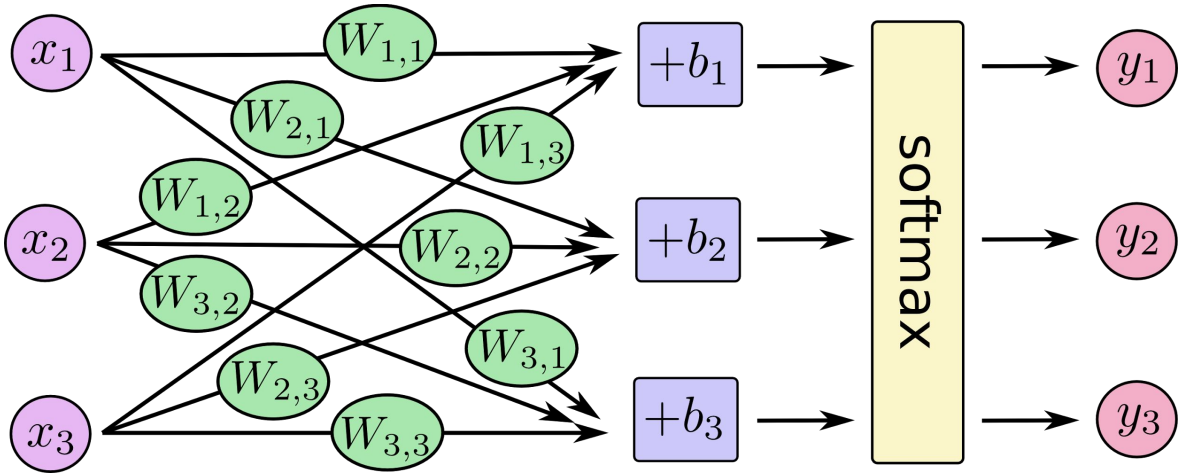
5000 Validation data

10000 Test data



URL: https://www.tensorflow.org/get_started/mnist/beginners

Tutorial for ML Beginner: MNIST & TensorFlow



URL: https://www.tensorflow.org/get_started/mnist/beginners

TensorFlow Installation

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

or

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

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

MNIST Model: TensorFlow + Python

```
x = tf.placeholder(tf.float32, [None, 784])  # Placeholder
W = tf.Variable(tf.zeros([784, 10]))  # Weight (W)
b = tf.Variable(tf.zeros([10]))  # Bias (b)

# Tensor Flow it!
# We can run it in CPU and GPU (let TensorFlow handle it)
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

MNIST Result & Comparison

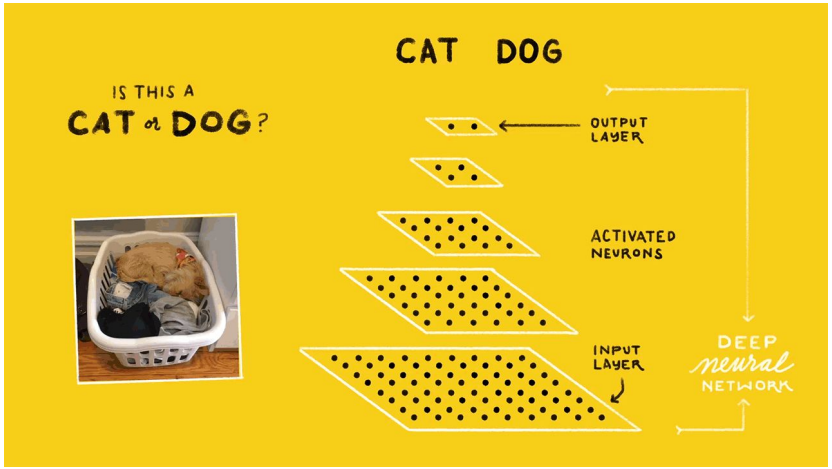
Multilayer Neural Network with Logistic Regression Acc. : ~ 91%
Speed (1000 iter, 0.01 learning rate): < 1 minute

