

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 x
1 """
2     MODIFIED FROM keras-yolo3 PACKAGE, https://github.com/qjwweee/keras-yolo3
3     Retrain the YOLO model for your own dataset.
4 """
5
6 import ...
7
8
9
10
11 def get_parent_dir(n=1):
12     """ returns the n-th parent directory of the current
13     working directory """
14     current_path = os.path.dirname(os.path.abspath(__file__))
15     for k in range(n):
16         current_path = os.path.dirname(current_path)
17     return current_path
18
19
20 src_path = os.path.join(get_parent_dir(0), "src")
21 sys.path.append(src_path)
22
23 utils_path = os.path.join(get_parent_dir(1), "Utils")
24 sys.path.append(utils_path)
25
26 import numpy as np
27 import keras.backend as K
28 from keras.layers import Input, Lambda
29 from keras.models import Model
30 from keras.optimizers import Adam
31 from keras.callbacks import (
32     TensorBoard,
33     ModelCheckpoint,
34     ReduceLROnPlateau,
35     EarlyStopping,
36 )
```

```
File Edit View Code Window Help Train_YOLO.py - ...\yolo_structure-master\2_Training
Train_YOLO.py x
69 log_dir = Model_Folder
70 anchors_path = os.path.join(keras_path, "model_data", "yolo_anchors.txt")
71 weights_path = os.path.join(keras_path, "yolo.h5")
72
73 FLAGS = None
74
75 if __name__ == "__main__":
76     # Delete all default flags
77     parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
78     """
79     Command line options
80     """
81
82     parser.add_argument(
83         "--annotation_file",
84         type=str,
85         default=YOLO_filename,
86         help="Path to annotation file for Yolo. Default is " + YOLO_filename,
87     )
88     parser.add_argument(
89         "--classes_file",
90         type=str,
91         default=YOLO_classname,
92         help="Path to YOLO classnames. Default is " + YOLO_classname,
93     )
94
95     parser.add_argument(
96         "--log_dir",
97         type=str,
98         default=log_dir,
99         help="Folder to save training logs and trained weights to. Default is "
100         + log_dir,
101     )
102
103 )
104
105 from keras_yolo3.yolo3.model import (
106     preprocess_true_boxes,
107     yolo_body,
108     tiny_yolo_body,
109     yolo_loss,
110 )
111
112 from keras_yolo3.yolo3.utils import get_random_data
113 from PIL import Image
