In [19]:
```
import sys
import keras
```

In [20]:
```
ls
```

```
0a1785dc6.jpg*                cropping1200Error.ipynb   test/
Boundingboxdataset.ipynb      'cropping2model .ipynb'   train/
Boundingboxdataset.pdf        imgrun/                   train.csv*
cli.txt                       network.ipynb             Whale/
create_boundingbox.csv.ipynb  sample_submission.csv*
create_boundingbox.csv.pdf    tails_coord.csv
```

In [21]:
```
with open('/home/lab2/DG foundation/Whale/models_csv/clicks.txt', 'rt')
data = [line.split(',') for line in data]
data = [(p,[(int(coord[i]),int(coord[i+1])) for i in range(0,len(coord)
data[1]
```

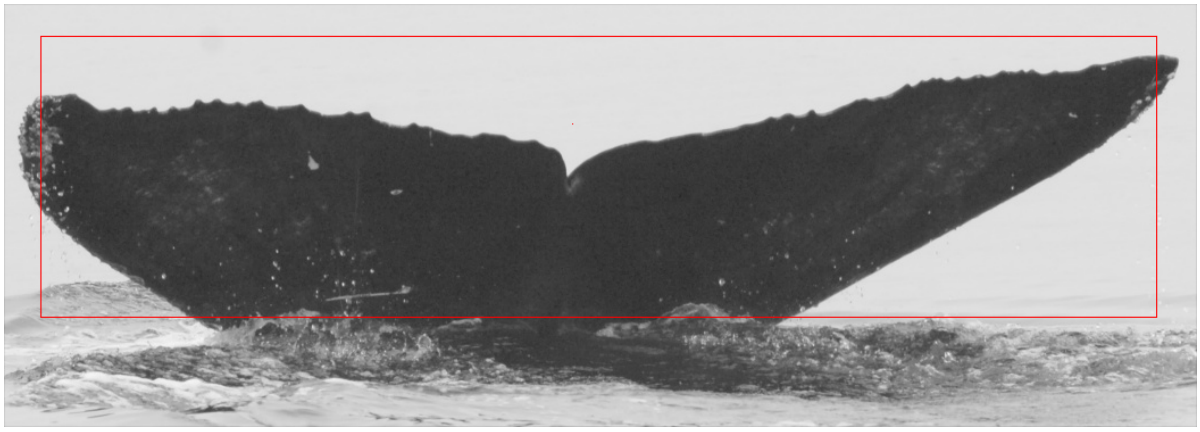
Out[21]:
```
('1fd140eec.jpg', [(114, 86), (879, 84), (846, 489), (311, 46)])
```

In [22]:
```
len(data)
```

Out[22]:
```
1190
```

In [23]:
```
from PIL import Image as pil_image
from PIL.ImageDraw import Draw
from os.path import isfile

def expand_path(p):
    if isfile('/home/lab2/DG foundation/train/' + p): return '/home/lab
    if isfile('/home/lab2/DG foundation/test/' + p): return '/home/lab2/
    return p

def read_raw_image(p):
    return pil_image.open(expand_path(p))

def draw_dot(draw, x, y):
    draw.ellipse(((x,y),(x,y)), fill='red', outline='red')

def draw_dots(draw, coordinates):
    for x,y in coordinates: draw_dot(draw, x, y)

def bounding_rectangle(list):
    x0, y0 = list[0]
    x1, y1 = x0, y0
    for x,y in list[1:]:
        x0 = min(x0, x)
        y0 = min(y0, y)
        x1 = max(x1, x)
        y1 = max(y1, y)
    return x0,y0,x1,y1

filename,coordinates = data[12]
box = bounding_rectangle(coordinates)
img = read_raw_image(filename)
draw = Draw(img)
draw_dots(draw, coordinates)
draw.rectangle(box, outline='red')
img
```

Out[23]:



