CS 3202 Kaggle Competition - Histopathologic Cancer Detection Spring 2021

Author: Sabine Hollatz

The Goal of the Competition

Train a binary image classifier that can detect metastatic cancer in histological images. The more accurate the classifiers prediction rate is, the better is the diagnostic precision. The task is clinically-relevant.

The Data

The images are taken from larger digital pathology scans. The dataset is very similar to the PatchCamelyon (PCam) benchmark dataset, but does not contain duplicates. It was provided by Bas Veeling, with additional input from Babak Ehteshami Bejnordi, Geert Litjens, and Jeroen van der Laak and is published under the CC0 License, following the license of Camelyon16.

The color images are 96x96px in size. The trainingsset contains 220025 images and the testset contains 57458 images. Either an image contains metastatic cancer cells or it does not.

In [1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from scipy import ndimage
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from PIL import Image
import glob
```

Loading the Data

In [2]:
```
labelpath = "/data/hollatz/deep_learning/datasets/tutorial_datasets/histopathologic_cancer_detection/train_labels.csv"
folderpath_train = "/data/hollatz/deep_learning/datasets/tutorial_datasets/histopathologic_cancer_detection/train/"
folderpath_test = "/data/hollatz/deep_learning/datasets/tutorial_datasets/histopathologic_cancer_detection/test/"

df_labels = pd.read_csv(labelpath,
                              skiprows=1,
                              sep=',',
                              names=['id', 'label'],
                              index_col='id')
df_labels.head()
```

Out[2]:

id	label
f38a6374c348f90b587e046aac6079959adf3835	0
c18f2d887b7ae4f6742ee445113fa1aef383ed77	1
755db6279dae599ebb4d39a9123cce439965282d	0
bc3f0c64fb968ff4a8bd33af6971ecae77c75e08	0
068aba587a4950175d04c680d38943fd488d6a9d	0

In [3]:
```
all_train_imgs = []
for filename, _ in df_labels.iterrows():
    img_path = folderpath_train + filename + ".tif"
    img_frame = Image.open(img_path)
    img_array = np.asarray(img_frame)
    all_train_imgs.append(img_array)
```

In [40]:
```
all_test_imgs = []
all_test_files = []
for img_path in glob.glob(folderpath_test + "*.tif"):
    img_frame = Image.open(img_path)
    img_array = np.asarray(img_frame)
    all_test_imgs.append(img_array)
    filename = img_path.split("/")[-1]
    file_id = filename.split(".")[0]
    all_test_files.append(file_id)
```

Exploratory Data Analysis

In [7]:
```
print(np.array(all_train_imgs).shape)
print(np.array(all_test_imgs).shape)
```

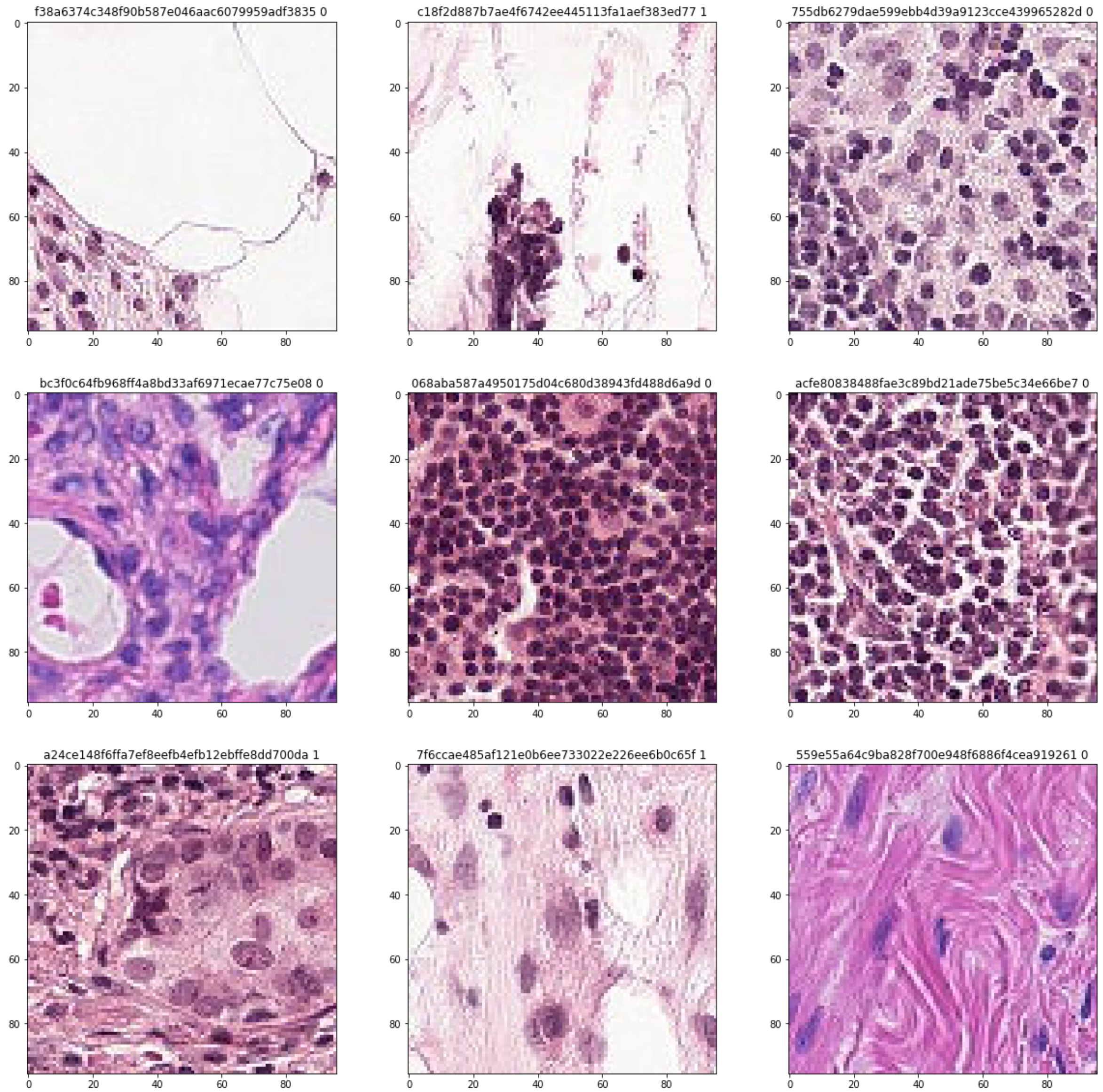
```
(220025, 96, 96, 3)
(57458, 96, 96, 3)
```

There are 220,025 rgb training images of size 96x96x3 pixels and 57,458 testing images of the same size.

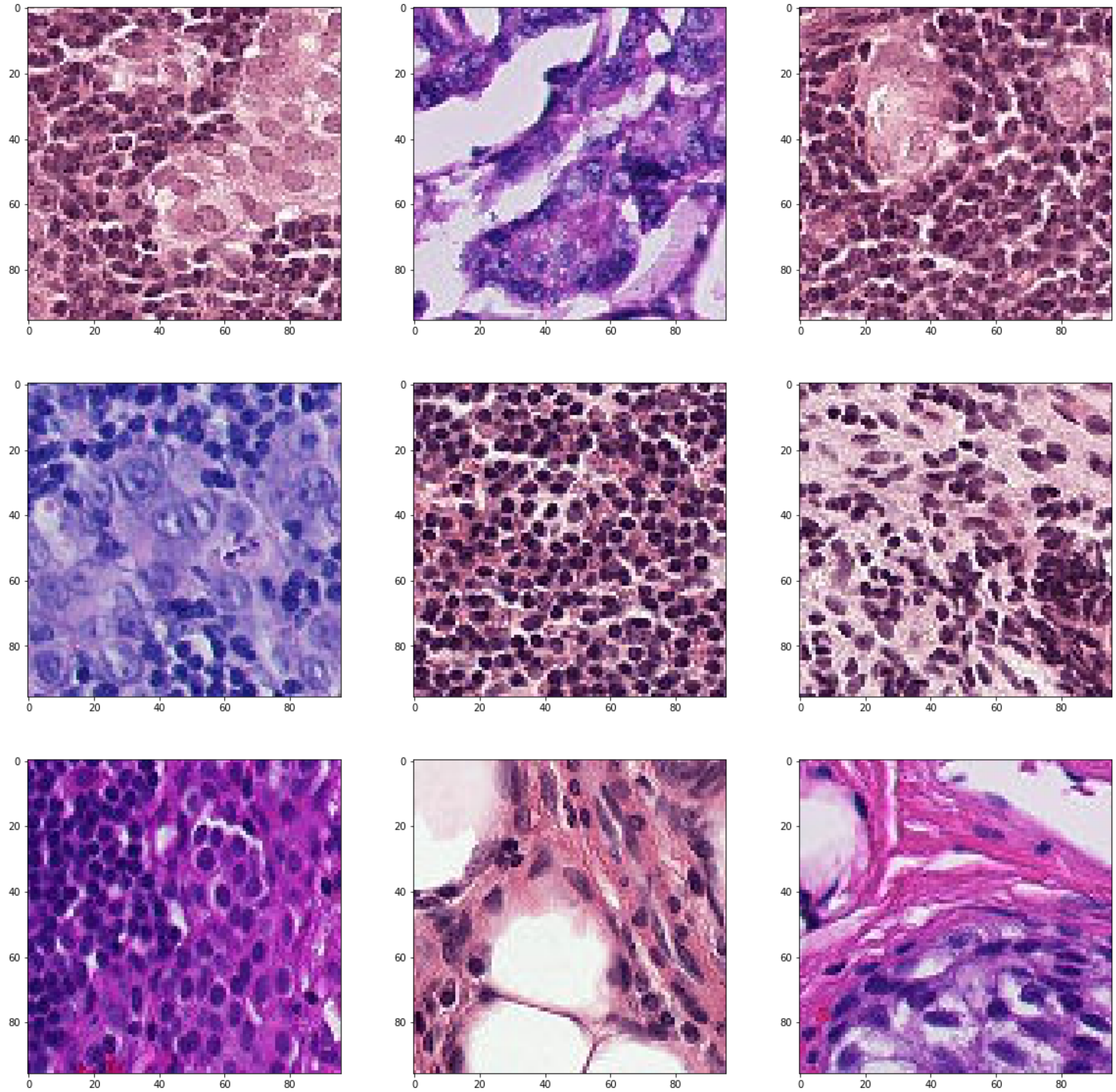
In [12]:
```
X_train = np.array(all_train_imgs)
y_train = df_labels
X_test = np.array(all_test_imgs)
print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)
```

