Modern Data Mining - Final Project

An Automated Method for Malaria Diagnosis

Aston Hamilton
Ines Gonzalez Del Mazo
Xiaowei Yan

Abstract: Malaria is a mosquito-borne infectious disease that can cause death. In 2016, there were 216 million cases of malaria worldwide resulting in an estimated 445,000 to 731,000 deaths. Approximately 90% of both cases and deaths occurred in Africa. Diagnosis of malaria is usually constrained by the availability of doctors and other medical resources, especially in less-developed areas where malaria is more common. Current diagnosis methods include microscopic examination of blood using blood films and antigen-based rapid diagnostic tests, both require the involvement of a skilled lab inspector for accurate detection of the illness. In this study, we explore different techniques to build a predictive model for malaria diagnosis based on thin blood smear slide images. We recommend a final model (accuracy $\approx 95\%$) to be used as a prototype of a timely, cost-effective way of diagnosing malaria by automating the diagnosis through image recognition and removing the heavy reliance on a skilled lab inspector. We aspire for the model to be a powerful tool for diagnosing malaria in countries and areas that lack medical resources – e.g. through incorporation into a mobile app.

Data: We use a repository of segmented cells from the thin blood smear slide images from the Malaria Screener research activity. Source of data: https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria

Contents

List of Figures

Background

Malaria is a mosquito-borne infectious disease that can cause death. In 2016, there were 216 million cases of malaria worldwide resulting in an estimated 445,000 to 731,000 deaths. Approximately 90% of both cases and deaths occurred in Africa. Diagnosis of malaria is usually constrained by the availability of doctors and other medical resources, especially in less-developed areas where malaria is more common.

Owing to the non-specific nature of the presentation of symptoms, current diagnosis confirmation relies on microscopic examination of blood using blood films or antigen-based rapid diagnostic tests. Microscopy is the most commonly used method to detect malarial parasite.

Despite its widespread usage, diagnosis by microscopy suffers from two main drawbacks:

1. many settings (especially rural) are not equipped to perform the test
2. the **accuracy** of the results depends on both the skill of the person examining the blood film and the levels of the parasite in the blood.

Description of problem

In this study, we build 5 different predictive models using 3 different modern data mining techniques (ElasticNet Regression, Random Forests, and Artificial Neural Networks) to perform malaria diagnosis based on a 50x50 picture of a thin blood smear slide (microscopy). To choose the best final model, we compare the predictive power of the models using the AUC and accuracy criteria on the validation dataset.

The final model can be a useful prototype of a timely, cost-effective way of diagnosing malaria by automating the diagnosis through image recognition. We aspire for the model to be a powerful tool for diagnosing malaria in countries and areas that lack medical resources – e.g. our model can be incorporated into a mobile phone app to diagnosing malaria with a mobile phone!! Our final model can also be used to improve the accuracy of diagnosis result as well as availability and quality consistency across regions.

Summary of Data

The dataset was obtained from Kaggle (originally taken from the official US National Institutes of Health Website) and contains a total of 27,558, which have been classified into two groups, "Infected" (13,778 images) or "Uninfected" (13,780 images). The raw images are in the PNG file format and are various resolutions, from 40x55 pixel to 241x394 pixels.

Data Cleanup and Extraction

Python data cleaning routines (Exhibit 1) were used to extract and clean up the images. Each image was pulled from the "tar" archive and analyzed to create a 3D matrix (w,h,d; where w=width of the image in pixels, h=heght of the image in pixels, d=3;). In the image matrix, the third dimension represented the components of the color channels for each pixel (i.e. the Red, Green, and Blue values). The below operations were then performed on these image matrices:

- Since the images needed to have been fed into different models that had an expected shape, we found it easier to first modify the images to make them have a square aspect ratio by removing the necessary pixels from either side of the dimension that was larger.
- Each image matrix had w $\times$ h $\times$ d numbers and for even a relatively small 100x100 pixel image represented 30,000 different numbers! Since these numbers were fed to our models, the models became very large, complex, and difficult to train. We thus decided to resize all our images to 50x50 pixels. Note that we considered removing the color channels (make the image grayscale) but we decided to

keep them since our observation of the images revealed that the infected images had a dot that was usually purple.

- The entire cleaned dataset was then divided into a training set (14,697 rows with 0.499 positive response proportion) – used to train all the models explored in this paper; a validation set (7,349 rows with 0.496 positive response proportion) – used to evaluate the models explored after they were trained and iteratively improved; a test set (5,512 rows with 0.507 positive response proportion) – never touched during the training or improvement of any of the explored model but instead used to compute the final evaluation metrics for the model recommended for implementation by this paper.
  - Note that all our datasets were very balanced (i.e. no response skew) so accuracy was an acceptable measure of performance for each explored model

The figure below shows a sample of the raw cell images before and after cleaning with the Python routines:

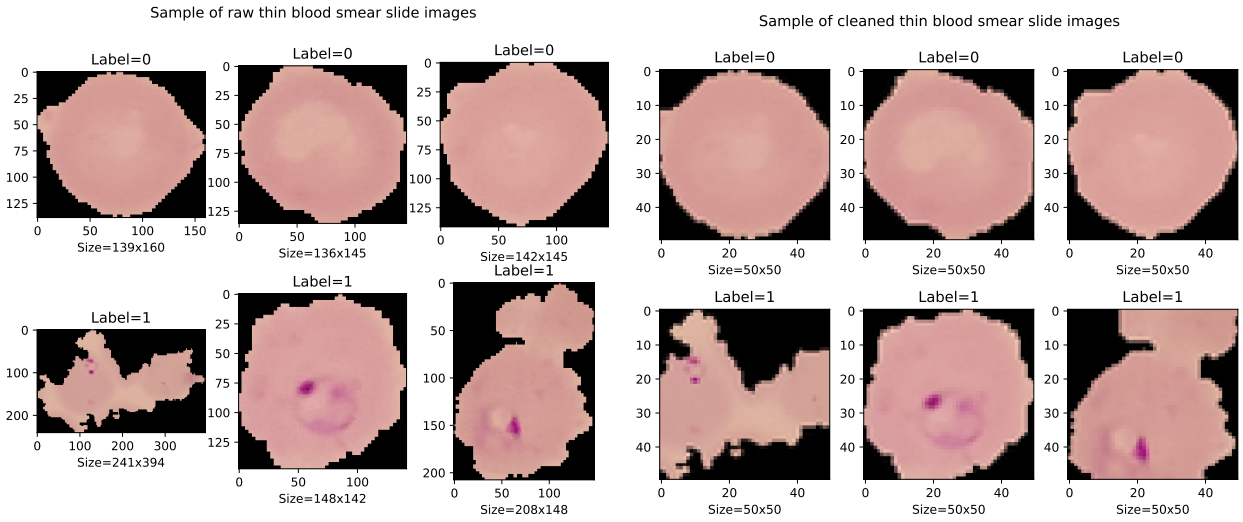

Figure 1: A sample of raw thin blood smear slide images before and after they were cleaned by our Python routines

Predictor Variables

For this study, the "predictor variables" were the matrices of the pixels of the image. Some of our models required a single dimensional vector (e.g. Random Forest), in which case we flattened each 3D matrix into a (1x7500) vector for training and evaluating the models.

The response variable was a binary indicator variable that was 1 if the image represented an infected cell or 0 if the cell wasn't infected.

Analysis

In this study, we took an approach to evaluate different models with different configurations. Each model's accuracy on the validation dataset was measured and used for evaluating the model. Additionally, the AUC was used to compare the models to avoid any bias from an assumed loss importance ratio in selecting a classification threshold when necessary. 5 of the explored models are presented in this study. The models used different data mining techniques, specifically: ElasticNet, Random Forest and Artificial Neural Network (3 different architectures)

ElasticNet

The ElasticNet algorithm was also used to analyze the images (Exhibit 2). The L1 L2 penalty was selected to be 0.95 and the optimal alpha was determined from analyzing the CV Error.

- The best lambda was determined to be 0.26004
- The model had an acceptable validation dataset accuracy of 0.679004, at a 0.5 classification threshold
- The model had an acceptable validation dataset AUC of 0.739080

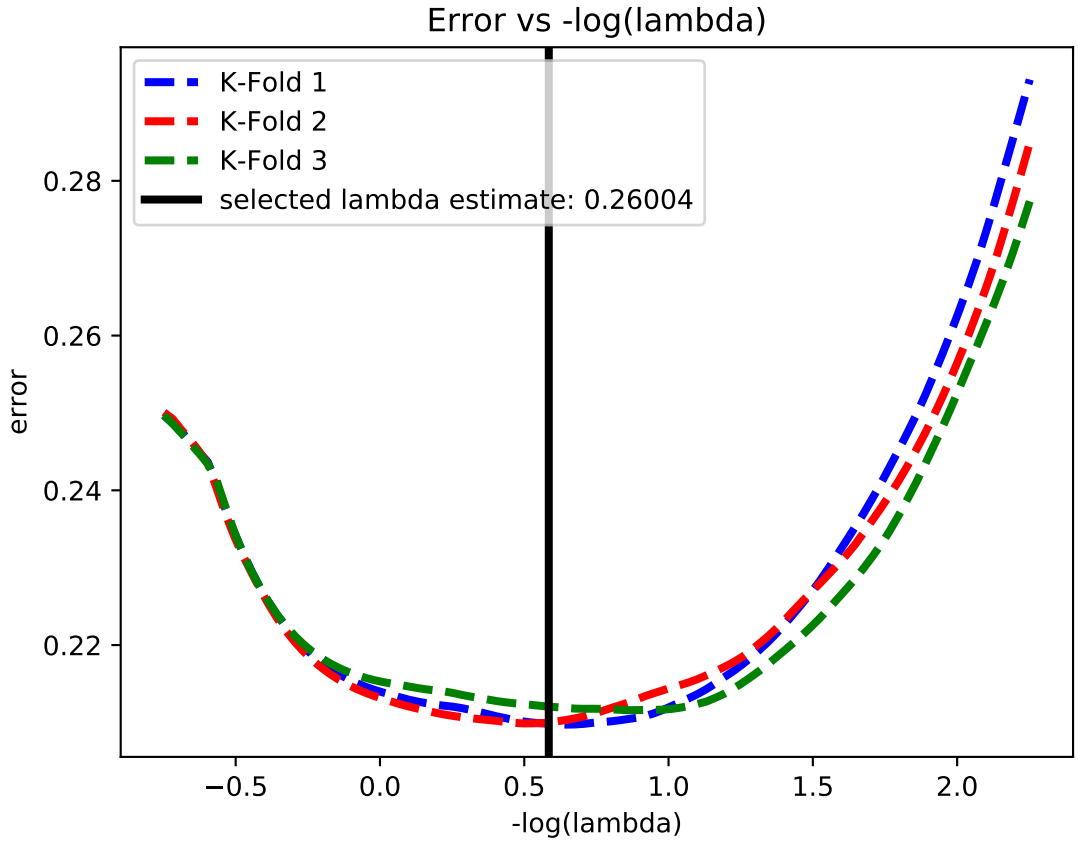

Figure 2: The CV Error as lambda was varied

Random Forest

The Random Forest algorithm was used to analyze the images (Exhibit 3). 200 trees were used in the implementation after running an extensive diagnosis of how the error varied with the number of trees. The $\sqrt{\#\text{ of features}}$ was used as the number of features to features to consider at each split.