Convolutional Neural Network (Deep Learning) Acc.: ~ 99%
Speed (20000 iter, 0.0001 learning rate):
~2700 seconds (without GPU), ~360 seconds (with GPU)

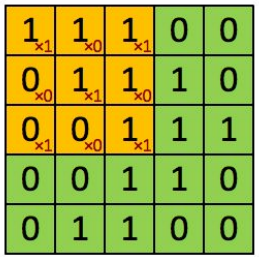
Deep Learning

Increasing number of iterations will get stagnated at certain point.

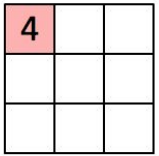
More layers! But it is slow :’(



Deep Learning: Convolution



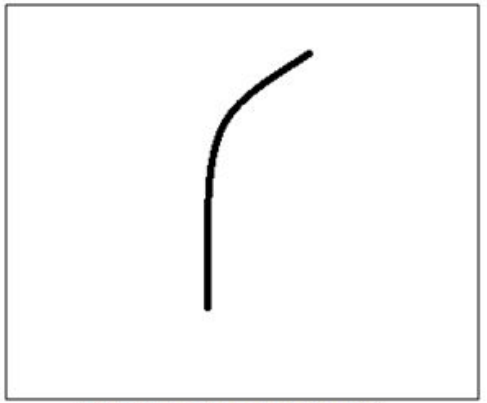
Image



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

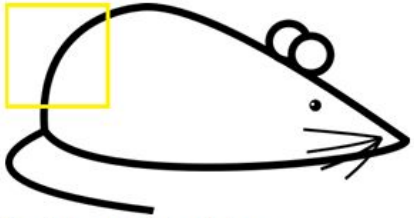
Pixel representation of filter



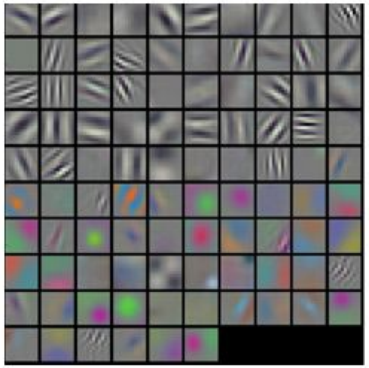
Visualization of a curve detector filter

Deep Learning: Convolution

Original image

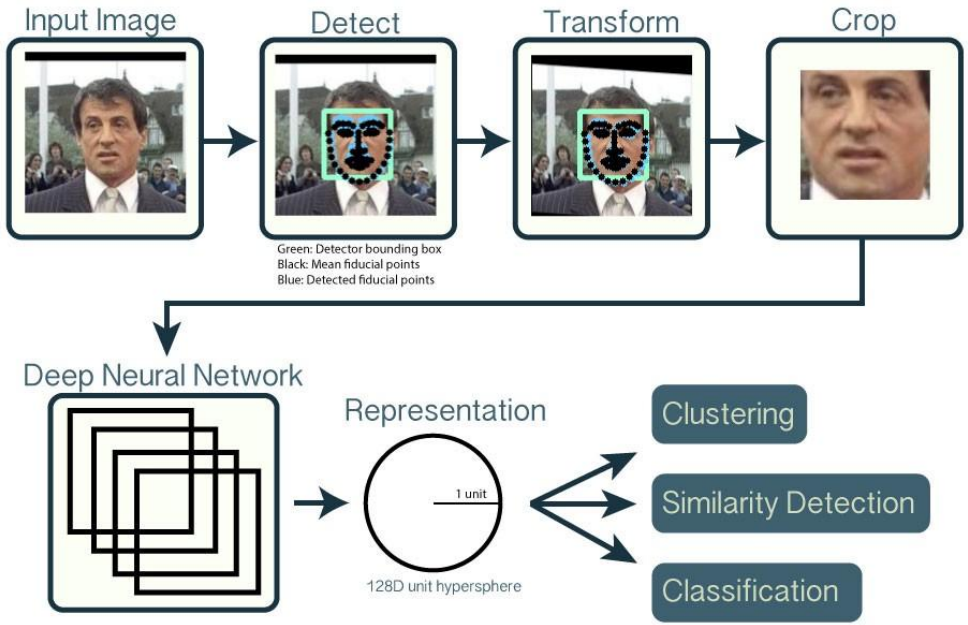


Visualization of the filter on the image



Visualizations of filters

Face Detection: Introduction



Face Detection (Human Face)

Adapted from https://github.com/shantnu/FaceDetect:

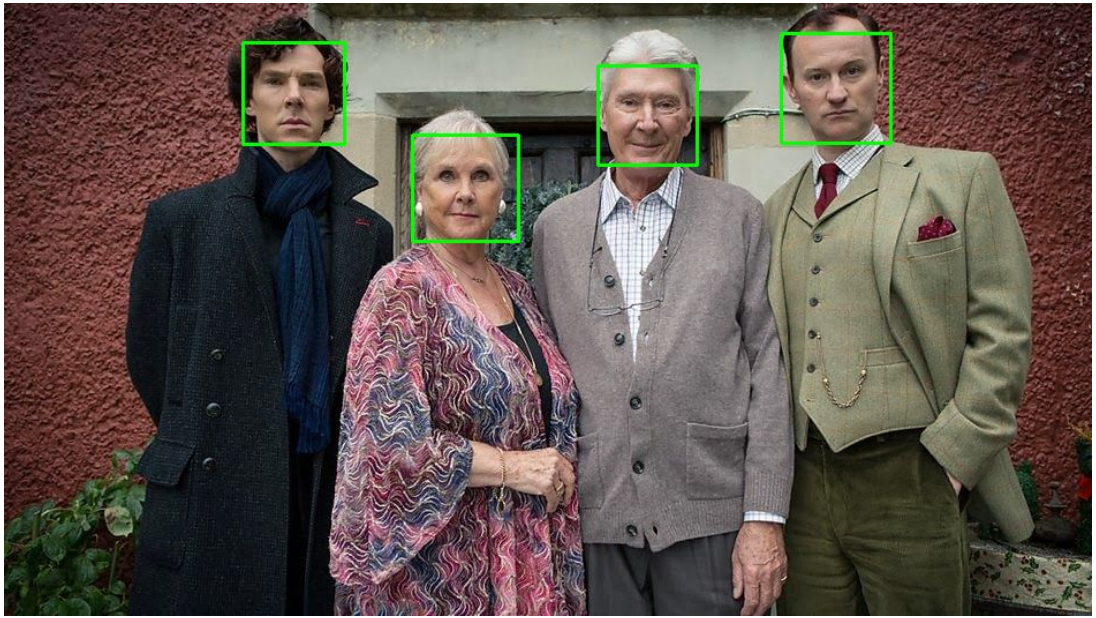
```
import cv2


faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1,
                                     minNeighbors=5, minSize=(30, 30))
```

Face Detection (Human Face)

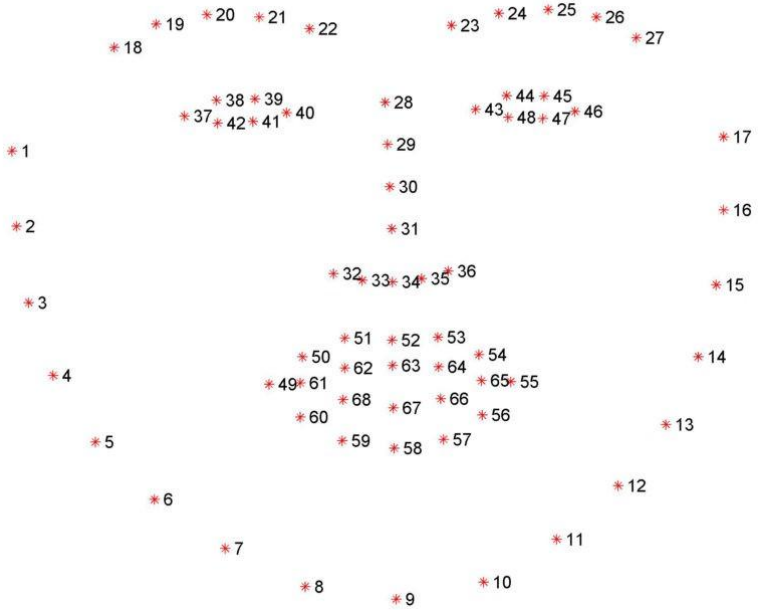


Face Detection (Same Model - Anime Face)

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

Facial features are different!

e.g.: 2D has no nose



Face Detection: Train New Model!

Adapted from https://github.com/nagadomi/lbpcascade_animeface:

```
import cv2


cascade = cv2.CascadeClassifier("lbpcascade_animeface.xml")

image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = cascade.detectMultiScale(gray, scaleFactor=1.1,
                                 minNeighbors=5, minSize=(24, 24))
```

Face Detection (Anime Face)

Face Detection (Same Model - Human Face)

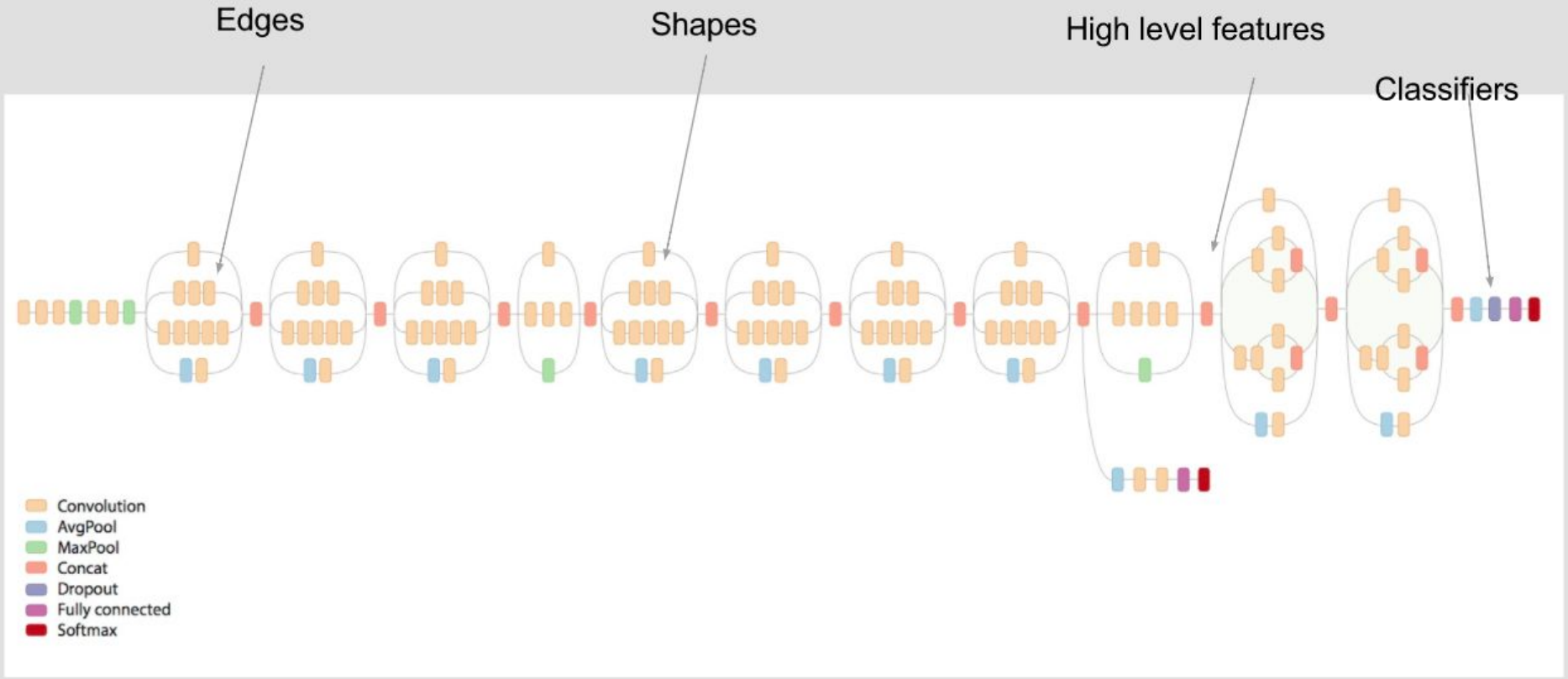
Face Recognition

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

What does the layers learn?