```
('X_train shape:', (220025, 96, 96, 3))
('y_train shape:', (220025, 1))
```

In [8]:
```
# Displaying a sample of the training images
fig = plt.figure(figsize=(20,20))
for o in range(9):
    fig.add_subplot(3, 3, o+1)
    file = df_labels.index[o]
    label = str(df_labels['label'][o])
    plt.title(file + " " + label)
    plt.imshow(all_train_imgs[o].reshape(all_train_imgs[o].shape[0],
all_train_imgs[o].shape[1], 3))
plt.show()
```



In [10]:
```
# Displaying a sample of the testing images
fig = plt.figure(figsize=(20,20))
for o in range(9):
    fig.add_subplot(3, 3, o+1)
    plt.imshow(all_test_imgs[o].reshape(all_test_imgs[o].shape[0], al
l_test_imgs[o].shape[1], 3))
plt.show()
```

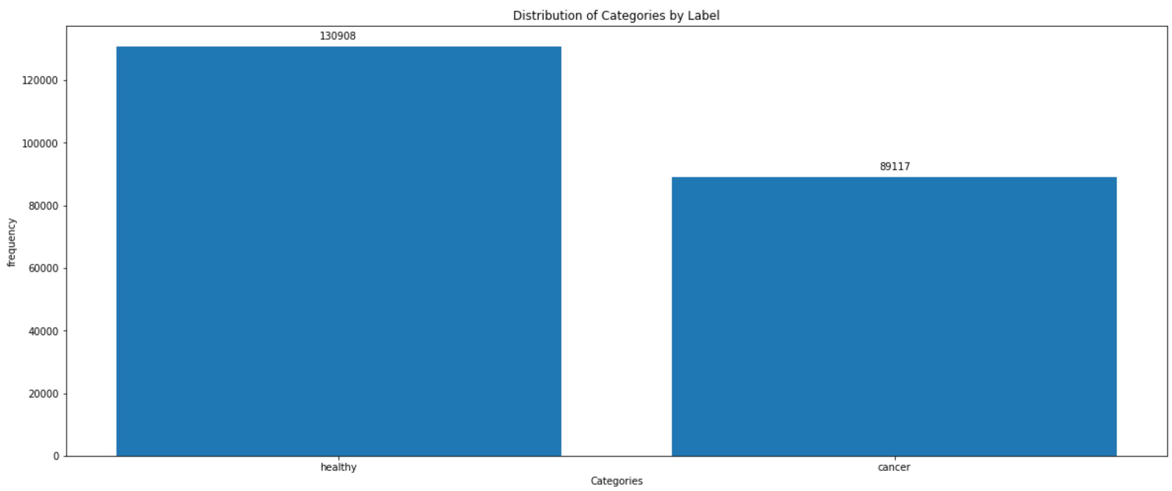


In [14]:
```
# Checking for an even distribution of images over the binary categor
y.
def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its
 height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 5),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

def fill_counter(df, cats):
    counter = np.zeros(len(cats), dtype=int)
    for i, cat in enumerate(cats):
        counter[i] = df[cat][df[cat] == 1].count()
    return counter

counter = np.zeros(2, dtype=int)
counter[0] = df_labels['label'][df_labels['label'] == 0].count()
counter[1] = df_labels['label'][df_labels['label'] == 1].count()

fig, ax = plt.subplots(figsize=(20,8))
rects = plt.bar(['healthy', 'cancer'], counter, label="distribution o
f categories by label")
autolabel(rects)
plt.title("Distribution of Categories by Label")
plt.xticks([0, 1], ['healthy', 'cancer'])
plt.ylabel('frequency')
plt.xlabel('Categories');
```



There are 130,908 images that do not show metastatic cancer and 89,117 images that do show cancer cells. The difference of more than 50,000 images is probably fine, because the sample size for each outcome is still large with at least 89,117 images.

In [15]:
```
df_labels.isna().sum()
```

Out[15]:
```
label    0
dtype: int64
```

In [16]:
```
df_labels.isnull().sum()
```

Out[16]:
```
label    0
dtype: int64
```

All training images have a valid label.

Preprocessing the Data

In [13]:
```
# Partitioning for cross-validation
xTrain, xVal, yTrain, yVal = train_test_split(X_train, y_train)
print(xTrain.shape, yTrain.shape)
print(xVal.shape, yVal.shape)
```

