Humpback Whale Identification Challenge

School of Engineering

kaggle

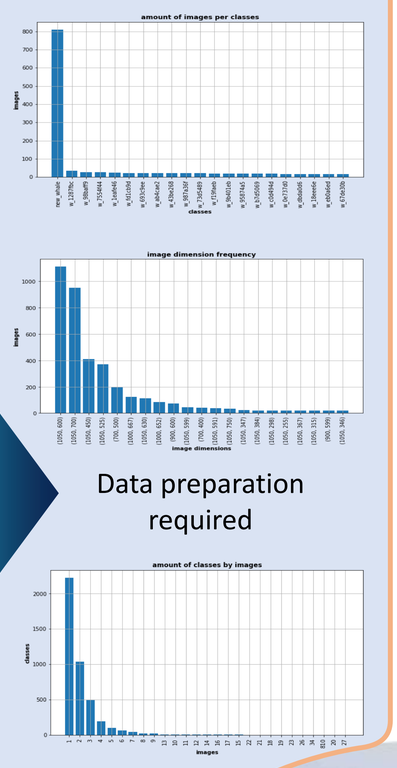
Data Analysis

- Dataset is not unitary. Images vary heavily in size, shape, resolution and color vs. greyscale
- All unidentified animals are assigned to the same class new_whale. This class includes almost 10% of the training data
- 4251 classes

	Image	Id
0	00022e1a.jpg	w_e15442c
1	000466c4.jpg	w_1287fbc
2	00087b01.jpg	w_da2efe0
3	001296d5.jpg	w_19e5482
4	0014cfdf.jpg	w_f22f3e3
5	0025e8c2.jpg	w_8b1ca89

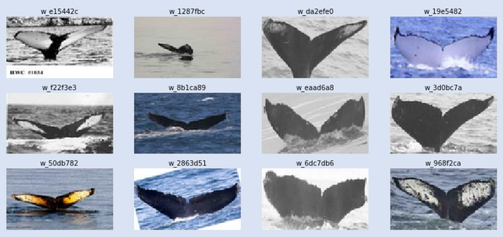
- Partially images are double assigned to two different classes.
- Partially the images include dedicated text

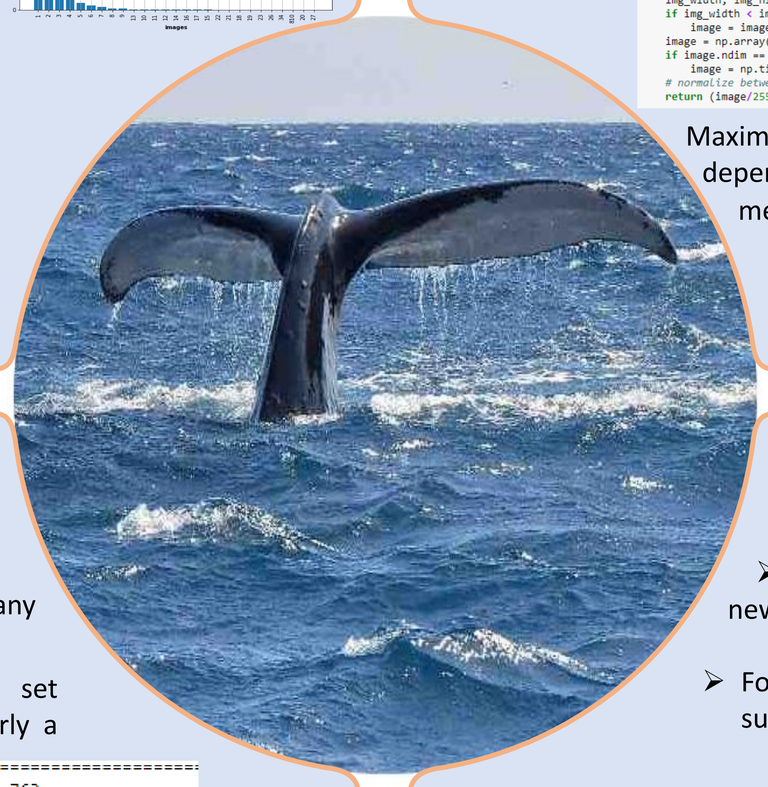
15610 test images, thereof 3453 greyscale



Data preparation required

9850 training images, thereof 1469 greyscale





Data Augmentation & Cloning

Due to the diverse data and in general few labeled pictures for each class, an extended preprocessing is essential to reach acceptable results.

```
Index(['Image', 'Id'], dtype='object')
amount of train files: 9850
amount of classes: 4251
```

```
Index(['Image', 'Id'], dtype='object')
amount of train files: 18231
amount of classes: 4251
```