In [24]:
```
img_shape  = (128,128,1)
anisotropy = 2.15
```

In [25]:
```
import random
import numpy as np
from scipy.ndimage import affine_transform
from keras.preprocessing.image import img_to_array

def read_array(p):
    img = read_raw_image(p).convert('L')
    return img_to_array(img)

def build_transform(rotation, shear, height_zoom, width_zoom, height_sh
    rotation        = np.deg2rad(rotation)
    shear           = np.deg2rad(shear)
    rotation_matrix = np.array([[np.cos(rotation), np.sin(rotation), 0]
    shift_matrix    = np.array([[1, 0, height_shift], [0, 1, width_shif
    shear_matrix    = np.array([[1, np.sin(shear), 0], [0, np.cos(shear
    zoom_matrix     = np.array([[1.0/height_zoom, 0, 0], [0, 1.0/width_
    shift_matrix    = np.array([[1, 0, -height_shift], [0, 1, -width_sh
    return np.dot(np.dot(rotation_matrix, shear_matrix), np.dot(zoom_ma

def center_transform(affine, input_shape):
    hi, wi = float(input_shape[0]), float(input_shape[1])
    ho, wo = float(img_shape[0]), float(img_shape[1])
    top, left, bottom, right = 0, 0, hi, wi
    if wi/hi/anisotropy < wo/ho: # input image too narrow, extend width
        w     = hi*wo/ho*anisotropy
        left  = (wi-w)/2
        right = left + w
    else: # input image too wide, extend height
        h      = wi*ho/wo/anisotropy
        top    = (hi-h)/2
        bottom = top + h
    center_matrix   = np.array([[1, 0, -ho/2], [0, 1, -wo/2], [0, 0, 1]
    scale_matrix    = np.array([[(bottom - top)/ho, 0, 0], [0, (right -
    decenter_matrix = np.array([[1, 0, hi/2], [0, 1, wi/2], [0, 0, 1]])
    return np.dot(np.dot(decenter_matrix, scale_matrix), np.dot(affine,

def transform_img(x, affine):
    matrix   = affine[:2,:2]
    offset   = affine[:2,2]
    x        = np.moveaxis(x, -1, 0)
    channels = [affine_transform(channel, matrix, offset, output_shape=
                                 mode='constant', cval=np.average(chann
    return np.moveaxis(np.stack(channels, axis=0), 0, -1)

def read_for_validation(p):
    x  = read_array(p)
    t  = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    t  = center_transform(t, x.shape)
    x  = transform_img(x, t)
    x -= np.mean(x, keepdims=True)
    x /= np.std(x, keepdims=True) + K.epsilon()
    return x,t

def read_for_training(p):
    x  = read_array(p)
    t  = build_transform(
```

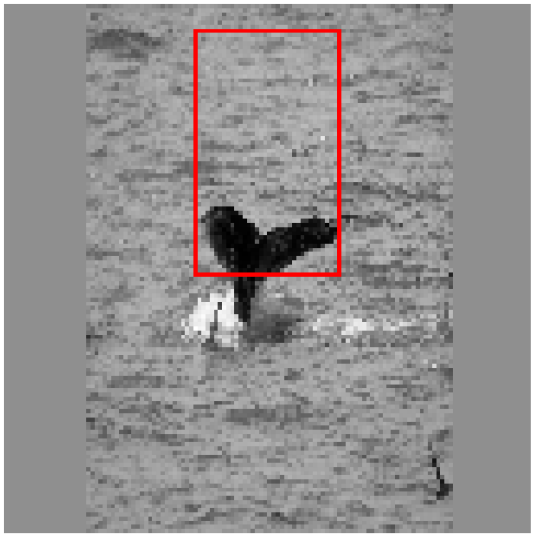
```
            random.uniform(-5, 5),
            random.uniform(-5, 5),
            random.uniform(0.9, 1.0),
            random.uniform(0.9, 1.0),
            random.uniform(-0.05*img_shape[0], 0.05*img_shape[0]),
            random.uniform(-0.05*img_shape[1], 0.05*img_shape[1]))
    t  = center_transform(t, x.shape)
    x  = transform_img(x, t)
    x -= np.mean(x, keepdims=True)
    x /= np.std(x, keepdims=True) + K.epsilon()
    return x,t

def coord_transform(list, trans):
    result = []
    for x,y in list:
        y,x,_ = trans.dot([y,x,1]).astype(np.int)
        result.append((x,y))
    return result
```