Transfer Learning

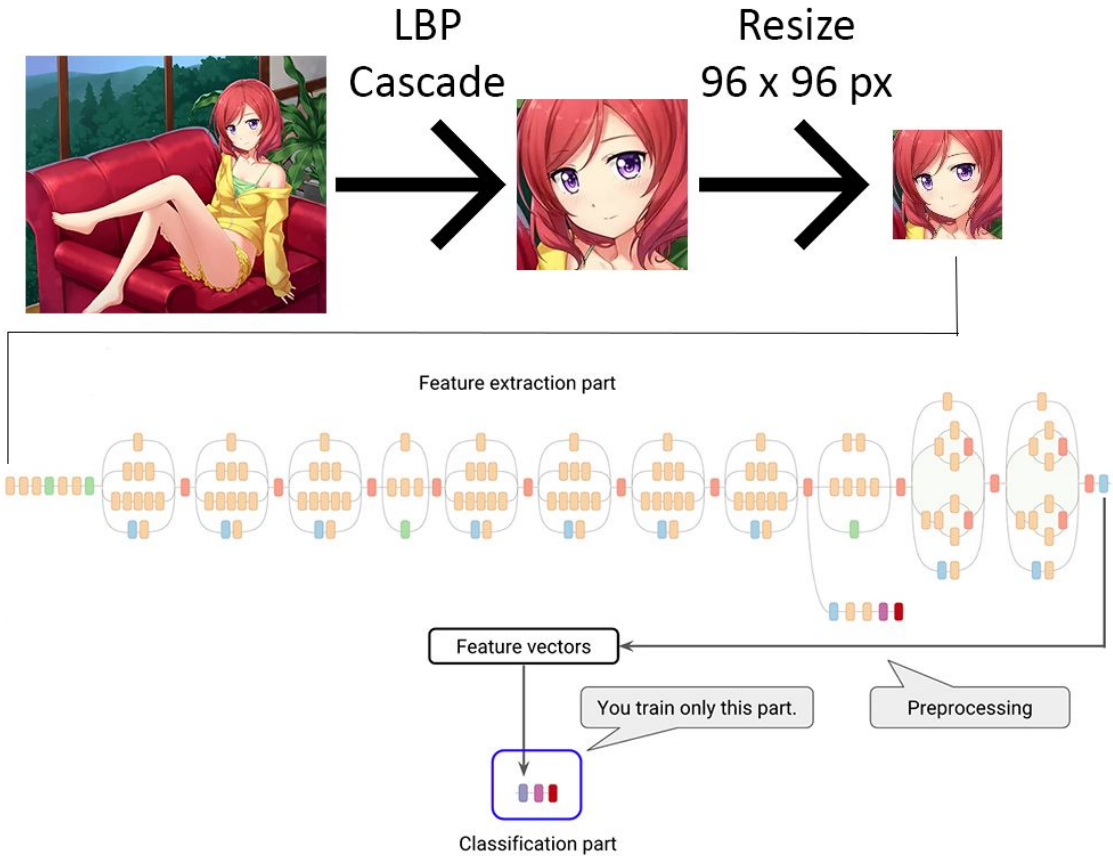
From certain Top-5 characters indexing website:

- 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.

URL: https://www.tensorflow.org/tutorials/image_retraining


LBP
Cascade
Resize
96 x 96 px
Feature extraction part
Feature vectors
You train only this part.
Preprocessing
Classification part

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

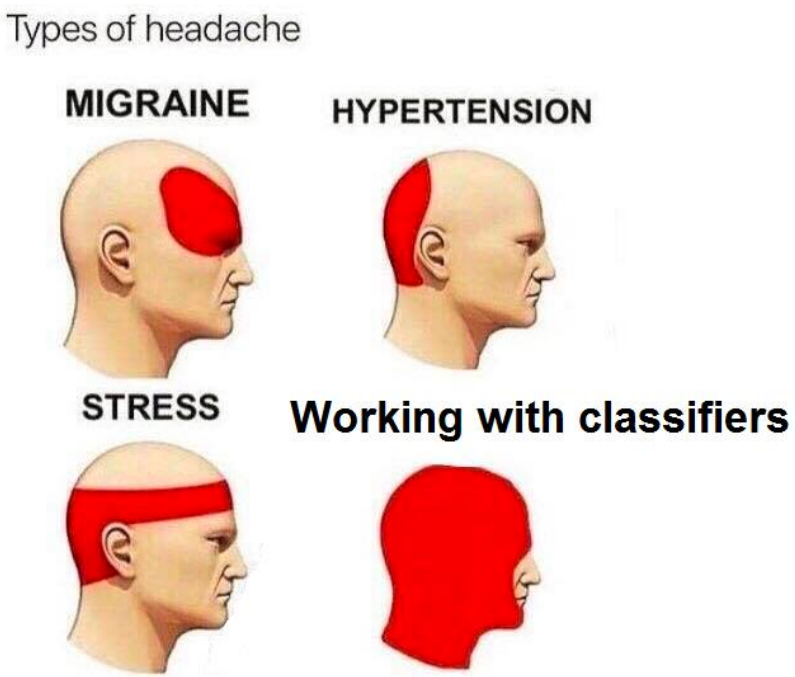
Build the retrainer:

```
$ 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

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

MoeFlow: Specification

→ Build with Sanic (Flask-like Python 3.5+ web server)

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

What it does:

- Run face detection with OpenCV

- Resize image to a fixed proportion

- Run classification with TensorFlow

```
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
```

MoeFlow: Use Retrained Model

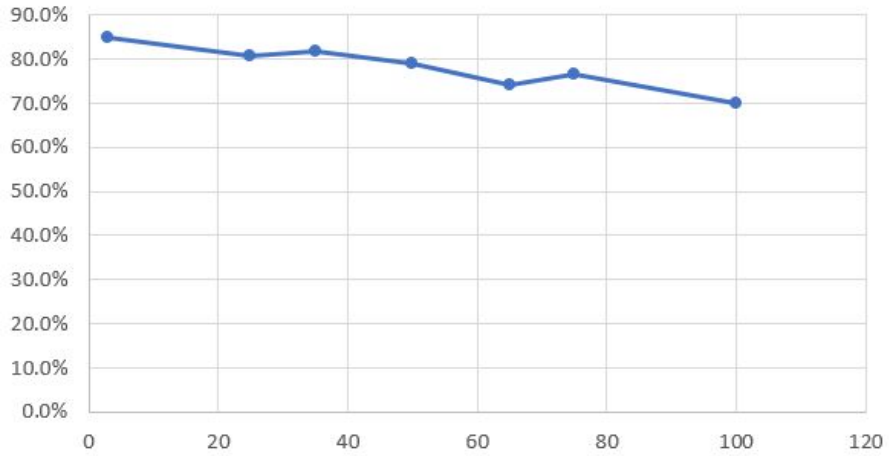
```
2017-12-06 01:51:53 - (network)[INFO][127.0.0.1:49154]: GET http://127.0.0.1:888
8/static/images/83a97094d7634333b5c9d7e8e0901da4.jpg  200 119582
INFO:network:
2017-12-06 01:51:53 - (network)[INFO][127.0.0.1:49156]: GET http://127.0.0.1:888
8/static/images/e96410fcf8114fbb88a4f33ede40205e.jpg  200 5902
INFO:network:
2017-12-06 05:48:33 - (network)[INFO][127.0.0.1:59920]: GET http://127.0.0.1:888
8/  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://127.0.0.1:88
88/  200 1565
INFO:network:
2017-12-06 05:49:39 - (network)[INFO][127.0.0.1:59972]: GET http://127.0.0.1:888
8/static/images/fda2e4333b194f30a845ed8d320bf449.jpg  200 294605
INFO:network:
2017-12-06 05:49:39 - (network)[INFO][127.0.0.1:59974]: GET http://127.0.0.1:888
8/static/images/d9a530d76c17404fbdbde7a2e5bf9c55.jpg  200 5807
INFO:network:
```

MoeFlow: Use Retrained Model

Test Results (Number of Class)

With 100 class and 60 images per class, it achieves 70.1% top-1 accuracy.

