Test Plan and Results

Overall Test Plan

We are testing with two primary aims. Firstly, we want to test individual modules to ensure they are functioning correctly. This will include testing image I/O modules to ensure that images are read and processed correctly. Testing the training pipeline to ensure sanity checks and everything is working appropriately. We will test both normal and abnormal cases to ensure that the pipelines can handle errors without crashing.

In the second phase of testing, we will test the performance of the modules. This include performance of scripts to download images, scripts to load images from disk and train the neural network.

Test Case Descriptions

T1.1	**Image Download Script Test 1**
T1.2	To test the script's ability to create file directories
T1.3	This test will ensure that given ImageNet API links in a text file and root file directory name, the script can create the appropriate file directories to save the images in
T1.4	Inputs: The inputs for the text will be ImageNet API links to download ImageNet images from
T1.5	Outputs: Correct file directory structure to save ImageNet images
T1.6	Normal
T1.7	Whitebox
T1.8	Functional
T1.9	Unit Test
T1.10	Results: File directories were created correctly

T2.1	**Image Download Script Test 2**
T2.2	To test the script's ability to download images in correct file format
T2.3	This test will ensure that given ImageNet API links in a text file and root file directory name, the script can correctly save images in correct file format
T2.4	Inputs: The inputs for the text will be ImageNet API links to download ImageNet images from
T2.5	Outputs: Images stored in correct file format
T2.6	Normal
T2.7	Whitebox
T2.8	Functional
T2.9	Unit Test
T2.10	Results: Images were saved in correct file format

T3.1	**Image Download Script Test 3**
T3.2	To test the script's ability to parallelly download images
T3.3	This test will ensure that the script can download images in parallel using multi-threading. Each thread is allocated roughly the same amount of CPU time and can run concurrently.
T3.4	Inputs: The inputs for the text will be ImageNet API links to download ImageNet images from
T3.5	Outputs: Images stored in correct file format
T3.6	Normal
T3.7	Whitebox
T3.8	Performance
T3.9	Unit Test
T3.10	Results: Each thread gets roughly the same amount of CPU time. Images were saved in correct file format
T4.1	**TFRecord Converter Script Test 1**
T4.2	To test the if the script can correctly store images as TFRecord files
T4.3	This test will ensure that the script can correctly store images in TFRecord format along with the information about it's class label, filename, image size.
T4.4	Inputs: The root directory name where images are stored
T4.5	Outputs: TFRecord file that you can reload images from
T4.6	Normal
T4.7	Whitebox
T4.8	Functional
T4.9	Unit Test
T4.10	Results: TFRecord files are of the correct format
T5.1	**TFRecord Converter Script Test 2**
T5.2	To test the if the script can correctly handle erroneous images
T5.3	This test will ensure that the script can correctly log erroneous images and continue write other images into TFRecord file
T5.4	Inputs: The root directory name where images are stored
T5.5	Outputs: TFRecord file that you can reload images from
T5.6	Abnormal
T5.7	Whitebox
T5.8	Functional
T5.9	Unit Test
T5.10	Results: Correctly outputs filenames of erroneous images and continues to write other images into TFRecord file
T6.1	**Dataset Input Pipeline Test 1**
T6.2	To test the pipeline if it can correctly read images from TFRecord files
T6.3	This test will ensure that the module can read images and class label information from TFRecord Files and prepare a Tensorflow Dataset
T6.4	Inputs: Filepath to TFRecord File
T6.5	Outputs: Tensorflow Dataset object
T6.6	Normal
T6.7	Whitebox

T6.8	Functional
T6.9	Unit Test
T6.10	Results: Tensorflow dataset object created correctly
T7.1	**Image Preprocessing Pipeline Test 1**
T7.2	To test the pipeline if it can correctly read images from TFRecord files
T7.3	This test will ensure that the module can pre-process images before feeding them to the neural network for training
T7.4	Inputs: Tensorflow Dataset Iterator
T7.5	Outputs: Pre-processed images and class labels
T7.6	Normal
T7.7	Whitebox
T7.8	Functional
T7.9	Unit Test
T7.10	Results: Correctly returns pre-processed images and class labels
T8.1	**Neural Network Initialization**
T8.2	To test the if the neural network is correctly initialized
T8.3	This test will ensure that the neural networks are correctly initialized with respect to input size and number of classes
T8.4	Inputs: Input size, number of classes, neural network hyperparameters
T8.5	Outputs: Vector of confidence scores whose length is equal to the number of classes
T8.6	Normal
T8.7	Blackbox
T8.8	Functional
T8.9	Unit Test
T8.10	Results: Neural Network is initialized correctly and outputs the correct number of classes
T9.1	**Training Pipeline Test 1**
T9.2	To test the entire training pipeline to make sure it functions correctly on GPU
T9.3	This test tests the entire training pipeline from loading images, pre-processing images, initializing neural network, training neural network and saving final neural network model
T9.4	Inputs: The filepath to train and validation TFRecord file
T9.5	Outputs: Trained Neural Network checkpoint files, History of training and validation metrics
T9.6	Normal
T9.7	Whitebox
T9.8	Functional
T9.9	Integration
T9.10	Results: TFRecord files are of the correct format

T10.1 **Training Pipeline Test 2**

T10.2 To test the performance entire training pipeline on GPU

T10.3 This test tests the entire training pipeline and records it performance for each epoch

T10.4 Inputs: The filepath to train and validation TFRecord file

T10.5 Outputs: Trained Neural Network checkpoint files, History of training and validation metrics

T10.6 Normal

T10.7 Whitebox

T10.8 Performance

T10.9 Integration

T10.10 Results: The training pipeline runs approximately 110s per epoch, which is an acceptable value for training the model.

T11.1 **Model Prediction**

T11.2 To test if final model works correctly

T11.3 This test will ensure that the final model can be loaded and run on images

T11.4 Inputs: Filepath of final model file, filepath of image to run

T11.5 Outputs: Confidence scores

T11.6 Normal

T11.7 Blackbox

T11.8 Functional

T11.9 Unit Test

T11.10 Results: Final model works correctly

Test Case Matrix

	Normal/Abnormal	**Blackbox/Whitebox**	**Functional/Performance**	**Unit/Integration**
T1	Normal	Whitebox	Functional	Unit
T2	Normal	Whitebox	Functional	Unit
T3	Normal	Whitebox	Performance	Unit
T4	Normal	Whitebox	Functional	Unit
T5	Abnormal	Whitebox	Functional	Unit
T6	Normal	Whitebox	Functional	Unit
T7	Normal	Whitebox	Functional	Unit
T8	Normal	Blackbox	Functional	Unit
T9	Normal	Whitebox	Functional	Integration
T10	Normal	Whitebox	Performance	Integration
T11	Normal	Blackbox	Functional	Unit