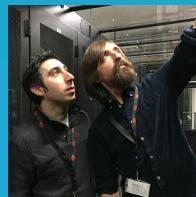




FOSDEM, FEB 2 2019

AI image search with Go & Tensorflow



Gildas Chabot

Leboncoin

 gildasch



AI is about
computers doing
magic

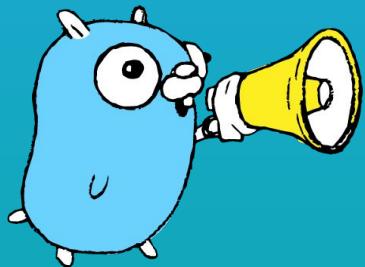


"Hi, I'm calling to book a women's haircut for a client."



01. AI, Tensorflow & Go





Today's (glorious) blather.

AI, Tensorflow & Go 01

Image classification 02

Face recognition 03

Search 04

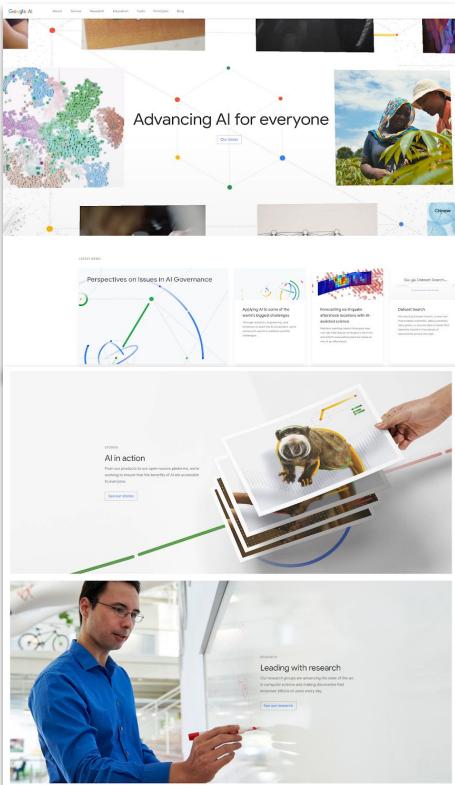
Conclusion 05



SECTION ONE

AI, Tensorflow & Go

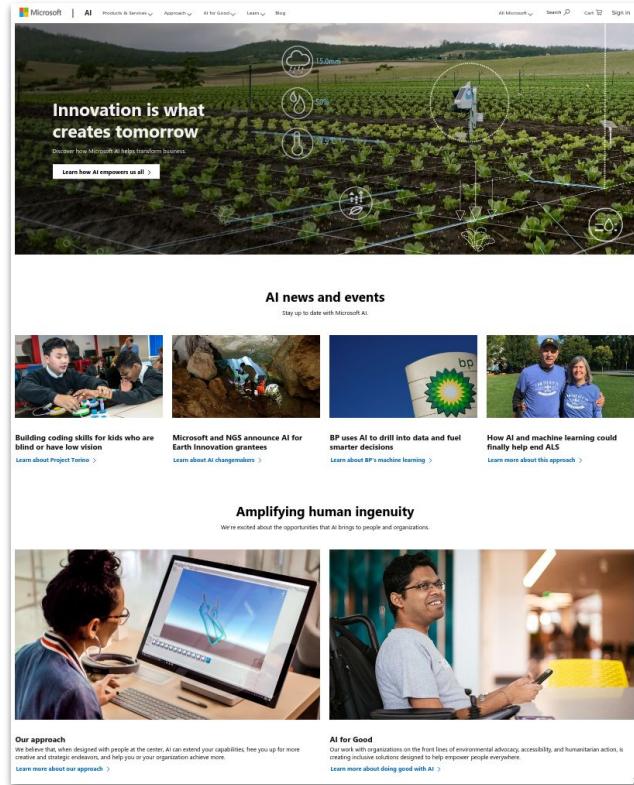
Google, Facebook, Microsoft, Baidu in the AI Race



The Google AI homepage features a collage of images related to AI, including a molecular structure, a map, and people working. A central banner reads "Advancing AI for everyone". Below it, a section titled "Perspectives on Issues in AI Governance" includes links to "Applying AI to some of the world's biggest challenges", "Promoting equitable approaches to AI research", and "Dataset search". Another section, "AI in action", shows a person interacting with a tablet displaying a 3D model of a baboon's brain. At the bottom, a man in a blue shirt is shown giving a presentation.

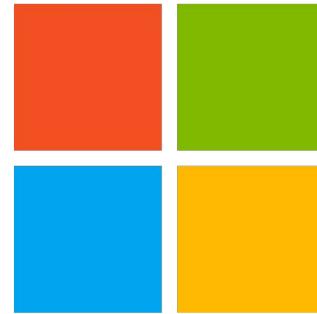


The Facebook AI Research homepage has a dark blue header with the "facebook research" logo and navigation links for Research Areas, Publications, People, Academic Programs, Downloads & Projects, Careers, and Blog. The main content area features the "Facebook AI Research" logo and the tagline "Advancing the field of machine intelligence". Below this, a video player shows a group photo of researchers, with the text "Leading with research" and "Our research groups are advancing the state of the art in AI, from computer vision to natural language processing, helping billions of users every day." A "Learn more" button is present.



The Microsoft AI homepage has a top navigation bar with links for AI, Products & Services, Approach, AI for Good, Learn, and Blog. The main visual is a photograph of a field with a circular overlay showing agricultural data like "15.00°C", "50%", "20°C", and "E0:00". The tagline "Innovation is what creates tomorrow" is displayed. Below the image, a button says "Learn how AI empowers us all". The "AI news and events" section features four cards: "Building coding skills for kids who are blind or have low vision", "Microsoft and NGS announce AI for Earth Innovation grants", "BP uses AI to drill into data and fuel smarter decisions", and "How AI and machine learning could finally help end ALS". The "Amplifying human ingenuity" section shows a woman working at a computer and a man in a wheelchair using a smartphone. The "Our approach" section discusses AI's potential for creative and strategic endeavors. The "AI for Good" section highlights work with organizations on environmental advocacy, accessibility, and humanitarian actions.

As many frameworks...



K Keras



... and models too!

TensorFlow Hub	
Text Embedding Image Classification Feature Vector Generator Other Video Classification Publishers Google DeepMind	<p>Text embedding</p> <div><p>universal-sentence-encoder By Google text-embedding DAN English Encoder of greater-than-word length text trained on a variety of data.</p><p>universal-sentence-encoder-large By Google text-embedding Transformer English Encoder of greater-than-word length text trained on a variety of data.</p><p>elmo By Google text-embedding billion-word-benchmark ELMO English Embeddings from a language model trained on the 1 Billion Word Benchmark.</p></div> <p>View more text embeddings</p> <p>Image feature vectors</p> <div><p>imagenet/inception_v3/feature_vector By Google image-feature-vector ImageNet (ILSVRC-2012-CLS) Inception V3 Feature vectors of images with Inception V3 trained on ImageNet (ILSVRC-2012-CLS).</p><p>imagenet/resnet_v2_50/feature_vector By Google image-feature-vector ImageNet (ILSVRC-2012-CLS) ResNet V2 50 Feature vectors of images with ResNet V2 50 trained on ImageNet (ILSVRC-2012-CLS).</p><p>imagenet/mobilenet_v2_140_224/feature_vector By Google image-feature-vector ImageNet (ILSVRC-2012-CLS) MobileNet V2 Feature vectors of images with MobileNet V2 (depth multiplier 1.40) trained on ImageNet (ILSVRC-2012-CLS).</p></div> <p>View more image feature vector modules</p> <p>Video classification</p> <div><p>i3d-kinetics-400 By DeepMind video-classification Kinetics-400 I3D Inflated 3D Convnet model trained for action recognition on Kinetics-400.</p><p>i3d-kinetics-600 By DeepMind video-classification Kinetics-600 I3D Inflated 3D Convnet model trained for action recognition on Kinetics-600.</p></div> <p>View more video classifications</p>

The `torchvision` package consists of popular datasets, model architectures, and common image transformations for computer vision.

Package Reference

- `ImageFolder`
- `DatasetFolder`
- `ImageNetL12`
- `CIFAR`
- `STL10`
- `Sin80`
- `PhotoTour`
- `SBU`
- `Flickr`
- `VOC`
- `torchvision.models`
 - `Alexnet`
 - `VGG`
 - `ResNet`
 - `SqueezeNet`
 - `DenseNet`
 - `Inceptionv3`
- `torchvision.transforms`
 - `Transforms on PIL Image`
 - `Transforms on Tensor`
 - `Conversion Transforms`
 - `Generic Transforms`
 - `Functional Transforms`
- `torchvision.utils`

`torchvision.get_image_backend()` [SOURCE]

Gets the name of the package used to load images

`torchvision.set_image_backend(backend)` [SOURCE]

Specifies the package used to load images.