- The model had an acceptable validation dataset accuracy of 0.798340
- The model had an acceptable validation dataset AUC of 0.798453

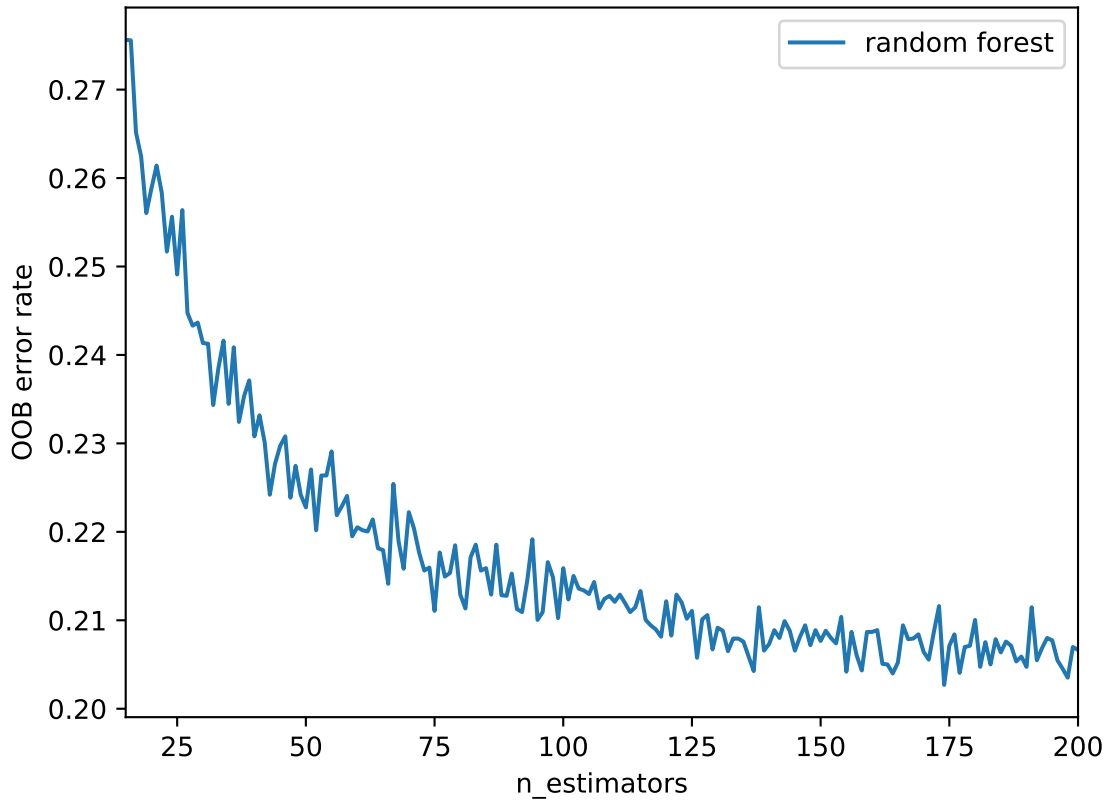

Figure 3: The error as number of trees was varied

Artificial Neural Network

The Artificial Neural Network algorithm was also used to analyze the images. We explored the three different architectures: Multi-layer Perceptron, Convolutional Neural Networks and Google's Inception v3 Architecture. For all the ANN architectures, the output node with the highest value is used as the prediction of the classification.

Multi-layer Perceptron

This was the simplest of the Neural Network Architectures. We used three hidden layers (1024, 256, 128 neurons respectively). The input image matrices were flattened to 7,500 unit vectors and used as the input layer of the network and the output layer had 2 neurons (1 for each class). The model had a total of 7,976,578 parameters.

- The model had an acceptable validation dataset accuracy of 0.663355
- As shown in the plots below, by the end of the training, the training dataset error was high, and this (combined with a high validation set error) prompted us that the model was not sufficiently complex or not converging effectively.

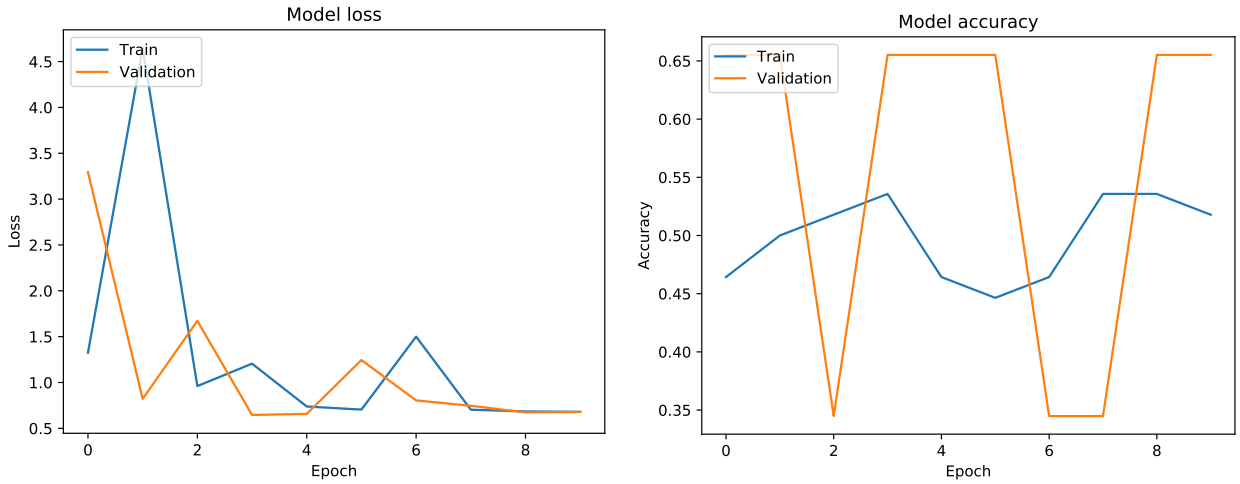

Figure 4: MLP arch. performance on the training and validation datasets over 10 training iterations

Convolutional Neural Netoworks

We designed another architecture that leveraged Convolutional layers. Convolutional layers are described by research to be very effective for applying Artificial Neural Networks to image recognition tasks. These layers are good for making the network robust to a translation of important image features, which applies very well to our task, where we observed that the images tagged as infected primarily had a "purple dot" at different parts in the image. The model had a total of 2,425,538 parameters.

- The model had an acceptable validation dataset accuracy of 0.953326
- The plots below show that the training loss was decreasing which indicates that the data is being modeled well. The low loss of the validation set also shows that the model wasn't overfitting. The simultaneously high model accuracies by the end of training (i.e. on the training and validation set simultaneously) also support that this model architecture offers good performance and generalizability.

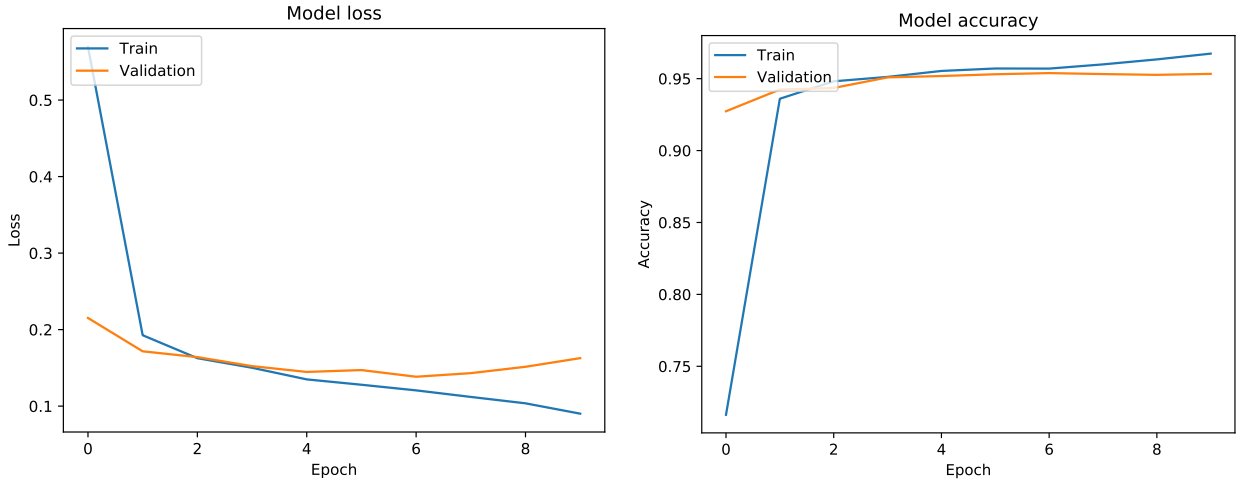

Figure 5: Convolutional arch. performance on the training and validation datasets over 10 training iterations

Inception v3 Architecture

For the third model, we explored Google's proposed Inception architecture (version 3). This architecture also leverages convolutional layers and other advanced complexities that enable it to excelConvolutional layers are described by research to be very effective for applying Artificial Neural Networks to image recognition tasks. These layers are good for making the network robust to a translation of important image features, which applies very well to our task, where we observed that the images tagged as infected primarily had a "purple dot" at different parts in the image. The model had a total of 22,327,842 parameters.

- The model had an acceptable validation dataset accuracy of 0.951966
- The plots below show that the training loss was decreasing to very low amounts with the validation dataset loss, which indicates that this model's architecture is modeling the data very well and with good generalizability. The anomalous drop in the validation set accuracy around iteration 5 is probably just an anomaly in the interaction of the data and the training weights, but the immediate recovery in the subsequent iterations gives us confidence that we can ignore this drop. Thus we conclude that this model performs well with good generalizability.