Additional greyscale images are created by cloning the original RGB image to increase the amount of training images

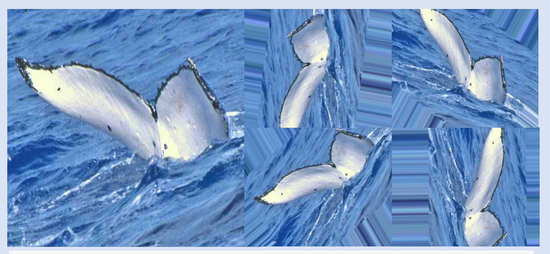
```
def add_greyscale_images(data, path, new_path):
    for i in data['Image']:
        new_image_filename = i.replace(path, new_path)
        image = Image.open(i)
        if not check_greyscale(image):
            if not os.path.exists(new_image_filename):
                image = image.convert('L')
                image.save(new_image_filename)
            row = data.loc[(data['Image'] == i), ['Image', 'Id']]
            row['Image'] = new_image_filename
            data = data.append(row)
    return data
```

- All images are resized to a common size with using the most common shape. Images in portrait format are rotated to the landscape format

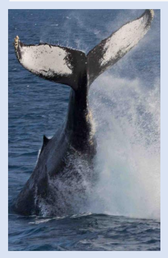
```
def pre_process(image, width = 64, height = 64):
    # if greyscale add dummy layer
    # resize to given width and height
    img_width, img_hight = image.size
    if img_width < img_hight:
        image = image.rotate(90, expand=True)
    image = np.array(image.resize((width, height), Image.ANTIALIAS))
    if image.ndim == 2:
        image = np.tile(image, (3,1,1)).transpose((1,2,0))
    # normalize between 0..1
    return (image/255)
```

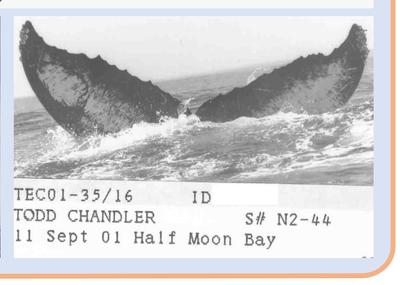
Maximum image size is heavily depending on the amount of memory

During the CNN training, the images are also processed using the *Keras ImageDataGenerator*

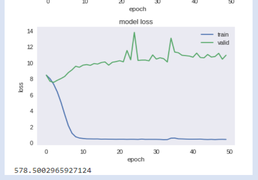


```
def data_generator(X_train, Y_train, batch_size):
    datagen = ImageDataGenerator(
        rotation_range = 30,
        width_shift_range = 0.15,
        height_shift_range = 0.15,
        horizontal_flip = True,
        #vertical_flip = True,
        #zoom_range = 0.1
        )
    train_generator = datagen.flow(
        x = X_train,
        y = Y_train,
        batch_size = batch_size,
        shuffle = True)
    return train_generator
```





Approach & Iteration

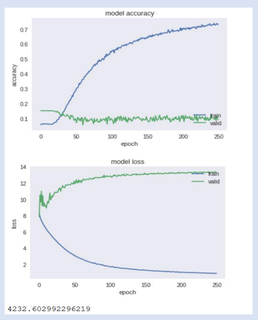


One of the first test run with a minimal CNN model. With data harmonization but without any additional data generation.

The model is heavily overfitting. Training set learning rate ist great but validation is nearly a flatline.

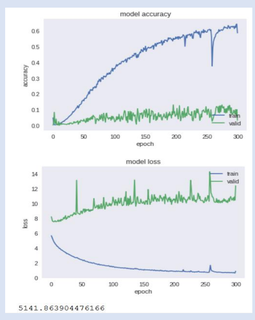
Deeper networks did not reduce the overfitting.

```
=================================================
Total params: 7,934,763
Trainable params: 7,933,547
Non-trainable params: 1,216
```



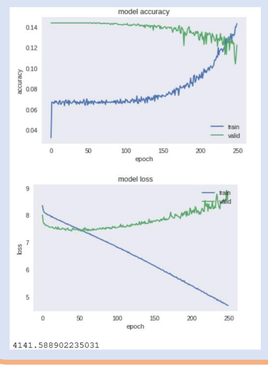
ADAM Optimizer:
Adjusting Learning Rate (LR) did not result in the anticipated profit.

LR = 0.01 «In this Example»
Default = 0.001



ADAM Optimizer with additional greyscale Images.