```
((165018, 96, 96, 3), (165018, 1))
((55007, 96, 96, 3), (55007, 1))
```

In [17]:
```
# rescaling to keep the neural network computations with smaller numb
ers.
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    data_format = "channels_last")

val_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    data_format = "channels_last")
```

The Model Architecture

Convolutional Neural Networks (CNNs) are well suited for image data because they preserve the spatial pixel information in an image. The convolutional neural layers provide kernels that can learn and filter the input image for particular patterns. I have started out with the established model VGG16, but could not achieve a validation accuracy larger than 0.60 and the model did not show proper learning behavior. I reduced the number of convolutional layers to 4 and could see a better starting loss and some learning behavior, but the validation accuracy did not improve significantly. When I added Dropout and L2 regularization to the classifier, I started to see proper learning behavor and an improved accuracy. Since this is just a binary classification task, I kept the number of convolutional layers and the number of filters relatively small. MaxPooling enables the preservation of the strongest contrast within a particular area of the image, while downsampling the feature maps. Dropout and L2 regularization prevent overfitting, so that the bias can be kept as low as possible. The sigmoid activation function at the end keeps the output between 0 and 1, which can be used in binary classification with a threshold of 0.5 for example. Any image with a predictive value larger than 0.5 is interpreted as showing metastatic cancer. By trial and error was the learning rate set to 0.00005. However, there does not seem to be a major performance difference with learning rates between 0.001 and 0.00005. But the learning behavior looked better with a smaller learning rate.

In [18]:
```
import tensorflow.keras as k

model = k.Sequential()
model.add(k.layers.Conv2D(32, (3, 3), input_shape=(96, 96, 3)))
model.add(k.layers.Activation('relu'))
model.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(k.layers.Conv2D(32, (3, 3)))
model.add(k.layers.Activation('relu'))
model.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(k.layers.Conv2D(64, (3, 3)))
model.add(k.layers.Activation('relu'))
model.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(k.layers.Conv2D(64, (3, 3)))
model.add(k.layers.Activation('relu'))
model.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(k.layers.Flatten())
model.add(k.layers.Dense(2048, kernel_regularizer='l2'))
model.add(k.layers.Activation("relu"))
model.add(k.layers.Dropout(0.5))
model.add(k.layers.Dense(64, kernel_regularizer='l2'))
model.add(k.layers.Activation('sigmoid'))
model.add(k.layers.Dropout(0.5))
model.add(k.layers.Dense(1))

callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=
5)

model.compile(optimizer=tf.keras.optimizers.Adam(0.00005),
              loss='binary_crossentropy',
              metrics=["accuracy"])
```

```
WARNING:tensorflow:From /usr/lib/python2.7/site-packages/tensorflow/p
ython/ops/resource_variable_ops.py:435: colocate_with (from tensorflo
w.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/lib/python2.7/site-packages/tensorflow/p
ython/keras/layers/core.py:143: calling dropout (from tensorflow.pyth
on.ops.nn_ops) with keep_prob is deprecated and will be removed in a
future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate
= 1 - keep_prob`.
```

In [19]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 94, 94, 32)        896       
_________________________________________________________________
activation (Activation)      (None, 94, 94, 32)        0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 47, 47, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 45, 45, 32)        9248      
_________________________________________________________________
activation_1 (Activation)    (None, 45, 45, 32)        0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 64)        18496     
_________________________________________________________________
activation_2 (Activation)    (None, 20, 20, 64)        0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          36928     
_________________________________________________________________
activation_3 (Activation)    (None, 8, 8, 64)          0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 64)          0         
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0         
_________________________________________________________________
dense (Dense)                (None, 2048)              2099200   
_________________________________________________________________
activation_4 (Activation)    (None, 2048)              0         
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                131136    
_________________________________________________________________
activation_5 (Activation)    (None, 64)                0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65        
=================================================================
Total params: 2,295,969
Trainable params: 2,295,969
Non-trainable params: 0
_________________________________________________________________
```

Training and further Hypterparameter Tuning

2 different models have been trained with the same architecture. One with the Adam optimizer and a fixed learning rate and the other with the Adam optimizer and a learning rate scheduler. The following is conducted with the simple learning rate of lr= 0.00005.

In [20]:
```
epochs = 100
batchsize = 64
history = model.fit(train_datagen.flow(xTrain, yTrain, batch_size=bat
chsize),
                    validation_data=val_datagen.flow(xVal, yVal, batc
h_size=batchsize),
                    steps_per_epoch=len(xTrain) / batchsize,
                    shuffle=True,
                    callbacks=[callback],
                    epochs=epochs)
```

```
WARNING:tensorflow:From /usr/lib/python2.7/site-packages/tensorflow/p
ython/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math
_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/100
860/860 [==============================] - 6s 7ms/step - loss: 2.8428
- acc: 0.5955
2579/2579 [==============================] - 53s 21ms/step - loss: 6.
1943 - acc: 0.5482 - val_loss: 2.8428 - val_acc: 0.5955
Epoch 2/100
860/860 [==============================] - 7s 8ms/step - loss: 2.0441
- acc: 0.6337
2579/2579 [==============================] - 40s 16ms/step - loss: 2.
4681 - acc: 0.6065 - val_loss: 2.0441 - val_acc: 0.6337
Epoch 3/100
860/860 [==============================] - 6s 8ms/step - loss: 1.7236
- acc: 0.5959
2579/2579 [==============================] - 34s 13ms/step - loss: 1.
7703 - acc: 0.6796 - val_loss: 1.7236 - val_acc: 0.5959
Epoch 4/100
860/860 [==============================] - 6s 7ms/step - loss: 1.1239
- acc: 0.7483
2579/2579 [==============================] - 35s 14ms/step - loss: 1.
3767 - acc: 0.6812 - val_loss: 1.1239 - val_acc: 0.7483
Epoch 5/100
860/860 [==============================] - 6s 7ms/step - loss: 0.9019
- acc: 0.7679
2579/2579 [==============================] - 34s 13ms/step - loss: 1.
0976 - acc: 0.7121 - val_loss: 0.9019 - val_acc: 0.7679
Epoch 6/100
860/860 [==============================] - 6s 7ms/step - loss: 0.7843
- acc: 0.7114
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
8985 - acc: 0.7171 - val_loss: 0.7843 - val_acc: 0.7114
Epoch 7/100
860/860 [==============================] - 7s 8ms/step - loss: 0.6302
- acc: 0.7717
2579/2579 [==============================] - 35s 14ms/step - loss: 0.
7724 - acc: 0.7000 - val_loss: 0.6302 - val_acc: 0.7717
Epoch 8/100
860/860 [==============================] - 10s 12ms/step - loss: 0.62
16 - acc: 0.7214
2579/2579 [==============================] - 38s 15ms/step - loss: 0.
6767 - acc: 0.7261 - val_loss: 0.6216 - val_acc: 0.7214
Epoch 9/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5742
- acc: 0.7506
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6510 - acc: 0.7220 - val_loss: 0.5742 - val_acc: 0.7506
Epoch 10/100
860/860 [==============================] - 6s 7ms/step - loss: 0.6487
- acc: 0.6364
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6399 - acc: 0.7196 - val_loss: 0.6487 - val_acc: 0.6364
Epoch 11/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5535
```

```
- acc: 0.7786
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6610 - acc: 0.7069 - val_loss: 0.5535 - val_acc: 0.7786
Epoch 12/100
860/860 [==============================] - 9s 10ms/step - loss: 0.674
1 - acc: 0.6069
2579/2579 [==============================] - 37s 14ms/step - loss: 0.
6848 - acc: 0.6690 - val_loss: 0.6741 - val_acc: 0.6069
Epoch 13/100
860/860 [==============================] - 7s 8ms/step - loss: 0.5424
- acc: 0.7793
2579/2579 [==============================] - 35s 14ms/step - loss: 0.
6341 - acc: 0.7197 - val_loss: 0.5424 - val_acc: 0.7793
Epoch 14/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5298
- acc: 0.7798
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6014 - acc: 0.7443 - val_loss: 0.5298 - val_acc: 0.7798
Epoch 15/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5361
- acc: 0.7766
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6193 - acc: 0.7286 - val_loss: 0.5361 - val_acc: 0.7766
Epoch 16/100
860/860 [==============================] - 7s 8ms/step - loss: 0.5264
- acc: 0.7778
2579/2579 [==============================] - 35s 14ms/step - loss: 0.
6045 - acc: 0.7381 - val_loss: 0.5264 - val_acc: 0.7778
Epoch 17/100
860/860 [==============================] - 9s 10ms/step - loss: 0.513
8 - acc: 0.7867
2579/2579 [==============================] - 35s 14ms/step - loss: 0.
5929 - acc: 0.7496 - val_loss: 0.5138 - val_acc: 0.7867
Epoch 18/100
860/860 [==============================] - 7s 8ms/step - loss: 0.6042
- acc: 0.7110
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6154 - acc: 0.7226 - val_loss: 0.6042 - val_acc: 0.7110
Epoch 19/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5322
- acc: 0.7762
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6083 - acc: 0.7418 - val_loss: 0.5322 - val_acc: 0.7762
Epoch 20/100
860/860 [==============================] - 7s 8ms/step - loss: 0.5165
- acc: 0.7841
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
5954 - acc: 0.7487 - val_loss: 0.5165 - val_acc: 0.7841
Epoch 21/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5306
- acc: 0.7890
2579/2579 [==============================] - 32s 13ms/step - loss: 0.
5872 - acc: 0.7557 - val_loss: 0.5306 - val_acc: 0.7890
Epoch 22/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4981
- acc: 0.7901
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
```