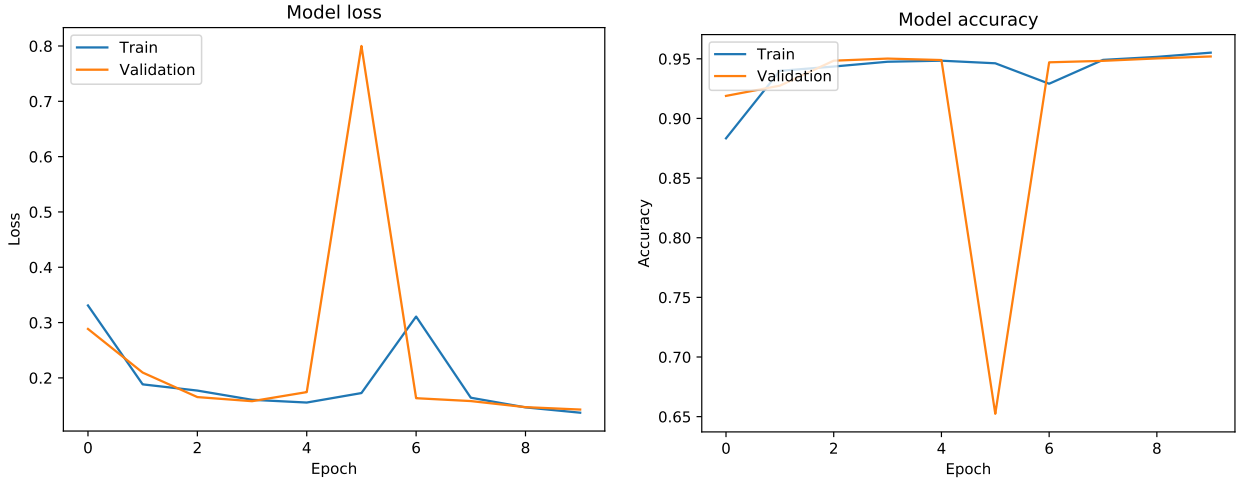

Figure 6: Inception v3 arch. performance on the training and validation datasets over 10 training iterations

Conclusion

We recommend moving forward with the Convolutional Architecture (Exhibit 7). The performance was very similar to the state of the art Inception v3 architecture but the Convolutional architecture that we used is much simpler – it's relative simplicity was important considering that our intended use of the model is integration into low-end devices in developing countries where Malaria is a big problem.

- The final model's test set accuracy is 0.950653
- The confusion matrix for the test dataset performance of the final model is shown below. The misclassified errors are very low and indicates the model is very performant. A sample of the misclassified images in the test dataset is shown below, and we think misclassified samples of these types are acceptable for moving forward with the model.

Next Steps: We recommend continuing to extend the model to not just classify when an image is indicative of being infected with Malaria but also to identify which one of the 5 types of malaria is indicated in the image.

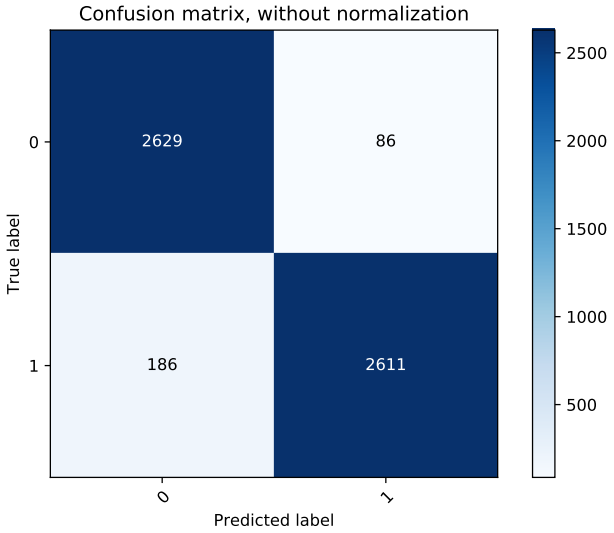

Figure 7: Confusion Matrix of the final model on the test dataset

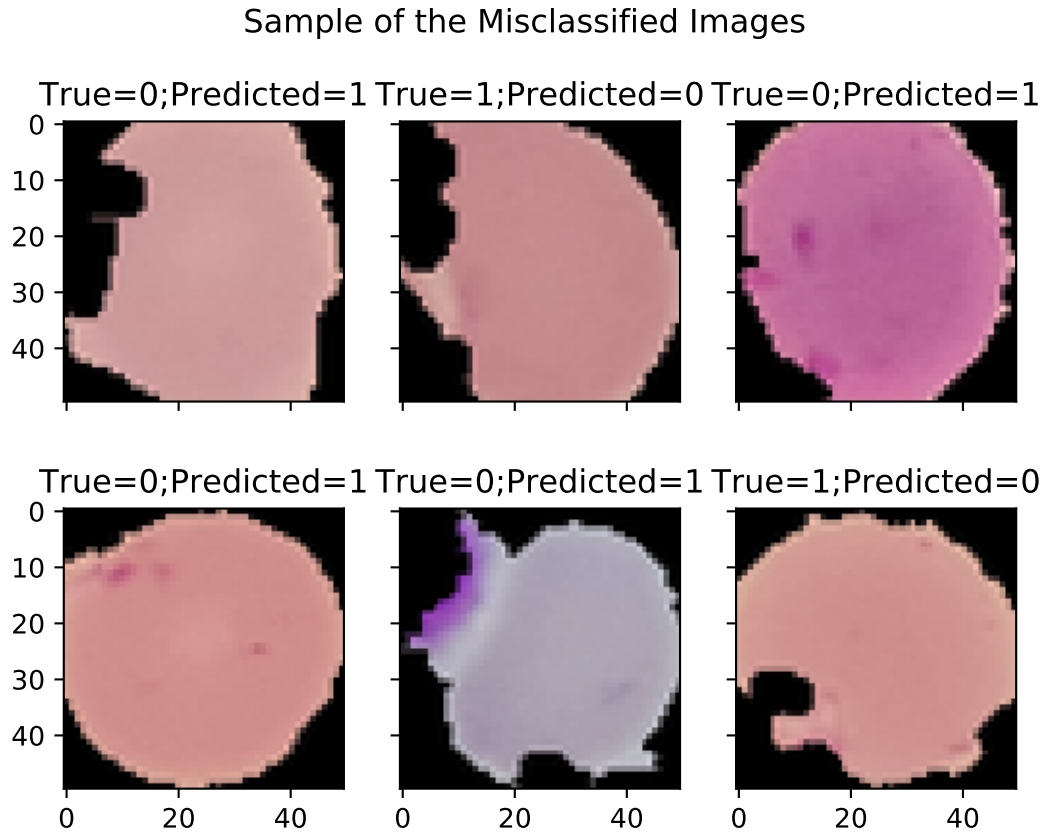

Figure 8: Sample of misclassified images using the final model on the test dataset

Appendix

Exhibit 1

- The Python code used to extract and clean the thin blood smear slide images:

```
import cv2
import numpy as np
import tarfile
import scipy.io
import math
from PIL import Image

from sklearn.model_selection import train_test_split

# Configuration variables for the script
IMAGE_RESIZE_TARGET_HINT = 50
IN_TAR_FILE_FN = 'data/cell_images_full.tar.gz'
OUT_FILE_FN = 'data/cell_images_full_32.mat'


# read filenames directly from tar
def get_all_filenames(tar_fn):
    with tarfile.open(tar_fn) as f:
        return [m.name for m in f.getmembers() if m.isfile()]


# reads bytes directly from tar by filename
def read_raw_from_tar(tar_fn, fn):
    with tarfile.open(tar_fn) as f:
        m = f.getmember(fn)
        return f.extractfile(m).read()


# creates RGB 3D matrix from raw image data
def decode_image_from_raw_bytes(raw_bytes):
    img = cv2.imdecode(np.asarray(bytearray(raw_bytes), dtype=np.uint8), 1)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return img


# standardizes the images to a uniform size
def standardize_image_pixels(pixels, img_size):
    # centers the image and crops it so that it is square
    def _helper_center_crop_image(img):
        h, w = img.shape[0], img.shape[1]
        shift = math.floor(np.abs(h - w) / 2.0)
        if w > h:
            return img[:, shift:shift + h, :]

        return img[shift:shift + w, :, :]

    centered_cropped = _helper_center_crop_image(pixels)
    return np.array(Image.fromarray(centered_cropped)
```

```
                    .resize([img_size, img_size], Image.ANTIALIAS))


# generator to yield each image from tar raw_bytes, label, and file_path
def gn_load_image_observations_from_tar(tar_fn):
    with tarfile.open(tar_fn) as f:
        for m in f.getmembers():
            if m.isfile() and m.name.endswith('.png'):
                raw_bytes = f.extractfile(m).read()
                img = decode_image_from_raw_bytes(raw_bytes)
                yield (
                  img, (1 if m.name.startswith('cell_images/infected/') else 0),
                  m.name
                )

# entry point
def _kernel():
    # %%
    # pull raw images from the tar archive
    image_dataset = list(gn_load_image_observations_from_tar(IN_TAR_FILE_FN))

    # find the smallest image and scale everything to that size
    # or use preconfigured size if it is defined
    IMG_SIZE = IMAGE_RESIZE_TARGET_HINT if IMAGE_RESIZE_TARGET_HINT is not None else (
      math.ceil(
          min([d for e in image_dataset for d in e[0].shape[0:2]]) / 25
      ) * 25)

    # %%
    # standardize all images (e.g. uniform size)
    image_dataset = [
        (standardize_image_pixels(e[0], img_size=IMG_SIZE),) + e[1:]
        for e in image_dataset
    ]

    # split training and test set
    image_dataset_train, image_dataset_test = train_test_split(
        image_dataset,
        test_size=2 / 10,
        random_state=42
    )

    # split optional training and validation set
    image_dataset_model_train, image_dataset_train_model_val = train_test_split(
        image_dataset_train,
        test_size=2 / 6,
        random_state=73
    )

    # package everything for export
    scipy.io.savemat(OUT_FILE_FN, mdict=dict(
        img_size=IMG_SIZE,
```

```
        nrow_train=len(image_dataset_train),
        train_X=[e[0] for e in image_dataset_train],
        train_Y=[e[1] for e in image_dataset_train],
        train_fns=[e[2] for e in image_dataset_train],

        nrow_model_train=len(image_dataset_model_train),
        model_train_X=[e[0] for e in image_dataset_model_train],
        model_train_Y=[e[1] for e in image_dataset_model_train],
        model_train_fns=[e[2] for e in image_dataset_model_train],

        nrow_model_val=len(image_dataset_train_model_val),
        model_val_X=[e[0] for e in image_dataset_train_model_val],
        model_val_Y=[e[1] for e in image_dataset_train_model_val],
        model_val_fns=[e[2] for e in image_dataset_train_model_val],

        nrow_test=len(image_dataset_test),
        test_X=[e[0] for e in image_dataset_test],
        test_Y=[e[1] for e in image_dataset_test],
        test_fns=[e[2] for e in image_dataset_test]
    ), do_compression=True, oned_as='column')

    # print summary
    print('Cleaned data summary:')
    print(dict(
        img_size=IMG_SIZE,
        nrow_train=len(image_dataset_train),
        nrow_model_train=len(image_dataset_model_train),
        nrow_model_val=len(image_dataset_train_model_val),
        nrow_test=len(image_dataset_test)
    ))


# %%
# boot
_kernel()
```

