SPRINT IV

Date	19 November 2022
Team ID	PNT2022TMID03996
Project Name	Project - AI - Based localisation and classification of skin disease with Erythema
Maximum Marks	-

TRAINING USING YOLO MODEL:

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
"""
MODIFIED FROM keras-yolo3 PACKAGE, https://github.com/qqwweee/keras-yolo3
Retrain the YOLO model for your own dataset.
"""

import ...

def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


src_path = os.path.join(get_parent_dir(0), "src")
sys.path.append(src_path)

utils_path = os.path.join(get_parent_dir(1), "Utils")
sys.path.append(utils_path)

import numpy as np
import keras.backend as K
from keras.layers import Input, Lambda
from keras.models import Model
from keras.optimizers import Adam
from keras.callbacks import (
    TensorBoard,
    ModelCheckpoint,
    ReduceLROnPlateau,
    EarlyStopping,
)
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
log_dir = Model_Folder
anchors_path = os.path.join(keras_path, "model_data", "yolo_anchors.txt")
weights_path = os.path.join(keras_path, "yolo.h5")

FLAGS = None

if __name__ == "__main__":
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--annotation_file",
        type=str,
        default=YOLO_filename,
        help="Path to annotation file for Yolo. Default is " + YOLO_filename,
    )
    parser.add_argument(
        "--classes_file",
        type=str,
        default=YOLO_classname,
        help="Path to YOLO classnames. Default is " + YOLO_classname,
    )

    parser.add_argument(
        "--log_dir",
        type=str,
        default=log_dir,
        help="Folder to save training logs and trained weights to. Default is "
        + log_dir,
    )
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
)
from keras_yolo3.yolo3.model import (
    preprocess_true_boxes,
    yolo_body,
    tiny_yolo_body,
    yolo_loss,
)
from keras_yolo3.yolo3.utils import get_random_data
from PIL import Image
from time import time
import tensorflow.compat.v1 as tf
import pickle

from Train_Utils import (
    get_classes,
    get_anchors,
    create_model,
    create_tiny_model,
    data_generator,
    data_generator_wrapper,
    ChangeToOtherMachine,
)


keras_path = os.path.join(src_path, "keras_yolo3")
Data_Folder = os.path.join(get_parent_dir(1), "Data")
Image_Folder = os.path.join(Data_Folder, "Source_Images", "Training_Images")
VoTT_Folder = os.path.join(Image_Folder, "vott-csv-export")
YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")

Model_Folder = os.path.join(Data_Folder, "Model_Weights")
YOLO_classname = os.path.join(Model_Folder, "data_classes.txt")

log_dir = Model_Folder
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
    parser.add_argument(
        "--anchors_path",
        type=str,
        default=anchors_path,
        help="Path to YOLO anchors. Default is " + anchors_path,
    )

    parser.add_argument(
        "--weights_path",
        type=str,
        default=weights_path,
        help="Path to pre-trained YOLO weights. Default is " + weights_path,
    )
    parser.add_argument(
        "--val_split",
        type=float,
        default=0.1,
        help="Percentage of training set to be used for validation. Default is 10%.",
    )
    parser.add_argument(
        "--is_tiny",
        default=False,
        action="store_true",
        help="Use the tiny Yolo version for better performance and less accuracy. Default is False.",
    )
    parser.add_argument(
        "--random_seed",
        type=float,
        default=None,
        help="Random seed value to make script deterministic. Default is 'None', i.e. non-deterministic.",
    )
    parser.add_argument(
        "--epochs",
        type=float,
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
        default=51,
        help="Number of epochs for training last layers and number of epochs for fine-tuning layers. Default is 51.",
    )
    parser.add_argument(
        "--warnings",
        default=False,
        action="store_true",
        help="Display warning messages. Default is False.",
    )

