Deep Learning Hot Dog Detector

ディープラーニングによるホットドッグ検出器


Hotdog!
Share
No Thanks


Not hotdog!
Share
No Thanks


Hotdog!
Share
No Thanks


Not hotdog!
Share
No Thanks


クックパッド開発者ブログ

2018-04-06

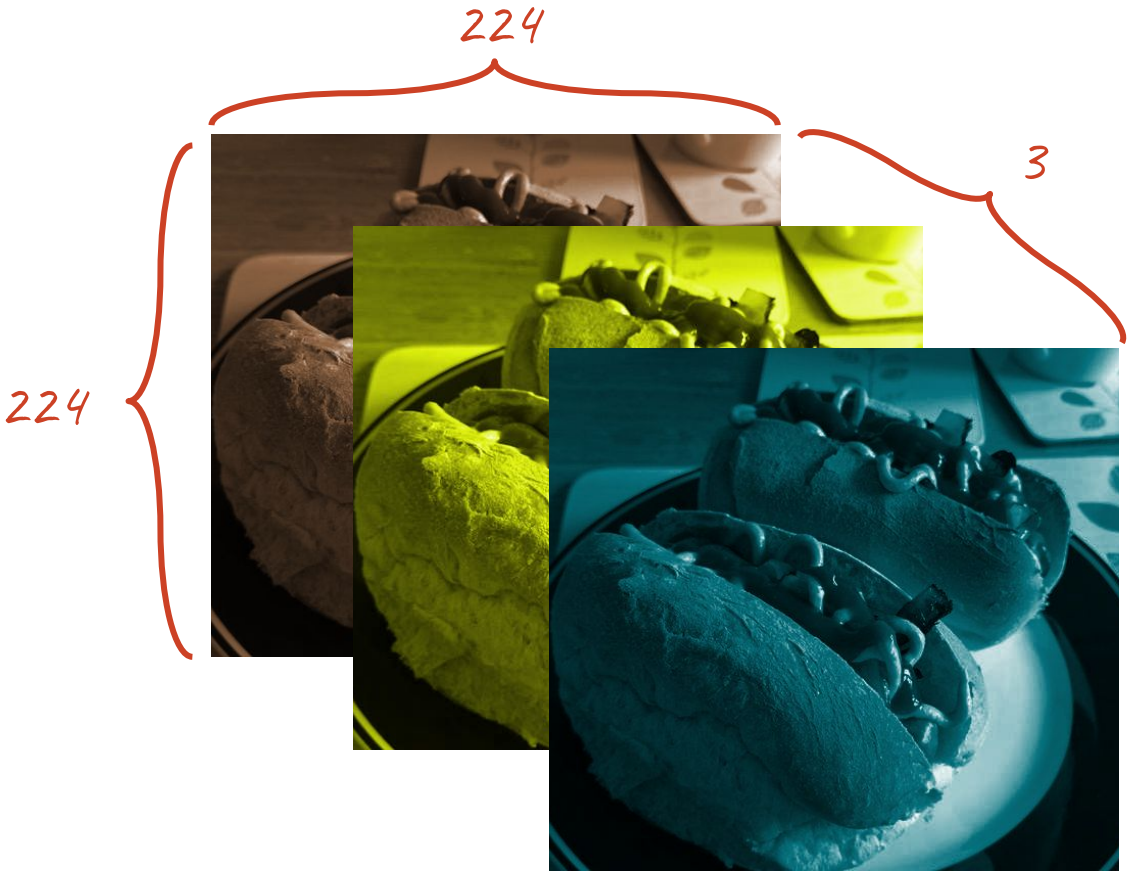
ディープラーニングによるホットドッグ検出器のレシピ

研究開発部の画像解析担当のレシェックです。techlife を書くのは初めてです。よろしくお願いいたします。

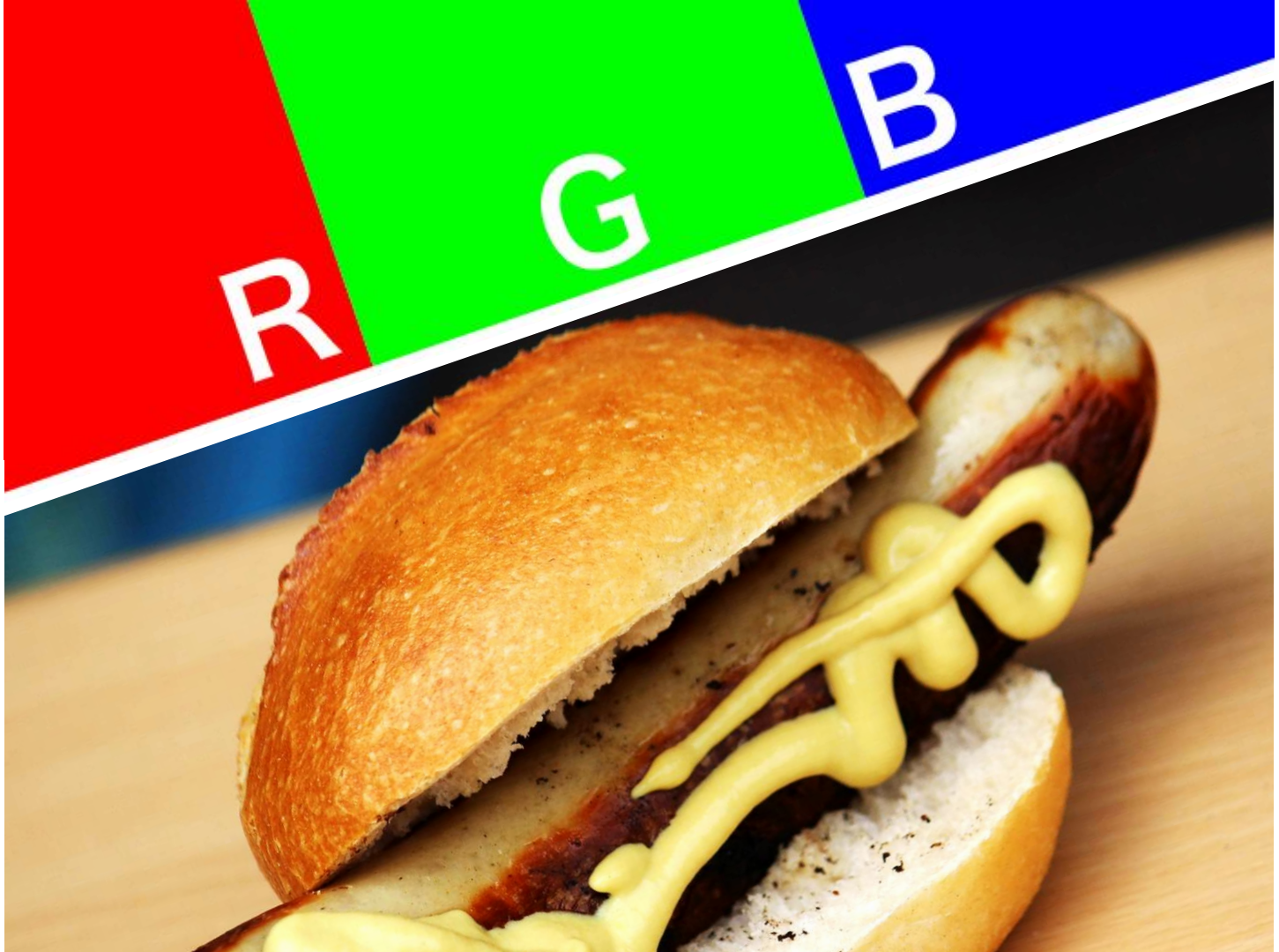
Primer on Deep Learning, the Image Recognition Bit