Exhibit 2

- The Python code used to train a ElasticNet Regression analyzer to classify the cell images:

```
import scipy.io
import sklearn
from sklearn.linear_model import ElasticNetCV


TRAIN_DATASET_FN = 'data/cell_images_full_32.mat'


# Flatten an matrix for ElasticNet Regression
def _helper_flatten_matrix(arr):
    return arr.reshape(arr.shape[0], -1)

# entry point
```

```
def _kernel():
    # %%
    image_dataset = scipy.io.loadmat(
        TRAIN_DATASET_FN,
        variable_names=(
            'img_size',
            'nrow_model_train',
            'model_train_X', 'model_train_Y', 'model_train_fns',
            'nrow_model_val',
            'model_val_X', 'model_val_Y', 'model_train_fns'
        ),
        squeeze_me=True
    )

    # %%
    print('Building ElasticNet:')
    clf = ElasticNetCV(l1_ratio=0.95, verbose=2)
    clf.fit(
        _helper_flatten_matrix(image_dataset['model_train_X']),
        image_dataset['model_train_Y']
    )
    print('Model training complete!!')

    # %%
    # compute validation metric
    val_Y_pred = clf.predict(_helper_flatten_matrix(
        image_dataset['model_val_X']
    ))

    val_Y_pred_classes = (val_Y_pred > 0.5).astype(int)

    print("Validation accuracy:", sklearn.metrics.accuracy_score(
        image_dataset['model_val_Y'], val_Y_pred_classes
    ))

    fpr, tpr, thresholds = sklearn.metrics.roc_curve(
        image_dataset['model_val_Y'], val_Y_pred, pos_label=1
    )
    print('AUC Score:', sklearn.metrics.auc(fpr, tpr))


# %%
# boot
_kernel()
```

Exhibit 3

- The Python code used to train a Random Forest analyzer to classify the cell images:

```
import scipy.io
import sklearn
from sklearn.ensemble import RandomForestClassifier

TRAIN_DATASET_FN = 'data/cell_images_full_32.mat'

# Flatten an matrix for Random Forest input
def _helper_flatten_matrix(arr):
    return arr.reshape(arr.shape[0], -1)

# entry point
def _kernel():
    # %%
    image_dataset = scipy.io.loadmat(
        TRAIN_DATASET_FN,
        variable_names=(
            'img_size',
            'nrow_model_train',
            'model_train_X', 'model_train_Y', 'model_train_fns',
            'nrow_model_val',
            'model_val_X', 'model_val_Y', 'model_train_fns'
        ),
        squeeze_me=True
    )

    # %%
    # Train the model
    print('Training model:')
    clf = RandomForestClassifier(n_estimators=200, verbose=2)
    clf.fit(
        _helper_flatten_matrix(image_dataset['model_train_X']),
        image_dataset['model_train_Y']
    )
    print('Model training complete!!')

    # %%
    # compute validation metric
    val_Y_pred = clf.predict(_helper_flatten_matrix(
        image_dataset['model_val_X']
    ))
    print("Validation accuracy:", sklearn.metrics.accuracy_score(
        image_dataset['model_val_Y'], val_Y_pred
    ))

    fpr, tpr, thresholds = sklearn.metrics.roc_curve(
        image_dataset['model_val_Y'], val_Y_pred, pos_label=1
    )
    print('AUC Score:', sklearn.metrics.auc(fpr, tpr))

# %%
```

```
# boot
_kernel()
```

Exhibit 4

- The Python code used to train a MLP ANN to classify the cell images:

```
import numpy as np
import scipy.io
import math

import tensorflow as tf
import keras
import sklearn

TRAIN_DATASET_FN = 'data/cell_images_full_32.mat'

TRAIN_BATCH_SIZE = 50
TRAIN_EPOCHS = 10

FLG_SHIM_FLATTEN_MODEL_INPUTS = True


# this decorator will selectively flatten the model inputs
# based on the noted flag
def _deco_flatten_model_input_matrix(arr):
    if FLG_SHIM_FLATTEN_MODEL_INPUTS:
        return _helper_flatten_matrix(arr)
    return arr


# Flatten an matrix for a dense MLP
def _helper_flatten_matrix(arr):
    return arr.reshape(arr.shape[0], -1)


# Build deep net with a few layers
def build_model_mlp(input_shape):
    model = keras.models.Sequential()
    model.add(keras.layers.Dense(1024, input_shape=(np.prod(input_shape),)))
    model.add(keras.layers.Activation('sigmoid'))
    model.add(keras.layers.Dense(256))
    model.add(keras.layers.Activation('sigmoid'))
    model.add(keras.layers.Dense(128))
    model.add(keras.layers.Activation('sigmoid'))
    model.add(keras.layers.Dense(2))
    model.add(keras.layers.Activation('softmax'))
    return model


# entry point
def _kernel():
    # %%
```

```
image_dataset = scipy.io.loadmat(
    TRAIN_DATASET_FN,
    variable_names=(
        'img_size',
        'nrow_model_train',
        'model_train_X', 'model_train_Y', 'model_train_fns',
        'nrow_model_val',
        'model_val_X', 'model_val_Y', 'model_train_fns'
    ),
    squeeze_me=True
)

# %%
n_training_batches = math.ceil(image_dataset['nrow_model_train'] / TRAIN_BATCH_SIZE)

# Print some statistics on the training data
print('Training data shape:', image_dataset['model_train_X'].shape)
print('Number training batches:', n_training_batches)

print(
    "Proportion of training 1's:",
    sum(image_dataset['model_train_Y']),
    sum(image_dataset['model_train_Y']) / image_dataset['nrow_model_train']
)

# Print some statistics on the validation data
print('Validation data shape:', image_dataset['model_val_X'].shape)
print('Number validation batches:', n_training_batches)

print(
    "Proportion of validation 1's:",
    sum(image_dataset['model_val_Y']),
    sum(image_dataset['model_val_Y']) / image_dataset['nrow_model_val']
)

# %%
# helper functions
# Resets the TF graph
def helper_reset_tf_session():
    curr_session = tf.get_default_session()
    # if current session exists then close it
    if curr_session is not None:
        curr_session.close()
    # reset the tf graph
    keras.backend.clear_session()
    # create a new tf session
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True  # use GPU memory as needed
    s = tf.InteractiveSession(config=config)
    keras.backend.set_session(s)
    return s

# %%
```

```
    helper_reset_tf_session()

    # %%
    # A small abstraction to enable switching between models
    model = build_model_mlp((image_dataset['img_size'], image_dataset['img_size'], 3))

    # %%
    # compile the model
    model.compile(
        loss='categorical_crossentropy',
        optimizer=keras.optimizers.adamax(lr=1e-2),
        metrics=['accuracy']  # report accuracy during training
    )

    # %%
    # train the model on the dataset
    model_fit_history = model.fit(
        _deco_flatten_model_input_matrix(np.stack(map(
            keras.applications.inception_v3.preprocess_input,
            image_dataset['model_train_X']
        ))),
        keras.utils.np_utils.to_categorical(image_dataset['model_train_Y'], 2),
        epochs=TRAIN_EPOCHS,
        batch_size=TRAIN_BATCH_SIZE,
        validation_data=(
            _deco_flatten_model_input_matrix(np.stack(map(
                keras.applications.inception_v3.preprocess_input,
                image_dataset['model_val_X']
            ))),
            keras.utils.np_utils.to_categorical(image_dataset['model_val_Y'], 2)
        ),
        verbose=0
    )

    print('Model training complete!!')

# %%
# boot
_kernel()
```

Exhibit 5

- The architecture of the trained MLP ANN:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1024)              7681024
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               262400
_________________________________________________________________
```

```
activation_2 (Activation)    (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 258
_________________________________________________________________
activation_4 (Activation)    (None, 2)                 0
=================================================================
Total params: 7,976,578
```

Exhibit 6

- The Python code used to train the Convolutional ANN to classify the cell images:

```
import numpy as np
import scipy.io
import math

import tensorflow as tf
import keras
import sklearn

TRAIN_DATASET_FN = 'data/cell_images_full_32.mat'

TRAIN_BATCH_SIZE = 50
TRAIN_EPOCHS = 10

FLG_SHIM_FLATTEN_MODEL_INPUTS = False


# this decorator will selectively flatten the model inputs
# based on the noted flag
def _deco_flatten_model_input_matrix(arr):
    if FLG_SHIM_FLATTEN_MODEL_INPUTS:
        return _helper_flatten_matrix(arr)
    return arr


# Flatten an matrix for a dense MLP
def _helper_flatten_matrix(arr):
    return arr.reshape(arr.shape[0], -1)


# Build deep net with a few convolutional layers
def build_model_conv_net(input_shape):
    model = keras.Sequential()
    model.add(keras.layers.Conv2D(
      filters=16, kernel_size=3, strides=1, padding='same', input_shape=input_shape
    ))
    model.add(keras.layers.LeakyReLU(0.1))
    model.add(keras.layers.Conv2D(filters=32, kernel_size=3, strides=1, padding='same'))
```

```
    model.add(keras.layers.LeakyReLU(0.1))
    model.add(keras.layers.AveragePooling2D())
    model.add(keras.layers.Dropout(0.25))
    model.add(keras.layers.Conv2D(filters=32, kernel_size=3, strides=1, padding='same'))
    model.add(keras.layers.LeakyReLU(0.1))
    model.add(keras.layers.Conv2D(filters=64, kernel_size=3, strides=1, padding='same'))
    model.add(keras.layers.LeakyReLU(0.1))
    model.add(keras.layers.AveragePooling2D())
    model.add(keras.layers.Dropout(0.25))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(256))
    model.add(keras.layers.LeakyReLU(0.1))
    model.add(keras.layers.Dense(128))
    model.add(keras.layers.LeakyReLU(0.1))
    model.add(keras.layers.Dense(2))
    model.add(keras.layers.Activation('softmax'))
    return model

# entry point
def _kernel():
    # %%
    image_dataset = scipy.io.loadmat(
        TRAIN_DATASET_FN,
        variable_names=(
            'img_size',
            'nrow_model_train',
            'model_train_X', 'model_train_Y', 'model_train_fns',
            'nrow_model_val',
            'model_val_X', 'model_val_Y', 'model_train_fns'
        ),
        squeeze_me=True
    )