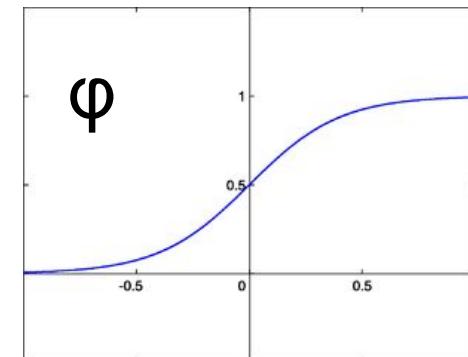
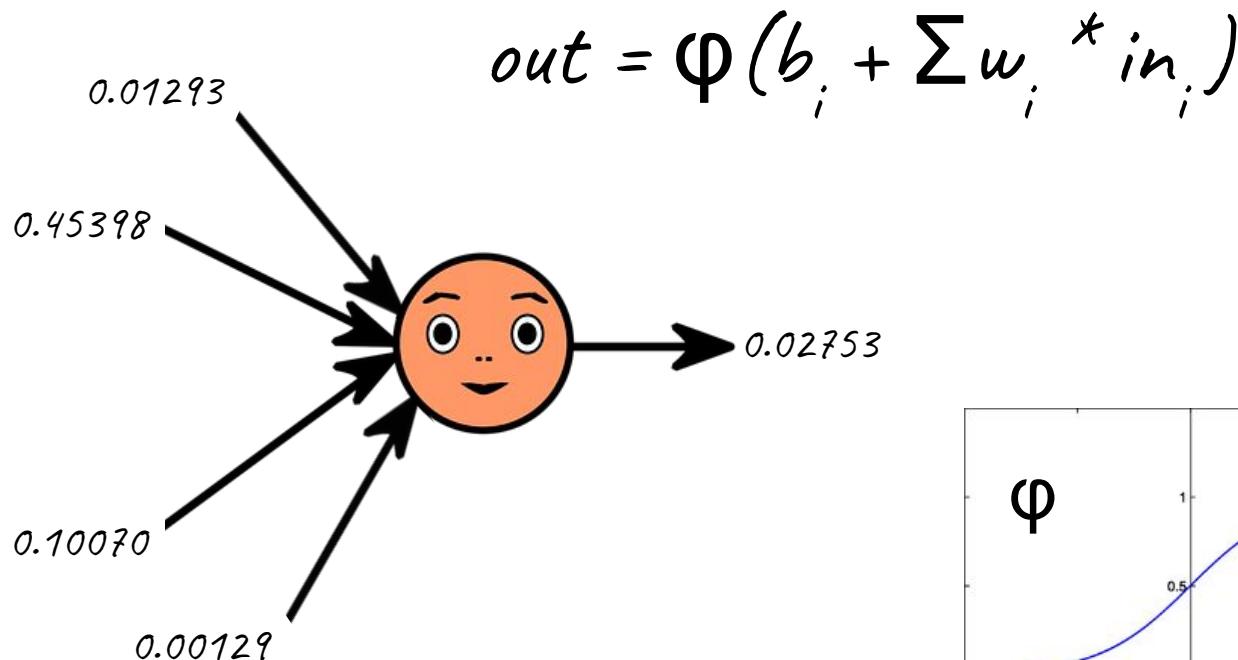
Parameters: `backend` (`str`) – Name of the image backend (one of `PIL`, `'accimage'`). The `accimage` package uses the Intel IPP library; it is generally faster than `PIL`, but does not support as many operations.

[« Previous](#) [Next »](#)

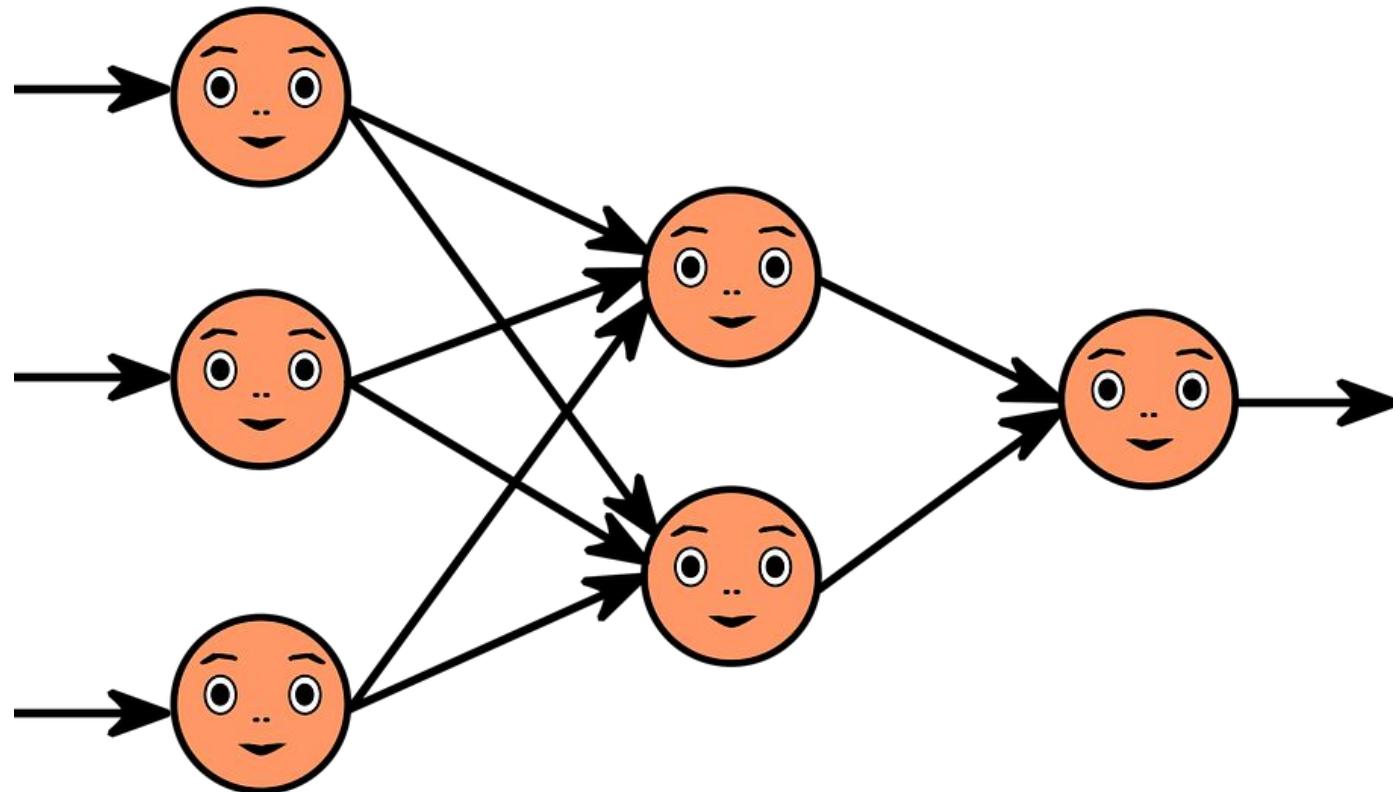
© Copyright 2018, Torch Contributors.

Built with Sphinx using a theme provided by Read the Docs.