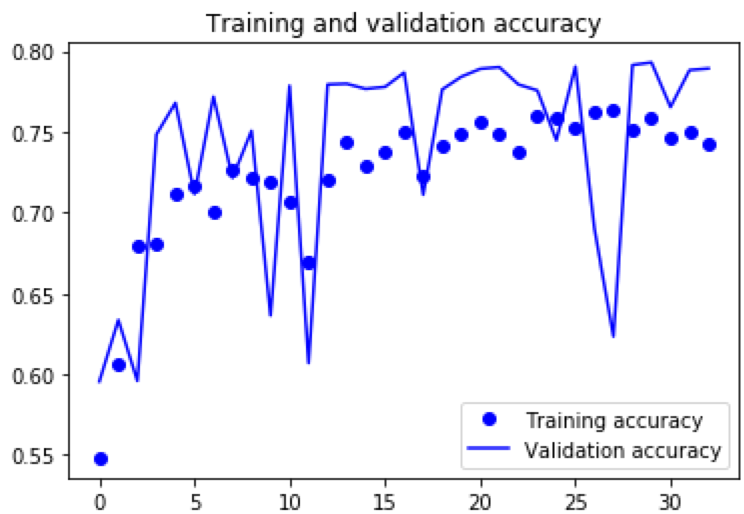
```
5864 - acc: 0.7481 - val_loss: 0.4981 - val_acc: 0.7901
Epoch 23/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5287
- acc: 0.7793
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5993 - acc: 0.7377 - val_loss: 0.5287 - val_acc: 0.7793
Epoch 24/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5193
- acc: 0.7756
2579/2579 [==============================] - 32s 12ms/step - loss: 0.
5760 - acc: 0.7598 - val_loss: 0.5193 - val_acc: 0.7756
Epoch 25/100
860/860 [==============================] - 6s 8ms/step - loss: 0.5750
- acc: 0.7447
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5687 - acc: 0.7581 - val_loss: 0.5750 - val_acc: 0.7447
Epoch 26/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5164
- acc: 0.7904
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5726 - acc: 0.7519 - val_loss: 0.5164 - val_acc: 0.7904
Epoch 27/100
860/860 [==============================] - 6s 7ms/step - loss: 0.6281
- acc: 0.6901
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5585 - acc: 0.7620 - val_loss: 0.6281 - val_acc: 0.6901
Epoch 28/100
860/860 [==============================] - 6s 7ms/step - loss: 0.6421
- acc: 0.6234
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5561 - acc: 0.7637 - val_loss: 0.6421 - val_acc: 0.6234
Epoch 29/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5104
- acc: 0.7914
2579/2579 [==============================] - 32s 13ms/step - loss: 0.
5764 - acc: 0.7505 - val_loss: 0.5104 - val_acc: 0.7914
Epoch 30/100
860/860 [==============================] - 6s 8ms/step - loss: 0.4976
- acc: 0.7930
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5699 - acc: 0.7589 - val_loss: 0.4976 - val_acc: 0.7930
Epoch 31/100
860/860 [==============================] - 7s 8ms/step - loss: 0.5614
- acc: 0.7653
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5834 - acc: 0.7462 - val_loss: 0.5614 - val_acc: 0.7653
Epoch 32/100
860/860 [==============================] - 6s 8ms/step - loss: 0.5256
- acc: 0.7882
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5765 - acc: 0.7505 - val_loss: 0.5256 - val_acc: 0.7882
Epoch 33/100
860/860 [==============================] - 6s 7ms/step - loss: 0.5119
- acc: 0.7892
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5843 - acc: 0.7423 - val_loss: 0.5119 - val_acc: 0.7892
```

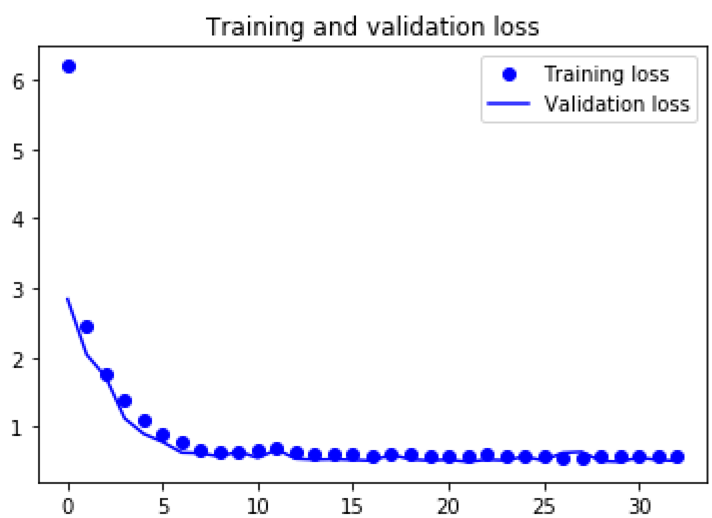
In [22]:
```
testEval = model.evaluate(xVal, yVal, verbose=0)
print('Test loss:', testEval[0])
print('Test accuracy:', testEval[1])
```

```
('Test loss:', 0.5332510370224037)
('Test accuracy:', 0.78884506)
```