In [27]:
```
from sklearn.model_selection import train_test_split

train, val = train_test_split(data, test_size=190, random_state=1)
train += train
train += train
train += train
#train += train
len(train),len(val)
```

Out[27]: (8000, 190)

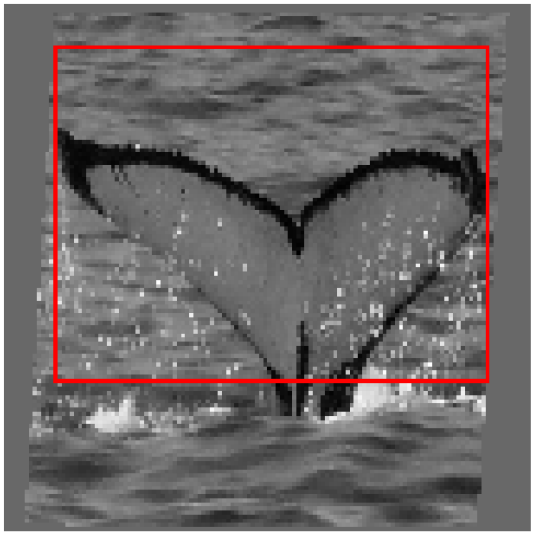
In [29]:
```
import matplotlib.pyplot as plt
from tqdm import tqdm, tqdm_notebook
from keras import backend as K
from keras.preprocessing.image import array_to_img
from numpy.linalg import inv as mat_inv

def show_whale(imgs, per_row=5):
    n          = len(imgs)
    rows       = (n + per_row - 1)//per_row
    cols       = min(per_row, n)
    fig, axes = plt.subplots(rows,cols, figsize=(24//per_row*cols,24//pe
    for ax in axes.flatten(): ax.axis('off')
    for i,(img,ax) in enumerate(zip(imgs, axes.flatten())): ax.imshow(ir

val_a = np.zeros((len(val),)+img_shape,dtype=K.floatx())
val_b = np.zeros((len(val),4),dtype=K.floatx())
for i,(p,coords) in enumerate(tqdm_notebook(val)):
    img,trans      = read_for_validation(p)
    coords         = coord_transform(coords, mat_inv(trans))
    x0,y0,x1,y1    = bounding_rectangle(coords)
    val_a[i,:,:,:] = img
    val_b[i,0]     = x0
    val_b[i,1]     = y0
    val_b[i,2]     = x1
    val_b[i,3]     = y1

idx  = 34
img  = array_to_img(val_a[idx])
img  = img.convert('RGB')
draw = Draw(img)
draw.rectangle(val_b[idx], outline='red')
show_whale([read_raw_image(val[idx][0]), img], per_row=2)
```

100% 190/190 [00:02<00:00, 93.59it/s]



In [16]:
```
from keras.utils import Sequence

class TrainingData(Sequence):
    def __init__(self, batch_size=32):
        super(TrainingData, self).__init__()
        self.batch_size = batch_size
    def __getitem__(self, index):
        start = self.batch_size*index;
        end   = min(len(train), start + self.batch_size)
        size  = end - start
        a     = np.zeros((size,) + img_shape, dtype=K.floatx())
        b     = np.zeros((size,4), dtype=K.floatx())
        for i,(p,coords) in enumerate(train[start:end]):
            img,trans   = read_for_training(p)
            coords      = coord_transform(coords, mat_inv(trans))
            x0,y0,x1,y1 = bounding_rectangle(coords)
            a[i,:,:,:]  = img
            b[i,0]      = x0
            b[i,1]      = y0
            b[i,2]      = x1
            b[i,3]      = y1
        return a,b
    def __len__(self):
        return (len(train) + self.batch_size - 1)//self.batch_size

random.seed(1)
a, b = TrainingData(batch_size=5)[1]
img  = array_to_img(a[0])
img  = img.convert('RGB')
draw = Draw(img)
draw.rectangle(b[0], outline='red')
show_whale([read_raw_image(train[0][0]), img], per_row=2)
```