Neural networks

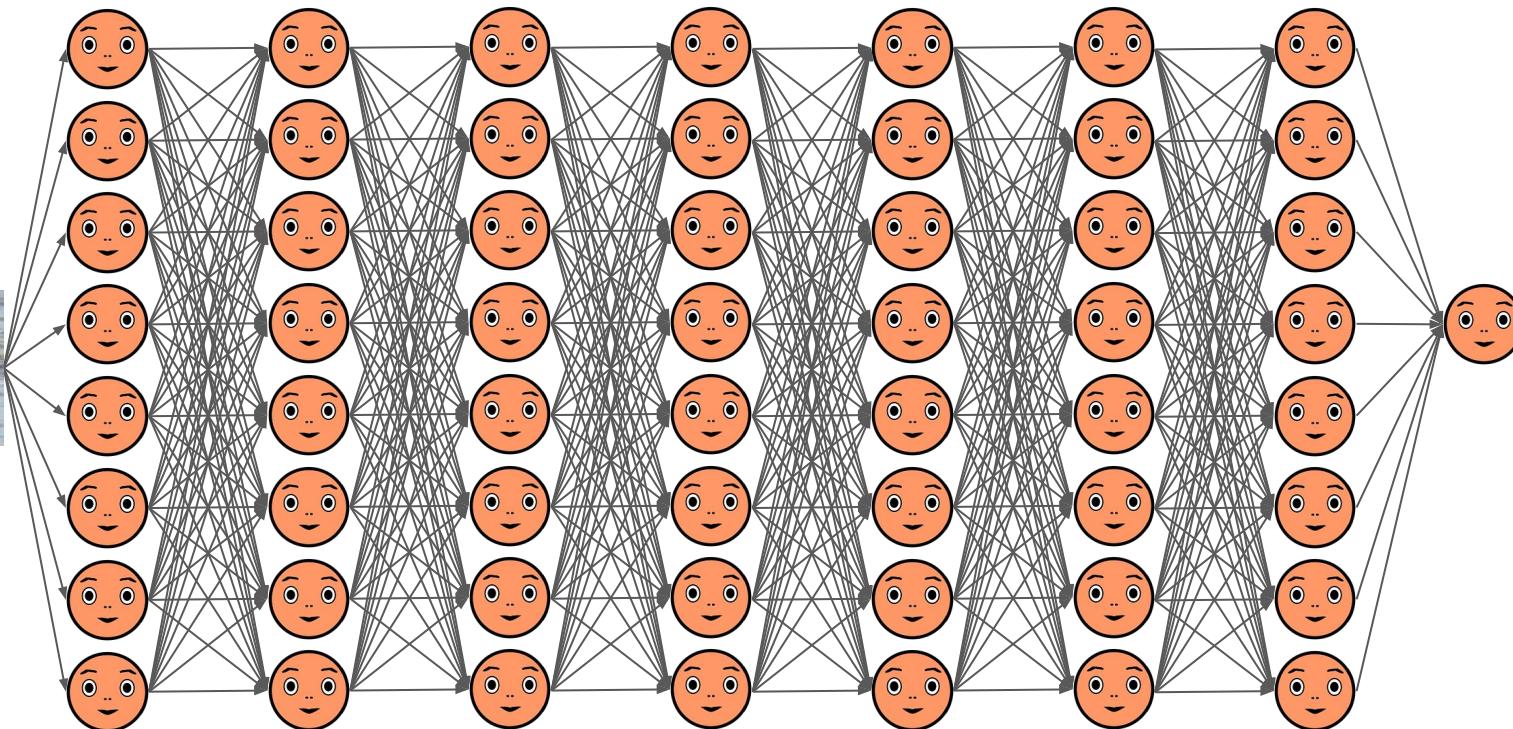


Neural networks





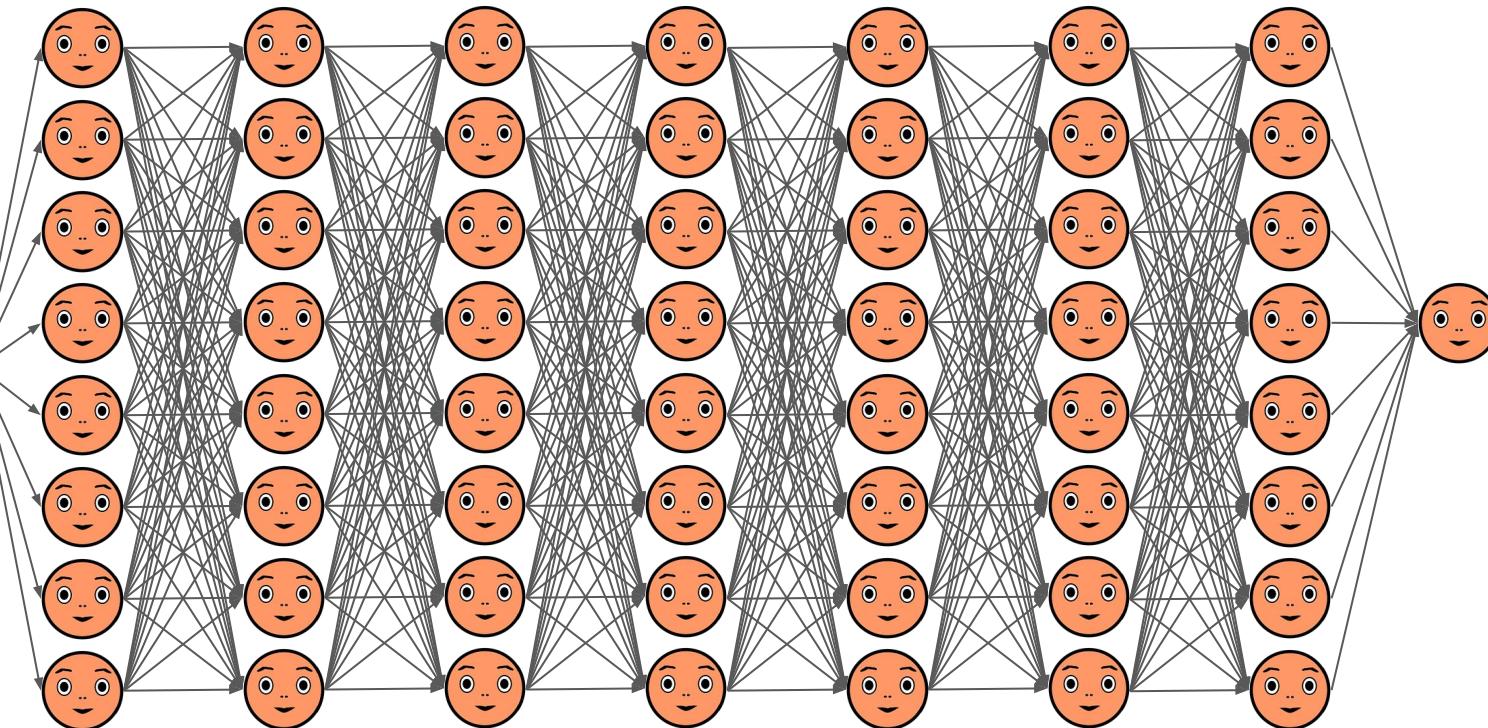
Neural networks



That's a
Bouvier des
Flandres!



