SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

Model Development Phase Template

Date	7th July 2025
Team ID	SWTID1750822736
Project Title	Fault detection using transfer learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The following section contains the initial model training code extracted from the Jupyter notebook, including data loading, preprocessing, model configuration, and training.

Initial Model Training Code:

Code Block 1

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

import os
import pathlib
import PIL
import cv2
import skimage
from IPython.display import Image, display
from matplotlib.image import imread
import matplotlib.cm as cm

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image

import random
seed = 0
random.seed(seed)
np.random.seed(seed)
tf.random.set_seed(seed)
```

Code Block 2

```
# import dataset
dataset_url = "/kaggle/input/real-life-industrial-dataset-of-casting-product/casting_512x512/casting_512x512/"
data_dir = pathlib.Path(dataset_url)
data_dir
```

Output:

```
PosixPath('/kaggle/input/real-life-industrial-dataset-of-casting-product/casting_512x512/casting_512x512')
```

Code Block 3
```
# check the number of rows / number of .jpeg files
image_count = len(list(data_dir.glob('*/*.jpeg')))
print(image_count)
```
Output:
```
1300
```
Code Block 4
```
# open the image using the PIL library => for defective_front
def_front = list(data_dir.glob('def_front/*'))
PIL.Image.open(def_front[0])
```
Output:
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=512x512>
```
Code Block 5
```
# ok_front list of products
ok_front = list(data_dir.glob('ok_front/*'))
PIL.Image.open(ok_front[0])
```
Output:
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=512x512>
```
Code Block 6
```
sample1= imread(ok_front[0])
sample1.shape
```
Output:
```
(512, 512, 3)
```
Code Block 7
```
batch_size = 64
epochs = 200
img_height = 299
img_width = 299
img_size = (img_height, img_width)
```
Code Block 8
```
train_set = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    class_names =  ['ok_front', 'def_front'],
    subset="training",
    seed=seed,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```
Output:
```
Found 1300 files belonging to 2 classes.
Using 1040 files for training.
```
Code Block 9
```
val_set = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    class_names =  ['ok_front', 'def_front'],
    subset="validation",
    seed=seed,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```
Output:
```
Found 1300 files belonging to 2 classes.
Using 260 files for validation.
```
Code Block 10
```
class_names = train_set.class_names
print(class_names)
```
Output:
```
['ok_front', 'def_front']
```

Code Block 11

```
plt.figure(figsize=(10, 10))
for images, labels in train_set.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Code Block 12

```
for images, labels in train_set:
  print(images.shape)
  print(labels.shape)
  break
```

Output:

```
(64, 299, 299, 3)
(64,)
```

Code Block 13

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_set.cache().shuffle(1300).prefetch(buffer_size=AUTOTUNE)
val_ds = val_set.cache().prefetch(buffer_size=AUTOTUNE)
```

Code Block 14

```
data_augmentation = keras.Sequential(
  [

layers.RandomFlip("horizontal_and_vertical",input_shape=(img_height,img_width,3),
seed = seed ),
    layers.RandomZoom(0.1, seed = seed ),
    layers.RandomContrast(0.3, seed = seed  )
  ]
)
```

Code Block 15

```
custom_model = Sequential([

  layers.Rescaling(1./255),
  data_augmentation,

  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),


  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(1, activation = 'sigmoid')
])
```

Code Block 16

```
custom_model.compile(optimizer='adam',
                     loss='binary_crossentropy',
                     metrics=['accuracy'])
```

Code Block 17

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('accuracy') == 1.0 and logs.get('val_accuracy') == 1.0 :
            print("\nReached 100% accuracy so cancelling training!")
            self.model.stop_training = True
```

```
terminate_callback =  myCallback()
```

Code Block 18

```
# why reduceLROnPlateau() => takes bigger steps
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5,
verbose=1,min_delta=0.01,
                              patience=5, min_lr=0.000001)
```

Code Block 19

```
history1 = custom_model.fit(train_ds,
                            validation_data=val_ds,
                            epochs=epochs,
                            callbacks= [reduce_lr, terminate_callback]
)
```