In [26]:
```
accuracy = history.history['acc']
valAccuracy = history.history['val_acc']
loss = history.history['loss']
valLoss = history.history['val_loss']
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, valAccuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, valLoss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





The training loss and validation loss curves look fine, but the validation accuracy curve looks very bumpy. It seems that the model does not converge completely, but alternates between certain values. The learning rate of 0.00005 still seems to be too large close to the local minima, so that the error bounces forth and back. However, the results from this model will be submitted to the kaggle competition, to get an idea of its relative performance.

In [44]:
```
predictions = np.rint(model.predict(X_test))
predictions
```

Out[44]:
```
array([[1.],
       [0.],
       [0.],
       ...,
       [0.],
       [1.],
       [0.]], dtype=float32)
```

In [45]:
```
predSubmission = pd.DataFrame(predictions.astype('int'),
                              columns = ['label'])

predSubmission.index.name = 'id'
predSubmission.index = all_test_files

predSubmission
```

Out[45]:

	label
00006537328c33e284c973d7b39d340809f7271b	1
0000ec92553fda4ce39889f9226ace43cae3364e	0
00024a6dee61f12f7856b0fc6be20bc7a48ba3d2	0
000253dfaa0be9d0d100283b22284ab2f6b643f6	0
000270442cc15af719583a8172c87cd2bd9c7746	0
000309e669fa3b18fb0ed6a253a2850cce751a95	0
000360e0d8358db520b5c7564ac70c5706a0beb0	0
00040095a4a671280aeb66cb0c9231e6216633b5	1
000698b7df308d75ec9559ef473a588c513a68aa	1
0006e1af5670323331d09880924381d67d79eda0	1
000997a6038fa7441aa0111ac456255060a354c4	0
000c8db3e09f1c0f3652117cf84d78aae100e5a7	0
000de14191f3bab4d2d6a7384ca0e5aa5dc0dffe	1
000e6341cf18365d35b40f4991002fec8834afc0	0
0010e2887e0b977fcdfdf4c50564fafbbc2b6208	0
0010e7eaa3d8e14203cd3900b739d8bf0f0b67f0	0
001161a2eca200f565f12870048a78fa5b320dee	0
0011807dd1e3306ff3f7a755fd3efbefa2901dce	0
00118bec91b7fae175791896f7011ff506b3d7dd	0
0011f0596a038fc8daec4fde71465e347515392e	0
0014fdb3da986174f9a1d7ae95f3b75a2d025a57	0
00153be8e27526f9c2f035aff25ca9264db0a2ed	0
0016dddb2797d3fafb38c2651c5589d92030e835	0
00179c97cd2aaeafcbce352aa387db1b76616a53	1
00186a82134c399e8bb77d038475741d696cb72b	0
001d59af39e296eb8323a190f58cf498f66fb08d	0
001da402e229c18631c3dd7b8953e6f7e5d3ce99	0
001e385106b858d95c122a1cb3a930d3f30e67de	1
001e92257a59ac137d99f52419884ca60e0b91ee	0
0020b6b5a91d83fe3ba41ea6b89113db0848dac4	1
...	...
ffe7f2205d750486ba4b9ec56a9e7011c29633a0	0
ffe937550aed66e3373d31abd16608c7a0481337	0
ffeb0ee2add600e97d47379862626b18af7a07a2	0
ffecc7469b67e16e16456e464a99bb5620f0cd9f	0

	label
ffee4360d4273351e7283db5494fa1a6a6ca64b0	1
ffeea8c422d6db1b243738eb11343712ca368327	0
ffeec967ee78e98a8e55645c41f13b8ff5e97d0e	1
fff03179712a2c3dabf9b8c7bbb5ce1c767d6f17	0
fff204670ae06ad729b2fd57b65cd8a2074626c2	0
fff3711c84005cea2cb1adfa9ff47541d49567bd	1
fff44d16df02bd8e3d8cd01de693c2cbc80dfd47	0
fff48816ec621b3b2b4444279ac01382b74b5058	1
fff49294e77f806ea924eff8484356ab75feef7b	0
fff590742f1679a2d82945956e944463333ac7e0	0
fff69b7e90d979503222591557d5191dc74e9c9a	1
fff6aef707a38f6976ec5979c01d295671a7236e	0
fff726415ca9dbc3da87715e68787153b7a06b99	0
fff8c257c9fa44cd2604ff1887f25cc337565147	0
fff933e08e23a51cc6401d116e3c18999330b8dd	0
fffa681f68c58839f9fad90be2ac9cba6ab6f6a5	0
fffb109428e7c1ff5cfaa8ba49c657fc13828bf8	0
fffb3f6c8be9fb4d19245e6d5dbe003b03c70771	0
fffbaf2d17fc95a855fa833bd08dfc20aa18d232	0
fffbc065982ddf01c39e4c74e62d1868551d0b25	1
fffbfb279c9c378af5c362363b841a38bbee3294	0
fffdd1cbb1ac0800f65309f344dd15e9331e1c53	0
fffdf4b82ba01f9cae88b9fa45be103344d9f6e3	0
fffec7da56b54258038b0d382b3d55010eceb9d7	0
ffff276d06a9e3fffc456f2a5a7a3fd1a2d322c6	1
ffffeb4c0756098c7f589b7beec08ef1899093b5	0

57458 rows × 1 columns

In [46]:
```
predSubmission.to_csv('SHollatz_submission.csv')
```

The predictions reached a competition score of 0.8007 which matches rank 1024. I will try to improve the convergence and overall performance of the model by adding a learning rate scheduler.

In [47]:
```
model2 = k.Sequential()
model2.add(k.layers.Conv2D(32, (3, 3), input_shape=(96, 96, 3)))
model2.add(k.layers.Activation('relu'))
model2.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model2.add(k.layers.Conv2D(32, (3, 3)))
model2.add(k.layers.Activation('relu'))
model2.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model2.add(k.layers.Conv2D(64, (3, 3)))
model2.add(k.layers.Activation('relu'))
model2.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model2.add(k.layers.Conv2D(64, (3, 3)))
model2.add(k.layers.Activation('relu'))
model2.add(k.layers.MaxPooling2D(pool_size=(2, 2)))

model2.add(k.layers.Flatten())
model2.add(k.layers.Dense(2048, kernel_regularizer='l2'))
model2.add(k.layers.Activation("relu"))
model2.add(k.layers.Dropout(0.5))
model2.add(k.layers.Dense(64, kernel_regularizer='l2'))
model2.add(k.layers.Activation('sigmoid'))
model2.add(k.layers.Dropout(0.5))
model2.add(k.layers.Dense(1))

early_stopping = tf.keras.callbacks.EarlyStopping(monitor='loss', pat
ience=5)
```

In [50]:
```
import math
def step_decay(epoch):
    initial_lrate = 0.001
    drop = 0.5
    epochs_drop = 5.0
    lrate = initial_lrate * math.pow(drop,
                                     math.floor((1+epoch)/epochs_drop
))
    return lrate

lrate = tf.keras.callbacks.LearningRateScheduler(step_decay, verbose=
1)

model2.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='binary_crossentropy',
              metrics=["accuracy"])
```

In [51]:
```
epochs = 100
batchsize = 64
history2 = model2.fit(train_datagen.flow(xTrain, yTrain, batch_size=b
atchsize),
                    validation_data=val_datagen.flow(xVal, yVal, batc
h_size=batchsize),
                    steps_per_epoch=len(xTrain) / batchsize,
                    shuffle=True,
                    callbacks=[early_stopping, lrate],
                    epochs=epochs)
```

```
Epoch 00001: LearningRateScheduler reducing learning rate to 0.001.
Epoch 1/100
860/860 [==============================] - 6s 7ms/step - loss: 0.7610
- acc: 0.5955
2579/2579 [==============================] - 35s 14ms/step - loss: 2.
4802 - acc: 0.5763 - val_loss: 0.7610 - val_acc: 0.5955

Epoch 00002: LearningRateScheduler reducing learning rate to 0.001.
Epoch 2/100
860/860 [==============================] - 6s 7ms/step - loss: 0.6882
- acc: 0.5955
2579/2579 [==============================] - 38s 15ms/step - loss: 0.
7534 - acc: 0.5736 - val_loss: 0.6882 - val_acc: 0.5955

Epoch 00003: LearningRateScheduler reducing learning rate to 0.001.
Epoch 3/100
860/860 [==============================] - 6s 7ms/step - loss: 0.6928
- acc: 0.5955
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
7018 - acc: 0.5909 - val_loss: 0.6928 - val_acc: 0.5955

Epoch 00004: LearningRateScheduler reducing learning rate to 0.001.
Epoch 4/100
860/860 [==============================] - 7s 8ms/step - loss: 0.6939
- acc: 0.5955
2579/2579 [==============================] - 35s 13ms/step - loss: 0.
7029 - acc: 0.5909 - val_loss: 0.6939 - val_acc: 0.5955

Epoch 00005: LearningRateScheduler reducing learning rate to 0.0005.
Epoch 5/100
860/860 [==============================] - 7s 8ms/step - loss: 0.6831
- acc: 0.5955
2579/2579 [==============================] - 39s 15ms/step - loss: 0.
6909 - acc: 0.5926 - val_loss: 0.6831 - val_acc: 0.5955

Epoch 00006: LearningRateScheduler reducing learning rate to 0.0005.
Epoch 6/100
860/860 [==============================] - 7s 8ms/step - loss: 0.6784
- acc: 0.5955
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
6819 - acc: 0.5942 - val_loss: 0.6784 - val_acc: 0.5955

Epoch 00007: LearningRateScheduler reducing learning rate to 0.0005.
Epoch 7/100
860/860 [==============================] - 6s 8ms/step - loss: 0.6794
- acc: 0.5955
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6800 - acc: 0.5945 - val_loss: 0.6794 - val_acc: 0.5955

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0005.
Epoch 8/100
860/860 [==============================] - 7s 8ms/step - loss: 0.6799
- acc: 0.5955
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6804 - acc: 0.5961 - val_loss: 0.6799 - val_acc: 0.5955

Epoch 00009: LearningRateScheduler reducing learning rate to 0.0005.
```

```
Epoch 9/100
860/860 [==============================] - 6s 7ms/step - loss: 0.6779
- acc: 0.5955
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
6797 - acc: 0.5938 - val_loss: 0.6779 - val_acc: 0.5955

Epoch 00010: LearningRateScheduler reducing learning rate to 0.00025.
Epoch 10/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4804
- acc: 0.7882
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5771 - acc: 0.7102 - val_loss: 0.4804 - val_acc: 0.7882

Epoch 00011: LearningRateScheduler reducing learning rate to 0.00025.
Epoch 11/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4717
- acc: 0.7952
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5087 - acc: 0.7841 - val_loss: 0.4717 - val_acc: 0.7952

Epoch 00012: LearningRateScheduler reducing learning rate to 0.00025.
Epoch 12/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4770
- acc: 0.7944
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5206 - acc: 0.7791 - val_loss: 0.4770 - val_acc: 0.7944

Epoch 00013: LearningRateScheduler reducing learning rate to 0.00025.
Epoch 13/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4787
- acc: 0.7945
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5363 - acc: 0.7640 - val_loss: 0.4787 - val_acc: 0.7945

Epoch 00014: LearningRateScheduler reducing learning rate to 0.00025.
Epoch 14/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4574
- acc: 0.8039
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
5057 - acc: 0.7896 - val_loss: 0.4574 - val_acc: 0.8039

Epoch 00015: LearningRateScheduler reducing learning rate to 0.00012
5.
Epoch 15/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4829
- acc: 0.7950
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5075 - acc: 0.7886 - val_loss: 0.4829 - val_acc: 0.7950

Epoch 00016: LearningRateScheduler reducing learning rate to 0.00012
5.
Epoch 16/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4459
- acc: 0.8133
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4952 - acc: 0.7942 - val_loss: 0.4459 - val_acc: 0.8133
```

```
Epoch 00017: LearningRateScheduler reducing learning rate to 0.00012
5.
Epoch 17/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4720
- acc: 0.7985
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
5139 - acc: 0.7797 - val_loss: 0.4720 - val_acc: 0.7985

Epoch 00018: LearningRateScheduler reducing learning rate to 0.00012
5.
Epoch 18/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4468
- acc: 0.8106
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4962 - acc: 0.7891 - val_loss: 0.4468 - val_acc: 0.8106

Epoch 00019: LearningRateScheduler reducing learning rate to 0.00012
5.
Epoch 19/100
860/860 [==============================] - 9s 11ms/step - loss: 0.483
3 - acc: 0.7932
2579/2579 [==============================] - 37s 14ms/step - loss: 0.
4826 - acc: 0.8017 - val_loss: 0.4833 - val_acc: 0.7932

Epoch 00020: LearningRateScheduler reducing learning rate to 6.25e-0
5.
Epoch 20/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4386
- acc: 0.8146
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
4819 - acc: 0.7972 - val_loss: 0.4386 - val_acc: 0.8146

Epoch 00021: LearningRateScheduler reducing learning rate to 6.25e-0
5.
Epoch 21/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4624
- acc: 0.8128
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
4696 - acc: 0.8085 - val_loss: 0.4624 - val_acc: 0.8128

Epoch 00022: LearningRateScheduler reducing learning rate to 6.25e-0
5.
Epoch 22/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4309
- acc: 0.8289
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4565 - acc: 0.8195 - val_loss: 0.4309 - val_acc: 0.8289

Epoch 00023: LearningRateScheduler reducing learning rate to 6.25e-0
5.
Epoch 23/100
860/860 [==============================] - 7s 9ms/step - loss: 0.4081
- acc: 0.8362
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
4506 - acc: 0.8267 - val_loss: 0.4081 - val_acc: 0.8362

Epoch 00024: LearningRateScheduler reducing learning rate to 6.25e-0
```

```
5.
Epoch 24/100
860/860 [==============================] - 8s 10ms/step - loss: 0.412
5 - acc: 0.8377
2579/2579 [==============================] - 35s 13ms/step - loss: 0.
4547 - acc: 0.8232 - val_loss: 0.4125 - val_acc: 0.8377

Epoch 00025: LearningRateScheduler reducing learning rate to 3.125e-0
5.
Epoch 25/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4266
- acc: 0.8371
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4478 - acc: 0.8279 - val_loss: 0.4266 - val_acc: 0.8371

Epoch 00026: LearningRateScheduler reducing learning rate to 3.125e-0
5.
Epoch 26/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4034
- acc: 0.8426
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4402 - acc: 0.8326 - val_loss: 0.4034 - val_acc: 0.8426

Epoch 00027: LearningRateScheduler reducing learning rate to 3.125e-0
5.
Epoch 27/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4032
- acc: 0.8427
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4491 - acc: 0.8281 - val_loss: 0.4032 - val_acc: 0.8427

Epoch 00028: LearningRateScheduler reducing learning rate to 3.125e-0
5.
Epoch 28/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4014
- acc: 0.8435
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4361 - acc: 0.8339 - val_loss: 0.4014 - val_acc: 0.8435

Epoch 00029: LearningRateScheduler reducing learning rate to 3.125e-0
5.
Epoch 29/100
860/860 [==============================] - 8s 9ms/step - loss: 0.4028
- acc: 0.8432
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
4454 - acc: 0.8279 - val_loss: 0.4028 - val_acc: 0.8432

Epoch 00030: LearningRateScheduler reducing learning rate to 1.5625e-
05.
Epoch 30/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4235
- acc: 0.8291
2579/2579 [==============================] - 32s 13ms/step - loss: 0.
4344 - acc: 0.8348 - val_loss: 0.4235 - val_acc: 0.8291

Epoch 00031: LearningRateScheduler reducing learning rate to 1.5625e-
05.
```

```
Epoch 31/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3919
- acc: 0.8455
2579/2579 [==============================] - 32s 13ms/step - loss: 0.
4318 - acc: 0.8365 - val_loss: 0.3919 - val_acc: 0.8455

Epoch 00032: LearningRateScheduler reducing learning rate to 1.5625e-
05.
Epoch 32/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3989
- acc: 0.8424
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4315 - acc: 0.8362 - val_loss: 0.3989 - val_acc: 0.8424