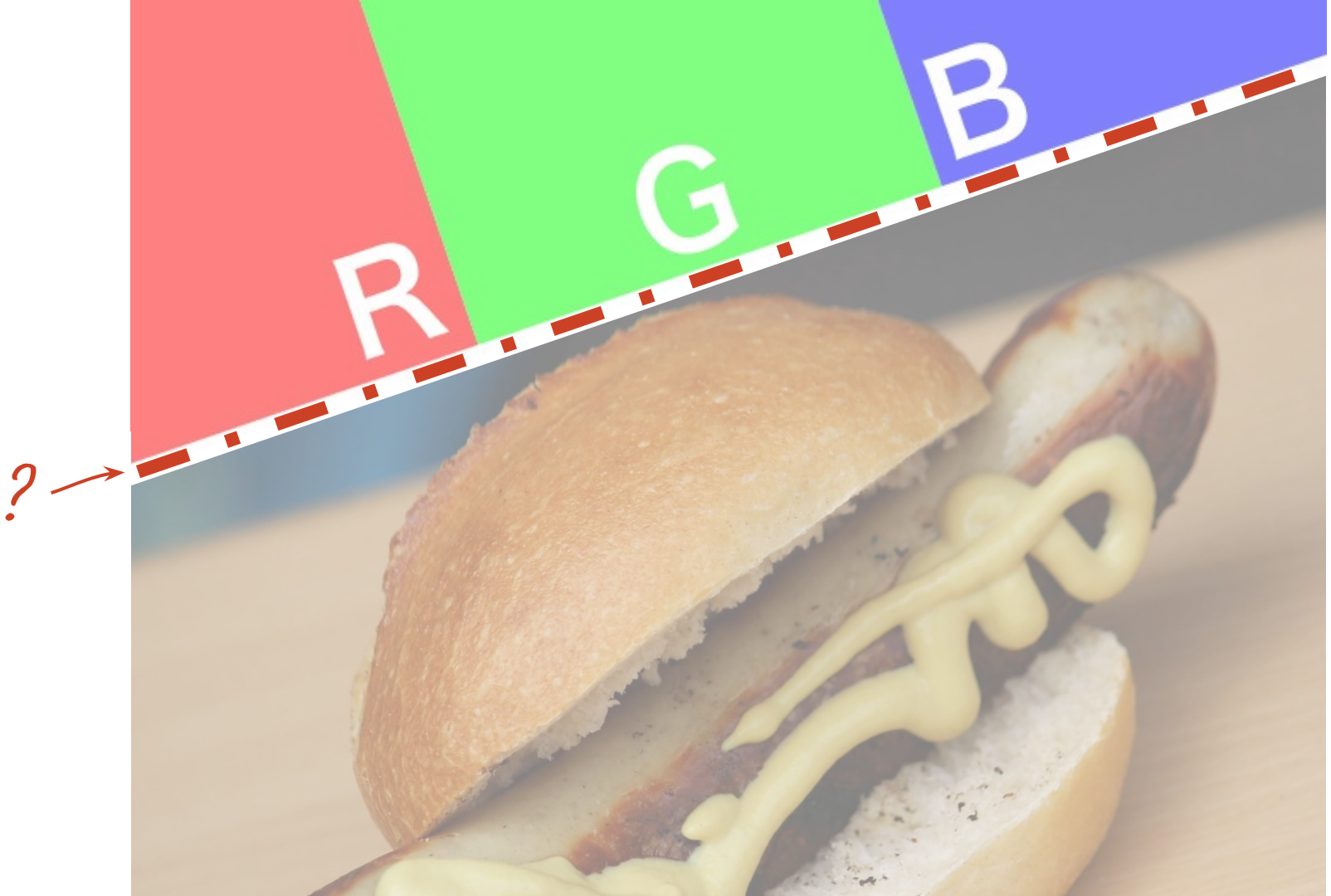
DLで画像解析、とは？


224px
224px


224
224
3


R
G
B
C
M
Y
K


R
G
B

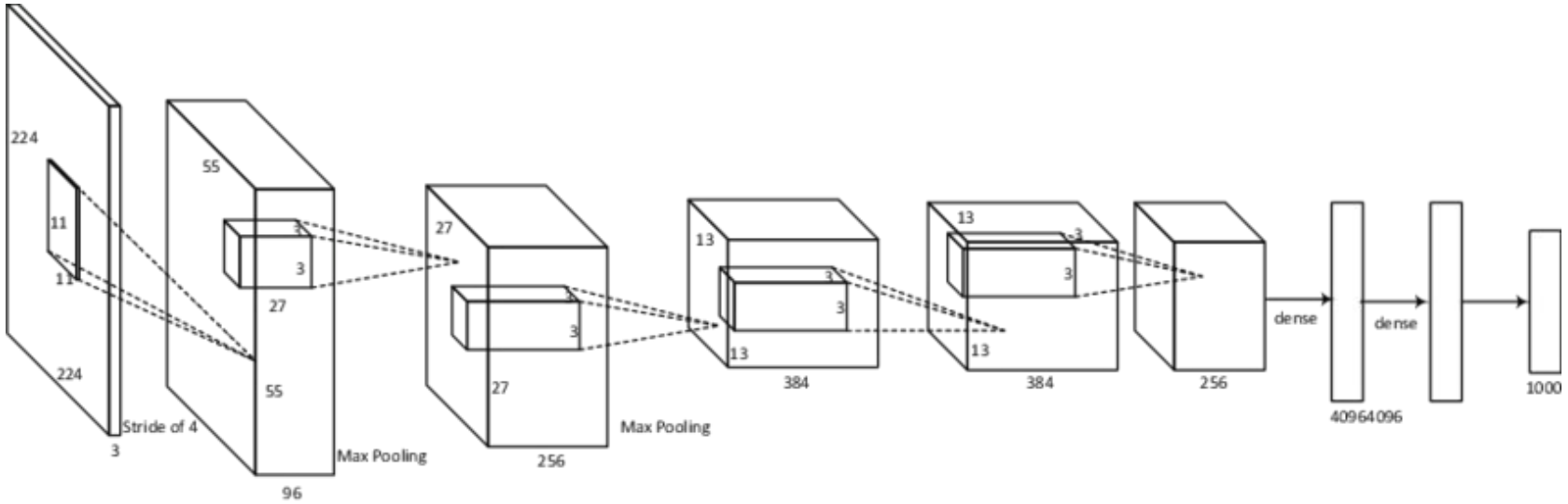

R
G
B
?

