

# A deep learning image classification workflow for the RAMP

Mehdi Cherti

# Classification task

We are given a training data of labeled examples:

$$D_{\text{train}} = \{(x_i, y_i), i = 1 \dots \text{nb\_examples}\}$$

where:

$$x_i \in \mathbb{R}^N, y_i \in \{1, 2, \dots, \text{nb\_classes}\}, x_i \sim p(x), y_i \sim p(y|x = x_i)$$

we want to find a model  $\mathbf{f}^*$  that minimizes prediction error:

$$f^* = \operatorname{argmin}_f \int L(f, x, y) p(x, y) dx dy$$
$$L(f, x, y) = 1 \text{ if } f(x) \neq y$$
$$L(f, x, y) = 0 \text{ if } f(x) = y$$

# Specifications of the workflow

- Deal with datasets that can't be stored in memory
- Support several deep learning frameworks
- Separate preprocessing and training/fitting
- Use CPU to load images into memory in parallel to GPU training

# Workflow - user perspective

- Users submit two files :
  - **image\_preprocessor.py**
  - **batch\_classifier.py**

# Image preprocessor

Purpose : preprocess images (**crop, resize, rotate, etc.**)

```
import numpy as np
from skimage.transform import resize


def transform(x):
    if x.shape[2] == 4:
        x = x[:, :, 0:3]
    x = resize(x, (32, 32), preserve_range=True)
    x = x.transpose((2, 0, 1))
    # 'RGB' -> 'BGR'
    x = x[::-1, :, :]
    # Zero-center by mean pixel
    x[0, :, :] -= 103.939
    x[1, :, :] -= 116.779
    x[2, :, :] -= 123.68
    return x
```



# Batch classifier

Purpose : train neural nets on **preprocessed images**

```
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import ZeroPadding2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.applications.vgg16 import VGG16
from keras.optimizers import Adam, SGD

class BatchClassifier(object):

    def __init__(self):
        self.model = build_model()  Builds train and valid generators

    def fit(self, gen_builder):
        gen_train, gen_valid, nb_train, nb_valid = gen_builder.get_train_valid_generators(batch_size=16, valid_ratio=0.1)
        self.model.fit_generator(
            gen_train,  Generator of tuples (X, y)
            samples_per_epoch=nb_train,
            nb_epoch=3,
            validation_data=gen_valid,  Generator of tuples (X, y)
            nb_val_samples=nb_valid,
            verbose=1)

    def predict_proba(self, X):
        return self.model.predict(X)

def build_model():
    vgg16 = VGG16(include_top=False, weights='imagenet')
    #vgg16.trainable = False
    inp = Input((3, 224, 224))
    x = vgg16(inp)
    x = Flatten(name='flatten')(x)
    x = Dense(4096, activation='relu', name='fc1')(x)
    x = Dense(4096, activation='relu', name='fc2')(x)
    out = Dense(209, activation='softmax', name='predictions')(x)
    model = Model(inp, out)
    model.compile(loss='categorical_crossentropy', optimizer=SGD(lr=1e-4), metrics=['accuracy'])
    return model
```

# Workflow - Internals

- 1) a chunk of (e.g. 1024) images are loaded into RAM
- 2) **transform** is applied to each image of the chunk
- 3) **fit**
  - loads a small mini-batch of (e.g. 32) images from the current chunk
  - Put the mini-batch into GPU memory
  - Updates neural net weights
  - repeat

# RAMP : Pollinating insects classification





# RAMP : Pollinating insects

- Three editions
  - **First** : 21K training examples with 18 classes (all images resized to 64x64)
    - **Accuracy** went from 30% to 71%
  - **Second** : 21K training examples with 18 classes
    - **Accuracy** went from 30% to 96%
  - **Third** : 68K training examples with 209 classes
    - **Accuracy** went from 33% to 83%

# RAMP : Pollinating insects

## Leaderboard of Second edition

Combined score: 0.959

Show  entries

Search:

team	submission	contributivity	historical contributivity	accuracy	nll	train time	test time	submitted at (UTC)
enzo.miller	starting_kitL	17	0	0.936	0.226	3249	558	2017-04-03 00:02:17 Mon
kegl	felix_crop	0	3	0.935	0.216	3321	590	2017-03-31 18:37:41 Fri
kegl	mathieu_crop	17	2	0.932	0.262	3218	579	2017-03-31 18:40:01 Fri
kegl	louis_crop_3	0	1	0.930	0.235	4273	578	2017-04-01 10:14:33 Sat
louis.remus	pytorch2++	0	3	0.926	0.261	3367	586	2017-03-31 13:13:47 Fri
kegl	jean_crop	0	0	0.925	0.270	4213	572	2017-03-31 18:41:56 Fri
felix.vogeli	starting_kit08	0	17	0.918	0.285	3258	572	2017-03-28 21:11:35 Tue
kegl	boyao_crop	17	2	0.916	0.300	11542	721	2017-04-01 19:02:25 Sat
mathieu.barre	pytorchResnet101	0	8	0.916	0.380	3195	595	2017-03-28 16:59:33 Tue
kegl	louis_crop	0	2	0.916	0.370	4329	579	2017-04-01 10:11:02 Sat
henri.hubert	best + invert	0	0	0.915	0.281	3241	574	2017-04-02 12:38:52 Sun
mathieu.barre	pytorchResnet	0	15	0.914	0.284	2089	454	2017-03-27 14:26:22 Mon
boyao.zhou	starting_4	0	12	0.913	0.319	11933	1118	2017-04-01 12:23:48 Sat
jean.langlois-meurin	-lr+epoch-batch	0	0	0.913	0.302	4261	566	2017-03-29 10:40:50 Wed
kegl	louis_crop_4	17	1	0.910	0.310	4649	722	2017-04-01 16:20:54 Sat
mhamed.hajaiej	starting_kit11	0	0	0.908	0.464	14061	595	2017-03-28 10:02:46 Tue

# RAMP : Pollinating insects


## Leaderboard of Third edition

Combined score: 0.831

Show  entries

Search:

Scores



team	submission	contributivity	historical contributivity	accuracy	nll	f1a	train time	test time	submitted at (UTC)
MH	baseline_more2come	33	35	0.812	0.709	0.83	32667	2335	2017-06-07 14:45:32 Wed
MH	baseline_rudy_DA	33	0	0.792	0.815	0.80	33317	2295	2017-06-08 03:45:47 Thu
rudyadam	best_aug_7ep_gamma	33	1	0.765	0.890	0.70	29437	2478	2017-06-07 15:57:49 Wed
rudyadam	best_aug	0	0	0.749	0.966	0.61	17761	3836	2017-06-07 06:13:51 Wed
mcherti	best	0	0	0.707	1.187	0.50	11379	1956	2017-05-26 09:44:56 Fri
DorraEA	fromTorchVgg	0	0	0.703	1.217	0.63	12904	2601	2017-06-07 17:22:45 Wed
tam	test_best0	0	0	0.699	1.278	0.50	11345	1905	2017-05-29 12:40:18 Mon
mcherti	pytorch_pretrained	0	0	0.614	1.714	0.32	11731	2347	2017-05-26 09:33:19 Fri
victor_estrade	inceptionv3_aug	0	0	0.595	2.339	0.37	17192	1836	2017-06-08 10:01:56 Thu
rudyadam	baseline	0	0	0.494	4.221	0.27	13514	1606	2017-06-06 04:51:42 Tue
rudyadam	baseline_299	0	0	0.420	6.954	0.16	18994	2258	2017-06-06 10:54:58 Tue
Xiwoieva	starting_kit_norm	0	0	0.401	2.799	0.10	8017	3159	2017-06-02 14:20:59 Fri
mcherti	pretrained_interm	0	0	0.393	2.815	0.11	12993	2107	2017-05-25 20:49:46 Thu
mcherti	pytorch	0	0	0.337	3.407	0.06	15457	1588	2017-05-26 09:03:22 Fri
rth	pretrained_v3	0	0	0.280	3.922	0.02	10710	2707	2017-05-30 17:23:25 Tue
deltabyte	2step_incep	0	0	0.273	7.403	0.02	16502	12346	2017-06-07 08:10:06 Wed
aboucaud	pretrained_vgg16	0	0	0.265	3.293	0.01	4879	954	2017-05-30 09:24:56 Tue
rudyadam	baselineNofrzln3Lrd	0	0	0.184	3.886	0.00	17752	2117	2017-06-08 20:48:31 Thu
kegl	starting_kit_test	0	0	0.009	33.876	0.00	579	883	2017-05-25 12:19:01 Thu
deltabyte	basic_incep	0	0	0.006	34.320	0.00	10505	9844	2017-06-06 14:34:48 Tue

# RAMP : Pollinating insects

- **Scores** used:

- Accuracy:  $L(f, x, y) = 1$  if  $f(x) = y$  else 0

- Negative log-likelihood :  $L(f, x, y) = -\log f(y|x), f(y|x) \in [0, 1]$

- F1a :  $F1(\text{class}) = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$

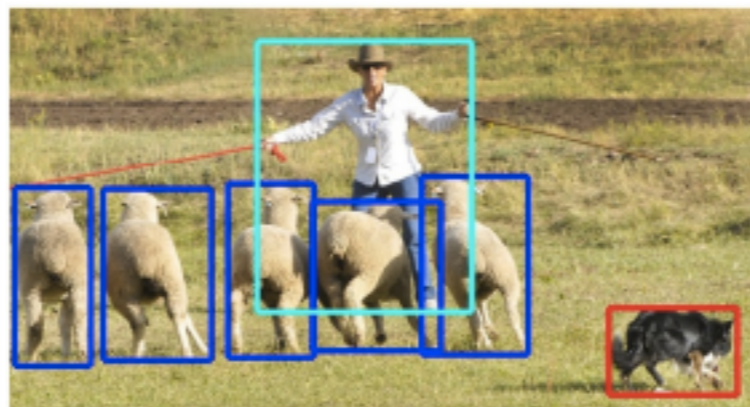
$$\text{precision} = \frac{\text{true\_positive}}{\text{true\_positive} + \text{false\_positive}} \quad \text{recall} = \frac{\text{true\_positive}}{\text{true\_positive} + \text{false\_negative}}$$

$F1a =$  proportion of classes with  $F1 > \theta$ , where  $\theta = 0.5$

# Object detection and segmentation in computer vision



(a) Image classification



(b) Object localization



(c) Semantic segmentation


# Object detection and segmentation in computer vision

Draw all unlabeled **person(s)** in the image.

- Find and draw on **all person(s)** that haven't been labeled.
- It's okay to overlap to labeled region.
- You need to label two images that contain unlabeled person(s) to complete
- Work will be rejected if **not carefully** drawn or unlabeled person(s) remain.

Submit No Unlabeled Person(s)

Draw (D) Erase (E) Zoom In (Z) Zoom Out (X)



© 2006 John A. Marsh - www.johnmarshphotography.com

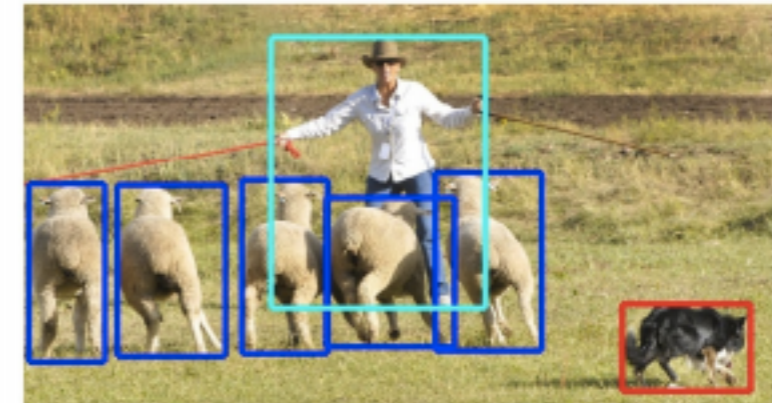
(e) Crowd Labeling

# Object detection score

**Input**



**Output**



- Algorithm outputs a list of bounding box detections with confidences
- A detection is considered correct if intersection over union (IoU) overlap with ground truth > threshold (0.5)
- Evaluated by average precision per object class

**Intersection over Union (IoU)**, measures amount of overlap between two sets (in particular, for bounding boxes):

$$\frac{A \cap B}{A \cup B}$$



# Segmentation score

Input



Output



**Metrics** We report four metrics from common semantic segmentation and scene parsing evaluations that are variations on pixel accuracy and region intersection over union (IU). Let  $n_{ij}$  be the number of pixels of class  $i$  predicted to belong to class  $j$ , where there are  $n_{cl}$  different classes, and let  $t_i = \sum_j n_{ij}$  be the total number of pixels of class  $i$ . We compute:

- pixel accuracy:  $\sum_i n_{ii} / \sum_i t_i$
- mean accuracy:  $(1/n_{cl}) \sum_i n_{ii} / t_i$
- mean IU:  $(1/n_{cl}) \sum_i n_{ii} / (t_i + \sum_j n_{ji} - n_{ii})$
- frequency weighted IU:  
 $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / (t_i + \sum_j n_{ji} - n_{ii})$



Thank you for listening