Lower start loss, but no improvement regarding overfitting



SGD Optimizer:
Default values are used except momentum

«Parameter that accelerates SGD in the relevant direction and dampens oscillations»

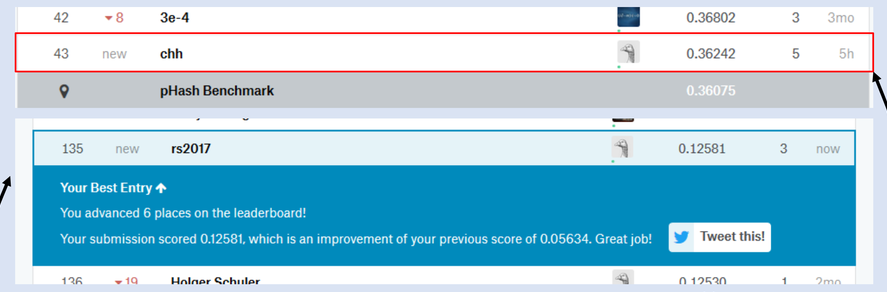
Add class weight to punish the classes with more images

```
def train_model(X_train, Y_train, X_valid, Y_valid, batch_size, number_of_epoch, log_path, class_weight, sample_weight):
    tensorboard = keras.callbacks.TensorBoard(
        log_dir=log_path,
        write_graph=True,
        histogram_freq=1)
    checkpointer = keras.callbacks.ModelCheckpoint(
        filepath = log_path + '/dl_whale_project_v4_ch_' + 'weights_epoch_{epoch:03d}-{val_loss:.2f}.hdf5',
        verbose = 1,
        save_best_only = False,
        period = 5)
    start = time.time()
    history = model.fit_generator(data_generator(X_train, Y_train, batch_size),
                                  steps_per_epoch = len(X_train) // batch_size,
                                  epochs = number_of_epoch,
                                  verbose=2,
                                  #class_weight = 'balanced',
                                  class_weight = class_weight,
                                  #sample_weight = sample_weight,
                                  validation_data=(X_valid, Y_valid),
                                  callbacks=[tensorboard, checkpointer]
                                  )
    end = time.time()
    return history, start, end
```

Outcome & Ideas

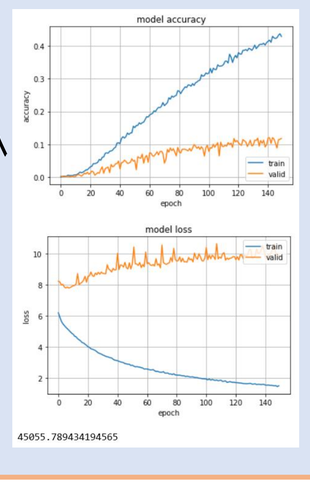
- As root cause for the bad generalization of the CNN, the dominant new_whale class with its high variety of different whales was identified.
- For further proceedings, this class was split into many smaller subclasses as well as completely ignored during the training.

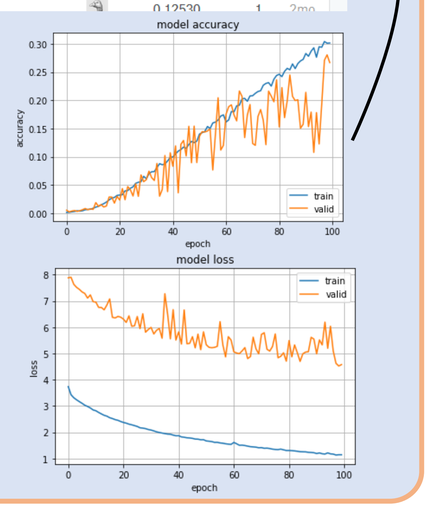
After many, many hours of modeling and trouble shooting:



Some more ideas to increase the score:

- Add more Data
- Detect and filter the wrong labeled images
- Ensembles





Data Set Description:

After centuries of intense whaling, recovering whale populations still have a hard time adapting to warming oceans and struggle to compete every day with the industrial fishing industry for food.

To aid whale conservation efforts, scientists use photo surveillance systems to monitor ocean activity. They use the shape of whales' tails and unique markings found in footage to identify what species of whale they're analyzing and meticulously log whale pod dynamics and movements. For the past 40 years, most of this work has been done manually by individual scientists, leaving a huge trove of data untapped and underutilized.

In this competition, you're challenged to build an algorithm to identifying whale species in images. You'll analyze Happy Whale's database of over 25,000 images, gathered from research institutions and public contributors. By contributing, you'll help to open rich fields of understanding for marine mammal population dynamics around the globe.

https://www.kaggle.com/c/whale-categorization-playground

Project Team:

Roger Schwyn, Christoph Hubmann and Hubert Keller