    FLAGS = parser.parse_args()

    if not FLAGS.warnings:
        tf.logging.set_verbosity(tf.logging.ERROR)
        os.environ['TF_CPP_MIN_LOG_LEVEL']='3'
        warnings.filterwarnings("ignore")

    np.random.seed(FLAGS.random_seed)

    log_dir = FLAGS.log_dir

    class_names = get_classes(FLAGS.classes_file)
    num_classes = len(class_names)
    anchors = get_anchors(FLAGS.anchors_path)
    weights_path = FLAGS.weights_path

    input_shape = (416, 416)  # multiple of 32, height, width
    epoch1, epoch2 = FLAGS.epochs, FLAGS.epochs

    is_tiny_version = len(anchors) == 6  # default setting
    if FLAGS.is_tiny:
        model = create_tiny_model(
            input_shape, anchors, num_classes, freeze_body=2, weights_path=weights_path
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
    else:
        model = create_model(
            input_shape, anchors, num_classes, freeze_body=2, weights_path=weights_path
        )  # make sure you know what you freeze

    log_dir_time = os.path.join(log_dir, "{}".format(int(time())))
    logging = TensorBoard(log_dir=log_dir_time)
    checkpoint = ModelCheckpoint(
        os.path.join(log_dir, "checkpoint.h5"),
        monitor="val_loss",
        save_weights_only=True,
        save_best_only=True,
        period=5,
    )
    reduce_lr = ReduceLROnPlateau(monitor="val_loss", factor=0.1, patience=3, verbose=1)
    early_stopping = EarlyStopping(
        monitor="val_loss", min_delta=0, patience=10, verbose=1
    )

    val_split = FLAGS.val_split
    with open(FLAGS.annotation_file) as f:
        lines = f.readlines()

    # This step makes sure that the path names correspond to the local machine
    # This is important if annotation and training are done on different machines (e.g. training on AWS)
    lines = ChangeToOtherMachine(lines, remote_machine="")
    np.random.shuffle(lines)
    num_val = int(len(lines) * val_split)
    num_train = len(lines) - num_val

    # Train with frozen layers first, to get a stable loss.
    # Adjust num epochs to your dataset. This step is enough to obtain a decent model.
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
    # Adjust num epochs to your dataset. This step is enough to obtain a decent model.
    if True:
        model.compile(
            optimizer=Adam(lr=1e-3),
            loss={
                # use custom yolo_loss Lambda layer.
                "yolo_loss": lambda y_true, y_pred: y_pred
            },
        )

        batch_size = 32
        print(
            "Train on {} samples, val on {} samples, with batch size {}.".format(
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
                lines[:num_train], batch_size, input_shape, anchors, num_classes
            ),
            steps_per_epoch=max(1, num_train // batch_size),
            validation_data=data_generator_wrapper(
                lines[num_train:], batch_size, input_shape, anchors, num_classes
            ),
            validation_steps=max(1, num_val // batch_size),
            epochs=epoch1,
            initial_epoch=0,
            callbacks=[logging, checkpoint],
        )
        model.save_weights(os.path.join(log_dir, "trained_weights_stage_1.h5"))

        step1_train_loss = history.history["loss"]