When the number of class is relatively small (~35), it can achieve 80%+ top-1 accuracy.



URL: https://github.com/freedomofkeima/MoeFlow/blob/master/100_class_traning_note.md

Test Results (Dataset size)

100 class experiment:

→ 30 images per class: **60.3%** accuracy

→ 60 images per class: **70.1%** accuracy

All tests are done with images which are not in training / validation set.

URL: https://github.com/freedomofkeima/MoeFlow/blob/master/100_class_traning_note.md

Secure | https://freedomofkeima.com/moeflow/

ec2-user@ip-172-31-48-241:~/MoeFlow

```
2017-12-02 10:38:45 - (network)[INFO][127.0.0.1:35584]: POST http://127.0.0.1:88
88/  200 1577
INFO:network:
2017-12-02 10:38:45 - (network)[INFO][127.0.0.1:35586]: GET http://127.0.0.1:888
8/static/images/bdb9c730b1074a4c8506fc7e0d733c7d.jpg  200 160714
INFO:network:
2017-12-02 10:38:45 - (network)[INFO][127.0.0.1:35588]: GET http://127.0.0.1:888
8/static/images/f16e3eb03197475c865a0237fe0c7db7.jpg  200 6142
INFO:network:
INFO:root:Height: 1574, Width: 2048
INFO:root:Input file is created at /tmp/tmpjelu9cy8.jpg
INFO:root:Processing classification
INFO:root:[('tedeza rize', 0.39948133), ('kasumigaoka utaha', 0.12605736), ('hyo
udou michiru', 0.068339251)]
INFO:root:Processing classification
INFO:root:[('kirima sharo', 0.97146475), ('hoto cocoa', 0.018899804), ('alice ca
rtelet', 0.0034658003)]
INFO:root:Processing classification
INFO:root:[('hoto cocoa', 0.94451982), ('kirima sharo', 0.028552018), ('yagami k
ou', 0.0080380896)]
2017-12-02 10:39:08 - (network)[INFO][127.0.0.1:35590]: POST http://127.0.0.1:88
88/  200 2013
INFO:network:
2017-12-02 10:39:08 - (network)[INFO][127.0.0.1:35592]: GET http://127.0.0.1:888
```

Output:

 Prediction: tedeza rize, kasumigaoka utaha, hyoudou michiru

 Prediction: kirima sharo, hoto cocoa, alice cartelet

 Prediction: hoto cocoa, kirima sharo, yagami kou

Results / Demo

Secure | https://freedomofkeima.com/moeflow/

Note: This operation will be very slow (around 15 seconds) if there are a lot of characters in a single image!

Choose File | No file chosen | Submit

Input:

Output:

Prediction: kousaka kirino, sakura nene, dekomori sanae

ec2-user@ip-172-31-48-241:~/MoeFlow

```
2017-12-02 11:25:42 - (network)[INFO][127.0.0.1:38124]: GET http://127.0.0.1:888
8/static/images/3775c305912b482693c7f32daafd52c1.jpg  200 6643
INFO:network:
2017-12-02 11:25:42 - (network)[INFO][127.0.0.1:38126]: GET http://127.0.0.1:888
8/static/images/4cca249295f4401eb45fc0e34da809c2.jpg  200 7175
INFO:network:
2017-12-02 11:25:42 - (network)[INFO][127.0.0.1:38128]: GET http://127.0.0.1:888
8/static/images/8e6db243c7aa458c98061de61c0496c6.jpg  200 6126
INFO:network:
INFO:root:Height: 800, Width: 534
INFO:root:Input file is created at /tmp/tmpl35gbz41.jpg
INFO:root:Processing classification
INFO:root:[('kousaka kirino', 0.42335555), ('sakura nene', 0.35790154), ('dekomo
ri sanae', 0.08103735)]
2017-12-02 11:26:55 - (network)[INFO][127.0.0.1:38178]: POST http://127.0.0.1:88
88/  200 1575
INFO:network:
2017-12-02 11:26:56 - (network)[INFO][127.0.0.1:38180]: GET http://127.0.0.1:888
8/static/images/dc40cb23b9534196a3226f9f843e8500.jpg  200 80962
INFO:network:
2017-12-02 11:26:56 - (network)[INFO][127.0.0.1:38182]: GET http://127.0.0.1:888
8/static/images/a2dc5e325342434f898495ec5b8cf64f.jpg  200 5796
INFO:network:
```