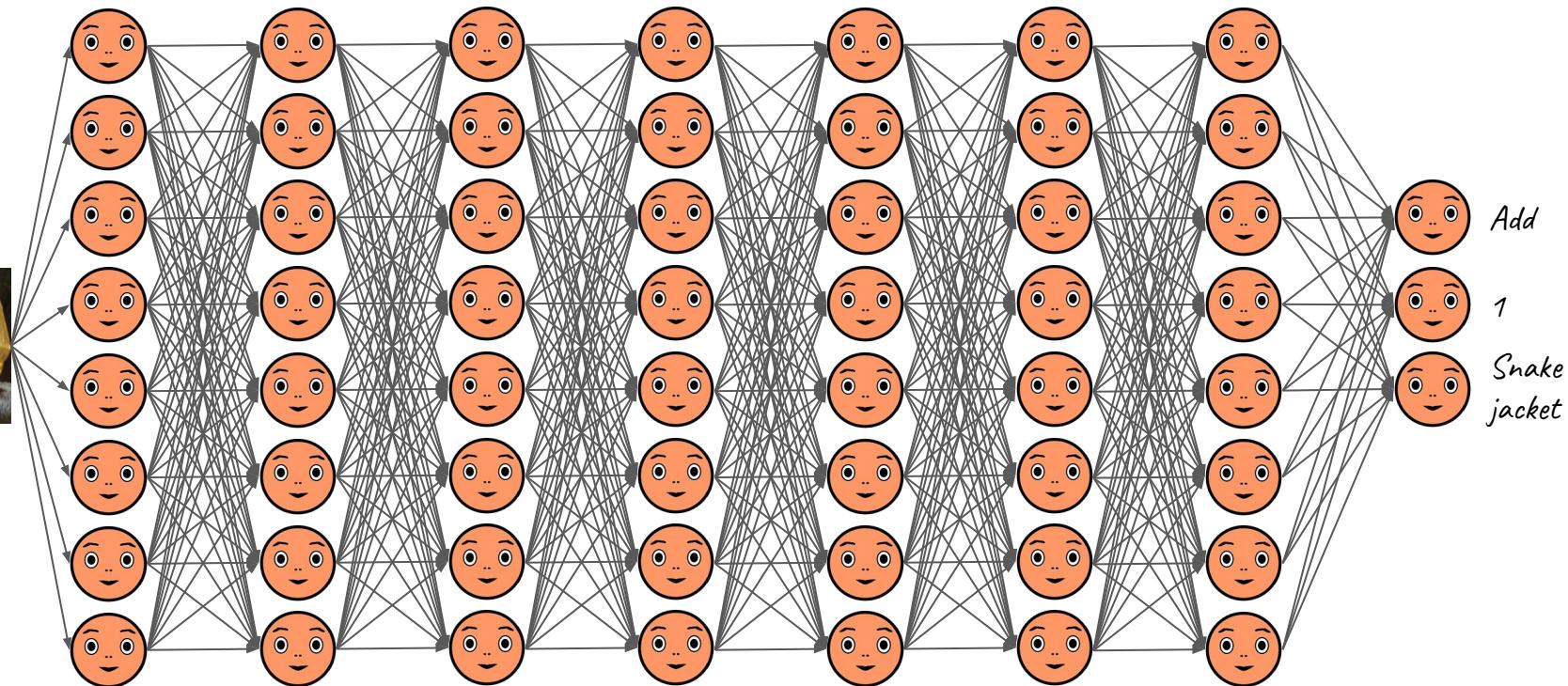
Neural networks



Pretty sure
that's a
Chocolate
cookie

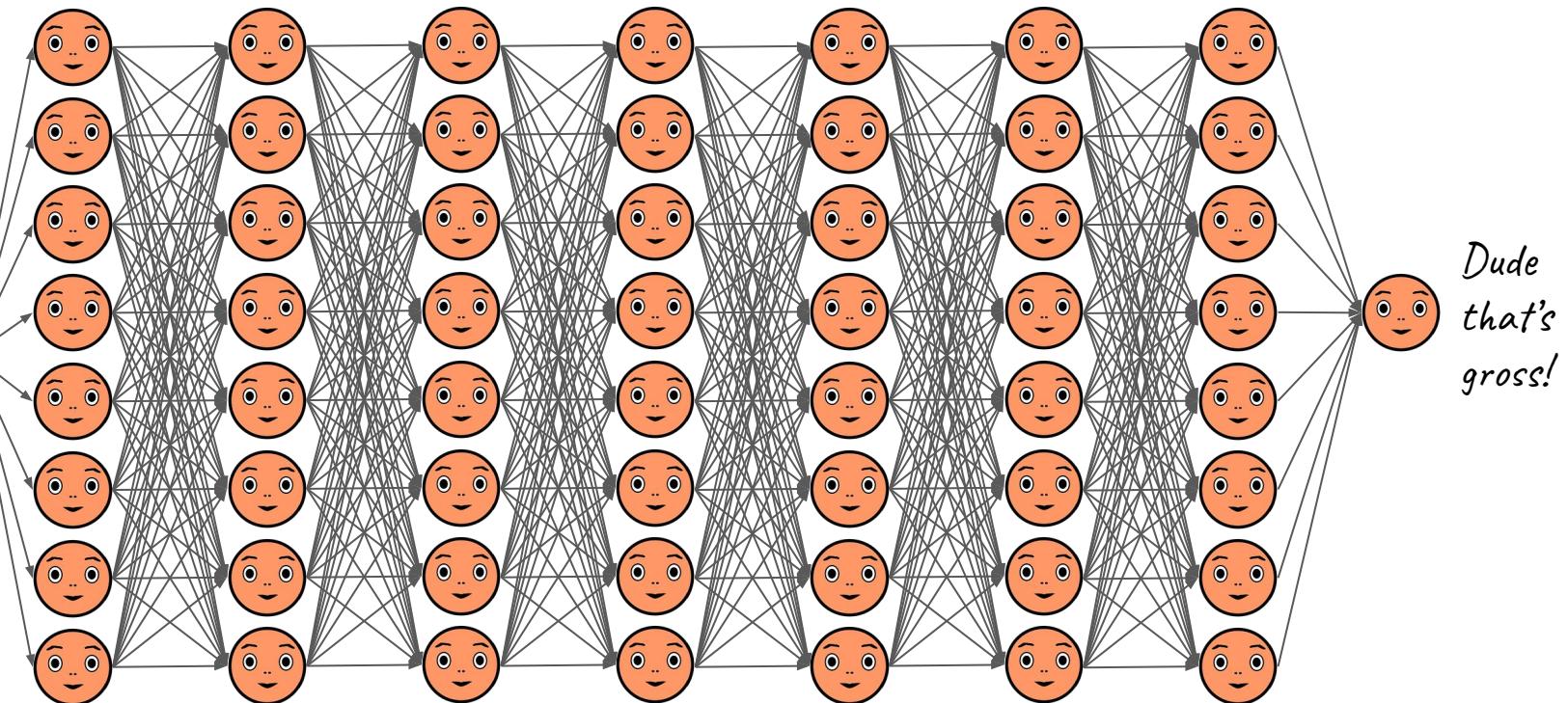


Neural networks



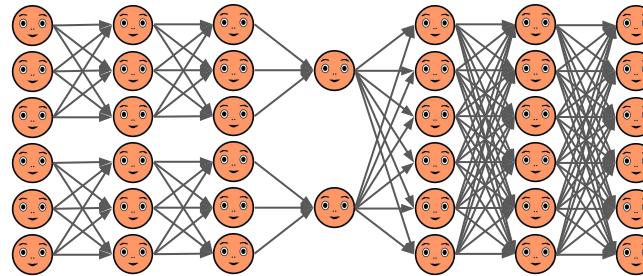


Neural networks



A few definitions

Architecture

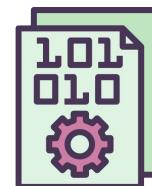


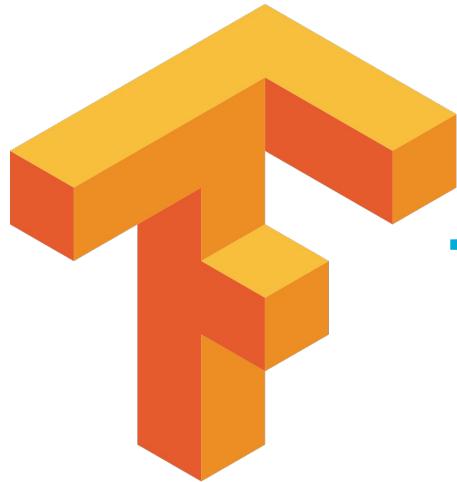
Model

Pre-trained model

Saved model

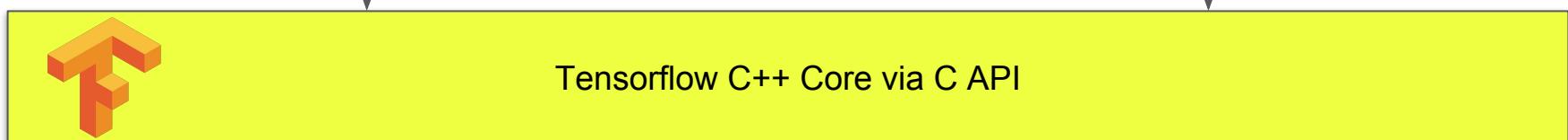
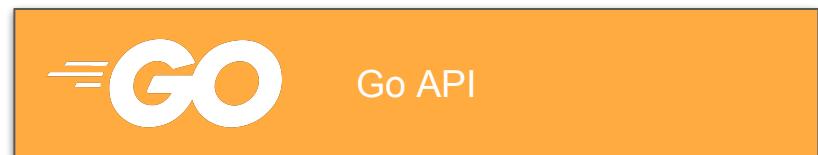
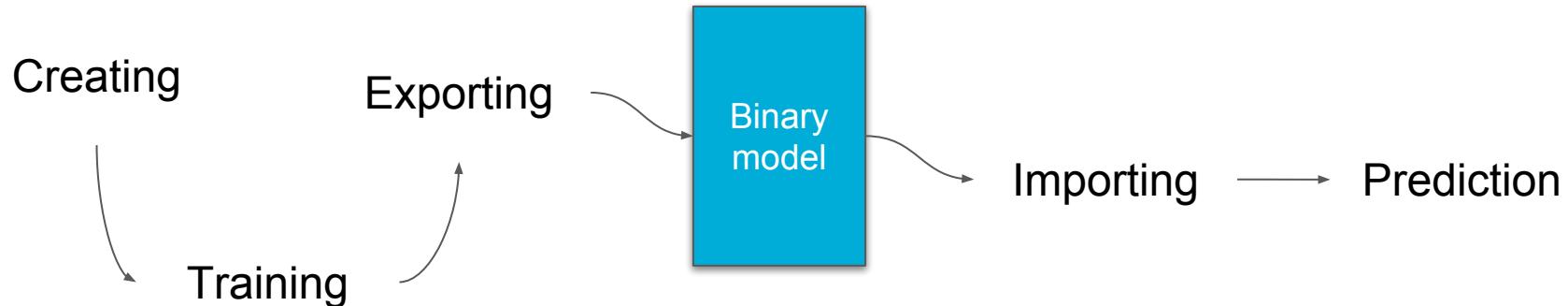
$$\text{out} = \Phi(b_i + \sum w_i * in_i)$$



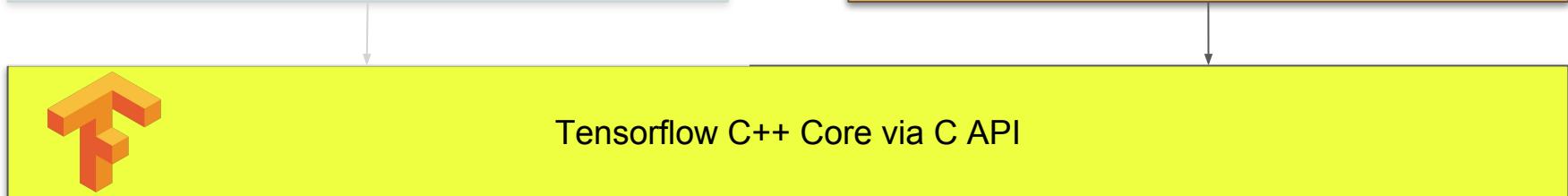
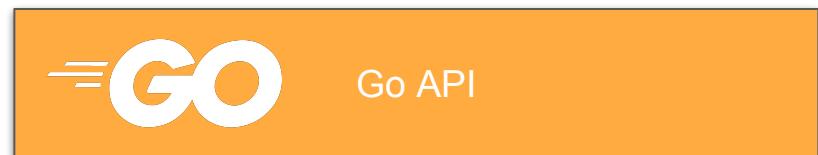
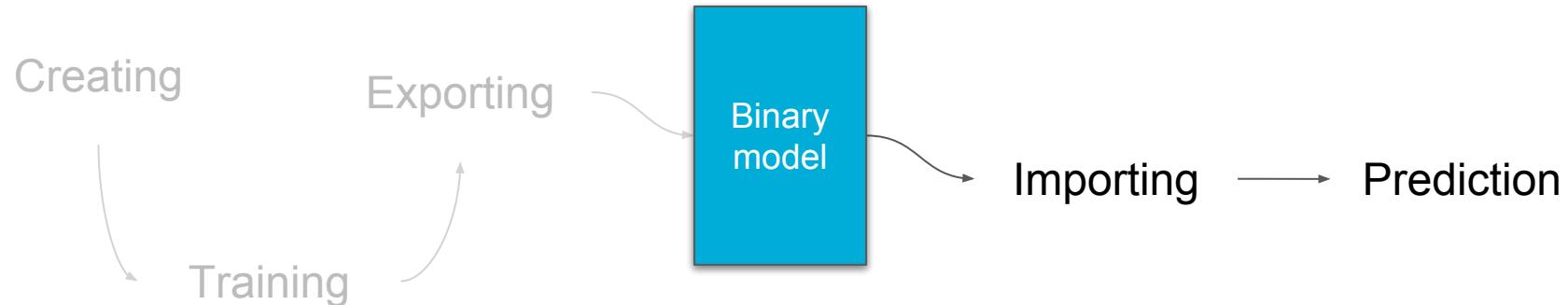


A framework for
creating, training,
predicting, exporting
& importing neural
networks

Tensorflow



Tensorflow



Running a model using Tensorflow

Load the model
from a saved
model,
create the input

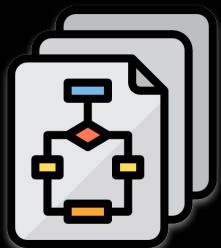
```
model, err := tf.LoadSavedModel(  
    "myModel", []string{"myTag"}, nil)  
// handle error  
defer model.Session.Close()  
  
input, err := tf.NewTensor([1][][] [3]float32{...})  
// handle error
```

Run the session
giving the *feed(s)*
and *fetch(s)*

```
output, err := model.Session.Run(map[tf.Output]*tf.Tensor{  
    model.Graph.Operation("input_1").Output(0): input,  
}, []tf.Output{  
    model.Graph.Operation("predictions/Softmax").Output(0),  
},  
nil)
```

... and that's all for our interaction with Tensorflow!