        file = open(os.path.join(log_dir_time, "step1_loss.npy"), "w")
```

File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

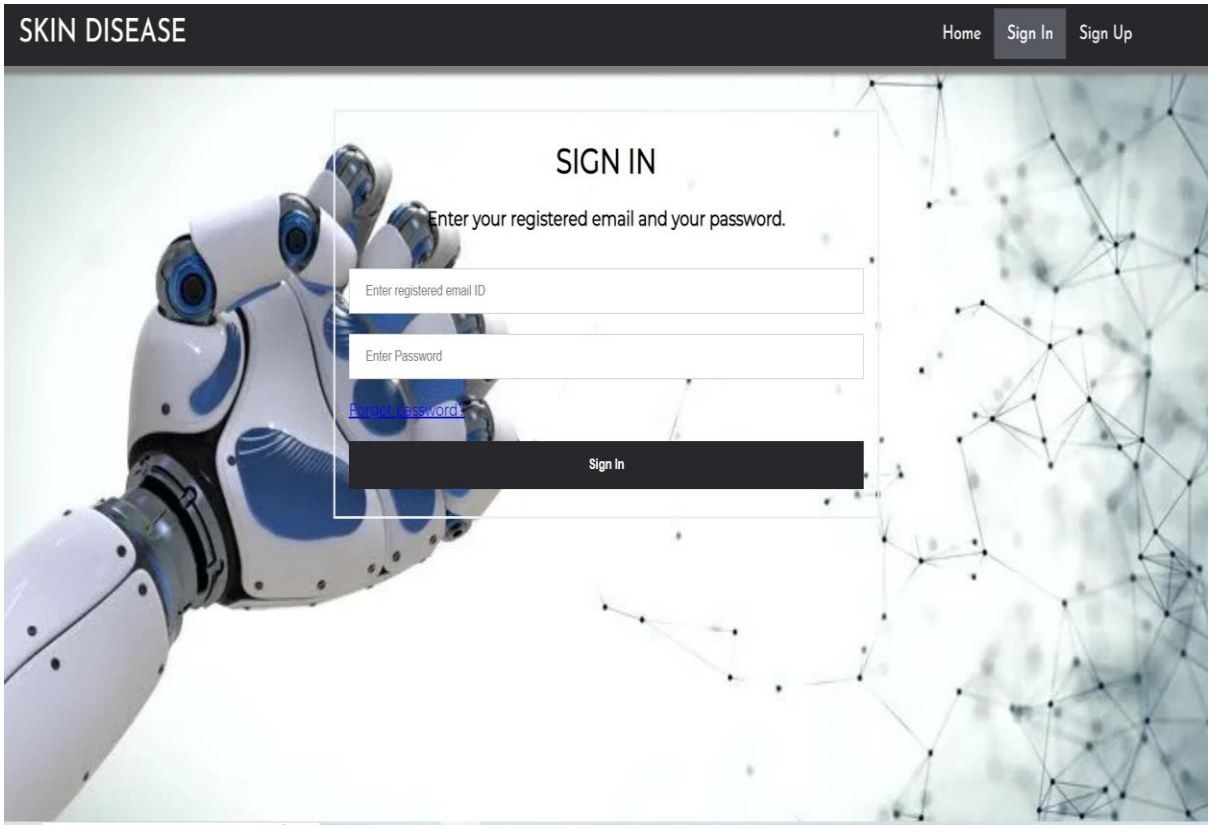
```
        with open(os.path.join(log_dir_time, "step1_loss.npy"), "w") as f:
            for item in step1_train_loss:
                f.write("%s\n" % item)
        file.close()

        step1_val_loss = np.array(history.history["val_loss"])

        file = open(os.path.join(log_dir_time, "step1_val_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step1_val_loss.npy"), "w") as f:
            for item in step1_val_loss:
                f.write("%s\n" % item)
        file.close()

    # Unfreeze and continue training, to fine-tune.
    # Train longer if the result is unsatisfactory.
    if True:
        for i in range(len(model.layers)):
            model.layers[i].trainable = True
        model.compile(
            optimizer=Adam(lr=1e-4), loss={"yolo_loss": lambda y_true, y_pred: y_pred}
        )  # recompile to apply the change
        print("Unfreeze all layers.")