    # %%
    n_training_batches = math.ceil(image_dataset['nrow_model_train'] / TRAIN_BATCH_SIZE)

    # Print some statistics on the training data
    print('Training data shape:', image_dataset['model_train_X'].shape)
    print('Number training batches:', n_training_batches)

    print(
        "Proportion of training 1's:",
        sum(image_dataset['model_train_Y']),
        sum(image_dataset['model_train_Y']) / image_dataset['nrow_model_train']
    )

    # Print some statistics on the validation data
    print('Validation data shape:', image_dataset['model_val_X'].shape)
    print('Number validation batches:', n_training_batches)

    print(
        "Proportion of validation 1's:",
        sum(image_dataset['model_val_Y']),
```

```
    sum(image_dataset['model_val_Y']) / image_dataset['nrow_model_val']
)

# %%
# helper functions
# Resets the TF graph
def helper_reset_tf_session():
    curr_session = tf.get_default_session()
    # if current session exists then close it
    if curr_session is not None:
        curr_session.close()
    # reset the tf graph
    keras.backend.clear_session()
    # create a new tf session
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True  # use GPU memory as needed
    s = tf.InteractiveSession(config=config)
    keras.backend.set_session(s)
    return s

# %%
helper_reset_tf_session()

# %%
# A small abstraction to enable switching between models
model = build_model_conv_net(
  (image_dataset['img_size'], image_dataset['img_size'], 3)
)

# %%
# compile the model
model.compile(
    loss='categorical_crossentropy',
    optimizer=keras.optimizers.adamax(lr=1e-2),
    metrics=['accuracy']  # report accuracy during training
)

# %%
# train the model on the dataset
model_fit_history = model.fit(
    _deco_flatten_model_input_matrix(np.stack(map(
        keras.applications.inception_v3.preprocess_input,
        image_dataset['model_train_X']
    ))),
    keras.utils.np_utils.to_categorical(image_dataset['model_train_Y'], 2),
    epochs=TRAIN_EPOCHS,
    batch_size=TRAIN_BATCH_SIZE,
    validation_data=(
        _deco_flatten_model_input_matrix(np.stack(map(
            keras.applications.inception_v3.preprocess_input,
            image_dataset['model_val_X']
        ))),
        keras.utils.np_utils.to_categorical(image_dataset['model_val_Y'], 2)
```

```
        ),
        verbose=0
    )

    print('Model training complete!!')

# %%
# boot
_kernel()
```

Exhibit 7

- The architecture of the trained Convolutional ANN:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 50, 50, 16)        448
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 50, 50, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 50, 50, 32)        4640
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 50, 50, 32)        0
_________________________________________________________________
average_pooling2d_1 (Average (None, 25, 25, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 25, 25, 32)        9248
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 25, 25, 64)        18496
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 25, 25, 64)        0
_________________________________________________________________
average_pooling2d_2 (Average (None, 12, 12, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2359552
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 258
_________________________________________________________________
```

```
activation_1 (Activation)    (None, 2)                 0
=================================================================
Total params: 2,425,538
```

Exhibit 8

- The Python code used to train the Inception v3 ANN to classify the cell images:

```
import numpy as np
import scipy.io
import math

import tensorflow as tf
import keras
import sklearn

TRAIN_DATASET_FN = 'data/cell_images_full_32.mat'

TRAIN_BATCH_SIZE = 50
TRAIN_EPOCHS = 10

FLG_SHIM_FLATTEN_MODEL_INPUTS = True


# this decorator will selectively flatten the model inputs
# based on the noted flag
def _deco_flatten_model_input_matrix(arr):
    if FLG_SHIM_FLATTEN_MODEL_INPUTS:
        return _helper_flatten_matrix(arr)
    return arr


# Flatten an matrix for a dense MLP
def _helper_flatten_matrix(arr):
    return arr.reshape(arr.shape[0], -1)


# Builds a Keras model by fine tuning inception V3 model
def build_model_inceptionV3_fine_tune(input_shape):
    # Exclude the final layer and add custum FC layer
    model = keras.applications.InceptionV3(
        include_top=False, input_shape=input_shape
    )

    # Add global max pooling
    final_layer = keras.layers.GlobalAveragePooling2D()(model.output)

    # Add new FC layer for predicting labels
    final_layer = keras.layers.Dense(256, activation='relu')(final_layer)

    # Add new output layer
    final_layer = keras.layers.Dense(2, activation='softmax')(final_layer)
```

```
    model = keras.engine.training.Model(model.inputs, final_layer)

    return model


# entry point
def _kernel():
    # %%
    image_dataset = scipy.io.loadmat(
        TRAIN_DATASET_FN,
        variable_names=(
            'img_size',
            'nrow_model_train',
            'model_train_X', 'model_train_Y', 'model_train_fns',
            'nrow_model_val',
            'model_val_X', 'model_val_Y', 'model_train_fns'
        ),
        squeeze_me=True
    )

    # %%
    n_training_batches = math.ceil(image_dataset['nrow_model_train'] / TRAIN_BATCH_SIZE)

    # Print some statistics on the training data
    print('Training data shape:', image_dataset['model_train_X'].shape)
    print('Number training batches:', n_training_batches)

    print(
        "Proportion of training 1's:",
        sum(image_dataset['model_train_Y']),
        sum(image_dataset['model_train_Y']) / image_dataset['nrow_model_train']
    )

    # Print some statistics on the validation data
    print('Validation data shape:', image_dataset['model_val_X'].shape)
    print('Number validation batches:', n_training_batches)

    print(
        "Proportion of validation 1's:",
        sum(image_dataset['model_val_Y']),
        sum(image_dataset['model_val_Y']) / image_dataset['nrow_model_val']
    )

    # %%
    # helper functions
    # Resets the TF graph
    def helper_reset_tf_session():
        curr_session = tf.get_default_session()
        # if current session exists then close it
        if curr_session is not None:
            curr_session.close()
        # reset the tf graph
        keras.backend.clear_session()
```

```
        # create a new tf session
        config = tf.ConfigProto()
        config.gpu_options.allow_growth = True  # use GPU memory as needed
        s = tf.InteractiveSession(config=config)
        keras.backend.set_session(s)
        return s

    # %%
    helper_reset_tf_session()

    # %%
    # A small abstraction to enable switching between models
    model = build_model_inceptionV3_fine_tune(
      (image_dataset['img_size'], image_dataset['img_size'], 3)
    )

    # %%
    # compile the model
    model.compile(
        loss='categorical_crossentropy',
        optimizer=keras.optimizers.adamax(lr=1e-2),
        metrics=['accuracy']  # report accuracy during training
    )

    # %%
    # train the model on the dataset
    model_fit_history = model.fit(
        _deco_flatten_model_input_matrix(np.stack(map(
            keras.applications.inception_v3.preprocess_input,
            image_dataset['model_train_X']
        ))),
        keras.utils.np_utils.to_categorical(image_dataset['model_train_Y'], 2),
        epochs=TRAIN_EPOCHS,
        batch_size=TRAIN_BATCH_SIZE,
        validation_data=(
            _deco_flatten_model_input_matrix(np.stack(map(
                keras.applications.inception_v3.preprocess_input,
                image_dataset['model_val_X']
            ))),
            keras.utils.np_utils.to_categorical(image_dataset['model_val_Y'], 2)
        ),
        verbose=0
    )

    print('Model training complete!!')

# %%
# boot
_kernel()
```