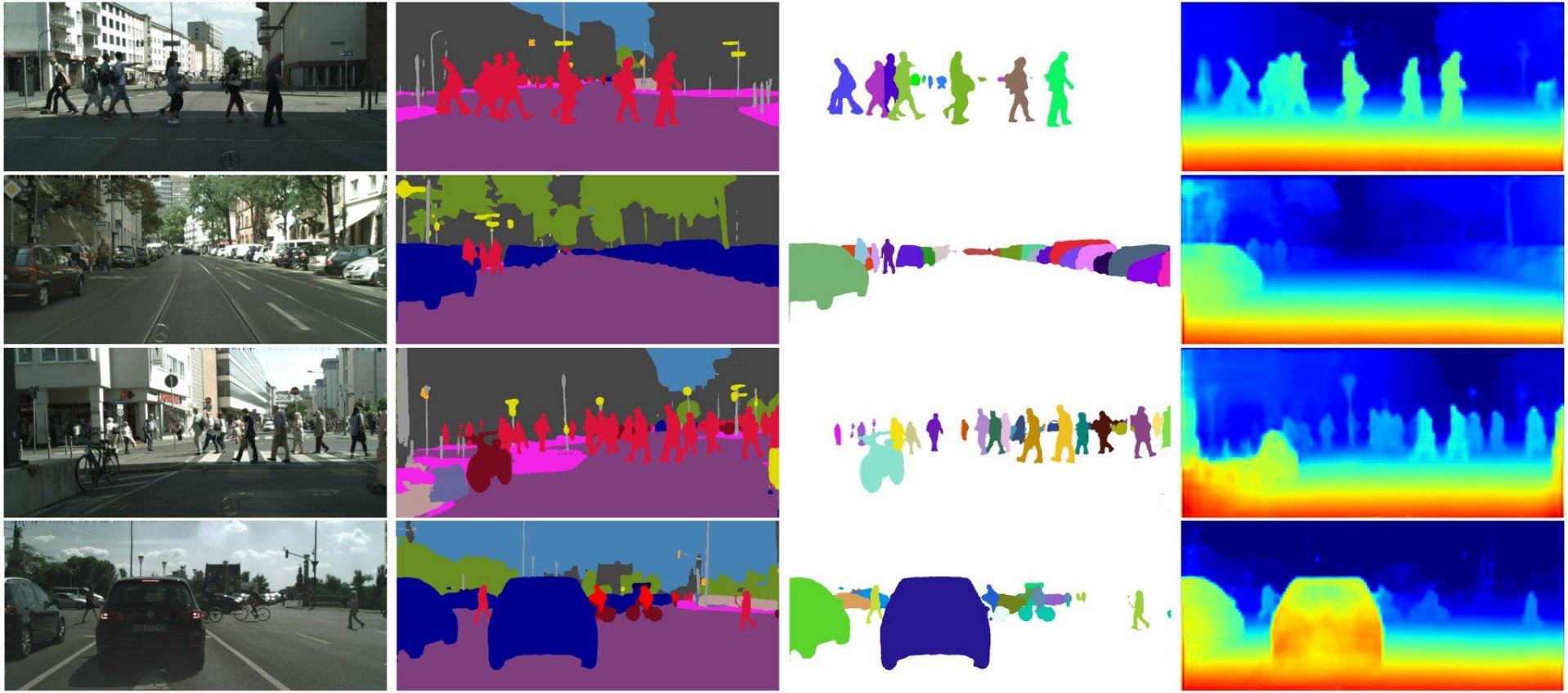
...and that is deep learning.

それはディープラーニングですよ。

...but seriously...

真面目な話。。。





(a) Input image (b) Segmentation output (c) Instance output (d) Depth output

Also, there's the LEARNING part.
それから学習のこともあるよ。

Prepare the Data
データを用意して

Dataset

Hot Dog - Not Hot Dog

https://www.kaggle.com/dansbecker/hot-dog-not-hot-dog

DanB • last updated 6 months ago

9 voters

Data | Overview | Kernels | Discussion | Activity | Download (45 MB) | New Kernel