114 from time import time
115 import tensorflow.compat.v1 as tf
116 import pickle
117
118 from Train_Utils import (
119     get_classes,
120     get_anchors,
121     create_model,
122     create_tiny_model,
123     data_generator,
124     data_generator_wrapper,
125     ChangeToOtherMachine,
126 )
127
128
129 keras_path = os.path.join(src_path, "keras_yolo3")
130 Data_Folder = os.path.join(get_parent_dir(1), "Data")
131 Image_Folder = os.path.join(Data_Folder, "Source_Images", "Training_Images")
132 VoTT_Folder = os.path.join(Image_Folder, "vott-csv-export")
133 YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")
134
135 Model_Folder = os.path.join(Data_Folder, "Model_Weights")
136 YOLO_classname = os.path.join(Model_Folder, "data_classes.txt")
137
138 log_dir = Model_Folder
```

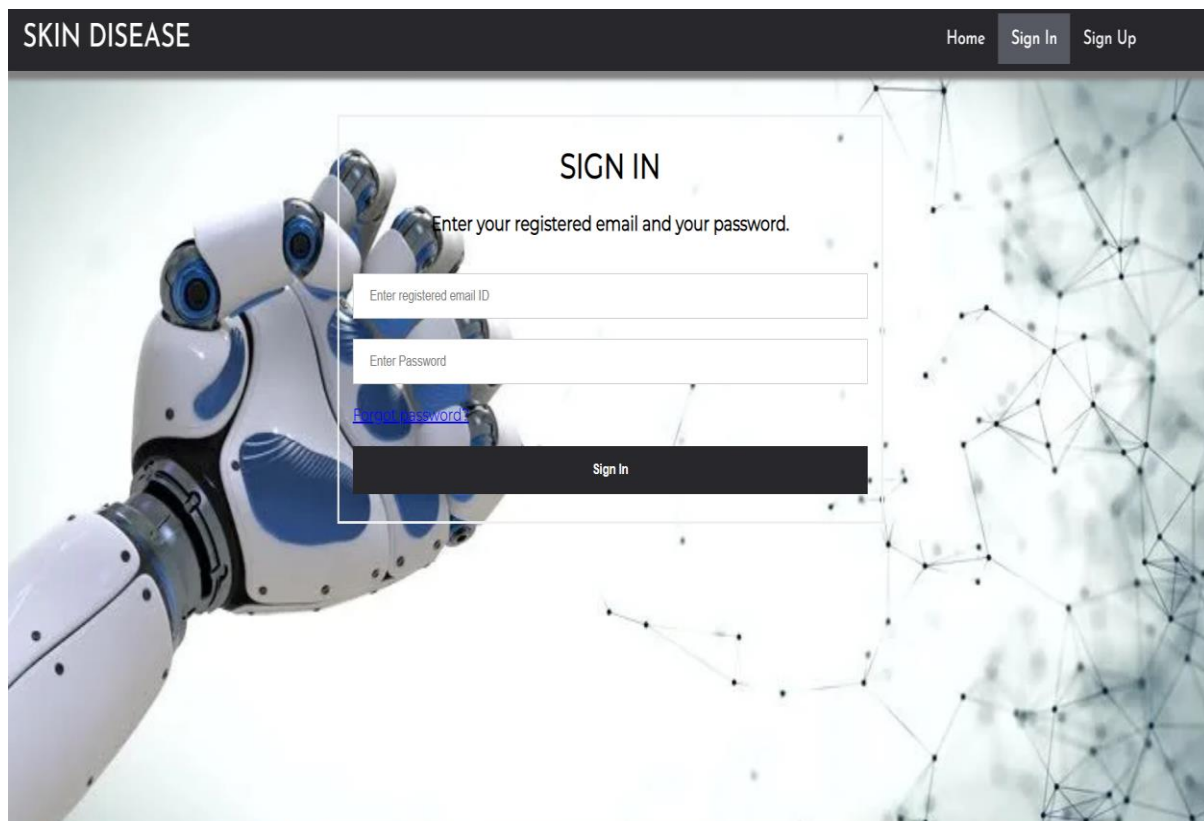
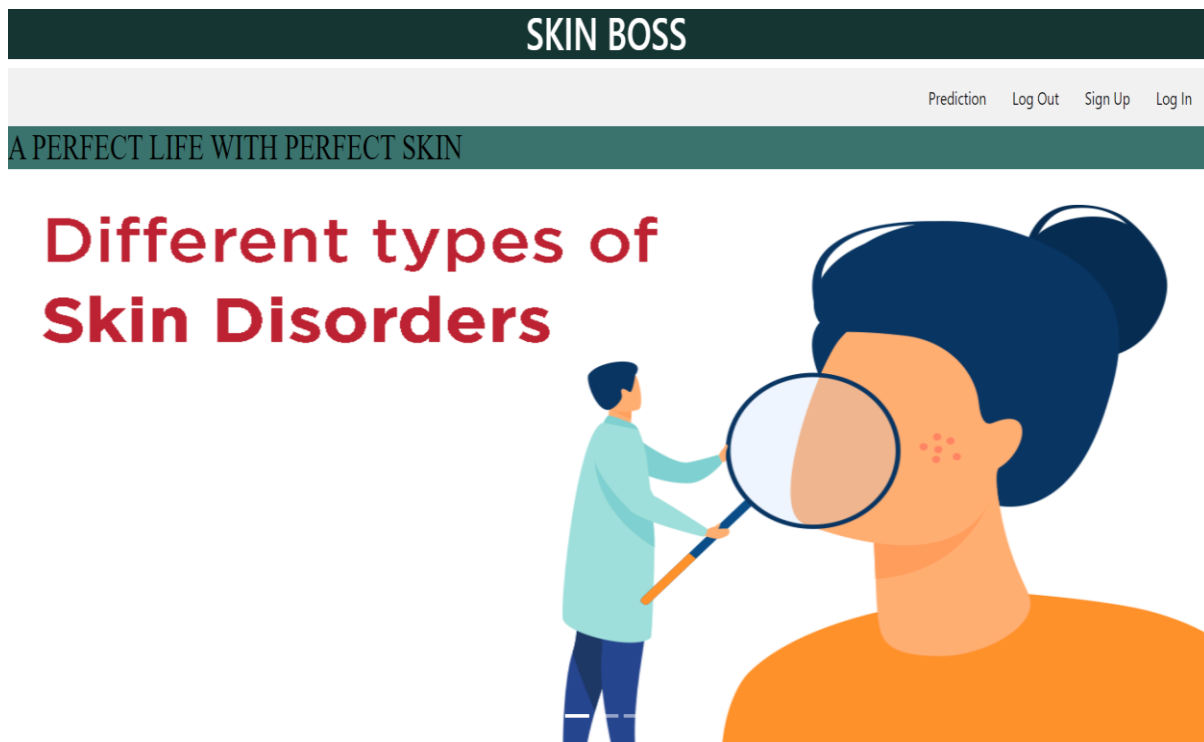
```
File Edit View Code Window Help Train_YOLO.py - ...yolo_structure-master\2_Training
Train_YOLO.py x
103     parser.add_argument(
104         "--anchors_path",
105         type=str,
106         default=anchors_path,
107         help="Path to YOLO anchors. Default is " + anchors_path,
108     )
109
110     parser.add_argument(
111         "--weights_path",
112         type=str,
113         default=weights_path,
114         help="Path to pre-trained YOLO weights. Default is " + weights_path,
115     )
116     parser.add_argument(
117         "--val_split",
118         type=float,
119         default=0.1,
120         help="Percentage of training set to be used for validation. Default is 10%. ",
121     )
122     parser.add_argument(
123         "--is_tiny",
124         default=False,
125         action="store_true",
126         help="Use the tiny Yolo version for better performance and less accuracy. Default is False.",
127     )
128     parser.add_argument(
129         "--random_seed",
130         type=float,
131         default=None,
132         help="Random seed value to make script deterministic. Default is 'None', i.e. non-deterministic.",
133     )
134     parser.add_argument(
135         "--epochs",
136         type=float,
137         default=51,
138         help="Number of epochs for training last layers and number of epochs for fine-tuning layers. Default is 51.",
139     )
140     parser.add_argument(
141         "--warnings",