Exhibit 9

- The architecture of the trained Inception v3 ANN:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 75, 75, 3)	0	
conv2d_1 (Conv2D)	(None, 37, 37, 32)	864	input_1[0][0]
batch_normalization_1 (BatchNor	(None, 37, 37, 32)	96	conv2d_1[0][0]
activation_1 (Activation)	(None, 37, 37, 32)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 35, 35, 32)	9216	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 35, 35, 32)	96	conv2d_2[0][0]
activation_2 (Activation)	(None, 35, 35, 32)	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, 35, 35, 64)	18432	activation_2[0][0]
batch_normalization_3 (BatchNor	(None, 35, 35, 64)	192	conv2d_3[0][0]
activation_3 (Activation)	(None, 35, 35, 64)	0	batch_normalization_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 17, 17, 64)	0	activation_3[0][0]
conv2d_4 (Conv2D)	(None, 17, 17, 80)	5120	max_pooling2d_1[0][0]
batch_normalization_4 (BatchNor	(None, 17, 17, 80)	240	conv2d_4[0][0]
activation_4 (Activation)	(None, 17, 17, 80)	0	batch_normalization_4[0][0]
conv2d_5 (Conv2D)	(None, 15, 15, 192)	138240	activation_4[0][0]
batch_normalization_5 (BatchNor	(None, 15, 15, 192)	576	conv2d_5[0][0]
activation_5 (Activation)	(None, 15, 15, 192)	0	batch_normalization_5[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 192)	0	activation_5[0][0]
conv2d_9 (Conv2D)	(None, 7, 7, 64)	12288	max_pooling2d_2[0][0]
batch_normalization_9 (BatchNor	(None, 7, 7, 64)	192	conv2d_9[0][0]
activation_9 (Activation)	(None, 7, 7, 64)	0	batch_normalization_9[0][0]
conv2d_7 (Conv2D)	(None, 7, 7, 48)	9216	max_pooling2d_2[0][0]
conv2d_10 (Conv2D)	(None, 7, 7, 96)	55296	activation_9[0][0]
batch_normalization_7 (BatchNor	(None, 7, 7, 48)	144	conv2d_7[0][0]

```
batch_normalization_10 (BatchNo (None, 7, 7, 96)     288         conv2d_10[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, 7, 7, 48)     0           batch_normalization_7[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, 7, 7, 96)     0           batch_normalization_10[0][0]
__________________________________________________________________________________________________
average_pooling2d_1 (AveragePoo (None, 7, 7, 192)    0           max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 7, 7, 64)     12288       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 7, 7, 64)     76800       activation_7[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 7, 7, 96)     82944       activation_10[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 7, 7, 32)     6144        average_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 7, 7, 64)     192         conv2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 7, 7, 64)     192         conv2d_8[0][0]
__________________________________________________________________________________________________
batch_normalization_11 (BatchNo (None, 7, 7, 96)     288         conv2d_11[0][0]
__________________________________________________________________________________________________
batch_normalization_12 (BatchNo (None, 7, 7, 32)     96          conv2d_12[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 7, 7, 64)     0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 7, 7, 64)     0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
activation_11 (Activation)      (None, 7, 7, 96)     0           batch_normalization_11[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, 7, 7, 32)     0           batch_normalization_12[0][0]
__________________________________________________________________________________________________
mixed0 (Concatenate)            (None, 7, 7, 256)    0           activation_6[0][0]
                                                                 activation_8[0][0]
                                                                 activation_11[0][0]
                                                                 activation_12[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 7, 7, 64)     16384       mixed0[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, 7, 7, 64)     192         conv2d_16[0][0]
__________________________________________________________________________________________________
activation_16 (Activation)      (None, 7, 7, 64)     0           batch_normalization_16[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 7, 7, 48)     12288       mixed0[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 7, 7, 96)     55296       activation_16[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, 7, 7, 48)     144         conv2d_14[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 7, 7, 96)     288         conv2d_17[0][0]
__________________________________________________________________________________________________
activation_14 (Activation)      (None, 7, 7, 48)     0           batch_normalization_14[0][0]
__________________________________________________________________________________________________
```

```
activation_17 (Activation)      (None, 7, 7, 96)     0           batch_normalization_17[0][0]
__________________________________________________________________________________________
average_pooling2d_2 (AveragePoo (None, 7, 7, 256)    0           mixed0[0][0]
__________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 7, 7, 64)     16384       mixed0[0][0]
__________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 7, 7, 64)     76800       activation_14[0][0]
__________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 7, 7, 96)     82944       activation_17[0][0]
__________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 7, 7, 64)     16384       average_pooling2d_2[0][0]
__________________________________________________________________________________________
batch_normalization_13 (BatchNo (None, 7, 7, 64)     192         conv2d_13[0][0]
__________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, 7, 7, 64)     192         conv2d_15[0][0]
__________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 7, 7, 96)     288         conv2d_18[0][0]
__________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, 7, 7, 64)     192         conv2d_19[0][0]
__________________________________________________________________________________________
activation_13 (Activation)      (None, 7, 7, 64)     0           batch_normalization_13[0][0]
__________________________________________________________________________________________
activation_15 (Activation)      (None, 7, 7, 64)     0           batch_normalization_15[0][0]
__________________________________________________________________________________________
activation_18 (Activation)      (None, 7, 7, 96)     0           batch_normalization_18[0][0]
__________________________________________________________________________________________
activation_19 (Activation)      (None, 7, 7, 64)     0           batch_normalization_19[0][0]
__________________________________________________________________________________________
mixed1 (Concatenate)            (None, 7, 7, 288)    0           activation_13[0][0]
                                                                 activation_15[0][0]
                                                                 activation_18[0][0]
                                                                 activation_19[0][0]
__________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 7, 7, 64)     18432       mixed1[0][0]
__________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, 7, 7, 64)     192         conv2d_23[0][0]
__________________________________________________________________________________________
activation_23 (Activation)      (None, 7, 7, 64)     0           batch_normalization_23[0][0]
__________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 7, 7, 48)     13824       mixed1[0][0]
__________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 7, 7, 96)     55296       activation_23[0][0]
__________________________________________________________________________________________
batch_normalization_21 (BatchNo (None, 7, 7, 48)     144         conv2d_21[0][0]
__________________________________________________________________________________________
batch_normalization_24 (BatchNo (None, 7, 7, 96)     288         conv2d_24[0][0]
__________________________________________________________________________________________
activation_21 (Activation)      (None, 7, 7, 48)     0           batch_normalization_21[0][0]
__________________________________________________________________________________________
activation_24 (Activation)      (None, 7, 7, 96)     0           batch_normalization_24[0][0]
__________________________________________________________________________________________
average_pooling2d_3 (AveragePoo (None, 7, 7, 288)    0           mixed1[0][0]
__________________________________________________________________________________________
```

```
conv2d_20 (Conv2D)              (None, 7, 7, 64)     18432       mixed1[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 7, 7, 64)     76800       activation_21[0][0]
__________________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, 7, 7, 96)     82944       activation_24[0][0]
__________________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 7, 7, 64)     18432       average_pooling2d_3[0][0]
__________________________________________________________________________________________________
batch_normalization_20 (BatchNo (None, 7, 7, 64)     192         conv2d_20[0][0]
__________________________________________________________________________________________________
batch_normalization_22 (BatchNo (None, 7, 7, 64)     192         conv2d_22[0][0]
__________________________________________________________________________________________________
batch_normalization_25 (BatchNo (None, 7, 7, 96)     288         conv2d_25[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, 7, 7, 64)     192         conv2d_26[0][0]
__________________________________________________________________________________________________
activation_20 (Activation)      (None, 7, 7, 64)     0           batch_normalization_20[0][0]
__________________________________________________________________________________________________
activation_22 (Activation)      (None, 7, 7, 64)     0           batch_normalization_22[0][0]
__________________________________________________________________________________________________
activation_25 (Activation)      (None, 7, 7, 96)     0           batch_normalization_25[0][0]
__________________________________________________________________________________________________
activation_26 (Activation)      (None, 7, 7, 64)     0           batch_normalization_26[0][0]
__________________________________________________________________________________________________
mixed2 (Concatenate)            (None, 7, 7, 288)    0           activation_20[0][0]
                                                                 activation_22[0][0]
                                                                 activation_25[0][0]
                                                                 activation_26[0][0]
__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 7, 7, 64)     18432       mixed2[0][0]
__________________________________________________________________________________________________
batch_normalization_28 (BatchNo (None, 7, 7, 64)     192         conv2d_28[0][0]
__________________________________________________________________________________________________
activation_28 (Activation)      (None, 7, 7, 64)     0           batch_normalization_28[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 7, 7, 96)     55296       activation_28[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 7, 7, 96)     288         conv2d_29[0][0]
__________________________________________________________________________________________________
activation_29 (Activation)      (None, 7, 7, 96)     0           batch_normalization_29[0][0]
__________________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, 3, 3, 384)    995328      mixed2[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 3, 3, 96)     82944       activation_29[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, 3, 3, 384)    1152        conv2d_27[0][0]
__________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 3, 3, 96)     288         conv2d_30[0][0]
__________________________________________________________________________________________________
activation_27 (Activation)      (None, 3, 3, 384)    0           batch_normalization_27[0][0]
__________________________________________________________________________________________________
activation_30 (Activation)      (None, 3, 3, 96)     0           batch_normalization_30[0][0]
__________________________________________________________________________________________________
```

```
max_pooling2d_3 (MaxPooling2D)  (None, 3, 3, 288)    0           mixed2[0][0]
__________________________________________________________________________________________________
mixed3 (Concatenate)            (None, 3, 3, 768)    0           activation_27[0][0]
                                                                 activation_30[0][0]
                                                                 max_pooling2d_3[0][0]
__________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 3, 3, 128)    98304       mixed3[0][0]
__________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, 3, 3, 128)    384         conv2d_35[0][0]
__________________________________________________________________________________________________
activation_35 (Activation)      (None, 3, 3, 128)    0           batch_normalization_35[0][0]
__________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 3, 3, 128)    114688      activation_35[0][0]
__________________________________________________________________________________________________
batch_normalization_36 (BatchNo (None, 3, 3, 128)    384         conv2d_36[0][0]
__________________________________________________________________________________________________
activation_36 (Activation)      (None, 3, 3, 128)    0           batch_normalization_36[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 3, 3, 128)    98304       mixed3[0][0]
__________________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, 3, 3, 128)    114688      activation_36[0][0]
__________________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 3, 3, 128)    384         conv2d_32[0][0]
__________________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 3, 3, 128)    384         conv2d_37[0][0]
__________________________________________________________________________________________________
activation_32 (Activation)      (None, 3, 3, 128)    0           batch_normalization_32[0][0]
__________________________________________________________________________________________________
activation_37 (Activation)      (None, 3, 3, 128)    0           batch_normalization_37[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 3, 3, 128)    114688      activation_32[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 3, 3, 128)    114688      activation_37[0][0]
__________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 3, 3, 128)    384         conv2d_33[0][0]
__________________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, 3, 3, 128)    384         conv2d_38[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, 3, 3, 128)    0           batch_normalization_33[0][0]
__________________________________________________________________________________________________
activation_38 (Activation)      (None, 3, 3, 128)    0           batch_normalization_38[0][0]
__________________________________________________________________________________________________
average_pooling2d_4 (AveragePoo (None, 3, 3, 768)    0           mixed3[0][0]
__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 3, 3, 192)    147456      mixed3[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 3, 3, 192)    172032      activation_33[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 3, 3, 192)    172032      activation_38[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 3, 3, 192)    147456      average_pooling2d_4[0][0]
__________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 3, 3, 192)    576         conv2d_31[0][0]
```

