UAT INITIATION AND DESIGN

PROJECT TITLE	Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy
PROJECT ID	PNT2022TMID20731

Description	Total time (s)	Estimated time(s)	Accuracy	Loss
Epoch 1/40	25s	8s	10.0052	0.6667
Epoch 2/40	20s	6s	7.2062	0.5938
Epoch 3/40	18s	6s	5.6776	0.6771
Epoch 4/40	20s	6s	4.9279	0.5104
Epoch 5/40	18s	6s	3.1746	0.6562
Epoch 6/40	20s	6s	4.4157	0.6875
Epoch 7/40	19s	6s	3.2014	0.6458
Epoch 8/40	19s	6s	2.8938	0.7188
Epoch 9/40	18s	6s	4.0036	0.6562
Epoch 10/40	19s	6s	4.5460	0.5833
Epoch 11/40	19s	6s	5.5394	0.6562
Epoch 12/40	22s	8s	6.3243	0.6250
Epoch 13/40	20s	6s	3.5716	0.5521
Epoch 14/40	19s	6s	2.9709	0.7083
Epoch 15/40	18s	6s	2.9485	0.7083
Epoch 16/40	17s	5s	4.3814	0.6250
Epoch 17/40	18s	6s	4.3419	0.5000
Epoch 18/40	18s	5s	3.4202	0.6042
Epoch 19/40	20s	6s	3.9091	0.6667
Epoch 20/40	19s	6s	3.1840	0.6562
Epoch 21/40	19s	6s	0.7396	2.8528
Epoch 22/40	17s	6s	0.6771	4.2681
Epoch 23/40	17s	6s	0.7083	3.0671
Epoch 24/40	18s	6s	0.6875	3.4036
Epoch 25/40	18s	6s	0.7292	3.1001
Epoch 26/40	18s	6s	0.7917	2.8740
Epoch 27/40	18s	6s	0.6979	2.8779
Epoch 28/40	19s	6s	0.8021	3.0000
Epoch 29/40	18s	6s	0.6562	3.6661
Epoch 30/40	20s	7s	0.6875	3.6861
Epoch 31/40	19s	6s	0.6562	5.8091
Epoch 32/40	15s	4s	0.7692	1.5989
Epoch 33/40	21s	6s	0.6250	5.7354
Epoch 34/40	18s	5s	0.7500	5.2212
Epoch 35/40	20s	7s	0.7083	5.8717
Epoch 36/40	17s	5s	0.7396	2.6432
Epoch 37/40	20s	6s	0.5312	6.1494
Epoch 38/40	20s	6s	0.7812	2.3932
Epoch 39/40	20s	6s	0.7500	6.5562
Epoch 40/40	17s	5s	0.7188	5.6489

UAT REPORT SUBMISSION

```
Epoch 1/30
3/3 [==============================] - 57s 17s/step - loss: 8.6389 - accuracy: 0.4479
Epoch 2/30
3/3 [==============================] - 52s 16s/step - loss: 12.4857 - accuracy: 0.5312
Epoch 3/30
3/3 [==============================] - 48s 14s/step - loss: 9.0234 - accuracy: 0.4375
Epoch 4/30
3/3 [==============================] - 50s 14s/step - loss: 6.4987 - accuracy: 0.6042
Epoch 5/30
3/3 [==============================] - 48s 14s/step - loss: 7.9572 - accuracy: 0.5729
Epoch 6/30
3/3 [==============================] - 49s 15s/step - loss: 5.8569 - accuracy: 0.6667
Epoch 7/30
3/3 [==============================] - 47s 14s/step - loss: 4.5726 - accuracy: 0.6458
Epoch 8/30
3/3 [==============================] - 49s 15s/step - loss: 4.0588 - accuracy: 0.6250
Epoch 9/30
3/3 [==============================] - 40s 16s/step - loss: 3.4954 - accuracy: 0.6667
Epoch 10/30
3/3 [==============================] - 47s 14s/step - loss: 3.4865 - accuracy: 0.6667
Epoch 11/30
3/3 [==============================] - 48s 14s/step - loss: 4.1423 - accuracy: 0.6458
Epoch 12/30
3/3 [==============================] - 47s 14s/step - loss: 2.9211 - accuracy: 0.6771
Epoch 13/30
3/3 [==============================] - 48s 14s/step - loss: 5.5823 - accuracy: 0.5729
Epoch 14/30
3/3 [==============================] - 47s 14s/step - loss: 3.2374 - accuracy: 0.6979
Epoch 15/30
3/3 [==============================] - 49s 14s/step - loss: 3.3727 - accuracy: 0.6146
Epoch 16/30
```

Highest accuracy:

```
Epoch 25/30
3/3 [==============================] - 48s 14s/step - loss: 2.2073 - accuracy:
0.8021
```

UTILIZATION OF TESTING TOOLS

Tested in Jupyter notebook

Py Test:

Possibly the most widely-used test framework, Py Test is another great option. It's easy to use, well-documented, and open-source, and comes with many useful features that have had giants like Dropbox changing from Py Unit to Py Test

In [ ]:
```
train_datagen = ImageDataGenerator (rescale=1./255,
                                    shear_range= 0.2,
                                    zoom_range = 0.2,
                                    horizontal_flip = True)
test_datagen = ImageDataGenerator (rescale = 1./255)
```

In [ ]:
```
training_set = train_datagen.flow_from_directory(trainPath,
                                                 target_size = (299,299),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
test_set = test_datagen.flow_from_directory(testPath,
                                                 target_size =(299,299),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

UNIT TESTING:

Although TF2.0 and Keras provide a set of well-implemented algorithms. In many cases, we still need to make our hands dirty and implement our own models and layers. Implementations involve a set of mathematical transformations. This makes correctness testing necessary. Unit testing is suitable in this when we want to test the correctness of our own implementation.

In this section, I will introduce how to conduct unit testing in TF 2.0 with examples. I will cover how generally unit testing works in TF 2.0. Then, I will implement a custom layer (Multi-head Attention Layer), and test its correctness by using a set of unit testing.

```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf
import numpy as np


class DenseLayerTest(tf.test.TestCase):

    def setUp(self):
        super(DenseLayerTest, self).setUp()
        self.my_dense = tf.keras.layers.Dense(2)
        self.my_dense.build((2, 2))

    def testDenseLayerOutput(self):
        self.my_dense.set_weights([
            np.array([[1, 0],
                      [2, 3]]),
            np.array([0.5, 0])
        ])
        input_x = np.array([[1, 2],
                            [2, 3]])
        output = self.my_dense(input_x)
        expected_output = np.array([[5.5, 6.],
                                    [8.5, 9]])

        self.assertAllEqual(expected_output, output)


tf.test.main()
```