Output:

```
Epoch 1/200
17/17 [==============================] - 6s 165ms/step - loss: 0.6474 - accuracy:
0.6279 - val_loss: 0.5668 - val_accuracy: 0.6885
Epoch 2/200
17/17 [==============================] - 2s 132ms/step - loss: 0.5933 - accuracy:
0.6875 - val_loss: 0.5353 - val_accuracy: 0.7615
Epoch 3/200
17/17 [==============================] - 2s 131ms/step - loss: 0.5671 - accuracy:
0.7356 - val_loss: 0.5108 - val_accuracy: 0.7615
Epoch 4/200
17/17 [==============================] - 2s 132ms/step - loss: 0.5426 - accuracy:
0.7510 - val_loss: 0.4686 - val_accuracy: 0.8000
Epoch 5/200
17/17 [==============================] - 2s 132ms/step - loss: 0.5430 - accuracy:
0.7308 - val_loss: 0.4757 - val_accuracy: 0.7923
Epoch 6/200
17/17 [==============================] - 2s 131ms/step - loss: 0.5019 - accuracy:
0.7750 - val_loss: 0.4346 - val_accuracy: 0.8038
Epoch 7/200
17/17 [==============================] - 2s 131ms/step - loss: 0.4447 - accuracy:
0.7942 - val_loss: 0.4125 - val_accuracy: 0.8038
Epoch 8/200
17/17 [==============================] - 2s 131ms/step - loss: 0.3964 - accuracy:
0.8202 - val_loss: 0.3537 - val_accuracy: 0.8346
Epoch 9/200
17/17 [==============================] - 2s 132ms/step - loss: 0.4322 - accuracy:
0.8010 - val_loss: 0.4009 - val_accuracy: 0.8192
Epoch 10/200
17/17 [==============================] - 2s 131ms/step - loss: 0.4397 - accuracy:
0.8048 - val_loss: 0.3221 - val_accuracy: 0.8654
Epoch 11/200
17/17 [==============================] - 2s 131ms/step - loss: 0.3717 - accuracy:
0.8298 - val_loss: 0.2866 - val_accuracy: 0.8423
Epoch 12/200
17/17 [==============================] - 2s 131ms/step - loss: 0.3053 - accuracy:
0.8673 - val_loss: 0.2613 - val_accuracy: 0.8769
Epoch 13/200
17/17 [==============================] - 2s 131ms/step - loss: 0.2935 - accuracy:
0.8769 - val_loss: 0.2180 - val_accuracy: 0.9038
Epoch 14/200
17/17 [==============================] - 2s 131ms/step - loss: 0.3101 - accuracy:
0.8673 - val_loss: 0.3209 - val_accuracy: 0.8692
Epoch 15/200
17/17 [==============================] - 2s 131ms/step - loss: 0.2879 - accuracy:
0.8894 - val_loss: 0.1932 - val_accuracy: 0.9077
Epoch 16/200
17/17 [==============================] - 2s 131ms/step - loss: 0.3480 - accuracy:
```

```
0.8365 - val_loss: 0.1875 - val_accuracy: 0.9385
Epoch 17/200
17/17 [==============================] - 2s 131ms/step - loss: 0.2932 - accuracy: 0.8625 - val_loss: 0.2613 - val_accuracy: 0.9038
Epoch 18/200
17/17 [==============================] - 2s 132ms/step - loss: 0.2729 - accuracy: 0.8875 - val_loss: 0.1857 - val_accuracy: 0.8923
Epoch 19/200
17/17 [==============================] - 2s 132ms/step - loss: 0.2237 - accuracy: 0.9029 - val_loss: 0.1262 - val_accuracy: 0.9615
Epoch 20/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1625 - accuracy: 0.9404 - val_loss: 0.1176 - val_accuracy: 0.9654
Epoch 21/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1430 - accuracy: 0.9404 - val_loss: 0.1377 - val_accuracy: 0.9231
Epoch 22/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1444 - accuracy: 0.9433 - val_loss: 0.0983 - val_accuracy: 0.9577
Epoch 23/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1943 - accuracy: 0.9212 - val_loss: 0.1117 - val_accuracy: 0.9615
Epoch 24/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1300 - accuracy: 0.9500 - val_loss: 0.0821 - val_accuracy: 0.9731
Epoch 25/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1291 - accuracy: 0.9413 - val_loss: 0.1631 - val_accuracy: 0.9269
Epoch 26/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1547 - accuracy: 0.9346 - val_loss: 0.1512 - val_accuracy: 0.9269
Epoch 27/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1072 - accuracy: 0.9615 - val_loss: 0.0542 - val_accuracy: 0.9846
Epoch 28/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0973 - accuracy: 0.9663 - val_loss: 0.0430 - val_accuracy: 0.9885
Epoch 29/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0744 - accuracy: 0.9692 - val_loss: 0.0578 - val_accuracy: 0.9731
Epoch 30/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0880 - accuracy: 0.9596 - val_loss: 0.1289 - val_accuracy: 0.9462
Epoch 31/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0733 - accuracy: 0.9769 - val_loss: 0.0233 - val_accuracy: 0.9962
Epoch 32/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0833 - accuracy: 0.9692 - val_loss: 0.0342 - val_accuracy: 0.9846
Epoch 33/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0579 - accuracy: 0.9827 - val_loss: 0.0321 - val_accuracy: 0.9846
Epoch 34/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0509 - accuracy: 0.9856 - val_loss: 0.0172 - val_accuracy: 0.9962
Epoch 35/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0718 - accuracy: 0.9798 - val_loss: 0.0180 - val_accuracy: 0.9962
Epoch 36/200
17/17 [==============================] - 2s 131ms/step - loss: 0.1347 - accuracy: 0.9404 - val_loss: 0.1020 - val_accuracy: 0.9423
```

```
Epoch 00036: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 37/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0723 - accuracy: 0.9731 - val_loss: 0.0434 - val_accuracy: 0.9846
Epoch 38/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0464 - accuracy: 0.9885 - val_loss: 0.0301 - val_accuracy: 0.9885
Epoch 39/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0352 - accuracy: 0.9875 - val_loss: 0.0156 - val_accuracy: 0.9962
Epoch 40/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0311 - accuracy: 0.9913 - val_loss: 0.0064 - val_accuracy: 1.0000
Epoch 41/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0381 - accuracy: 0.9875 - val_loss: 0.0132 - val_accuracy: 0.9962
Epoch 42/200
17/17 [==============================] - 2s 137ms/step - loss: 0.0323 - accuracy: 0.9865 - val_loss: 0.0074 - val_accuracy: 0.9962
Epoch 43/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0247 - accuracy: 0.9923 - val_loss: 0.0038 - val_accuracy: 1.0000
Epoch 44/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0313 - accuracy: 0.9904 - val_loss: 0.0228 - val_accuracy: 0.9923
Epoch 45/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0166 - accuracy: 0.9942 - val_loss: 0.0034 - val_accuracy: 1.0000

Epoch 00045: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 46/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0101 - accuracy: 0.9971 - val_loss: 0.0023 - val_accuracy: 1.0000
Epoch 47/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0139 - accuracy: 0.9971 - val_loss: 0.0027 - val_accuracy: 1.0000
Epoch 48/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0095 - accuracy: 0.9981 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 49/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0095 - accuracy: 0.9971 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 50/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0172 - accuracy: 0.9942 - val_loss: 0.0015 - val_accuracy: 1.0000

Epoch 00050: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 51/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0085 - accuracy: 0.9990 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 52/200
17/17 [==============================] - 2s 131ms/step - loss: 0.0080 - accuracy: 0.9971 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 53/200
17/17 [==============================] - 2s 137ms/step - loss: 0.0088 - accuracy: 0.9981 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 54/200
17/17 [==============================] - 2s 132ms/step - loss: 0.0055 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000

Reached 100% accuracy so cancelling training!
```