Tags medium

Description

Context

Video Context

Description

2417.jpg
512 × 384

3690.jpg
384 × 512

5079.jpg
512 × 384

7896.jpg
512 × 384

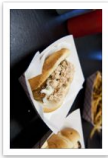
116486.jpg
382 × 512

117749.jpg
383 × 512

118875.jpg
512 × 384

120471.jpg
512 × 384

124323.jpg
512 × 384



124987.jpg
341 × 512

125329.jpg
384 × 512

127117.jpg
512 × 341

133012.jpg
382 × 512

133015.jpg
512 × 341

133245.jpg
512 × 384

135628.jpg
512 × 384

1164.jpg
512 × 343

1167.jpg
512 × 339

4176.jpg
512 × 384

4770.jpg
384 × 512

5762.jpg
512 × 384

5764.jpg
512 × 384

5813.jpg
512 × 384

6021.jpg
512 × 384

6261.jpg
512 × 384

6709.jpg
384 × 512

6926.jpg
512 × 384

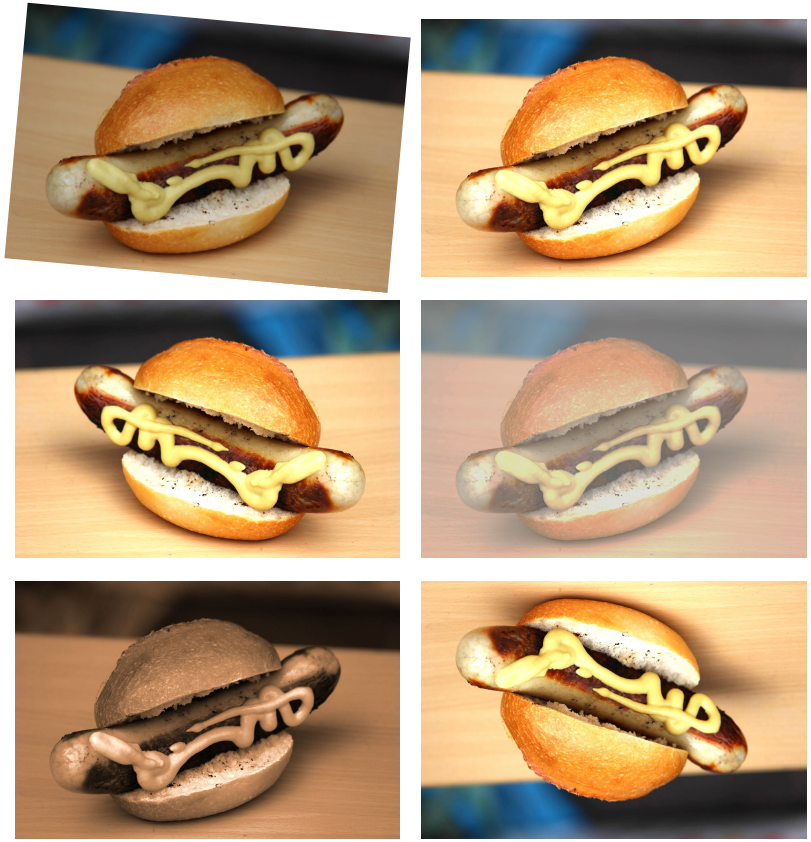
7056.jpg
512 × 384

8119.jpg
512 × 384

8350.jpg
512 × 384

8917.jpg
512 × 384

8984.jpg
512 × 384

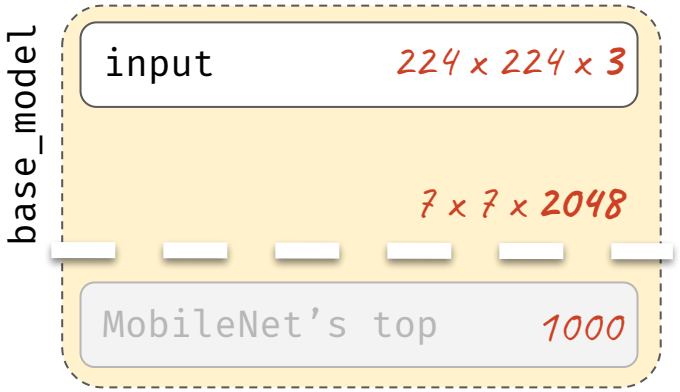


```
IMG_SIZE=[224, 224]

import keras.preprocessing.image

image_generator =
  keras.preprocessing.image.
    ImageDataGenerator(
        rescale=1./255,
        shear_range=0.0,
        width_shift_range=0.1,
        height_shift_range=0.1,
        rotation_range=10,
        fill_mode="wrap",
        vertical_flip=True,
        horizontal_flip=True
)
```

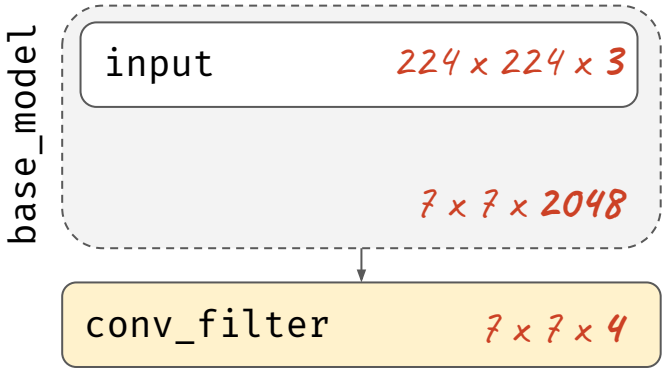
```
# unzip hot_dog_not_hot_dog.zip

# mkdir seefood/all

# cp -r seefood/test/* \
    seefood/train/* \
    seefood/all
```

```
train_generator =
  image_generator.flow_from_directory(
    "seefood/all",
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode="categorical",
    classes=["not_hot_dog", "hot_dog"]
)
```