NAHWC #0963 Fez YoNAH #2998



In [30]:
```
from keras.engine.topology import Input
from keras.layers import BatchNormalization, Concatenate, Conv2D, Dense
from keras.models import Model

def build_model(with_dropout=True):
    kwargs     = {'activation':'relu', 'padding':'same'}
    conv_drop  = 0.2
    dense_drop = 0.5
    inp        = Input(shape=img_shape)

    x = inp

    x = Conv2D(64, (9, 9), **kwargs)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = BatchNormalization()(x)
    if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, in

    x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = BatchNormalization()(x)
    if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, in

    x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = BatchNormalization()(x)
    if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, in

    x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = BatchNormalization()(x)
    if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, in

    x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = BatchNormalization()(x)
    if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, in

    x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = Conv2D(64, (3, 3), **kwargs)(x)
    x = BatchNormalization()(x)
    if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, in

    h = MaxPooling2D(pool_size=(1, int(x.shape[2])))(x)
    h = Flatten()(h)
    if with_dropout: h = Dropout(dense_drop)(h)
    h = Dense(16, activation='relu')(h)

    v = MaxPooling2D(pool_size=(int(x.shape[1]), 1))(x)
    v = Flatten()(v)
    if with_dropout: v = Dropout(dense_drop)(v)
    v = Dense(16, activation='relu')(v)
```

```
    x = Concatenate()([h,v])
    if with_dropout: x = Dropout(0.5)(x)
    x = Dense(4, activation='linear')(x)
    return Model(inp,x)

model = build_model(with_dropout=True)
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
==================================================================
input_2 (InputLayer)         (None, 128, 128, 1)       0

_________________________________________________________________
conv2d_18 (Conv2D)           (None, 128, 128, 64)      5248
input_2[0][0]
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 128, 128, 64)      36928
conv2d_18[0][0]
_________________________________________________________________
batch_normalization_7 (BatchNor (None, 128, 128, 64)      256
conv2d_19[0][0]
_________________________________________________________________
dropout_10 (Dropout)         (None, 128, 128, 64)      0
batch_normalization_7[0][0]
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 64, 64, 64)        16448
dropout_10[0][0]
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 64, 64, 64)        36928
conv2d_20[0][0]
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 64, 64, 64)        36928
conv2d_21[0][0]
_________________________________________________________________
batch_normalization_8 (BatchNor (None, 64, 64, 64)        256
conv2d_22[0][0]
_________________________________________________________________
dropout_11 (Dropout)         (None, 64, 64, 64)        0
batch_normalization_8[0][0]
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 32, 32, 64)        16448
dropout_11[0][0]
```

```
____________________________________________________________________________________________
____________________________
conv2d_24 (Conv2D)              (None, 32, 32, 64)   36928       conv2
d_23[0][0]
____________________________________________________________________________________________
____________________________
conv2d_25 (Conv2D)              (None, 32, 32, 64)   36928       conv2
d_24[0][0]
____________________________________________________________________________________________
____________________________
batch_normalization_9 (BatchNor (None, 32, 32, 64)   256         conv2
d_25[0][0]
____________________________________________________________________________________________
____________________________
dropout_12 (Dropout)            (None, 32, 32, 64)   0           batch
_normalization_9[0][0]