Code Block 20

```
custom_model.summary()
```

Output:

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
rescaling_4 (Rescaling)     (None, 299, 299, 3)       0
_________________________________________________________________
sequential_4 (Sequential)   (None, 299, 299, 3)       0
_________________________________________________________________
conv2d_16 (Conv2D)          (None, 299, 299, 64)      1792
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 149, 149, 64)     0
_________________________________________________________________
conv2d_17 (Conv2D)          (None, 149, 149, 64)      36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 74, 74, 64)       0
_________________________________________________________________
conv2d_18 (Conv2D)          (None, 74, 74, 64)        36928
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 37, 37, 64)       0
_________________________________________________________________
conv2d_19 (Conv2D)          (None, 37, 37, 32)        18464
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 18, 18, 32)       0
_________________________________________________________________
flatten_4 (Flatten)         (None, 10368)             0
_________________________________________________________________
dense_8 (Dense)             (None, 128)               1327232
_________________________________________________________________
dense_9 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,421,473
Trainable params: 1,421,473
Non-trainable params: 0
_________________________________________________________________
```

Code Block 21

```
acc = history1.history['accuracy']
val_acc = history1.history['val_accuracy']

loss = history1.history['loss']
val_loss = history1.history['val_loss']

epochs_range = range(len(acc))

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Code Block 22

```
base_model = keras.applications.Xception(
    weights='imagenet',  # Load weights pre-trained on ImageNet.
    input_shape=(img_height, img_width, 3),
    include_top=False)
```