Epoch 00033: LearningRateScheduler reducing learning rate to 1.5625e-
05.
Epoch 33/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3914
- acc: 0.8459
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4277 - acc: 0.8383 - val_loss: 0.3914 - val_acc: 0.8459

Epoch 00034: LearningRateScheduler reducing learning rate to 1.5625e-
05.
Epoch 34/100
860/860 [==============================] - 8s 9ms/step - loss: 0.4052
- acc: 0.8398
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
4274 - acc: 0.8386 - val_loss: 0.4052 - val_acc: 0.8398

Epoch 00035: LearningRateScheduler reducing learning rate to 7.8125e-
06.
Epoch 35/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3922
- acc: 0.8436
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4254 - acc: 0.8382 - val_loss: 0.3922 - val_acc: 0.8436

Epoch 00036: LearningRateScheduler reducing learning rate to 7.8125e-
06.
Epoch 36/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3876
- acc: 0.8472
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4272 - acc: 0.8400 - val_loss: 0.3876 - val_acc: 0.8472

Epoch 00037: LearningRateScheduler reducing learning rate to 7.8125e-
06.
Epoch 37/100
860/860 [==============================] - 8s 10ms/step - loss: 0.387
5 - acc: 0.8478
2579/2579 [==============================] - 35s 14ms/step - loss: 0.
4274 - acc: 0.8385 - val_loss: 0.3875 - val_acc: 0.8478

Epoch 00038: LearningRateScheduler reducing learning rate to 7.8125e-
06.
Epoch 38/100
```

```
860/860 [==============================] - 6s 7ms/step - loss: 0.3955
- acc: 0.8478
2579/2579 [==============================] - 32s 12ms/step - loss: 0.
4255 - acc: 0.8381 - val_loss: 0.3955 - val_acc: 0.8478

Epoch 00039: LearningRateScheduler reducing learning rate to 7.8125e-
06.
Epoch 39/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3885
- acc: 0.8462
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4247 - acc: 0.8406 - val_loss: 0.3885 - val_acc: 0.8462

Epoch 00040: LearningRateScheduler reducing learning rate to 3.90625e
-06.
Epoch 40/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3858
- acc: 0.8482
2579/2579 [==============================] - 32s 13ms/step - loss: 0.
4236 - acc: 0.8401 - val_loss: 0.3858 - val_acc: 0.8482

Epoch 00041: LearningRateScheduler reducing learning rate to 3.90625e
-06.
Epoch 41/100
860/860 [==============================] - 6s 7ms/step - loss: 0.4157
- acc: 0.8228
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4271 - acc: 0.8398 - val_loss: 0.4157 - val_acc: 0.8228

Epoch 00042: LearningRateScheduler reducing learning rate to 3.90625e
-06.
Epoch 42/100
860/860 [==============================] - 7s 8ms/step - loss: 0.4015
- acc: 0.8364
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4253 - acc: 0.8398 - val_loss: 0.4015 - val_acc: 0.8364

Epoch 00043: LearningRateScheduler reducing learning rate to 3.90625e
-06.
Epoch 43/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3881
- acc: 0.8488
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4222 - acc: 0.8411 - val_loss: 0.3881 - val_acc: 0.8488

Epoch 00044: LearningRateScheduler reducing learning rate to 3.90625e
-06.
Epoch 44/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3856
- acc: 0.8482
2579/2579 [==============================] - 32s 13ms/step - loss: 0.
4230 - acc: 0.8413 - val_loss: 0.3856 - val_acc: 0.8482

Epoch 00045: LearningRateScheduler reducing learning rate to 1.953125
e-06.
Epoch 45/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3852
```

```
- acc: 0.8484
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4213 - acc: 0.8411 - val_loss: 0.3852 - val_acc: 0.8484

Epoch 00046: LearningRateScheduler reducing learning rate to 1.953125
e-06.
Epoch 46/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3852
- acc: 0.8479
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4252 - acc: 0.8413 - val_loss: 0.3852 - val_acc: 0.8479

Epoch 00047: LearningRateScheduler reducing learning rate to 1.953125
e-06.
Epoch 47/100
860/860 [==============================] - 5s 6ms/step - loss: 0.3861
- acc: 0.8472
2579/2579 [==============================] - 32s 12ms/step - loss: 0.
4232 - acc: 0.8415 - val_loss: 0.3861 - val_acc: 0.8472

Epoch 00048: LearningRateScheduler reducing learning rate to 1.953125
e-06.
Epoch 48/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3861
- acc: 0.8475
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4200 - acc: 0.8412 - val_loss: 0.3861 - val_acc: 0.8475

Epoch 00049: LearningRateScheduler reducing learning rate to 1.953125
e-06.
Epoch 49/100
860/860 [==============================] - 8s 9ms/step - loss: 0.3843
- acc: 0.8482
2579/2579 [==============================] - 34s 13ms/step - loss: 0.
4206 - acc: 0.8416 - val_loss: 0.3843 - val_acc: 0.8482

Epoch 00050: LearningRateScheduler reducing learning rate to 9.765625
e-07.
Epoch 50/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3844
- acc: 0.8489
2579/2579 [==============================] - 32s 12ms/step - loss: 0.
4213 - acc: 0.8420 - val_loss: 0.3844 - val_acc: 0.8489

Epoch 00051: LearningRateScheduler reducing learning rate to 9.765625
e-07.
Epoch 51/100
860/860 [==============================] - 6s 7ms/step - loss: 0.3839
- acc: 0.8478
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4225 - acc: 0.8423 - val_loss: 0.3839 - val_acc: 0.8478

Epoch 00052: LearningRateScheduler reducing learning rate to 9.765625
e-07.
Epoch 52/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3830
- acc: 0.8491
```

```
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4244 - acc: 0.8422 - val_loss: 0.3830 - val_acc: 0.8491

Epoch 00053: LearningRateScheduler reducing learning rate to 9.765625
e-07.
Epoch 53/100
860/860 [==============================] - 7s 8ms/step - loss: 0.3826
- acc: 0.8487
2579/2579 [==============================] - 33s 13ms/step - loss: 0.
4235 - acc: 0.8416 - val_loss: 0.3826 - val_acc: 0.8487
```

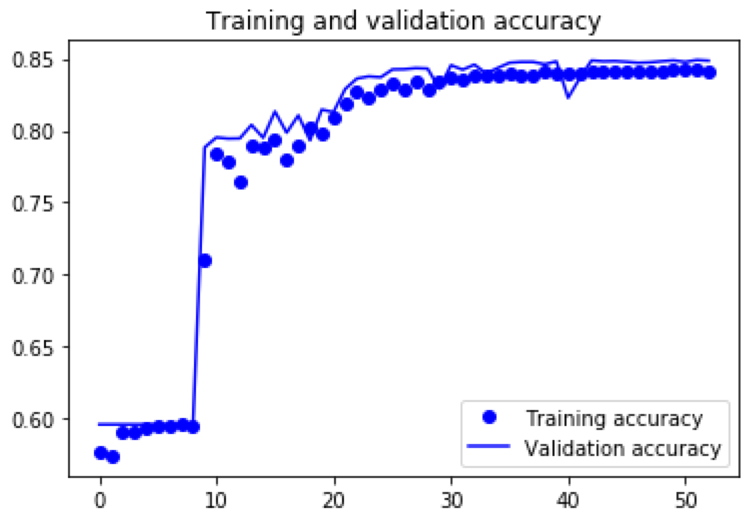
In [52]:

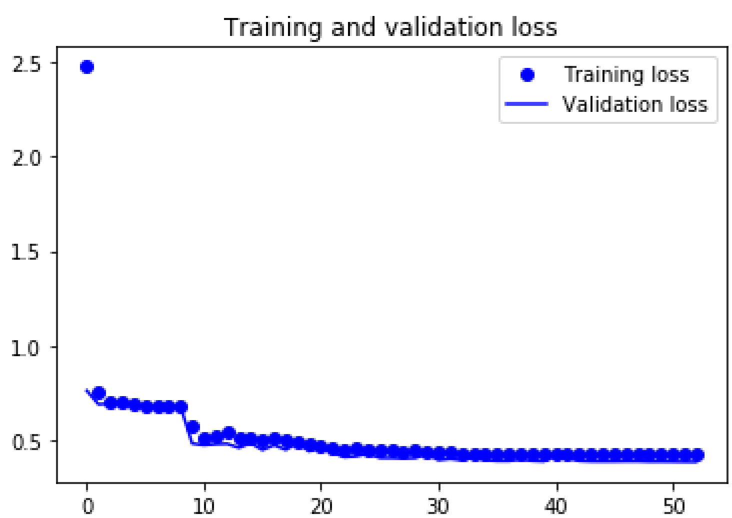
```python
testEval2 = model2.evaluate(xVal, yVal, verbose=0)
print('Test loss:', testEval2[0])
print('Test accuracy:', testEval2[1])
```

