

# 実データで学ぶ人工知能講座

## 成果発表

秋月佑太

The image displays four brain surface maps, likely representing the left and right hemispheres of a human brain, arranged in a 2x2 grid. Each map is rendered in a dark gray color, highlighting the complex folds and sulci of the cerebral cortex. Overlaid on these maps are areas of activation, color-coded in a gradient from red to yellow. These colored regions are primarily located in the posterior and lateral areas of the brain, which are typically associated with visual processing. The top-left map shows a large, central yellow/orange area with surrounding red patches. The top-right map shows more fragmented red and orange activation. The bottom-left map features a prominent yellow/orange area in the posterior region. The bottom-right map shows extensive yellow/orange activation across a large portion of the posterior and lateral cortex. The text '視覚刺激画像に基づくエンコーダー' is centered horizontally across the middle of the four brain maps.

視覚刺激画像に基づくエンコーダー

# 視覚刺激画像に基づくエンコーダー

特徴量の組み合わせを  $3 \cdot 6 \cdot (16 + 64 + 256) = 6048$  種類用意する。

```
freqs = [i/max(PIX_W,PIX_H) for i in [16, 32, 64]]
```

```
dirs = [math.pi * i for i in [0, 1/3, 3/2, 1/2, 4/3, 5/3]]
```

```
grids = [4, 8, 16]
```

```
def get_X(params, grid_h, grid_w):  
  
    X_tr = fb.G2_getfeatures(ims=raw_tr, fil_paras=params,  
                             gridshape=( grid_h, grid_w),  
                             mode="reflect", cval=0)  
  
    X_te = fb.G2_getfeatures(ims=raw_te, fil_paras=params,  
                             gridshape=(grid_h, grid_w),  
                             mode="reflect", cval=0)  
  
    return X_tr, X_te  
  
X_tr_tmp, X_te_tmp = get_X(myparas, mygrid_h, mygrid_w)  
  
X_tr = np.concatenate((X_tr, X_tr_tmp), axis=1)  
X_te = np.concatenate((X_te, X_te_tmp), axis=1)
```

計算が終わりそうにないことが判明  
フィルターを絞っていきましょう



# 轉移學習



# 転移学習

- モデルを変える (VGG16→InceptionV3)
- FC層を増やす (2層→3層)
- バッチサイズを変える (64→16, 32, 128)
- 目的関数を変える (SGD→Adam, RMSprop)

Accuracyが95%を超えず、劇的な精度向上に繋がらなかった。

普通にやっても罅が開かなそう



学習率の減衰がないSGDをまわし続ければ

いつかは95%を超える？



# ループを回して探索

- モデル: VGG16
- FC層: 3 層(1024, 512, 256)
- バッチサイズ: `int(random.uniform(1, 32))`
- 目的関数: SGD(学習率は`random.uniform(0.001, 0.00001)`)

```
for i in range(200):

    model = model_maker()

    lr      = random.uniform(0.001, 0.00001)
    batch_size = int(random.uniform(1, 32))

    model.compile(loss='categorical_crossentropy', metrics=['accuracy'],
                  optimizer=optimizers.SGD(lr=lr, momentum=0.0, decay=0.0))

    hist = model.fit_generator(
        train_generator, steps_per_epoch = nb_train_samples/batch_size,
        validation_steps = nb_validation_samples/batch_size,
        nb_epoch = nb_epoch, validation_data = validation_generator,
        callbacks=[checkpointer])

    model.save("ex1/model" + str(i) + ".h5")

    f.write(', '.join(str(i), str(lr), str(batch_size), str(checkpointer.best), '\n'))
```

i	learning_rate	batch_size	val_acc
0	0.0005985245440408503	18	0.9399999988079
1	0.0007297833731549363	5	0.941666670144
2	0.0007835998516742715	22	0.941666663289
3	0.0007997032000723217	22	0.941666665475
...			
10	0.0009461718658244601	21	0.946666670144
...			

これでも0.95の壁は越えられないのでモデルを変えて検証中..



# ヒートマップ

Food101<sup>1</sup>のデータから**15種類**の食べ物を抜き出して転移学習。  
それぞれ750枚の学習データがある。

---

<sup>1</sup> Food-101 – Mining Discriminative Components with Random Forests [website](#)