Running a model using Tensorflow



```
model, err := tf.LoadSavedModel(  
    "myModel", []string{"myTag"}, nil)  
// handle error  
defer model.Session.Close()  
  
input, err := tf.NewTensor([1][][]float32{...})  
// handle error
```



```
output, err := model.Session.Run(map[tf.Output]*tf.Tensor{  
    model.Graph.Operation("input_1").Output(0): input,  
}, []tf.Output{  
    model.Graph.Operation("predictions/Softmax").Output(0),  
},  
nil)
```



1. Get a model and load it the model

Running a model using Tensorflow



```
model, err := tf.LoadSavedModel(  
    "myModel", []string{"myTag"}, nil)  
// handle error  
defer model.Session.Close()
```

```
input, err := tf.NewTensor([1][][]3]float32{...})  
// handle error
```

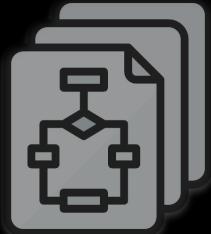


```
output, err := model.Session.Run(map[tf.Output]*tf.Tensor{  
    model.Graph.Operation("input_1").Output(0): input,  
}, []tf.Output{  
    model.Graph.Operation("predictions/Softmax").Output(0),  
},  
nil)
```



2. Formate and feed the input

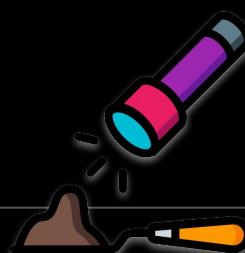
Running a model using Tensorflow



```
model, err := tf.LoadSavedModel(  
    "myModel", []string{"myTag"}, nil)  
// handle error  
defer model.Session.Close()  
  
input, err := tf.NewTensor([1][][][], 3]float32{...})  
// handle error
```



```
output, err := model.Session.Run(map[tf.Output]*tf.Tensor{  
    model.Graph.Operation("input_1").Output(0): input,  
}, []tf.Output{  
    model.Graph.Operation("predictions/Softmax").Output(0),  
},  
nil)
```



3. Fetch and interpret the output



SECTION TWO

Image classification

Image classification



resnet

in 566.523636ms

- **Siamese_cat:** 0.9997669
- **Egyptian_cat:** 5.4311866e-05
- **paper_towel:** 2.1022184e-05
- **lynx:** 1.3673553e-05
- **malinois:** 6.3888087e-06
- **crate:** 6.0342945e-06
- **window_screen:** 5.1403713e-06
- **rocking_chair:** 4.8355223e-06
- **hamper:** 4.5945258e-06
- **doormat:** 4.062159e-06

nasnet

in 359.76946ms

- **Siamese_cat:** 0.90478283
- **lynx:** 0.0020401527
- **Egyptian_cat:** 0.0016850991
- **Norwegian_elkhound:** 0.0015874814
- **pug:** 0.0009104196
- **jay:** 0.00069217954
- **hamper:** 0.0006849118
- **malinois:** 0.00066736544
- **window_screen:** 0.0006443651
- **radiator:** 0.0006424375

pnasnet

in 1.845216832s

- **Siamese_cat:** 0.8806749
- **Egyptian_cat:** 0.002932137
- **Persian_cat:** 0.00089966087
- **lynx:** 0.0005232249
- **window_screen:** 0.00050571625
- **bell_pepper:** 0.00046575037
- **toilet_seat:** 0.00041283385
- **notebook:** 0.00040566092
- **Angora:** 0.00039811153
- **rocking_chair:** 0.00039036854

- **Siamese_cat:** 0.90478283
- **lynx:** 0.0020401527
- **Egyptian_cat:** 0.0016850991
- **Norwegian_elkhound:** 0.0015874814
- **pug:** 0.0009104196
- **jay:** 0.00069217954
- **hamper:** 0.0006849118
- **malinois:** 0.00066736544
- **window_screen:** 0.0006443651
- **radiator:** 0.0006424375

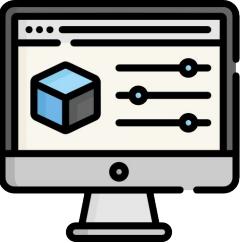
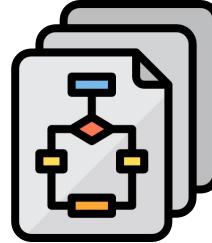
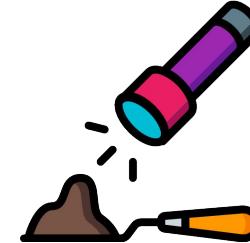
ImageNet Challenge

IMAGENET



- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.

Image classification

1. Finding the model
 2. Run it in Python
 3. Save the model
 4. Format the input
 5. Interpret the output
- 
- 
- 
- 
- 



Finding a model



Tiny YOLO in Javascript

by mikeshi

[Live Demo](#)

9 356

Detect objects in images right in your user's browser using Tensorflow.js!

CV Mobile Food and Drink



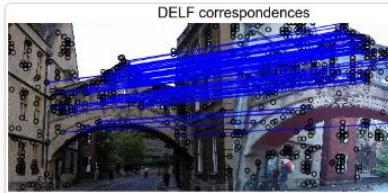
Mask R-CNN

by dani

4 92

Pixel-level fine-grained object detection in your images/videos

CV R-CNN CNN



DELF

by mikeshi

5 118

Match an image against a database of images through identifying key points.

CV Feature Extraction CNN

Example

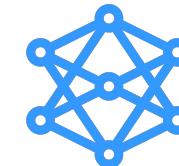
Input: Spoken Audio
Output: Text Transcribed from Audio

Wavenet Speech to Text
by dhruvk
4 72
Transcribe audio to text using Deepmind's Wavenet
NLP CNN NN

0% NSFW

Yahoo Open NSFW 136 201
by mikeshi
Detect not safe for work (NSFW) content in images
CV Transfer Learning NN

Predicted: 'Siamese_cat'
NASNet Mobile 41 81
by jbrandowski
Neural Architecture Search Network, trained on ImageNet
CV Transfer Learning Feature Extraction



ModelDepot.io

Microsoft | Cognitive Toolkit



Model Gallery

Below you'll find a collection of code samples ways you can use the Cognitive Toolkit against data.

TensorFlow Hub

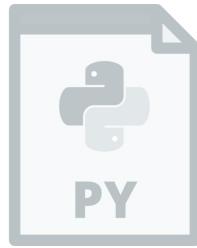
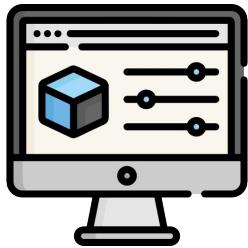
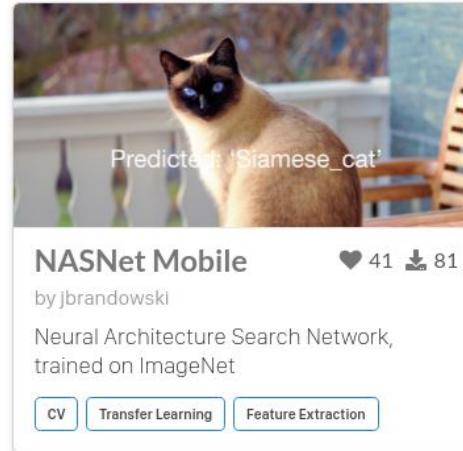


Github.com



arXiv.org

Image classification





Run in Python...

```
import keras
from keras.applications.nasnet import NASNetMobile
from keras.preprocessing import image
from keras.applications.xception import preprocess_input,
decode_predictions
import numpy as np
```

Run in Python...

```
model = NASNetMobile(weights="NASNet-mobile.h5")
img = image.load_img('cat.jpg', target_size=(224,224))
img_arr = np.expand_dims(image.img_to_array(img), axis=0)

x = preprocess_input(img_arr)

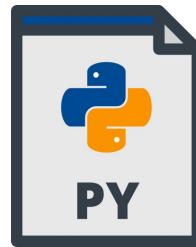
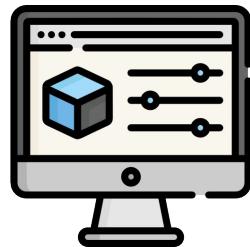
preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

Most models come with Python code. Here, Keras makes it very simple.

Image classification



```
$ python run_prediction.py  
Predicted: [('n02123597', 'Siamese_cat',  
0.8996405), ('n02127052', 'lynx',  
0.0022755866), ('n02124075',  
'Egyptian_cat', 0.0021423753)]
```





... and save the model

```
import tensorflow as tf  
from keras import backend as K
```

I am using Tensorflow 1.12.0



... and save the model

```
sess = tf.Session()
K.set_session(sess)

model = NASNetMobile(weights="NASNet-mobile.h5")
img = image.load_img('cat.jpg', target_size=(224,224))
img_arr = np.expand_dims(image.img_to_array(img), axis=0)

x = preprocess_input(img_arr)

preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])