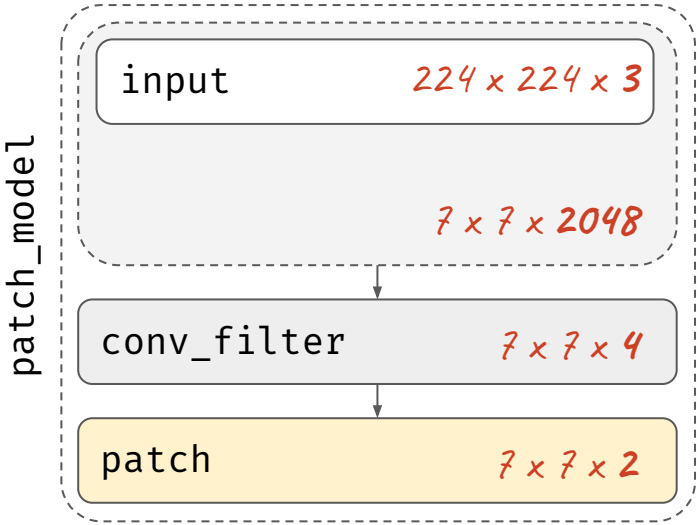
Build the Model
モデルを作って



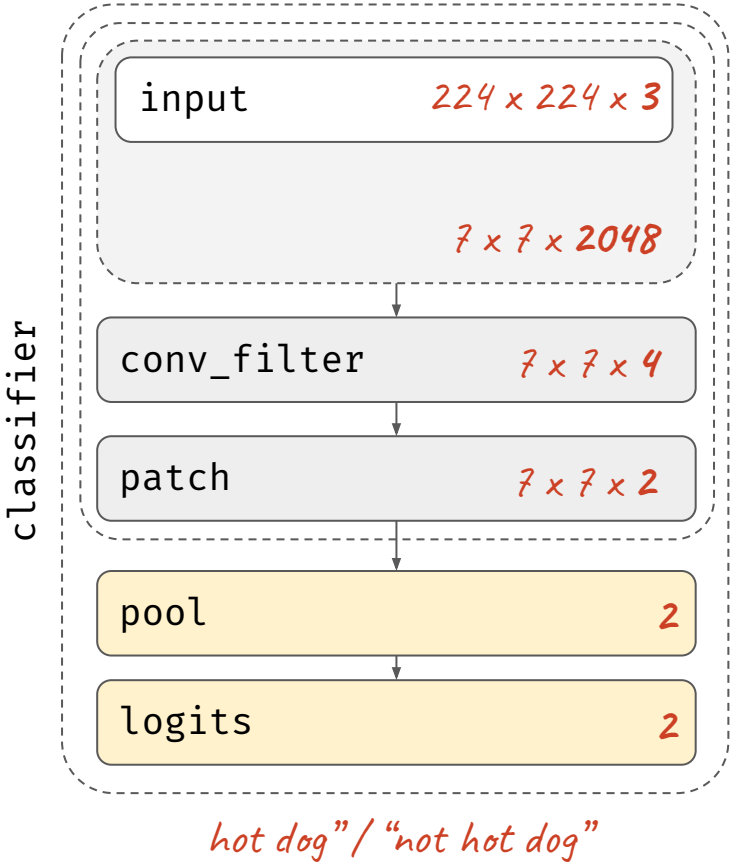

```python
import keras


base_model =
  keras.applications.mobilenet.MobileNet(
    input_shape=IMG_SIZE + [3],
    weights="imagenet",
    include_top=False
  )
```




```
 # add 30% noise during training
drop1 = keras.layers.SpatialDropout2D(0.3)
    (base_model.output)

conv_filter = keras.layers.convolutional.Conv2D(
    4, (1,1),
    activation="relu",
    use_bias=True,
    kernel_regularizer =
       keras.regularizers.l2(0.001)
)(drop1)
```

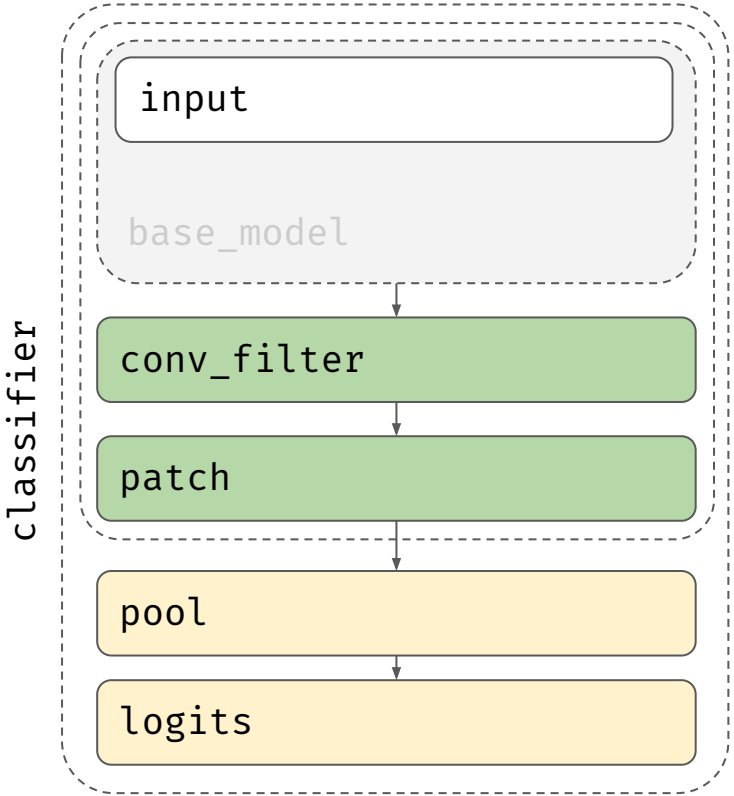



```
# add 30% noise during training
drop2 = keras.layers.SpatialDropout2D(0.3)
    (conv_filter)

patch = keras.layers.convolutional.Conv2D(
    2, (3, 3),
    name="patch",
    activation="softmax",
    use_bias=True,
    padding="same",
    kernel_regularizer=
          keras.regularizers.l2(0.001)
)(drop2)

patch_model = keras.models.Model(
    inputs=base_model.input,
    outputs=patch
)
```

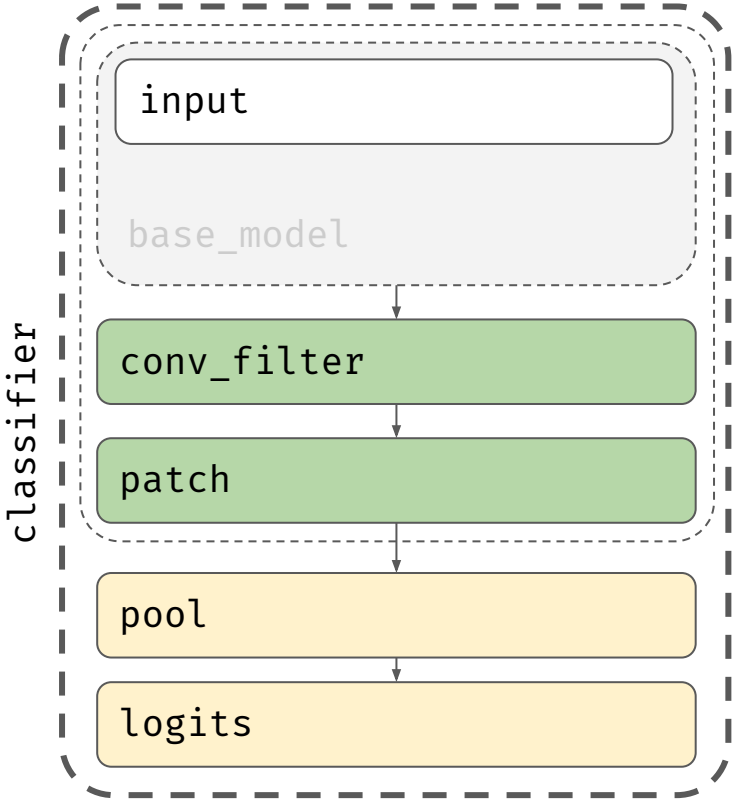



```
pool = keras.layers.GlobalAveragePooling2D()
    (patch)

logits = keras.layers.Activation("softmax")
    (pool)


classifier = keras.models.Model(
    inputs=base_model.input,
    outputs=logits
)
```

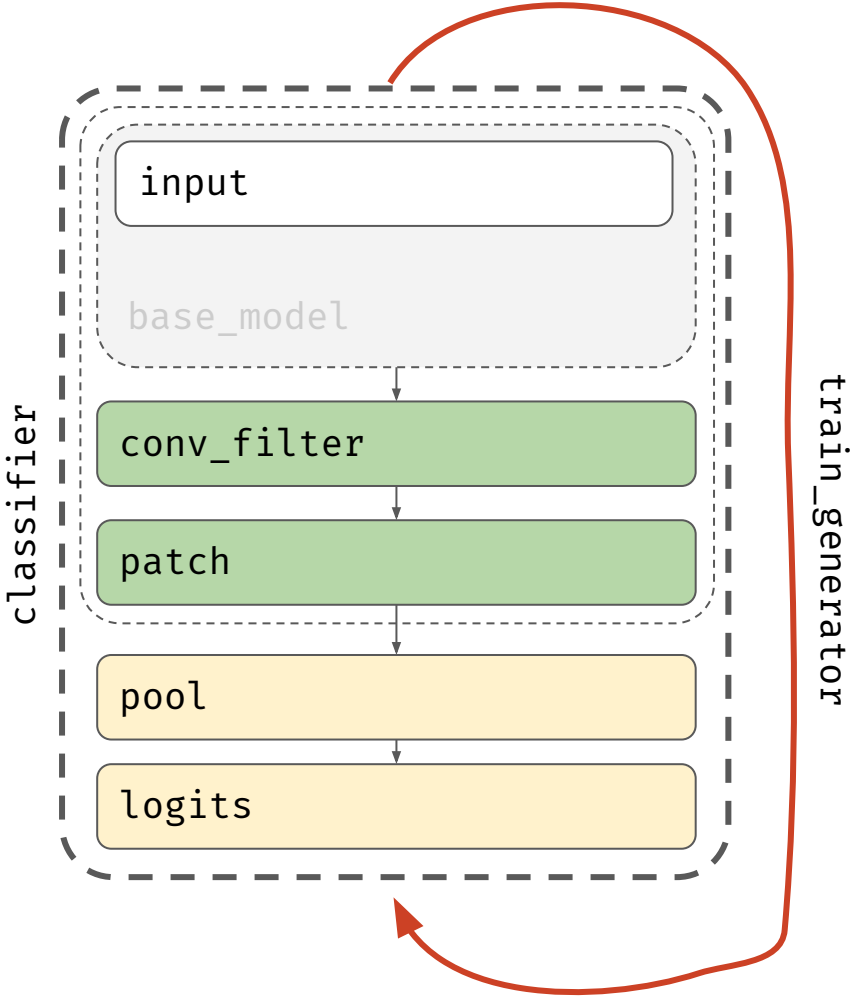
Train the Model
モデルを学習させて




```
for layer in base_model.layers:
    layer.trainable = False
```

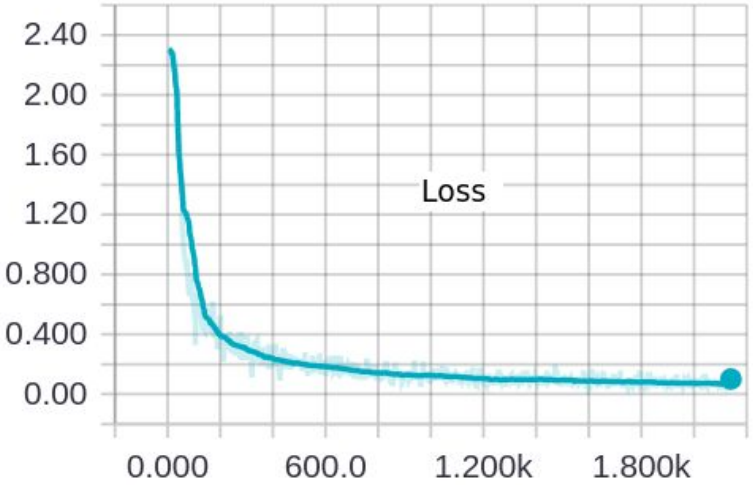


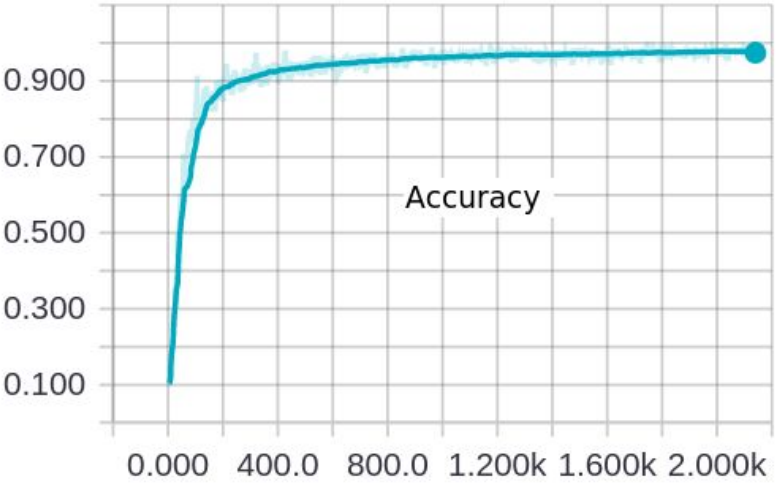

```
for layer in base_model.layers:
    layer.trainable = False


classifier.compile(
  optimizer="rmsprop",
  loss="categorical_crossentropy",
  metrics=["accuracy"]
)
```

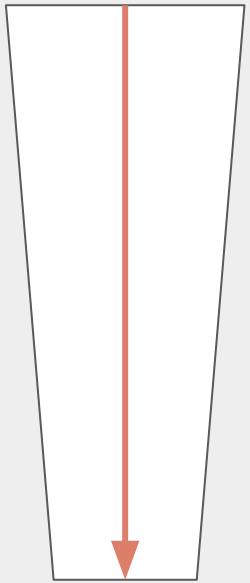



```
for layer in base_model.layers:
    layer.trainable = False


classifier.compile(
  optimizer="rmsprop",
  loss="categorical_crossentropy",
  metrics=["accuracy"]
)


classifier.fit_generator(
    train_generator,
    class_weight={0: .75, 1: .25},
    epochs=10
)
```

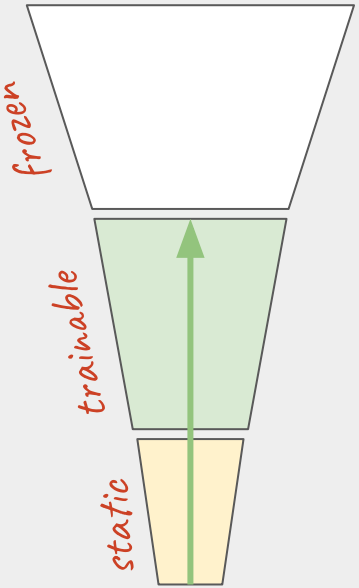
```
Epoch 1/10
32/32 [==============================] - 148s 5s/step - loss: 0.3157 - acc: 0.5051
Epoch 2/10
32/32 [==============================] - 121s 4s/step - loss: 0.3017 - acc: 0.5051
Epoch 3/10
32/32 [==============================] - 122s 4s/step - loss: 0.2961 - acc: 0.5010
Epoch 4/10
32/32 [==============================] - 121s 4s/step - loss: 0.2791 - acc: 0.5862
Epoch 5/10
32/32 [==============================] - 122s 4s/step - loss: 0.2681 - acc: 0.6380
Epoch 6/10
32/32 [==============================] - 123s 4s/step - loss: 0.2615 - acc: 0.6876
Epoch 7/10
32/32 [==============================] - 121s 4s/step - loss: 0.2547 - acc: 0.6790
Epoch 8/10
32/32 [==============================] - 122s 4s/step - loss: 0.2522 - acc: 0.7052
Epoch 9/10
32/32 [==============================] - 123s 4s/step - loss: 0.2522 - acc: 0.7045
Epoch 10/10
32/32 [==============================] - 145s 5s/step - loss: 0.2486 - acc: 0.7164
CPU times: user 1h 4min 20s, sys: 2min 35s, total: 1h 6min 56s
Wall time: 21min 8s
```







60% hot dog
40% not hot dog



+ 40%
- 40%

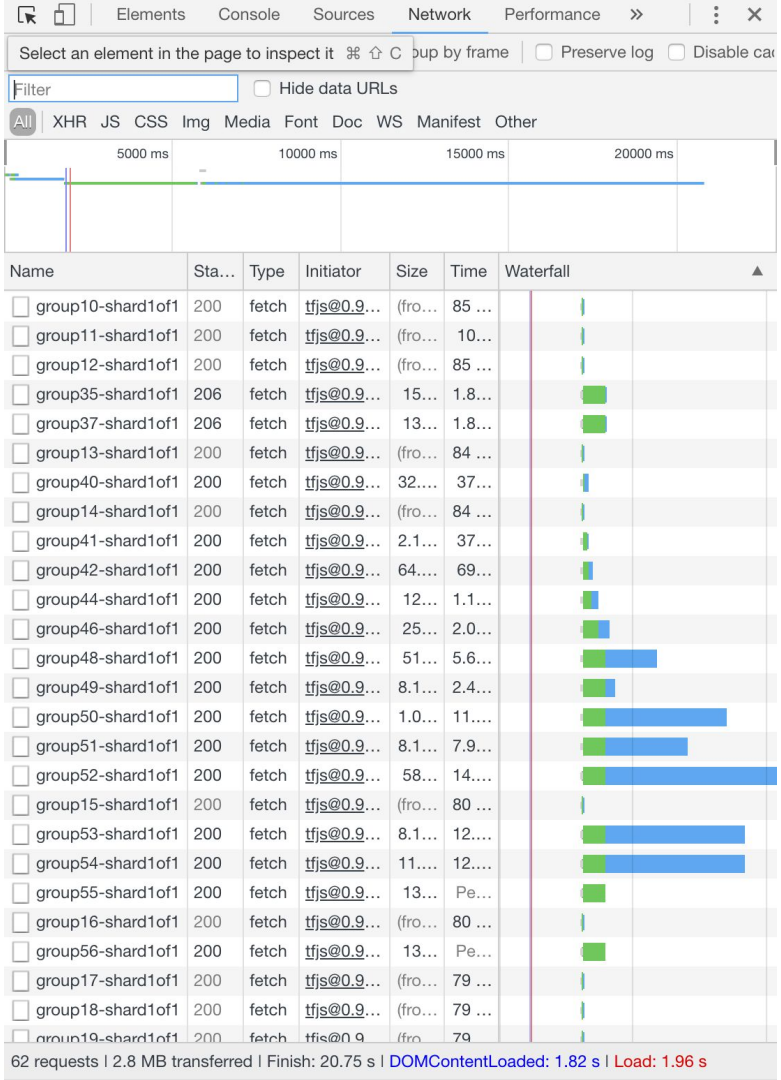
Save and export the model
モデルを保存して


group1-shard1of1
group2-shard1of1
group3-shard1of1
group4-shard1of1
group5-shard1of1
group6-shard1of1
group7-shard1of1
group8-shard1of1
group9-shard1of1
group10-shard1of1
group11-shard1of1
group12-shard1of1
group13-shard1of1
group14-shard1of1
group15-shard1of1
group16-shard1of1
group17-shard1of1
group18-shard1of1
group19-shard1of1
group20-shard1of1
group21-shard1of1
group22-shard1of1
group23-shard1of1
group24-shard1of1
group25-shard1of1
group26-shard1of1
group27-shard1of1
group28-shard1of1
group29-shard1of1
group30-shard1of1
group31-shard1of1
group32-shard1of1
group33-shard1of1
group34-shard1of1
group35-shard1of1
group36-shard1of1
group37-shard1of1
group38-shard1of1
group39-shard1of1
group40-shard1of1
group41-shard1of1
group42-shard1of1
group43-shard1of1
group44-shard1of1
group45-shard1of1
group46-shard1of1
group47-shard1of1
group48-shard1of1
group49-shard1of1
group50-shard1of1
group51-shard1of1
group52-shard1of1
group53-shard1of1
group54-shard1of1
group55-shard1of1
group56-shard1of1
JSON
model.json

Visualize

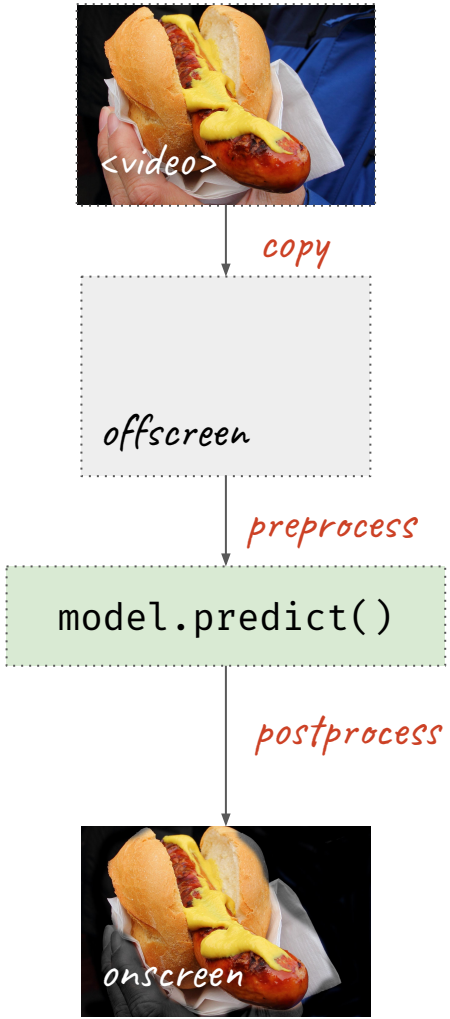
ビジュアライズして

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.9.0"> </script>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  </head>