____________________________________________________________________________________________
____________________________
conv2d_26 (Conv2D)              (None, 16, 16, 64)   16448       dropo
ut_12[0][0]
____________________________________________________________________________________________
____________________________
conv2d_27 (Conv2D)              (None, 16, 16, 64)   36928       conv2
d_26[0][0]
____________________________________________________________________________________________
____________________________
conv2d_28 (Conv2D)              (None, 16, 16, 64)   36928       conv2
d_27[0][0]
____________________________________________________________________________________________
____________________________
batch_normalization_10 (BatchNo (None, 16, 16, 64)   256         conv2
d_28[0][0]
____________________________________________________________________________________________
____________________________
dropout_13 (Dropout)            (None, 16, 16, 64)   0           batch
_normalization_10[0][0]
____________________________________________________________________________________________
____________________________
conv2d_29 (Conv2D)              (None, 8, 8, 64)     16448       dropo
ut_13[0][0]
____________________________________________________________________________________________
____________________________
conv2d_30 (Conv2D)              (None, 8, 8, 64)     36928       conv2
d_29[0][0]
____________________________________________________________________________________________
____________________________
conv2d_31 (Conv2D)              (None, 8, 8, 64)     36928       conv2
d_30[0][0]
____________________________________________________________________________________________
____________________________
batch_normalization_11 (BatchNo (None, 8, 8, 64)     256         conv2
d_31[0][0]
____________________________________________________________________________________________
____________________________
dropout_14 (Dropout)            (None, 8, 8, 64)     0           batch
_normalization_11[0][0]
____________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 4, 4, 64)     16448       dropo
ut_14[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 4, 4, 64)     36928       conv2
d_32[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 4, 4, 64)     36928       conv2
d_33[0][0]
__________________________________________________________________________________________________
batch_normalization_12 (BatchNo (None, 4, 4, 64)     256         conv2
d_34[0][0]
__________________________________________________________________________________________________
dropout_15 (Dropout)            (None, 4, 4, 64)     0           batch
_normalization_12[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 4, 1, 64)     0           dropo
ut_15[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 1, 4, 64)     0           dropo
ut_15[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 256)          0           max_p
ooling2d_3[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 256)          0           max_p
ooling2d_4[0][0]
__________________________________________________________________________________________________
dropout_16 (Dropout)            (None, 256)          0           flatt
en_3[0][0]
__________________________________________________________________________________________________
dropout_17 (Dropout)            (None, 256)          0           flatt
en_4[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 16)           4112        dropo
ut_16[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 16)           4112        dropo
ut_17[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 32)           0           dense
_4[0][0]
                                                                 dense
_5[0][0]
```

```
_____
dropout_18 (Dropout)            (None, 32)           0           conca
tenate_2[0][0]
_____
dense_6 (Dense)                 (None, 4)            132         dropo
ut_18[0][0]
==========================================================================
==============================
Total params: 503,588
Trainable params: 502,820
Non-trainable params: 768
_____
_____
```

In [31]:
```
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnP
from keras.optimizers import Adam

for num in range(1, 3):
    model_name = 'cropping-%1d.h5' % num
    print(model_name)
    model.compile(Adam(lr=0.04), loss='mean_squared_error')
    model.fit_generator(
        TrainingData(), epochs=30, max_queue_size=12, workers=4, verbos
        validation_data=(val_a, val_b),
        callbacks=[
            EarlyStopping(monitor='val_loss', patience=9, min_delta=0.1
            ReduceLROnPlateau(monitor='val_loss', patience=3, min_delta
            ModelCheckpoint(model_name, save_best_only=True, save_weigh
        ])
    model.load_weights(model_name)
    model.evaluate(val_a, val_b, verbose=0)
```

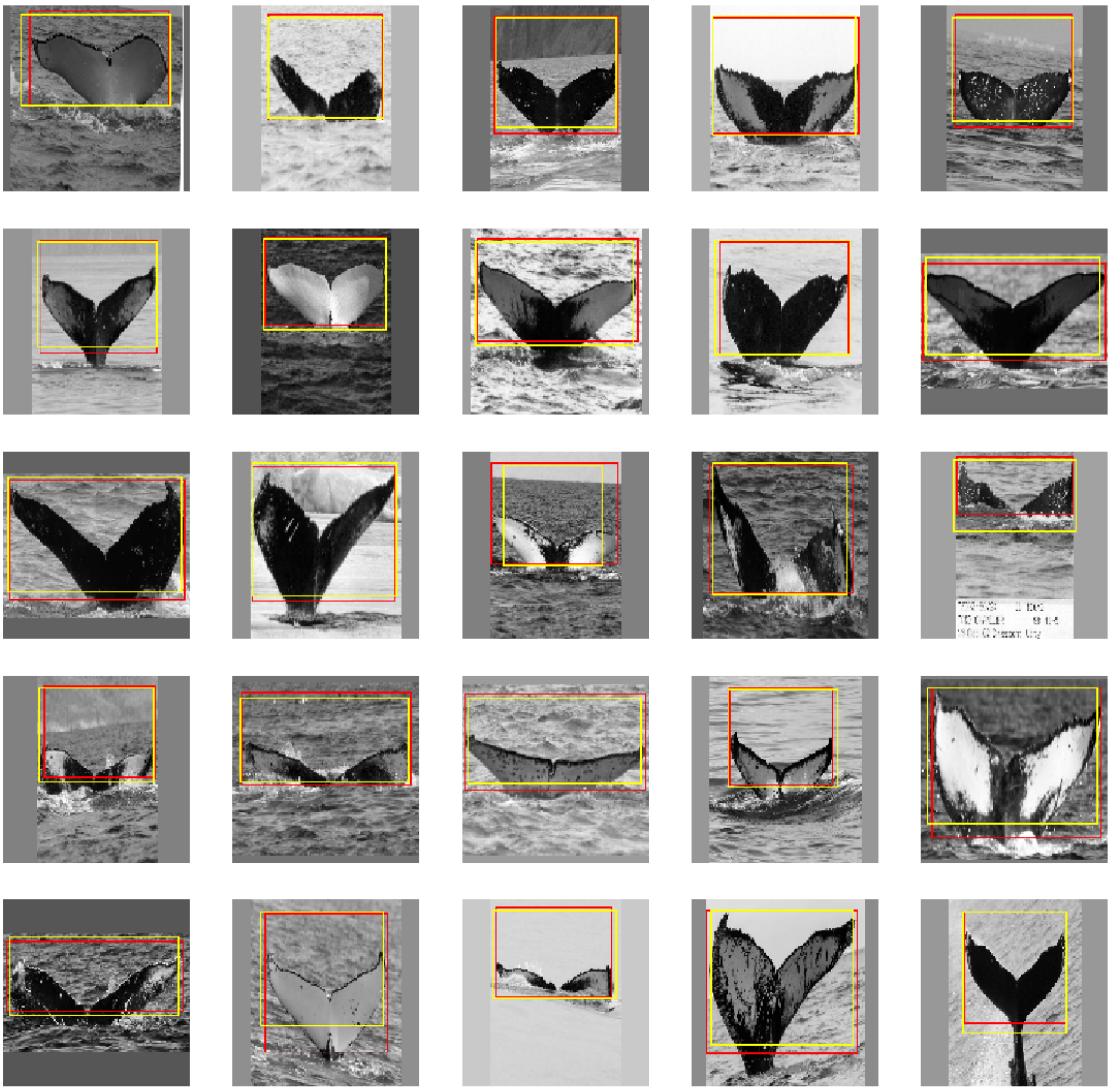
```
250/250 [==============================] - 998s 4s/step - loss: 62.66
76 - val_loss: 34.6407
Epoch 7/30
250/250 [==============================] - 998s 4s/step - loss: 58.95
70 - val_loss: 30.0171
Epoch 8/30
250/250 [==============================] - 996s 4s/step - loss: 58.33
31 - val_loss: 4016049435474.9614

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0099999997
76482582.
Epoch 9/30
250/250 [==============================] - 998s 4s/step - loss: 49.31
79 - val_loss: 24.2646
Epoch 10/30
250/250 [==============================] - 994s 4s/step - loss: 46.13
09 - val_loss: 23.8010
Epoch 11/30
250/250 [==============================] - 994s 4s/step - loss: 45.69
10 - val_loss: 25.8821
```

In [32]:
```
model.load_weights('cropping-1.h5')
loss1 = model.evaluate(val_a, val_b, verbose=0)
model.load_weights('cropping-2.h5')
loss2 = model.evaluate(val_a, val_b, verbose=0)
# model.load_weights('cropping-3.h5')
# loss3 = model.evaluate(val_a, val_b, verbose=0)
model_name = 'cropping-1.h5'
if loss2 <= loss1: model_name = 'cropping-2.h5'
# if loss3 <= loss1 and loss3 <= loss2: model_name = 'cropping-3.h5'
model.load_weights(model_name)
loss1, loss2, model_name
```

Out[32]:
```
(30.10952337164628, 19.577757905658924, 'cropping-2.h5')
```

In [33]:
```
model.save('crop.model')
```

In [34]:
```
images = []
for i,(p,coords) in enumerate(val[:25]):
    a          = val_a[i:i+1]
    rect1      = val_b[i]
    rect2      = model.predict(a).squeeze()
    img        = array_to_img(a[0]).convert('RGB')
    draw       = Draw(img)
    draw.rectangle(rect1, outline='red')
    draw.rectangle(rect2, outline='yellow')
    images.append(img)
show_whale(images)
```



In [35]:
```
from pandas import read_csv
tagged = [p for _,p,_ in read_csv('/home/lab2/DG foundation/train.csv')
submit = [p for _,p,_ in read_csv('/home/lab2/DG foundation/sample_subm
join = tagged + submit
len(join)
```

Out[35]: 33321

In [36]:
```
p2bb = {}
for p in tqdm_notebook(join):
    if p not in p2bb:
        img,trans         = read_for_validation(p)
        a                 = np.expand_dims(img, axis=0)
        x0, y0, x1, y1    = model.predict(a).squeeze()
        (u0, v0),(u1, v1) = coord_transform([(x0,y0),(x1,y1)], trans)
        p2bb[p]           = (u0, v0, u1, v1)
```

100% 33321/33321 [21:17<00:00, 24.26it/s]

In [37]:
```
print("model1 finish")
```

```
model1 finish
```

In [ ]:
```
model2 = build_model(with_dropout=False)
model2.load_weights(model_name)
model2.summary()
```

In [ ]:
```
model2.compile(Adam(lr=0.002), loss='mean_squared_error')
model2.evaluate(val_a, val_b, verbose=0)
```

In [ ]:
```
for layer in model2.layers:
    if not isinstance(layer, BatchNormalization):
        layer.trainable = False
model2.compile(Adam(lr=0.002), loss='mean_squared_error')
model2.fit_generator(TrainingData(), epochs=1, max_queue_size=12, worke
for layer in model2.layers:
    if not isinstance(layer, BatchNormalization):
        layer.trainable = True
model2.compile(Adam(lr=0.002), loss='mean_squared_error')
model2.save('cropping.model')
```

In [ ]:
```
model2.evaluate(val_a, val_b, verbose=0)
```

In [ ]:
```
images = []
for i,(p,coords) in enumerate(val[:25]):
    a         = val_a[i:i+1]
    rect1     = val_b[i]
    rect2     = model2.predict(a).squeeze()
    img       = array_to_img(a[0]).convert('RGB')
    draw      = Draw(img)
    draw.rectangle(rect1, outline='red')
    draw.rectangle(rect2, outline='yellow')
    images.append(img)
show_whale(images)
```

In [ ]:
```
from pandas import read_csv

tagged = [p for _,p,_ in read_csv('/home/lab2/DG foundation/train.csv')
submit = [p for _,p,_ in read_csv('/home/lab2/DG foundation/sample_subm
join = tagged + submit
len(join)
```

In [ ]:
```
p2bb = {}
for p in tqdm_notebook(join):
    if p not in p2bb:
        img,trans         = read_for_validation(p)
        a                 = np.expand_dims(img, axis=0)
        x0, y0, x1, y1    = model2.predict(a).squeeze()
        (u0, v0),(u1, v1) = coord_transform([(x0,y0),(x1,y1)], trans)
        p2bb[p]           = (u0, v0, u1, v1)
```

In [ ]:
```

```