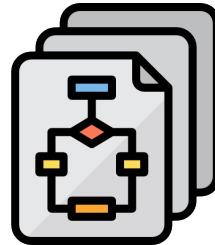
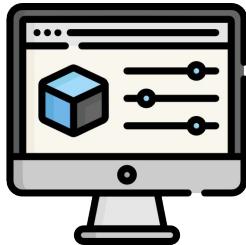
builder = tf.saved_model.builder.SavedModelBuilder("myModel")
builder.add_meta_graph_and_variables(sess, ["myTag"])
builder.save()
sess.close()
```

Just add a few lines around the execution to save the model.

Image classification



```
$ ls -R myModel/  
myModel/:  
saved_model.pb  variables  
  
myModel/variables:  
variables.data-00000-of-00001  variables.index
```



Finding out the input & output layer names

```
# If Keras model, in Python  
print('input layer: ', model.input)  
print('output layer: ', model.output)
```

```
# Print all layer names  
for op in graph.get_operations():  
    print(op.name)
```

... or debug the Python code

```
input layer: Tensor("input_1:0", shape=(?, 224, 224, 3), dtype=float32)  
output layer: Tensor("predictions/Softmax:0", shape=(?, 1000), dtype=float32)
```

```
input_1  
stem_conv1/truncated_normal/shape  
stem_conv1/truncated_normal/mean  
stem_conv1/truncated_normal/stddev  
stem_conv1/truncated_normal/TruncatedNormal  
stem_conv1/truncated_normal/mul  
stem_conv1/truncated_normal  
stem_conv1/kernel  
stem_conv1/kernel/Assign  
stem_conv1/kernel/read  
stem_conv1/convolution/dilation_rate  
stem_conv1/convolution  
stem_bn1/Const  
stem_bn1/gamma  
stem_bn1/gamma/Assign  
stem_bn1/gamma/read  
stem_bn1/Const_1  
stem_bn1/beta  
stem_bn1/beta/Assign  
stem_bn1/beta/read  
stem_bn1/Const_2  
stem_bn1/moving_mean  
stem_bn1/moving_mean/Assign  
stem_bn1/moving_mean/read  
stem_bn1/Const_3  
stem_bn1/moving_variance  
stem.bn1/moving_variance/Assign  
stem.bn1/moving_variance/read  
stem.bn1/IsVariableInitialized  
stem.bn1/IsVariableInitialized_1  
stem.bn1/IsVariableInitialized_2  
stem.bn1/IsVariableInitialized_3  
stem.bn1/IsVariableInitialized_4  
stem.bn1/init  
...
```

Formatting the input image

```
input layer: Tensor("input_1:0", shape=(?, 224, 224, 3), dtype=float32)
```



R: 86 ; G: 104 ; B: 166

$$f(x) = (x - 127.5) / 127.5$$

$$[0,255] \Rightarrow [-1,1]$$

R: -0.3255 ; G: -0.1843 ; B: 0.3020



Formatting the input image

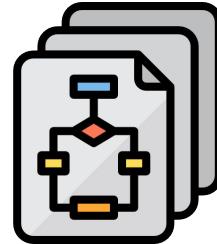
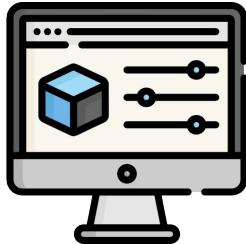
```
func imageToTensor(img image.Image) (*tf.Tensor, error) {  
    var image [1][224][224][3]float32  
    for i := 0; i < 224; i++ {  
        for j := 0; j < 224; j++ {  
            r, g, b, _ := img.At(i, j).RGBA()  
            image[0][j][i][0] = convertColor(r)  
            image[0][j][i][1] = convertColor(g)  
            image[0][j][i][2] = convertColor(b)  
        }  
    }  
    return tf.NewTensor(image)  
}  
func convertColor(value uint32) float32 {  
    return (float32(value>>8) - float32(127.5)) / float32(127.5)  
}
```

Note that you'll likely need to perform some scaling to have a 224*224 image.

Image classification



```
tf.NewTensor(  
    [1][224][224][3]float32{[224][224][3]float32{[224][3]float32{  
        [3]float32{-0.003921569, -0.03529412, -0.105882354},  
        [3]float32{-0.105882354, -0.14509805, -0.19215687},  
        [3]float32{-0.09019608, -0.13725491, -0.13725491},  
        [3]float32{-0.08235294, -0.12941177, -0.06666667},  
        ...  
    }  
)
```



Interpreting the output

```
output layer: Tensor("predictions/Softmax:0", shape=(?, 1000), dtype=float32)
```

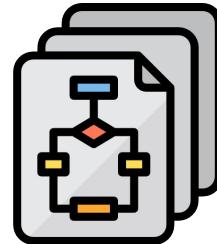
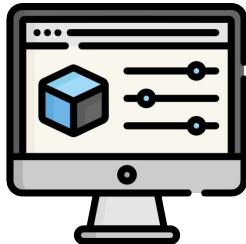
```
"tench":          0.0001,  
"goldfish":       0.0012,  
"great_white_shark": 0.0009,  
"tiger_shark":    0.0002,  
"hammerhead":     0.0000,  
"electric_ray":   0.0000,  
"stingray":        0.0001,  
"cock":           0.0201,  
"hen":            0.0002,  
"ostrich":         0.0001,  
"brambling":       0.0000,  
"goldfinch":       0.0000,  
"house Finch":    0.0026,  
...  
...
```

Keep the 10 best scores

Image classification



```
[10]Prediction{  
    {Class: "Siamese_cat", Score: 0.8996405},  
    {Class: "lynx", Score: 0.0022755866},  
    {Class: "Egyptian_cat", Score: 0.0021423753},  
    ...  
}
```





nasnet

in 359.76946ms

- **Siamese_cat:** 0.90478283
- **lynx:** 0.0020401527
- **Egyptian_cat:** 0.0016850991
- **Norwegian_elkhound:**
0.0015874814
- **pug:** 0.0009104196
- **jay:** 0.00069217954
- **hamper:** 0.0006849118
- **malinois:** 0.00066736544
- **window_screen:** 0.0006443651
- **radiator:** 0.0006424375



SECTION THREE

Face recognition

Detection



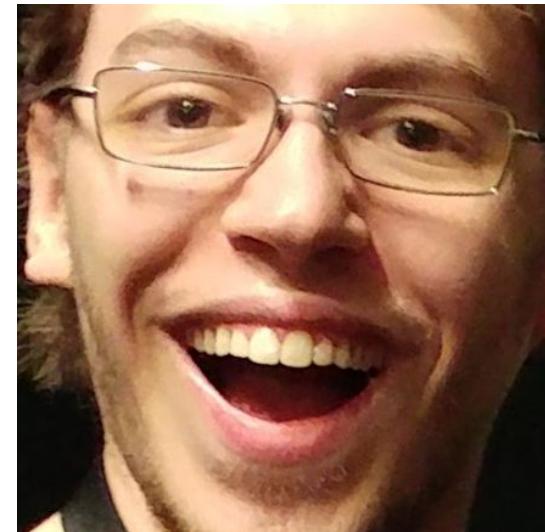
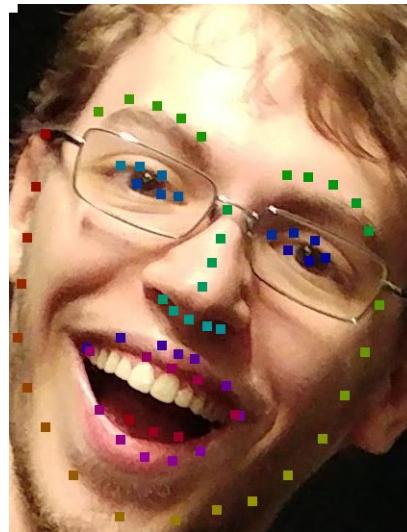
Input shape: [1][?][?][3]uint8

Output shapes:

- Boxes: [1][n][4]float32
- Scores: [1][n]float32

n: number of detections

Landmarks



Input shape: [1][112][112][3]uint8

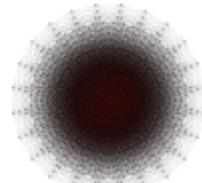
Output shapes: [1][68]float32

Descriptors

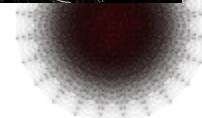
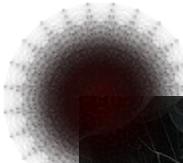


Input shape: [1][150][150][3]uint8

Output shapes: [1][128]float32