Code Block 23

```
base_model.trainable = False

# Create new model on top
inputs = keras.Input(shape=(img_height, img_width, 3))

x = data_augmentation(inputs)  # Apply random data augmentation

x = keras.layers.Rescaling(scale=1 / 255.0)(x)

x = base_model(x, training=False)

x = keras.layers.Flatten()(x)
x = keras.layers.Dense(128, activation = 'relu')(x)
outputs = keras.layers.Dense(1, activation = 'sigmoid')(x)
pretrained_model = keras.Model(inputs, outputs)
```

Code Block 24

```
pretrained_model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Code Block 25

```
history2 = pretrained_model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    callbacks= [reduce_lr, terminate_callback]
)
```

Output:

```
Epoch 1/200
17/17 [==============================] - 9s 364ms/step - loss: 6.5443 - accuracy:
0.7125 - val_loss: 1.6489 - val_accuracy: 0.7769
Epoch 2/200
17/17 [==============================] - 5s 324ms/step - loss: 0.7937 - accuracy:
0.8740 - val_loss: 0.6319 - val_accuracy: 0.8577
Epoch 3/200
17/17 [==============================] - 5s 325ms/step - loss: 0.2846 - accuracy:
0.9212 - val_loss: 0.0494 - val_accuracy: 0.9808
Epoch 4/200
17/17 [==============================] - 5s 324ms/step - loss: 0.1062 - accuracy:
0.9683 - val_loss: 0.0866 - val_accuracy: 0.9692
Epoch 5/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0556 - accuracy:
0.9808 - val_loss: 0.0141 - val_accuracy: 0.9962
Epoch 6/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0258 - accuracy:
0.9933 - val_loss: 0.0146 - val_accuracy: 0.9962
Epoch 7/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0102 - accuracy:
0.9981 - val_loss: 0.0062 - val_accuracy: 1.0000
Epoch 8/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0094 - accuracy:
0.9990 - val_loss: 0.0037 - val_accuracy: 1.0000
Epoch 9/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0100 - accuracy:
0.9981 - val_loss: 0.0058 - val_accuracy: 1.0000
Epoch 10/200
```

```
17/17 [==============================] - 5s 324ms/step - loss: 0.0049 - accuracy:
0.9990 - val_loss: 0.0092 - val_accuracy: 0.9962
Epoch 11/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0051 - accuracy:
1.0000 - val_loss: 0.0054 - val_accuracy: 0.9962
Epoch 12/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0059 - accuracy:
0.9990 - val_loss: 0.0081 - val_accuracy: 0.9962
Epoch 13/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0054 - accuracy:
0.9981 - val_loss: 0.0259 - val_accuracy: 0.9846

Epoch 00013: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 14/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0062 - accuracy:
0.9990 - val_loss: 0.0037 - val_accuracy: 1.0000
Epoch 15/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0065 - accuracy:
0.9981 - val_loss: 0.0107 - val_accuracy: 0.9962
Epoch 16/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0042 - accuracy:
0.9990 - val_loss: 0.0034 - val_accuracy: 1.0000
Epoch 17/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0045 - accuracy:
0.9990 - val_loss: 0.0044 - val_accuracy: 1.0000
Epoch 18/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0055 - accuracy:
0.9990 - val_loss: 0.0068 - val_accuracy: 0.9962

Epoch 00018: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 19/200
17/17 [==============================] - 5s 324ms/step - loss: 0.0046 - accuracy:
0.9981 - val_loss: 0.0033 - val_accuracy: 1.0000
Epoch 20/200
17/17 [==============================] - 5s 325ms/step - loss: 0.0034 - accuracy:
1.0000 - val_loss: 0.0042 - val_accuracy: 1.0000

Reached 100% accuracy so cancelling training!
```

Code Block 26

```
acc = history2.history['accuracy']
val_acc = history2.history['val_accuracy']

loss = history2.history['loss']
val_loss = history2.history['val_loss']

epochs_range = range(len(acc))

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Code Block 27

```
pretrained_model.summary()
```

Output:

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 299, 299, 3)]     0
_________________________________________________________________
sequential_4 (Sequential)    (None, 299, 299, 3)       0
_________________________________________________________________
rescaling_5 (Rescaling)      (None, 299, 299, 3)       0
_________________________________________________________________
xception (Functional)        (None, 10, 10, 2048)      20861480
_________________________________________________________________
flatten_5 (Flatten)          (None, 204800)            0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               26214528
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 129
=================================================================
Total params: 47,076,137
Trainable params: 26,214,657
Non-trainable params: 20,861,480
_________________________________________________________________
```