```
('Test loss:', 6.536827518815707)
('Test accuracy:', 0.5955242)
```

In [53]:
```
accuracy2 = history2.history['acc']
valAccuracy2 = history2.history['val_acc']
loss2 = history2.history['loss']
valLoss2 = history2.history['val_loss']
epochs2 = range(len(accuracy2))

plt.plot(epochs2, accuracy2, 'bo', label='Training accuracy')
plt.plot(epochs2, valAccuracy2, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs2, loss2, 'bo', label='Training loss')
plt.plot(epochs2, valLoss2, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





The cross validation loss and accuracy are better than in the previous training, however, the test loss and accuracy are both worse. Even though l2 regularization and dropout have been applied, the model seems to have overfit. However, the validation curve and the training curve do not separate yet. Lets run the model on the test set.

In [57]:
```
predictions2 = np.rint(model2.predict(X_test))
predictions2
```

Out[57]:
```
array([[-0.],
       [-0.],
       [-0.],
       ...,
       [-0.],
       [-0.],
       [-0.]], dtype=float32)
```

In [58]:
```
predSubmission2 = pd.DataFrame(predictions2.astype('int'),
                              columns = ['label'])

predSubmission2.index = all_test_files
predSubmission2.index.name = 'id'

predSubmission2
```

Out[58]:

id	label
00006537328c33e284c973d7b39d340809f7271b	0
0000ec92553fda4ce39889f9226ace43cae3364e	0
00024a6dee61f12f7856b0fc6be20bc7a48ba3d2	0
000253dfaa0be9d0d100283b22284ab2f6b643f6	0
000270442cc15af719583a8172c87cd2bd9c7746	0
000309e669fa3b18fb0ed6a253a2850cce751a95	0
000360e0d8358db520b5c7564ac70c5706a0beb0	0
00040095a4a671280aeb66cb0c9231e6216633b5	0
000698b7df308d75ec9559ef473a588c513a68aa	0
0006e1af5670323331d09880924381d67d79eda0	0
000997a6038fa7441aa0111ac456255060a354c4	0
000c8db3e09f1c0f3652117cf84d78aae100e5a7	0
000de14191f3bab4d2d6a7384ca0e5aa5dc0dffe	0
000e6341cf18365d35b40f4991002fec8834afc0	0
0010e2887e0b977fcdfdf4c50564fafbbc2b6208	0
0010e7eaa3d8e14203cd3900b739d8bf0f0b67f0	0
001161a2eca200f565f12870048a78fa5b320dee	0
0011807dd1e3306ff3f7a755fd3efbefa2901dce	0
00118bec91b7fae175791896f7011ff506b3d7dd	0
0011f0596a038fc8daec4fde71465e347515392e	0
0014fdb3da986174f9a1d7ae95f3b75a2d025a57	0
00153be8e27526f9c2f035aff25ca9264db0a2ed	0
0016dddb2797d3fafb38c2651c5589d92030e835	0
00179c97cd2aaeafcbce352aa387db1b76616a53	0
00186a82134c399e8bb77d038475741d696cb72b	0
001d59af39e296eb8323a190f58cf498f66fb08d	0
001da402e229c18631c3dd7b8953e6f7e5d3ce99	0
001e385106b858d95c122a1cb3a930d3f30e67de	0
001e92257a59ac137d99f52419884ca60e0b91ee	0
0020b6b5a91d83fe3ba41ea6b89113db0848dac4	0
...	...
ffe7f2205d750486ba4b9ec56a9e7011c29633a0	0
ffe937550aed66e3373d31abd16608c7a0481337	0
ffeb0ee2add600e97d47379862626b18af7a07a2	0

id	label
ffecc7469b67e16e16456e464a99bb5620f0cd9f	0
ffee4360d4273351e7283db5494fa1a6a6ca64b0	0
ffeea8c422d6db1b243738eb11343712ca368327	0
ffeec967ee78e98a8e55645c41f13b8ff5e97d0e	0
fff03179712a2c3dabf9b8c7bbb5ce1c767d6f17	0
fff204670ae06ad729b2fd57b65cd8a2074626c2	0
fff3711c84005cea2cb1adfa9ff47541d49567bd	0
fff44d16df02bd8e3d8cd01de693c2cbc80dfd47	0
fff48816ec621b3b2b4444279ac01382b74b5058	0
fff49294e77f806ea924eff8484356ab75feef7b	0
fff590742f1679a2d82945956e944463333ac7e0	0
fff69b7e90d979503222591557d5191dc74e9c9a	0
fff6aef707a38f6976ec5979c01d295671a7236e	0
fff726415ca9dbc3da87715e68787153b7a06b99	0
fff8c257c9fa44cd2604ff1887f25cc337565147	0
fff933e08e23a51cc6401d116e3c18999330b8dd	0
fffa681f68c58839f9fad90be2ac9cba6ab6f6a5	0
fffb109428e7c1ff5cfaa8ba49c657fc13828bf8	0
fffb3f6c8be9fb4d19245e6d5dbe003b03c70771	0
fffbaf2d17fc95a855fa833bd08dfc20aa18d232	0
fffbc065982ddf01c39e4c74e62d1868551d0b25	0
fffbfb279c9c378af5c362363b841a38bbee3294	0
fffdd1cbb1ac0800f65309f344dd15e9331e1c53	0
fffdf4b82ba01f9cae88b9fa45be103344d9f6e3	0
fffec7da56b54258038b0d382b3d55010eceb9d7	0
ffff276d06a9e3fffc456f2a5a7a3fd1a2d322c6	0
ffffeb4c0756098c7f589b7beec08ef1899093b5	0

57458 rows × 1 columns

In [59]:
```
predSubmission2.to_csv('SHollatz_submission_2.csv')
```

The score in the competition is only 0.5. The scheduled learning rate definitely seems to have overfit the data.

Conclusion

A binary classifier with a prediction accuracy of 78.9% was trained by designing a CNN with 4 convolutional layers, a maximum of 64 kernel per layer, maxpooling, and 2 dropout layers with a dropout rate of 0.5 and L2 regularization in the dense classifier part at the end. The Adam optimizer was used with a learning rate of 0.00005. The architecture was developed by trial and error, starting with a deeper CNN without dropout and regularization and a larger learning rate. However, these were the hyperparameter settings that lead to the best prediction accuracy on the test set. For future improvements, batch normalization could be tried as well as different batch sizes, number of filters per layer, different activation functions and also different preprocessing techniques for the input data.

In [ ]:

```

```