# 対象となる画像

pho



donuts



hot\_dog



omelette

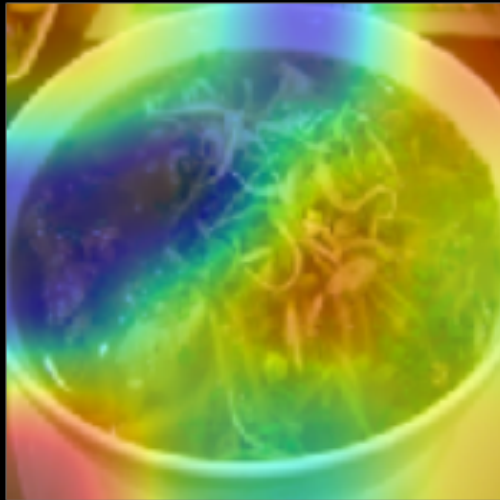


caesar\_salad

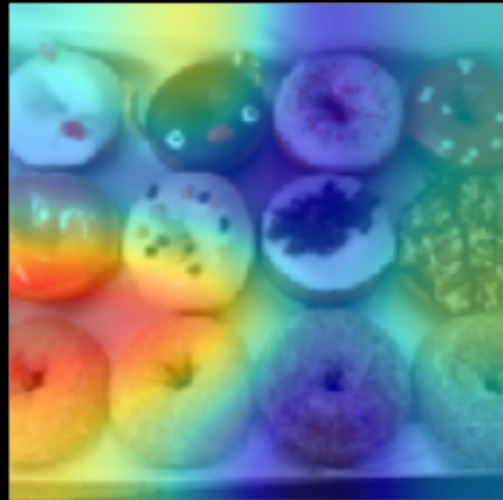


# vanilla

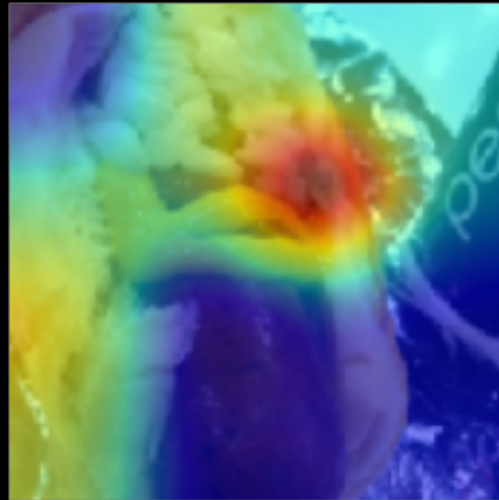
pho  
pho



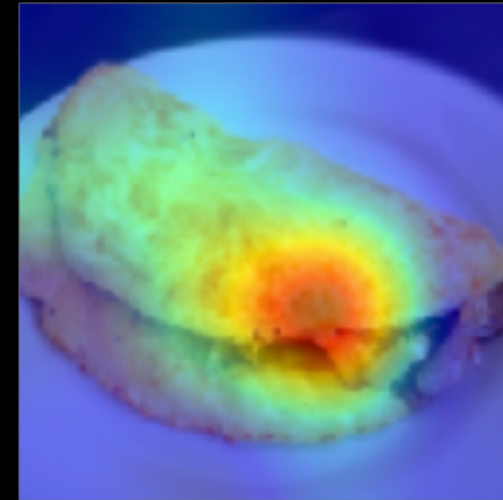
donuts  
donuts



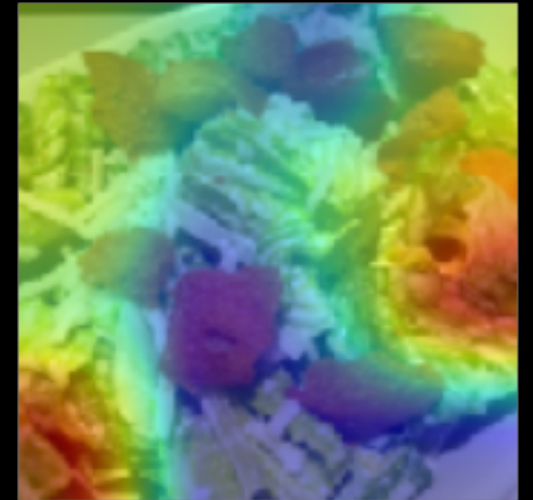
vanilla  
hot\_dog  
hot\_dog



lasagna  
omelette

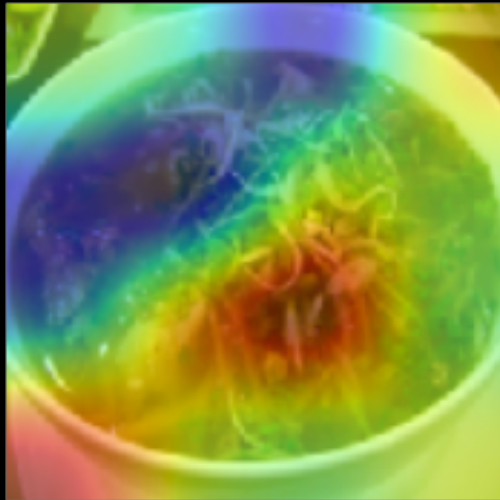


caesar\_salad  
caesar\_salad