  <body>

     <div id="app">
          <!-- stuff happening here -->
     </div>

     <script type="text/javascript" src="index.js"></script>
  </body>
</html>
```




```
this.loadModel()


loadModel: function() {
  return tf.loadModel('model/model.json')
    .then(loadedModel => {
      this.model = loadedModel
      return loadedModel
    })
}
```

loop()




```
loop: async function() {

  this.offscreen.drawImage(this.video, 0, 0, 640, 480)

  var imageData = this.offscreen
      .getImageData(0, 0, 640, 480)

  var pixeldata = tf.fromPixels(imageData)

  var response = await tf.tidy(() =>
      this.model.predict(preprocess(pixeldata))
  )

  responseData = await postprocess(
      response, pixeldata, 640, 480
  ).data()

  for (var i = 0; i < responseData.length; i+=1)
  {
    imageData.data[i] = responseData[i]
  }

  this.onscreen.putImageData(imageData, 0, 0)

  this.continue()

}
```

preprocess(pixeldata)

640px

480px

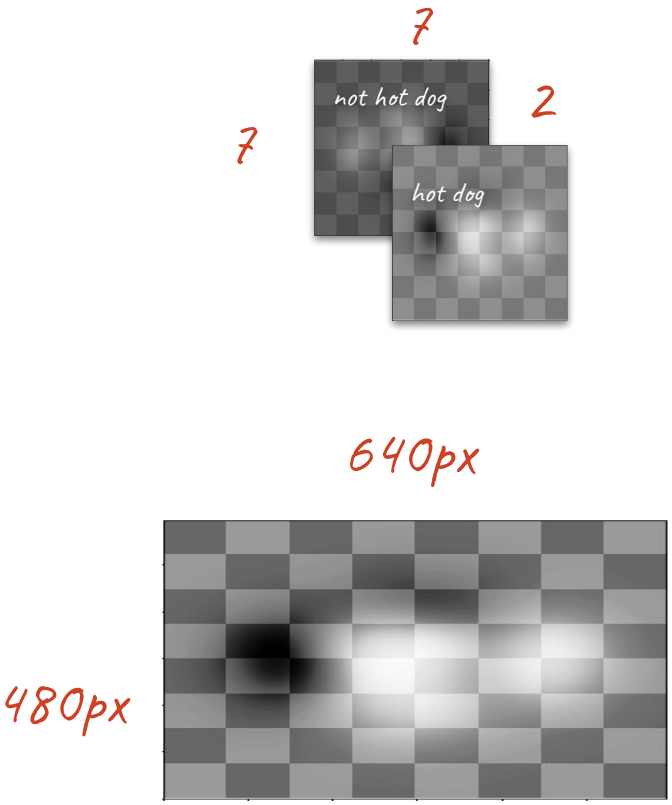
224px

224px

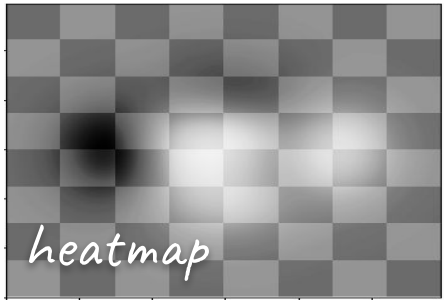
```
function preprocess(pixeldata) {

    // resize image for the model
    var contents = tf.image
      .resizeBilinear(pixeldata, [224, 224], false)

    // convert to float and make a one-image batch
    contents = contents
      .toFloat()
      .div(tf.scalar(255))
      .expandDims(0)

    return contents
}
```

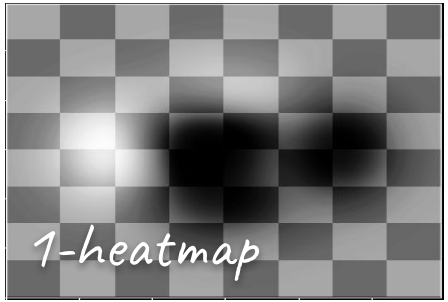
postprocess(tensor, pixeldata, width, height)

```
function postprocess(tensor, pixeldata, width, height) {

    var hotdog = 1
    var heatmap = tf.split(tensor.squeeze(), 2, 2)[hotdog]

    heatmap = tf.image.resizeBilinear(heatmap, [height, width], false)

    pixeldata = pixeldata.toFloat()
    var grayscale = pixeldata.mean(2).expandDims(2)

    grayscale = tf.onesLike(heatmap).sub(heatmap)
                .mul(grayscale).squeeze()
                .mul(tf.scalar(0.3))
    grayscaleStacked = tf.stack([grayscale, grayscale, grayscale]).transpose([1,2,0])


    composite = pixeldata.mul(heatmap).add(grayscaleStacked)

    var rgb = tf.split(composite, 3, 2)
    var alpha = tf.onesLike(rgb[0]).mul(tf.scalar(255))
    rgb.push(alpha)

    var composite = tf.stack(rgb, 2)

    return composite.toInt()
}
```




```
var hotdog = 1

var heatmap = tf.split(
     tensor.squeeze(),
     2, 2
   )[hotdog]

heatmap = tf.image.resizeBilinear(
   heatmap,
   [640, 480],
   false
)
```






```
pixeldata = pixeldata.toFloat()
var grayscale = pixeldata.mean(2).expandDims(2)

grayscale = tf.onesLike(heatmap).sub(heatmap)
                .mul(grayscale).squeeze()
                .mul(tf.scalar(0.3))

grayscaleStacked = tf.stack(
      [grayscale, grayscale, grayscale])
      .transpose([1,2,0])

composite = pixeldata
      .mul(heatmap)
      .add(grayscaleStacked)



var rgb = tf.split(composite, 3, 2)
var alpha = tf.onesLike(rgb[0])
      .mul(tf.scalar(255))
rgb.push(alpha)

var composite = tf.stack(rgb, 2)

return composite.toInt()
```

DEMO


1 REUBEN
2 HOUND
3 GREEK
4 ÉTOUFFÉE
5 MAC N CHEESE
6 CHILE RELLENO
7 SONORAN
8 BROCCOLI AND CHEESE
9 SWEET AND SOUR
10 RAPINI
11 POUTINE
12 PICKLED EGG SALAD
13 CAESAR
14 PORTLAND
15 BREAKFAST
16 CHICKEN FRIED
17 CREAMED CORN
18 PHILLY CHEESE DOG
19 ENCHILADA