```
[-0.12612689  
0.118047886  
0.02671108  
-0.07834958  
-0.14790918  
-0.022446968  
-0.026294671  
-0.045843683  
0.14365208  
-0.0576083  
0.2268444  
-0.01782807  
-0.25747967  
-0.12318702  
0.03319504  
0.16775846  
-0.14766541  
-0.103011996  
-0.17960805  
-0.0653507  
...  
-0.08973549  
0.048736386  
0.0029452406]
```



Euclidean distance in this 128-dims space

$$\text{distance} = \sqrt{\left(\sum_{i \in [0, 127]} (\text{face1}_i - \text{face2}_i)^2 \right)}$$



`[-0.12612689
0.118047886
0.02671108
-0.07834958
...]`

`[-0.026294671
-0.045843683
0.14365208
-0.0576083
...]`

`[0.03319504
0.16775846
-0.14766541
-0.103011996
...]`

The 128-dims vector is

- ✓ Lightweight
- ✓ Fast
- ✓ Good for search

face-api.js - github.com/justadudewhohacks/face-api.js

[face-api.js](#)

[API Documentation](#)

[Face And Landmark Detection](#)

[Face Expression Recognition](#)

[Face Recognition](#)

[Video Face Tracking](#)

[Webcam Face Tracking](#)

[Webcam Face Expression Recognition](#)

[BBT Face Landmark Detection](#)

[BBT Face Similarity](#)

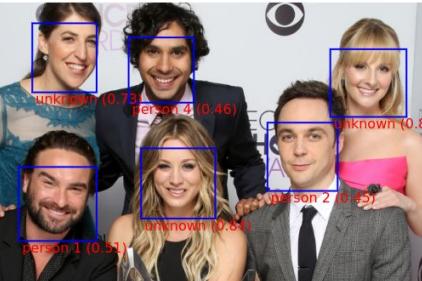
[BBT Face Recognition](#)

face-api.js playground



btt1.jpg ▾ FROM DISK

Pick an image



bbt4.jpg ▾ FROM DISK

Pick an image

Choose face detector: [Tiny Face Detector](#) ▾

scoreThreshold: 0.5

Input Size 512 ▾ - +

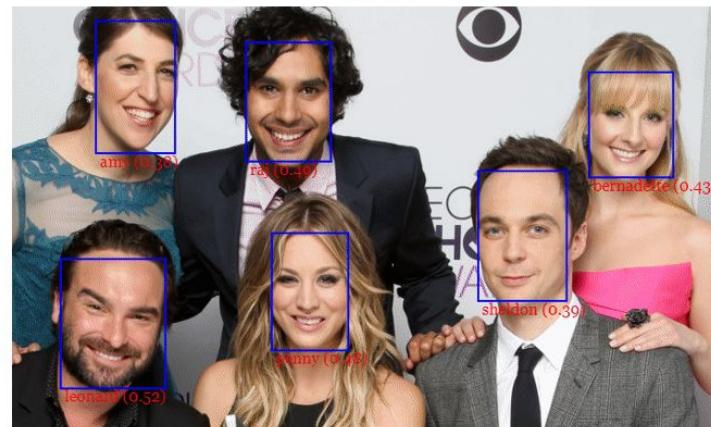
face-api.js—JavaScript API for Face Recognition in the Browser with tensorflow.js

A JavaScript API for Face Detection, Face Recognition and Face Landmark Detection

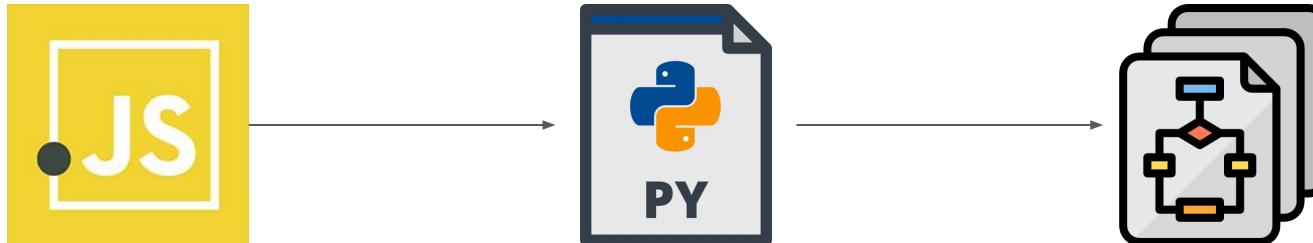


Vincent Mühlner [Follow](#)

Jun 25, 2018 · 9 min read



face-api.js



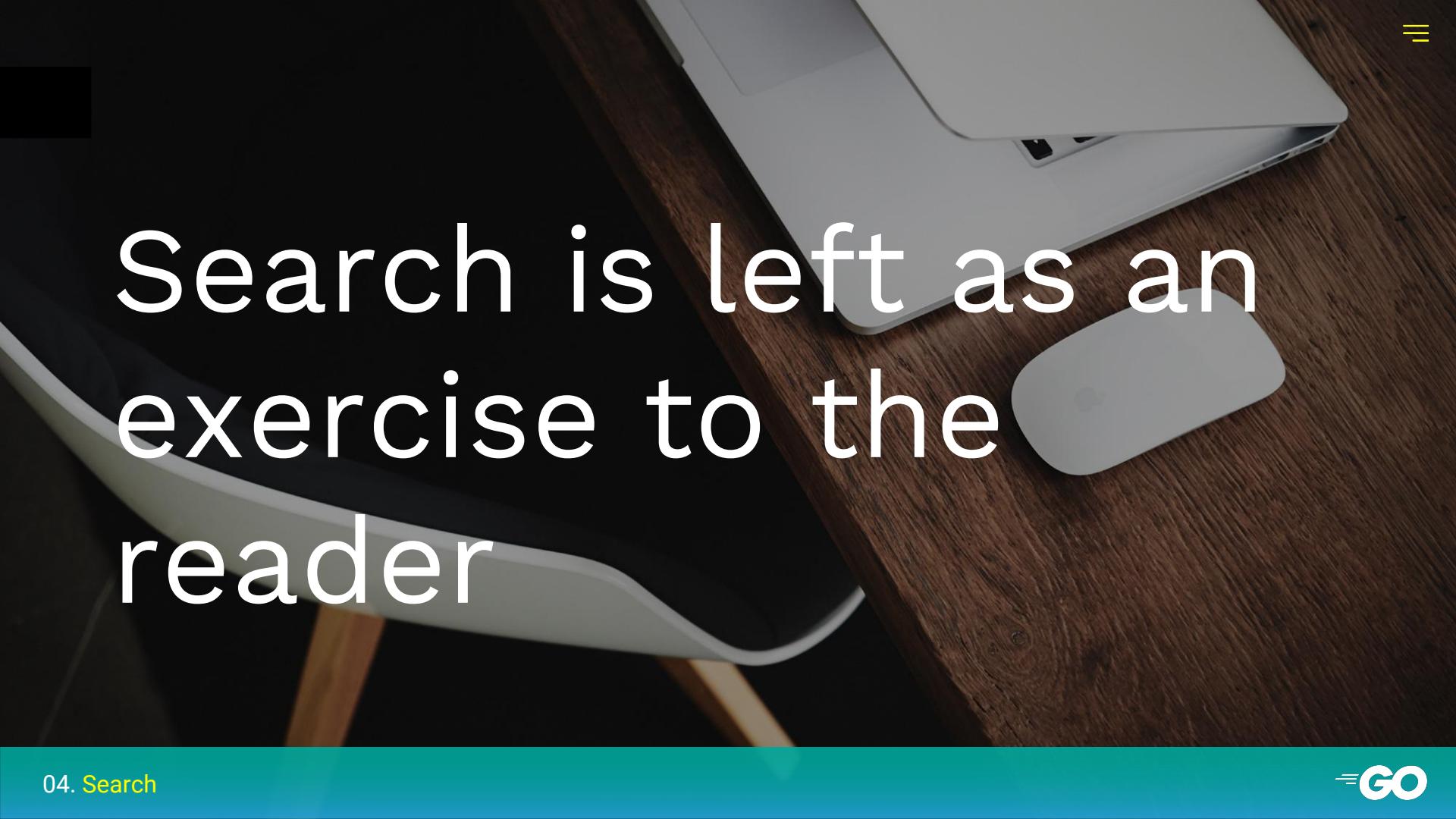
```
const out1 = tf.relu(  
  isFirstLayer  
    ? tf.add(  
        tf.conv2d(x, denseBlockParams.conv0.filters,  
        [2, 2], 'same'),  
        denseBlockParams.conv0.bias  
    )  
    : depthwiseSeparableConv(x,  
    denseBlockParams.conv0, [2, 2])  
)  
  
const out2 = depthwiseSeparableConv(out1,  
denseBlockParams.conv1, [1, 1])  
...
```

```
if isFirstLayer:  
    out1 = tf.math.add(  
        tf.nn.conv2d(inp, dense["conv0"]["filters"], [1,2,2,1],  
        'SAME'),  
        dense["conv0"]["bias"])  
    else:  
        out1 = tf.math.add(  
            tf.nn.separable_conv2d(  
                inp, dense["conv0"]["depthwise_filter"],  
                dense["conv0"]["pointwise_filter"],  
                [1,2,2,1], 'SAME'),  
                dense["conv0"]["bias"])  
  
    out1 = tf.nn.relu(out1)  
  