```
batch_normalization_34 (BatchNo (None, 3, 3, 192)    576         conv2d_34[0][0]
batch_normalization_39 (BatchNo (None, 3, 3, 192)    576         conv2d_39[0][0]
batch_normalization_40 (BatchNo (None, 3, 3, 192)    576         conv2d_40[0][0]
activation_31 (Activation)      (None, 3, 3, 192)    0           batch_normalization_31[0][0]
activation_34 (Activation)      (None, 3, 3, 192)    0           batch_normalization_34[0][0]
activation_39 (Activation)      (None, 3, 3, 192)    0           batch_normalization_39[0][0]
activation_40 (Activation)      (None, 3, 3, 192)    0           batch_normalization_40[0][0]
mixed4 (Concatenate)            (None, 3, 3, 768)    0           activation_31[0][0]
                                                                 activation_34[0][0]
                                                                 activation_39[0][0]
                                                                 activation_40[0][0]
conv2d_45 (Conv2D)              (None, 3, 3, 160)    122880      mixed4[0][0]
batch_normalization_45 (BatchNo (None, 3, 3, 160)    480         conv2d_45[0][0]
activation_45 (Activation)      (None, 3, 3, 160)    0           batch_normalization_45[0][0]
conv2d_46 (Conv2D)              (None, 3, 3, 160)    179200      activation_45[0][0]
batch_normalization_46 (BatchNo (None, 3, 3, 160)    480         conv2d_46[0][0]
activation_46 (Activation)      (None, 3, 3, 160)    0           batch_normalization_46[0][0]
conv2d_42 (Conv2D)              (None, 3, 3, 160)    122880      mixed4[0][0]
conv2d_47 (Conv2D)              (None, 3, 3, 160)    179200      activation_46[0][0]
batch_normalization_42 (BatchNo (None, 3, 3, 160)    480         conv2d_42[0][0]
batch_normalization_47 (BatchNo (None, 3, 3, 160)    480         conv2d_47[0][0]
activation_42 (Activation)      (None, 3, 3, 160)    0           batch_normalization_42[0][0]
activation_47 (Activation)      (None, 3, 3, 160)    0           batch_normalization_47[0][0]
conv2d_43 (Conv2D)              (None, 3, 3, 160)    179200      activation_42[0][0]
conv2d_48 (Conv2D)              (None, 3, 3, 160)    179200      activation_47[0][0]
batch_normalization_43 (BatchNo (None, 3, 3, 160)    480         conv2d_43[0][0]
batch_normalization_48 (BatchNo (None, 3, 3, 160)    480         conv2d_48[0][0]
activation_43 (Activation)      (None, 3, 3, 160)    0           batch_normalization_43[0][0]
```

```
activation_48 (Activation)      (None, 3, 3, 160)    0           batch_normalization_48[0][0]
__________________________________________________________________________________________________
average_pooling2d_5 (AveragePoo (None, 3, 3, 768)    0           mixed4[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 3, 3, 192)    147456      mixed4[0][0]
__________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, 3, 3, 192)    215040      activation_43[0][0]
__________________________________________________________________________________________________
conv2d_49 (Conv2D)              (None, 3, 3, 192)    215040      activation_48[0][0]
__________________________________________________________________________________________________
conv2d_50 (Conv2D)              (None, 3, 3, 192)    147456      average_pooling2d_5[0][0]
__________________________________________________________________________________________________
batch_normalization_41 (BatchNo (None, 3, 3, 192)    576         conv2d_41[0][0]
__________________________________________________________________________________________________
batch_normalization_44 (BatchNo (None, 3, 3, 192)    576         conv2d_44[0][0]
__________________________________________________________________________________________________
batch_normalization_49 (BatchNo (None, 3, 3, 192)    576         conv2d_49[0][0]
__________________________________________________________________________________________________
batch_normalization_50 (BatchNo (None, 3, 3, 192)    576         conv2d_50[0][0]
__________________________________________________________________________________________________
activation_41 (Activation)      (None, 3, 3, 192)    0           batch_normalization_41[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, 3, 3, 192)    0           batch_normalization_44[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 3, 3, 192)    0           batch_normalization_49[0][0]
__________________________________________________________________________________________________
activation_50 (Activation)      (None, 3, 3, 192)    0           batch_normalization_50[0][0]
__________________________________________________________________________________________________
mixed5 (Concatenate)            (None, 3, 3, 768)    0           activation_41[0][0]
                                                                 activation_44[0][0]
                                                                 activation_49[0][0]
                                                                 activation_50[0][0]
__________________________________________________________________________________________________
conv2d_55 (Conv2D)              (None, 3, 3, 160)    122880      mixed5[0][0]
__________________________________________________________________________________________________
batch_normalization_55 (BatchNo (None, 3, 3, 160)    480         conv2d_55[0][0]
__________________________________________________________________________________________________
activation_55 (Activation)      (None, 3, 3, 160)    0           batch_normalization_55[0][0]
__________________________________________________________________________________________________
conv2d_56 (Conv2D)              (None, 3, 3, 160)    179200      activation_55[0][0]
__________________________________________________________________________________________________
batch_normalization_56 (BatchNo (None, 3, 3, 160)    480         conv2d_56[0][0]
__________________________________________________________________________________________________
activation_56 (Activation)      (None, 3, 3, 160)    0           batch_normalization_56[0][0]
__________________________________________________________________________________________________
conv2d_52 (Conv2D)              (None, 3, 3, 160)    122880      mixed5[0][0]
__________________________________________________________________________________________________
conv2d_57 (Conv2D)              (None, 3, 3, 160)    179200      activation_56[0][0]
__________________________________________________________________________________________________
batch_normalization_52 (BatchNo (None, 3, 3, 160)    480         conv2d_52[0][0]
__________________________________________________________________________________________________
batch_normalization_57 (BatchNo (None, 3, 3, 160)    480         conv2d_57[0][0]
__________________________________________________________________________________________________
```

```
activation_52 (Activation)      (None, 3, 3, 160)    0           batch_normalization_52[0][0]
__________________________________________________________________________________________________
activation_57 (Activation)      (None, 3, 3, 160)    0           batch_normalization_57[0][0]
__________________________________________________________________________________________________
conv2d_53 (Conv2D)              (None, 3, 3, 160)    179200      activation_52[0][0]
__________________________________________________________________________________________________
conv2d_58 (Conv2D)              (None, 3, 3, 160)    179200      activation_57[0][0]
__________________________________________________________________________________________________
batch_normalization_53 (BatchNo (None, 3, 3, 160)    480         conv2d_53[0][0]
__________________________________________________________________________________________________
batch_normalization_58 (BatchNo (None, 3, 3, 160)    480         conv2d_58[0][0]
__________________________________________________________________________________________________
activation_53 (Activation)      (None, 3, 3, 160)    0           batch_normalization_53[0][0]
__________________________________________________________________________________________________
activation_58 (Activation)      (None, 3, 3, 160)    0           batch_normalization_58[0][0]
__________________________________________________________________________________________________
average_pooling2d_6 (AveragePoo (None, 3, 3, 768)    0           mixed5[0][0]
__________________________________________________________________________________________________
conv2d_51 (Conv2D)              (None, 3, 3, 192)    147456      mixed5[0][0]
__________________________________________________________________________________________________
conv2d_54 (Conv2D)              (None, 3, 3, 192)    215040      activation_53[0][0]
__________________________________________________________________________________________________
conv2d_59 (Conv2D)              (None, 3, 3, 192)    215040      activation_58[0][0]
__________________________________________________________________________________________________
conv2d_60 (Conv2D)              (None, 3, 3, 192)    147456      average_pooling2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_51 (BatchNo (None, 3, 3, 192)    576         conv2d_51[0][0]
__________________________________________________________________________________________________
batch_normalization_54 (BatchNo (None, 3, 3, 192)    576         conv2d_54[0][0]
__________________________________________________________________________________________________
batch_normalization_59 (BatchNo (None, 3, 3, 192)    576         conv2d_59[0][0]
__________________________________________________________________________________________________
batch_normalization_60 (BatchNo (None, 3, 3, 192)    576         conv2d_60[0][0]
__________________________________________________________________________________________________
activation_51 (Activation)      (None, 3, 3, 192)    0           batch_normalization_51[0][0]
__________________________________________________________________________________________________
activation_54 (Activation)      (None, 3, 3, 192)    0           batch_normalization_54[0][0]
__________________________________________________________________________________________________
activation_59 (Activation)      (None, 3, 3, 192)    0           batch_normalization_59[0][0]
__________________________________________________________________________________________________
activation_60 (Activation)      (None, 3, 3, 192)    0           batch_normalization_60[0][0]
__________________________________________________________________________________________________
mixed6 (Concatenate)            (None, 3, 3, 768)    0           activation_51[0][0]
                                                                 activation_54[0][0]
                                                                 activation_59[0][0]
                                                                 activation_60[0][0]
__________________________________________________________________________________________________
conv2d_65 (Conv2D)              (None, 3, 3, 192)    147456      mixed6[0][0]
__________________________________________________________________________________________________
batch_normalization_65 (BatchNo (None, 3, 3, 192)    576         conv2d_65[0][0]
__________________________________________________________________________________________________
activation_65 (Activation)      (None, 3, 3, 192)    0           batch_normalization_65[0][0]
__________________________________________________________________________________________________
```

```
conv2d_66 (Conv2D)              (None, 3, 3, 192)    258048      activation_65[0][0]
__________________________________________________________________________________________________
batch_normalization_66 (BatchNo (None, 3, 3, 192)    576         conv2d_66[0][0]
__________________________________________________________________________________________________
activation_66 (Activation)      (None, 3, 3, 192)    0           batch_normalization_66[0][0]
__________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, 3, 3, 192)    147456      mixed6[0][0]
__________________________________________________________________________________________________
conv2d_67 (Conv2D)              (None, 3, 3, 192)    258048      activation_66[0][0]
__________________________________________________________________________________________________
batch_normalization_62 (BatchNo (None, 3, 3, 192)    576         conv2d_62[0][0]
__________________________________________________________________________________________________
batch_normalization_67 (BatchNo (None, 3, 3, 192)    576         conv2d_67[0][0]
__________________________________________________________________________________________________
activation_62 (Activation)      (None, 3, 3, 192)    0           batch_normalization_62[0][0]
__________________________________________________________________________________________________
activation_67 (Activation)      (None, 3, 3, 192)    0           batch_normalization_67[0][0]
__________________________________________________________________________________________________
conv2d_63 (Conv2D)              (None, 3, 3, 192)    258048      activation_62[0][0]
__________________________________________________________________________________________________
conv2d_68 (Conv2D)              (None, 3, 3, 192)    258048      activation_67[0][0]
__________________________________________________________________________________________________
batch_normalization_63 (BatchNo (None, 3, 3, 192)    576         conv2d_63[0][0]
__________________________________________________________________________________________________
batch_normalization_68 (BatchNo (None, 3, 3, 192)    576         conv2d_68[0][0]
__________________________________________________________________________________________________
activation_63 (Activation)      (None, 3, 3, 192)    0           batch_normalization_63[0][0]
__________________________________________________________________________________________________
activation_68 (Activation)      (None, 3, 3, 192)    0           batch_normalization_68[0][0]
__________________________________________________________________________________________________
average_pooling2d_7 (AveragePoo (None, 3, 3, 768)    0           mixed6[0][0]
__________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, 3, 3, 192)    147456      mixed6[0][0]
__________________________________________________________________________________________________
conv2d_64 (Conv2D)              (None, 3, 3, 192)    258048      activation_63[0][0]
__________________________________________________________________________________________________
conv2d_69 (Conv2D)              (None, 3, 3, 192)    258048      activation_68[0][0]
__________________________________________________________________________________________________
conv2d_70 (Conv2D)              (None, 3, 3, 192)    147456      average_pooling2d_7[0][0]
__________________________________________________________________________________________________
batch_normalization_61 (BatchNo (None, 3, 3, 192)    576         conv2d_61[0][0]
__________________________________________________________________________________________________
batch_normalization_64 (BatchNo (None, 3, 3, 192)    576         conv2d_64[0][0]
__________________________________________________________________________________________________
batch_normalization_69 (BatchNo (None, 3, 3, 192)    576         conv2d_69[0][0]
__________________________________________________________________________________________________
batch_normalization_70 (BatchNo (None, 3, 3, 192)    576         conv2d_70[0][0]
__________________________________________________________________________________________________
activation_61 (Activation)      (None, 3, 3, 192)    0           batch_normalization_61[0][0]