Results / Demo

Secure | https://freedomofkeima.com/moeflow/

Input:

ec2-user@ip-172-31-48-241:~/MoeFlow

```
2017-12-02 11:00:43 - (network)[INFO][127.0.0.1:36820]: GET http://127.0.0.1:888
8/static/images/d6bfe2062fa844d88e7095cff241eb40.jpg  200 5372
INFO:network:
INFO:root:Height: 860, Width: 1280
INFO:root:Input file is created at /tmp/tmpish217do.jpg
INFO:root:Processing classification
INFO:root:[('suzukaze aoba', 0.90099531), ('kafuu chino', 0.030359417), ('yamada
 elf', 0.013034027)]
INFO:root:Processing classification
INFO:root:[('yagami kou', 0.89891928), ('ayase eli', 0.043274842), ('tomoe mami'
, 0.026960159)]
2017-12-02 11:00:59 - (network)[INFO][127.0.0.1:36822]: POST http://127.0.0.1:88
88/  200 1782
INFO:network:
2017-12-02 11:00:59 - (network)[INFO][127.0.0.1:36824]: GET http://127.0.0.1:888
8/static/images/02dfd53aeafc42d4b0189bd376f0f05b.jpg  200 264179
INFO:network:
2017-12-02 11:00:59 - (network)[INFO][127.0.0.1:36826]: GET http://127.0.0.1:888
8/static/images/7c65786b9db74bdba83a97f908ff9823.jpg  200 6871
INFO:network:
2017-12-02 11:01:00 - (network)[INFO][127.0.0.1:36828]: GET http://127.0.0.1:888
8/static/images/7a968786c4a24f01a8dfd998e4ba363e.jpg  200 6488
INFO:network:
```

Output:

Rotation / Axis problem

Prediction: suzukaze aoba, kafuu chino, yamada elf

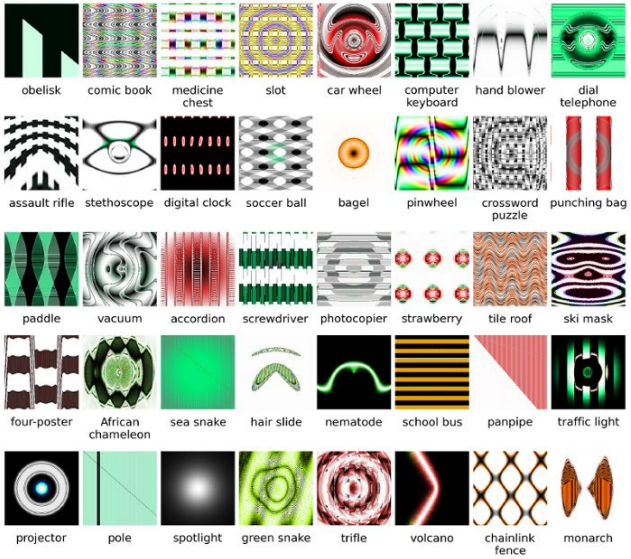
Prediction: yagami kou, ayase eli, tomoe mami

Problems (Example)

“Never-ending” Development

- 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**

Image Recognition as a Service

If you need image recognition features for production-ready environment and you don’t have any specific requirements to build your model from ground:

- Amazon Rekognition
- Computer Vision API in Cognitive Service (Azure)

Amazon Rekognition

Deep learning-based image recognition service
Search, verify, and organize millions of images

Object and Scene Detection

Facial Analysis

Face Comparison

Facial Recognition

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

My 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/pyconid2017.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/)
- ➔ 15% international people
- ➔ 6 people from Indonesia

Thank you!

freedomofkeima

***