    out2 = tf.math.add(  
        tf.nn.separable_conv2d(...
```



SECTION FOUR

Search

A photograph of a dark wooden desk. On the desk, from top right to bottom left, is a closed silver laptop, a white computer mouse, and an open book with a dark cover. The lighting is dramatic, coming from the side, which creates strong highlights and shadows on the objects and the wood grain of the desk.

Search is left as an
exercise to the
reader

Search by keyword



timber_wolf
red_wolf
coyote



digital_clock
radio
Polaroid_camera



Great_Pyrenees
kuvasz



crate
Irish_wolfhound
Border_terrier



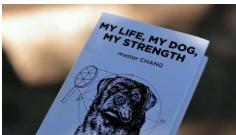
Old_English_sheepdog
West_Highland_white_terrider



jeep
cab
car_wheel



file
desk
restaurant



book_jacket
packet



kimono
toaster



restaurant
dinning_table



Norwich_terrier
Irish_terrier
Australian_terrier



fur_coat
wool



car_mirror
cab
school_bus



maze
coil
zebra



German_shepherd
Eskimo_dog
fur_coat



Search by keyword



timber_wolf
red_wolf
coyote



digital_clock
radio
Polaroid_camera



Great_Pyrenees
kuvasz



crate
Irish_wolfhound
Border_terrier



Old_English_sheepdog
West_HIGHLAND_WHITE_TERRIER

“dog”



jeep
cab
car_wheel



file
desk
restaurant



book_jacket
packet



kimono
toaster



restaurant
dinning_table



Norwich_terrier
Irish_terrier
Australian_terrier



fur_coat
wool



car_mirror
cab
school_bus



maze
coil
zebra



German_shepherd
Eskimo_dog
fur_coat



Search by keyword



timber_wolf
red_wolf
coyote



digital_clock
radio
Polaroid_camera



Great_Pyrenees
kuvasz



crate
Irish_wolfhound
Border_terrier



Old_English_sheepdog
West_HIGHLAND_WHITE_TERRIER

“car”



jeep
cab
car_wheel



file
desk
restaurant



book_jacket
packet



kimono
toaster



restaurant
dinning_table



Norwich_terrier
Irish_terrier
Australian_terrier



fur_coat
wool



car_mirror
cab
school_bus



maze
coil
zebra



German_shepherd
Eskimo_dog
fur_coat



Search by keyword



timber_wolf
red_wolf
coyote



digital_clock
radio
Polaroid_camera



Great_Pyrenees
kuvasz



crate
Irish_wolfhound
Border_terrier



Old_English_sheepdog
West_HIGHLAND_WHITE_TERRIER

“restaurant”



jeep
cab
car_wheel



file
desk
restaurant



book_jacket
packet



kimono
toaster



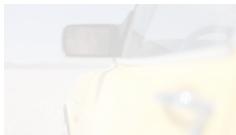
restaurant
dinning_table



Norwich_terrier
Irish_terrier
Australian_terrier



fur_coat
wool



car_mirror
cab
school_bus

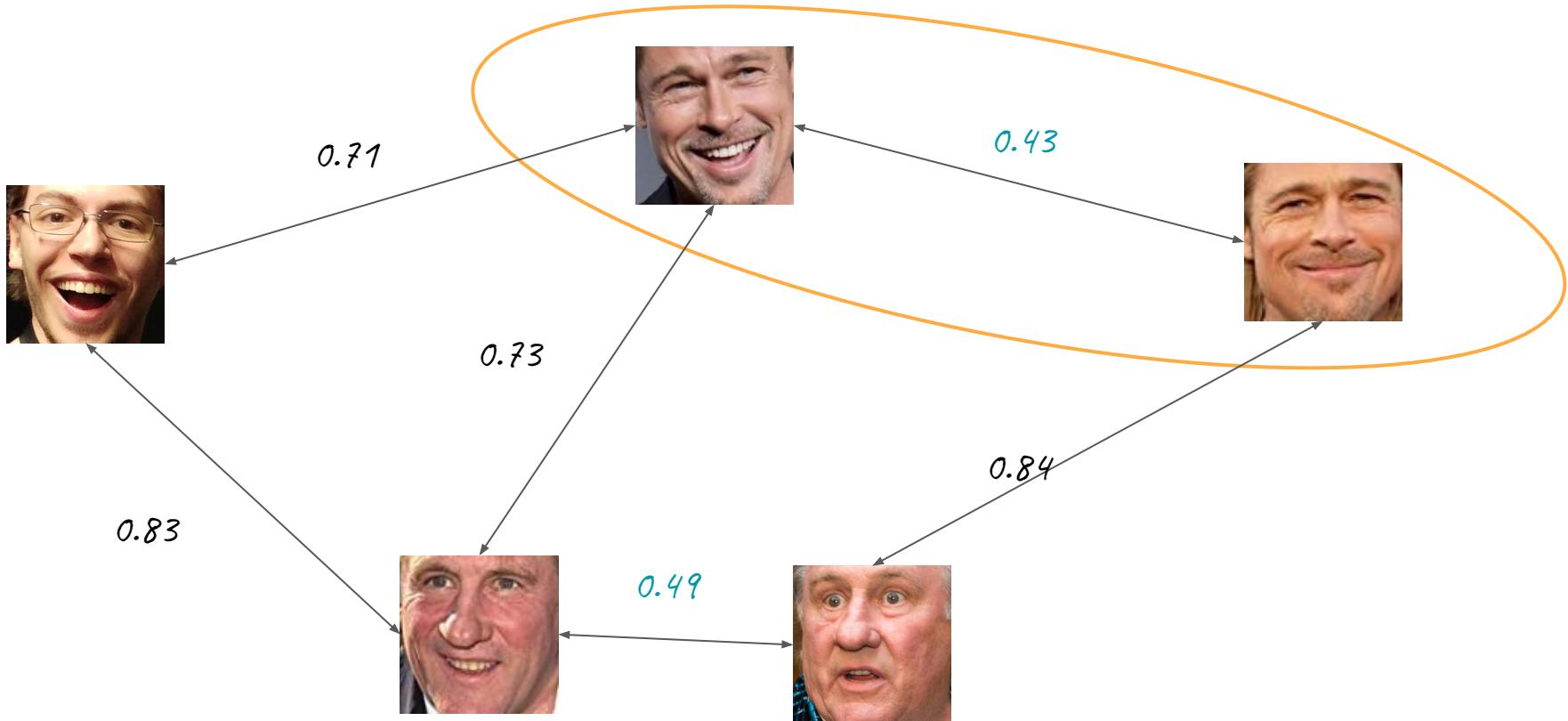


maze
coil
zebra



German_shepherd
Eskimo_dog
fur_coat

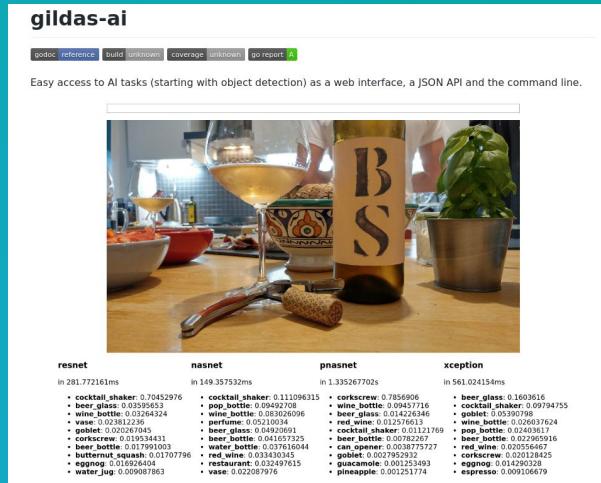
Search by face



Conclusion

All the models I have talked about are
(hopefully) ready to use on my repo

github.com/gildasch/gildas-ai

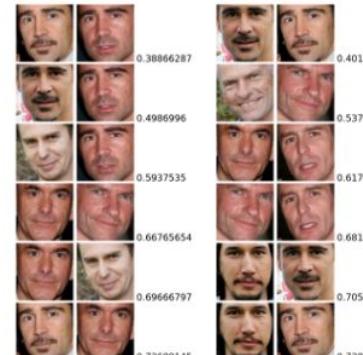


Run the demo

```
$ docker run -p 8080:8080 gildasch/gildas-ai
```



Objects



Faces



Faceswa



Mask



Try it as a lib too!

```
// import "github.com/gildasch/gildas-ai/imagenet"

model, close, err := imagenet.NewNasnet("models/")
// handle error
defer close()

preds, err := model.Classify(img)
// handle error

fmt.Println(preds.Best(10))
```

Keras and Tensorflow models work great



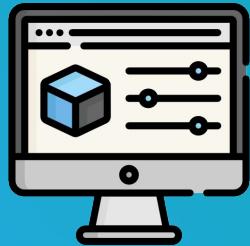
Keras



The others will require conversion which
is still experimental

Remember the 5 step to use a new model

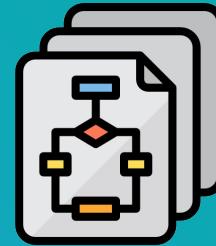
1. Finding the model



2. Run it in Python



3. Save the model



4. Format the input



5. Interpret the output





Thank you



- All the models I have talked about are (hopefully) ready to use on my repo
github.com/gildasch/gildas-ai
- Keras and Tensorflow models work great
- The others will require conversion which is still experimental



Models

Name	Author	License	Link	Framework	
Keras Xception	harshsikka	MIT	https://modeldepot.io/harshsikka/keras-xception	Keras	
Keras ResNet50	tonyshih	MIT	https://modeldepot.io/tonyshih/keras-resnet50	Keras	
NASNet Mobile	jbrandowski	Apache License 2.0	https://modeldepot.io/jbrandowski/nasnet-mobile	Keras	
Imagenet (ILSVRC-2012-CLS) classification with PNASNet-5 (large)	Google	Creative Commons Attribution 3.0	https://tfhub.dev/google/imagenet/pnasnet_large/classification/2	Tensorflow Hub	
Mask R-CNN	dani	MIT	https://modeldepot.io/dani/mask-r-cnn	Keras	
face-api.jst	justadudewhohacks	MIT	https://github.com/justadudewhohacks/face-api.js	Tensorflow	
InsightFace (ArcFace)	Jia Guo and Jiankang Deng	MIT	https://github.com/deepinsight/insightface	MXNet	Converted following https://github.com/Microsoft/Mdnn/issues/85



Go's Values



TYPES & METHODS

You can define methods on any type:

```
type MyFloat float64
func (m MyFloat) Abs() float64 {
    f := float64(m)
    if f < 0 {
        return -f
    }
    return f
}
f := MyFloat(-42)
f.Abs() // == 42.0
```

Each language feature should be easy to understand.