__________________________________________________________________________________________________
activation_64 (Activation)      (None, 3, 3, 192)    0           batch_normalization_64[0][0]
__________________________________________________________________________________________________
activation_69 (Activation)      (None, 3, 3, 192)    0           batch_normalization_69[0][0]
```

```
__________________________________________________________________________________________________
activation_70 (Activation)      (None, 3, 3, 192)    0           batch_normalization_70[0][0]
__________________________________________________________________________________________________
mixed7 (Concatenate)            (None, 3, 3, 768)    0           activation_61[0][0]
                                                                 activation_64[0][0]
                                                                 activation_69[0][0]
                                                                 activation_70[0][0]
__________________________________________________________________________________________________
conv2d_73 (Conv2D)              (None, 3, 3, 192)    147456      mixed7[0][0]
__________________________________________________________________________________________________
batch_normalization_73 (BatchNo (None, 3, 3, 192)    576         conv2d_73[0][0]
__________________________________________________________________________________________________
activation_73 (Activation)      (None, 3, 3, 192)    0           batch_normalization_73[0][0]
__________________________________________________________________________________________________
conv2d_74 (Conv2D)              (None, 3, 3, 192)    258048      activation_73[0][0]
__________________________________________________________________________________________________
batch_normalization_74 (BatchNo (None, 3, 3, 192)    576         conv2d_74[0][0]
__________________________________________________________________________________________________
activation_74 (Activation)      (None, 3, 3, 192)    0           batch_normalization_74[0][0]
__________________________________________________________________________________________________
conv2d_71 (Conv2D)              (None, 3, 3, 192)    147456      mixed7[0][0]
__________________________________________________________________________________________________
conv2d_75 (Conv2D)              (None, 3, 3, 192)    258048      activation_74[0][0]
__________________________________________________________________________________________________
batch_normalization_71 (BatchNo (None, 3, 3, 192)    576         conv2d_71[0][0]
__________________________________________________________________________________________________
batch_normalization_75 (BatchNo (None, 3, 3, 192)    576         conv2d_75[0][0]
__________________________________________________________________________________________________
activation_71 (Activation)      (None, 3, 3, 192)    0           batch_normalization_71[0][0]
__________________________________________________________________________________________________
activation_75 (Activation)      (None, 3, 3, 192)    0           batch_normalization_75[0][0]
__________________________________________________________________________________________________
conv2d_72 (Conv2D)              (None, 1, 1, 320)    552960      activation_71[0][0]
__________________________________________________________________________________________________
conv2d_76 (Conv2D)              (None, 1, 1, 192)    331776      activation_75[0][0]
__________________________________________________________________________________________________
batch_normalization_72 (BatchNo (None, 1, 1, 320)    960         conv2d_72[0][0]
__________________________________________________________________________________________________
batch_normalization_76 (BatchNo (None, 1, 1, 192)    576         conv2d_76[0][0]
__________________________________________________________________________________________________
activation_72 (Activation)      (None, 1, 1, 320)    0           batch_normalization_72[0][0]
__________________________________________________________________________________________________
activation_76 (Activation)      (None, 1, 1, 192)    0           batch_normalization_76[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 1, 1, 768)    0           mixed7[0][0]
__________________________________________________________________________________________________
mixed8 (Concatenate)            (None, 1, 1, 1280)   0           activation_72[0][0]
                                                                 activation_76[0][0]
                                                                 max_pooling2d_4[0][0]
__________________________________________________________________________________________________
conv2d_81 (Conv2D)              (None, 1, 1, 448)    573440      mixed8[0][0]
__________________________________________________________________________________________________
batch_normalization_81 (BatchNo (None, 1, 1, 448)    1344        conv2d_81[0][0]
__________________________________________________________________________________________________
```

```
activation_81 (Activation)      (None, 1, 1, 448)    0           batch_normalization_81[0][0]
__________________________________________________________________________________________________
conv2d_78 (Conv2D)              (None, 1, 1, 384)    491520      mixed8[0][0]
__________________________________________________________________________________________________
conv2d_82 (Conv2D)              (None, 1, 1, 384)    1548288     activation_81[0][0]
__________________________________________________________________________________________________
batch_normalization_78 (BatchNo (None, 1, 1, 384)    1152        conv2d_78[0][0]
__________________________________________________________________________________________________
batch_normalization_82 (BatchNo (None, 1, 1, 384)    1152        conv2d_82[0][0]
__________________________________________________________________________________________________
activation_78 (Activation)      (None, 1, 1, 384)    0           batch_normalization_78[0][0]
__________________________________________________________________________________________________
activation_82 (Activation)      (None, 1, 1, 384)    0           batch_normalization_82[0][0]
__________________________________________________________________________________________________
conv2d_79 (Conv2D)              (None, 1, 1, 384)    442368      activation_78[0][0]
__________________________________________________________________________________________________
conv2d_80 (Conv2D)              (None, 1, 1, 384)    442368      activation_78[0][0]
__________________________________________________________________________________________________
conv2d_83 (Conv2D)              (None, 1, 1, 384)    442368      activation_82[0][0]
__________________________________________________________________________________________________
conv2d_84 (Conv2D)              (None, 1, 1, 384)    442368      activation_82[0][0]
__________________________________________________________________________________________________
average_pooling2d_8 (AveragePoo (None, 1, 1, 1280)   0           mixed8[0][0]
__________________________________________________________________________________________________
conv2d_77 (Conv2D)              (None, 1, 1, 320)    409600      mixed8[0][0]
__________________________________________________________________________________________________
batch_normalization_79 (BatchNo (None, 1, 1, 384)    1152        conv2d_79[0][0]
__________________________________________________________________________________________________
batch_normalization_80 (BatchNo (None, 1, 1, 384)    1152        conv2d_80[0][0]
__________________________________________________________________________________________________
batch_normalization_83 (BatchNo (None, 1, 1, 384)    1152        conv2d_83[0][0]
__________________________________________________________________________________________________
batch_normalization_84 (BatchNo (None, 1, 1, 384)    1152        conv2d_84[0][0]
__________________________________________________________________________________________________
conv2d_85 (Conv2D)              (None, 1, 1, 192)    245760      average_pooling2d_8[0][0]
__________________________________________________________________________________________________
batch_normalization_77 (BatchNo (None, 1, 1, 320)    960         conv2d_77[0][0]
__________________________________________________________________________________________________
activation_79 (Activation)      (None, 1, 1, 384)    0           batch_normalization_79[0][0]
__________________________________________________________________________________________________
activation_80 (Activation)      (None, 1, 1, 384)    0           batch_normalization_80[0][0]
__________________________________________________________________________________________________
activation_83 (Activation)      (None, 1, 1, 384)    0           batch_normalization_83[0][0]
__________________________________________________________________________________________________
activation_84 (Activation)      (None, 1, 1, 384)    0           batch_normalization_84[0][0]
__________________________________________________________________________________________________
batch_normalization_85 (BatchNo (None, 1, 1, 192)    576         conv2d_85[0][0]
__________________________________________________________________________________________________
activation_77 (Activation)      (None, 1, 1, 320)    0           batch_normalization_77[0][0]
__________________________________________________________________________________________________
mixed9_0 (Concatenate)          (None, 1, 1, 768)    0           activation_79[0][0]
                                                                 activation_80[0][0]
__________________________________________________________________________________________________
```

```
concatenate_1 (Concatenate)     (None, 1, 1, 768)    0           activation_83[0][0]
                                                                 activation_84[0][0]
__________________________________________________________________________________________________
activation_85 (Activation)      (None, 1, 1, 192)    0           batch_normalization_85[0][0]
__________________________________________________________________________________________________
mixed9 (Concatenate)            (None, 1, 1, 2048)   0           activation_77[0][0]
                                                                 mixed9_0[0][0]
                                                                 concatenate_1[0][0]
                                                                 activation_85[0][0]
__________________________________________________________________________________________________
conv2d_90 (Conv2D)              (None, 1, 1, 448)    917504      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_90 (BatchNo (None, 1, 1, 448)    1344        conv2d_90[0][0]
__________________________________________________________________________________________________
activation_90 (Activation)      (None, 1, 1, 448)    0           batch_normalization_90[0][0]
__________________________________________________________________________________________________
conv2d_87 (Conv2D)              (None, 1, 1, 384)    786432      mixed9[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, 1, 1, 384)    1548288     activation_90[0][0]
__________________________________________________________________________________________________
batch_normalization_87 (BatchNo (None, 1, 1, 384)    1152        conv2d_87[0][0]
__________________________________________________________________________________________________
batch_normalization_91 (BatchNo (None, 1, 1, 384)    1152        conv2d_91[0][0]
__________________________________________________________________________________________________
activation_87 (Activation)      (None, 1, 1, 384)    0           batch_normalization_87[0][0]
__________________________________________________________________________________________________
activation_91 (Activation)      (None, 1, 1, 384)    0           batch_normalization_91[0][0]
__________________________________________________________________________________________________
conv2d_88 (Conv2D)              (None, 1, 1, 384)    442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_89 (Conv2D)              (None, 1, 1, 384)    442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 1, 1, 384)    442368      activation_91[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 1, 1, 384)    442368      activation_91[0][0]
__________________________________________________________________________________________________
average_pooling2d_9 (AveragePoo (None, 1, 1, 2048)   0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_86 (Conv2D)              (None, 1, 1, 320)    655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, 1, 1, 384)    1152        conv2d_88[0][0]
__________________________________________________________________________________________________
batch_normalization_89 (BatchNo (None, 1, 1, 384)    1152        conv2d_89[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, 1, 1, 384)    1152        conv2d_92[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, 1, 1, 384)    1152        conv2d_93[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 1, 1, 192)    393216      average_pooling2d_9[0][0]
__________________________________________________________________________________________________
batch_normalization_86 (BatchNo (None, 1, 1, 320)    960         conv2d_86[0][0]
__________________________________________________________________________________________________
activation_88 (Activation)      (None, 1, 1, 384)    0           batch_normalization_88[0][0]
```

```
__________________________________________________________________________________________________
activation_89 (Activation)      (None, 1, 1, 384)    0           batch_normalization_89[0][0]
__________________________________________________________________________________________________
activation_92 (Activation)      (None, 1, 1, 384)    0           batch_normalization_92[0][0]
__________________________________________________________________________________________________
activation_93 (Activation)      (None, 1, 1, 384)    0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
batch_normalization_94 (BatchNo (None, 1, 1, 192)    576         conv2d_94[0][0]
__________________________________________________________________________________________________
activation_86 (Activation)      (None, 1, 1, 320)    0           batch_normalization_86[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)          (None, 1, 1, 768)    0           activation_88[0][0]
                                                                 activation_89[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 1, 1, 768)    0           activation_92[0][0]
                                                                 activation_93[0][0]
__________________________________________________________________________________________________
activation_94 (Activation)      (None, 1, 1, 192)    0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 1, 1, 2048)   0           activation_86[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_2[0][0]
                                                                 activation_94[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 2048)         0           mixed10[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          524544      global_average_pooling2d_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 2)            514         dense_1[0][0]
==================================================================================================
Total params: 22,327,842
```