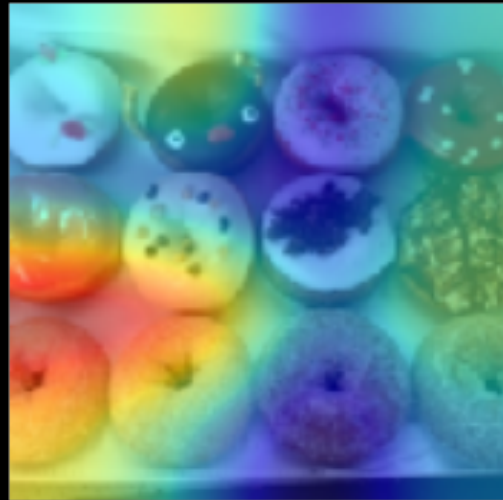


# guided

pho  
pho



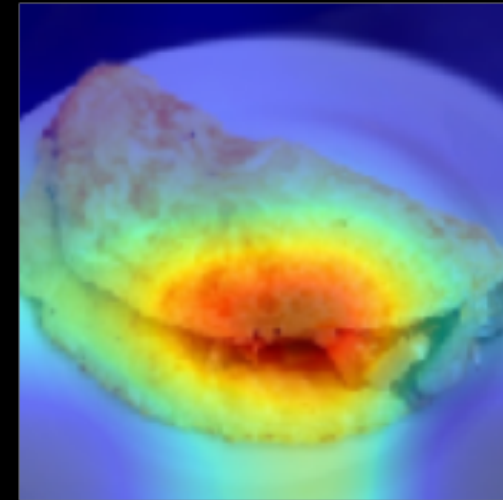
donuts  
donuts



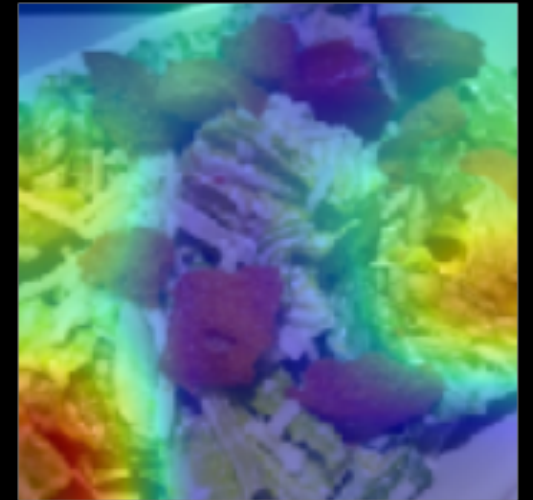
guided  
hot\_dog  
hot\_dog



lasagna  
omelette



caesar\_salad  
caesar\_salad





# relu

pho  
pho



donuts  
donuts



hot\_dog  
hot\_dog



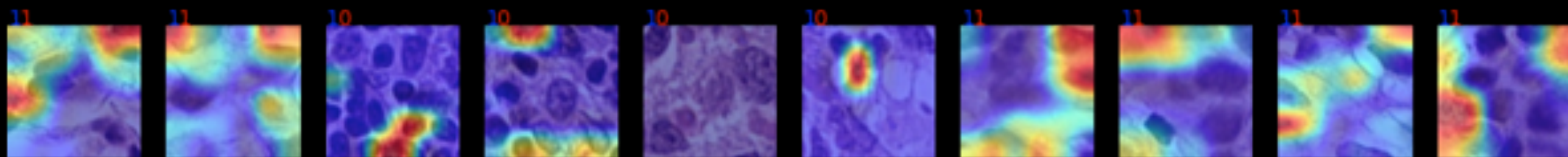
lasagna  
omelette



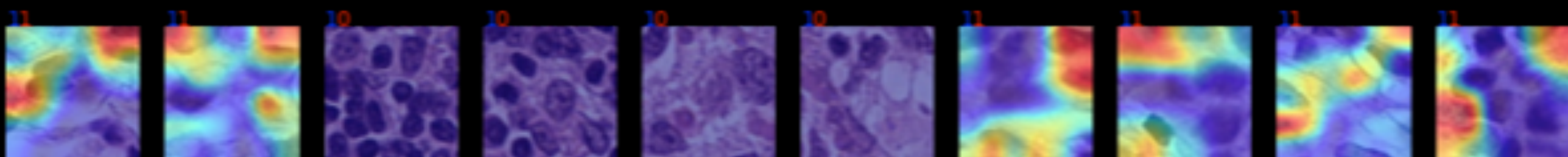
caesar\_salad  
caesar\_salad



vanilla



guided



relu