        batch_size = (
            4  # note that more GPU memory is required after unfreezing the body
        )
        print(
            "Train on {} samples, val on {} samples, with batch size {}.".format(
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
```

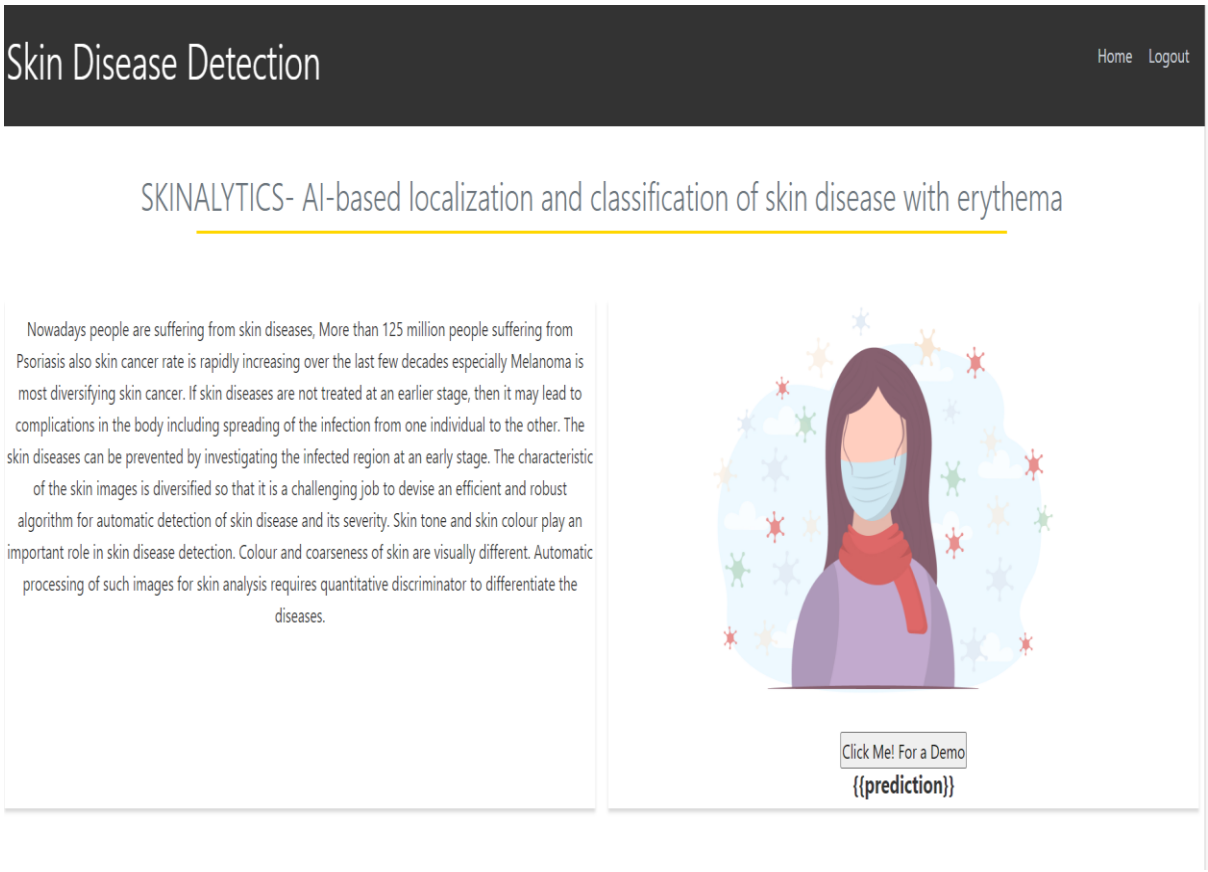
File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training

Train_YOLO.py

```
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
                lines[:num_train], batch_size, input_shape, anchors, num_classes
            ),
            steps_per_epoch=max(1, num_train // batch_size),
            validation_data=data_generator_wrapper(
                lines[num_train:], batch_size, input_shape, anchors, num_classes
            ),
            validation_steps=max(1, num_val // batch_size),
            epochs=epoch1 + epoch2,
            initial_epoch=epoch1,
            callbacks=[logging, checkpoint, reduce_lr, early_stopping],
        )
        model.save_weights(os.path.join(log_dir, "trained_weights_final.h5"))
        step2_train_loss = history.history["loss"]

        file = open(os.path.join(log_dir_time, "step2_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step2_loss.npy"), "w") as f:
            for item in step2_train_loss:
                f.write("%s\n" % item)
        file.close()

        step2_val_loss = np.array(history.history["val_loss"])

        file = open(os.path.join(log_dir_time, "step2_val_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step2_val_loss.npy"), "w") as f:
            for item in step2_val_loss:
                f.write("%s\n" % item)
        file.close()
```

OUTPUT RESULTS:





DATASETS USED FOR YOLO MODEL TRAINING: