Name that Fish and its Box

Saturday Hackathon

The data for this hackathon is at: https://datasets.univ.ai/kagglefish/ . This data required a NDA to sign, and it cannot be shared. Download the training zip file and the contents of the `bbox` folder. This will be about 800-900MB.

These are images of caught fish on boats from a kaggle competition. Your job is to classify the fish into the following categories, plus a "NoFish" and "Other" category.

Use `wandb init` with team `univai-ss19` and project `kagfish`.

1. Use Inception V3 and transfer learning to write a classifier to distinguish these 8 classes. You'll probably

need to retrain the entire network eventually using a real low learning rate.(10^{-4} and below). You might be able to simply pop the top and add a `GlobalAvgPool` layer, although some multi-layer perceptron with dropout action might be useful at the top. InceptionV3 is fully convolutional and should adjust to the size of the new images. (The images themselves might be of different sizes and it might just be worth rationalizing to the standard inception 299 x 299)

2. Using some of that multi-layer perceptron action, add a second head to the network to predict bounding boxes, and see if having two heads can improve the performance of your classifier. It should, but appropriate training parameters might be hard to find. Consider about 2 layers of perceptrons with BatchNorm (perhaps this will help) and dropout.

Consider using the Keras Functional API all through to make the second part possible

Some useful code

Here is some possibly useful code:

Show BBOX (from fast.ai 2017 notebooks)

```
anno_classes = ['alb', 'bet', 'dol', 'lag', 'other', 'shark', 'yft', 'NoF']
bb_json = {}
path = "bbox"
for c in anno_classes:
    if c == 'other': continue # no annotation file for "other" class
    j = json.load(open('{}/{}_labels.json'.format(path, c), 'r'))
    for l in j:
        if 'annotations' in l.keys() and len(l['annotations'])>0:
            bb_json[l['filename'].split('/')[-1]] = sorted(
                l['annotations'], key=lambda x: x['height']*x['width'])[-1]
bb_json['img_04908.jpg']
```

```
from keras import backend as K
bb_params = ['height', 'width', 'x', 'y']
config.width = 224 # or 299
config.height = 224 # or 299


def convert_bb(bb, size):
    bb = [bb[p] for p in bb_params]
    conv_x = (config.width / size[0])
    conv_y = (config.height / size[1])
    bb[0] = bb[0]*conv_y
    bb[1] = bb[1]*conv_x
```

```
    bb[2] = max(bb[2]*conv_x, 0)
    bb[3] = max(bb[3]*conv_y, 0)
    return bb

def create_rect(bb, color='red'):
    return plt.Rectangle((bb[2], bb[3]), bb[1], bb[0], color=color,
fill=False, lw=3)

  def to_plot(img):
    if K.image_dim_ordering() == 'tf':
        return np.rollaxis(img, 0, 1).astype(np.uint8)
    else:
        return np.rollaxis(img, 0, 3).astype(np.uint8)

def plotfish(img):
    plt.imshow(to_plot(img))

def show_bb(i):
    bb = val_bbox[i]
    plotfish(val[i])
    plt.gca().add_patch(create_rect(bb))
```

Useful code for the model

```
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator

# Define Model
model=...

model.compile(loss='categorical_crossentropy', optimizer = optimizer,
metrics = ['accuracy'])

# autosave best Model
best_model_file = "./weights.h5"
best_model = ModelCheckpoint(best_model_file, monitor='val_acc', verbose =
1, save_best_only = True)

train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.1,
        zoom_range=0.1,
        rotation_range=10.,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True)

```

```
# this is the augmentation configuration we will use for validation:
# only rescaling
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size = (config.width, config.height),
        batch_size = batch_size,
        shuffle = True,
        classes = anno_classes,
        class_mode = 'categorical')

validation_generator = val_datagen.flow_from_directory(
        val_data_dir,
        target_size=(config.width, config.height),
        batch_size=batch_size,
        shuffle = True,
        classes = anno_classes,
        class_mode = 'categorical')

model.fit_generator(
        train_generator,
        samples_per_epoch = config.n_train_samples,
        nb_epoch = config.epochs,
        validation_data = validation_generator,
        nb_val_samples = config.n_validation_samples,
        callbacks = [best_model, WanbCallback()])
```