142         default=False,
143         action="store_true",
144         help="Display warning messages. Default is False.",
145     )
146
147     FLAGS = parser.parse_args()
148
149     if not FLAGS.warnings:
150         tf.logging.set_verbosity(tf.logging.ERROR)
151         os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
152         warnings.filterwarnings("ignore")
153
154     np.random.seed(FLAGS.random_seed)
155
156     log_dir = FLAGS.log_dir
157
158     class_names = get_classes(FLAGS.classes_file)
159     num_classes = len(class_names)
160     anchors = get_anchors(FLAGS.anchors_path)
161     weights_path = FLAGS.weights_path
162
163     input_shape = (416, 416) # multiple of 32, height, width
164     epoch1, epoch2 = FLAGS.epochs, FLAGS.epochs
165
166     is_tiny_version = len(anchors) == 6 # default setting
167     if FLAGS.is_tiny:
168         model = create_tiny_model(
169             input_shape, anchors, num_classes, freeze_body=2, weights_path=weights_path
170         )
```

```
File Edit View Code Window Help Train_YOLO.py - ..\yolo_structure-master\2_Training
Train_YOLO.py x
171 else:
172     model = create_model(
173         input_shape, anchors, num_classes, freeze_body=2, weights_path=weights_path
174     ) # make sure you know what you freeze
175
176 log_dir_time = os.path.join(log_dir, "{}.format(int(time()))))
177 logging = TensorBoard(log_dir=log_dir_time)
178 checkpoint = ModelCheckpoint(
179     os.path.join(log_dir, "checkpoint.h5"),
180     monitor="val_loss",
181     save_weights_only=True,
182     save_best_only=True,
183     period=5,
184 )
185 reduce_lr = ReduceLROnPlateau(monitor="val_loss", factor=0.1, patience=3, verbose=1)
186 early_stopping = EarlyStopping(
187     monitor="val_loss", min_delta=0, patience=10, verbose=1
188 )
189
190 val_split = FLAGS.val_split
191 with open(FLAGS.annotation_file) as f:
192     lines = f.readlines()
193
194 # This step makes sure that the path names correspond to the local machine
195 # This is important if annotation and training are done on different machines (e.g. training on AWS)
196 lines = ChangeToOtherMachine(lines, remote_machine="")
197 np.random.shuffle(lines)
198 num_val = int(len(lines) * val_split)
199 num_train = len(lines) - num_val
200
201 # Train with frozen layers first, to get a stable loss.
202 # 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 x
202 # Adjust num epochs to your dataset. This step is enough to obtain a decent model.
203 if True:
204     model.compile(
205         optimizer=Adam(lr=1e-3),
206         loss={
207             # use custom yolo_loss Lambda layer.
208             "yolo_loss": lambda y_true, y_pred: y_pred
209         },
210     )
211
212 batch_size = 32
213 print(
214     "Train on {} samples, val on {} samples, with batch size {}".format(
215         num_train, num_val, batch_size
216     )
217 )
218 history = model.fit_generator(
219     data_generator_wrapper(
220         lines[:num_train], batch_size, input_shape, anchors, num_classes
221     ),
222     steps_per_epoch=max(1, num_train // batch_size),
223     validation_data=data_generator_wrapper(
224         lines[num_train:], batch_size, input_shape, anchors, num_classes
225     ),
226     validation_steps=max(1, num_val // batch_size),
227     epochs=epoch1,
228     initial_epoch=0,
229     callbacks=[logging, checkpoint],
230 )
231 model.save_weights(os.path.join(log_dir, "trained_weights_stage_1.h5"))
232
233 step1_train_loss = history.history["loss"]
234
235 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 x
236 with open(os.path.join(log_dir_time, "step1_loss.npy"), "w") as f:
237     for item in step1_train_loss:
238         f.write("%s\n" % item)
239     file.close()
240
241     step1_val_loss = np.array(history.history["val_loss"])
242
243     file = open(os.path.join(log_dir_time, "step1_val_loss.npy"), "w")
244     with open(os.path.join(log_dir_time, "step1_val_loss.npy"), "w") as f:
245         for item in step1_val_loss:
246             f.write("%s\n" % item)
247         file.close()
248
249     # Unfreeze and continue training, to fine-tune.
250     # Train longer if the result is unsatisfactory.
251     if True:
252         for i in range(len(model.layers)):
253             model.layers[i].trainable = True
254         model.compile(
255             optimizer=Adam(lr=1e-4), loss={"yolo_loss": lambda y_true, y_pred: y_pred}
256         ) # recompile to apply the change
257         print("Unfreeze all layers.")
258
259         batch_size = (
260             4 # note that more GPU memory is required after unfreezing the body
261         )
262         print(
263             "Train on {} samples, val on {} samples, with batch size {}".format(
264                 num_train, num_val, batch_size
265             )
266         )
267         history = model.fit_generator(
268             data_generator_wrapper(
269                 lines[:num_train], batch_size, input_shape, anchors, num_classes
270             ),
271             steps_per_epoch=max(1, num_train // batch_size),
272             validation_data=data_generator_wrapper(
273                 lines[num_train:], batch_size, input_shape, anchors, num_classes
274             ),
275             validation_steps=max(1, num_val // batch_size),
276             epochs=epoch1 + epoch2,
277             initial_epoch=epoch1,
278             callbacks=[logging, checkpoint, reduce_lr, early_stopping],
279         )
280     model.save_weights(os.path.join(log_dir, "trained_weights_final.h5"))
281     step2_train_loss = history.history["loss"]
282
283     file = open(os.path.join(log_dir_time, "step2_loss.npy"), "w")
284     with open(os.path.join(log_dir_time, "step2_loss.npy"), "w") as f:
285         for item in step2_train_loss:
286             f.write("%s\n" % item)
287     file.close()
288
289     step2_val_loss = np.array(history.history["val_loss"])
290
291     file = open(os.path.join(log_dir_time, "step2_val_loss.npy"), "w")
292     with open(os.path.join(log_dir_time, "step2_val_loss.npy"), "w") as f:
293         for item in step2_val_loss:
294             f.write("%s\n" % item)
295     file.close()
296
```

OUTPUT RESULTS:



SKINALYTICS- AI-based localization and classification of skin disease with erythema

Nowadays people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.



Click Me! For a Demo

{{prediction}}

DATASETS USED FOR YOLO MODEL TRAINING:

