National College of Ireland

Configuration Manual

Msc Research Project
Data Analytics

Rahul Manikrao Sane
Student ID: x21133964

School of Computing
National College of Ireland

Supervisor: Ass. Prof. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rahul Manikrao Sane
Student ID:	x21133964
Programme:	Data Analytics
Year:	2022
Module:	Msc Research Project
Supervisor:	Ass. Prof. Catherine Mulwa
Submission Due Date:	15/12/2022
Project Title:	**Configuration Manual**
Word Count:	1300
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rahul Manikrao Sane
x21133964
Msc Research Project

1 Introduction

This document serves as a configuration guide for the various parts of the research named - **"Identification and Detection of Skin Cancer Using Deep Learning"**. The document also describes the prototype model, implementation details, and model evaluation in addition to the requirements for hardware and software for carrying out this project.

2 System Configuration

2.1 Hardware

- **Processor:** Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
- **RAM:** 16GB
- **Operating System:** 64-bit Windows OS, x64-based processor
- **GPU:** Intel(R) UHD Graphics 620, 6GB
- **Storage:** 1 TB

2.2 Software

- **Google Collaboratory:** Google offers a free cloud-based platform called Google Collaboratory [1] for the development of machine learning. The platform offers Jupyter Notebook-like capabilities and offers an optional GPU and TPU hardware accelerator to accelerate code execution. Additionally, the GPU hardware accelerator handles huge datasets quite effectively.

[1] https://colab.research.google.com/

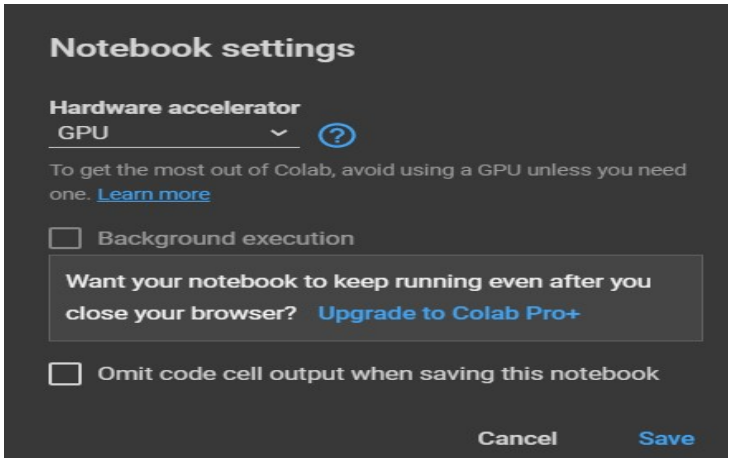

Figure 1: Google Collab: Hardware Accelerator Type

- **Microsoft Excel:** Microsoft software was used to produce the visuals and tables in the report.
- **Overleaf Latex:** Using the Overleaf platform,[2] a Latex project report and configuration manual were produced.

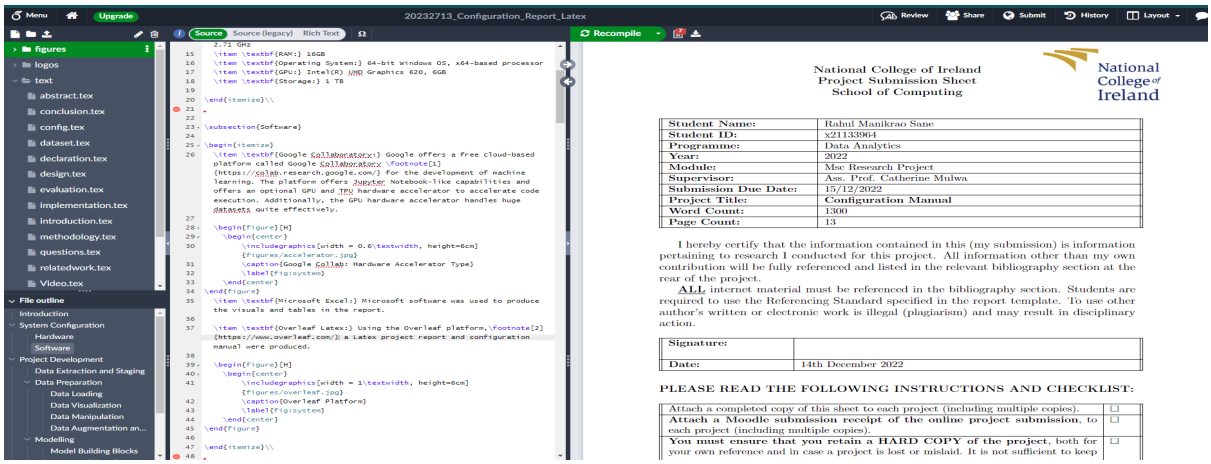

Figure 2: Overleaf Software

3 Project Development

Python coding was used to carry out the project. The course of the project can be broadly divided into three stages: data preparation, model implementation, and model evaluation.

[2]https://www.overleaf.com/

3.1 Data Extraction and Staging

The dataset is availed by international Skin Imaging Collaboration and the Society for Imaging Informatics in Medicine (SIIM-ISIC) competition hosted on Kaggle [3],publicly accessible platform.

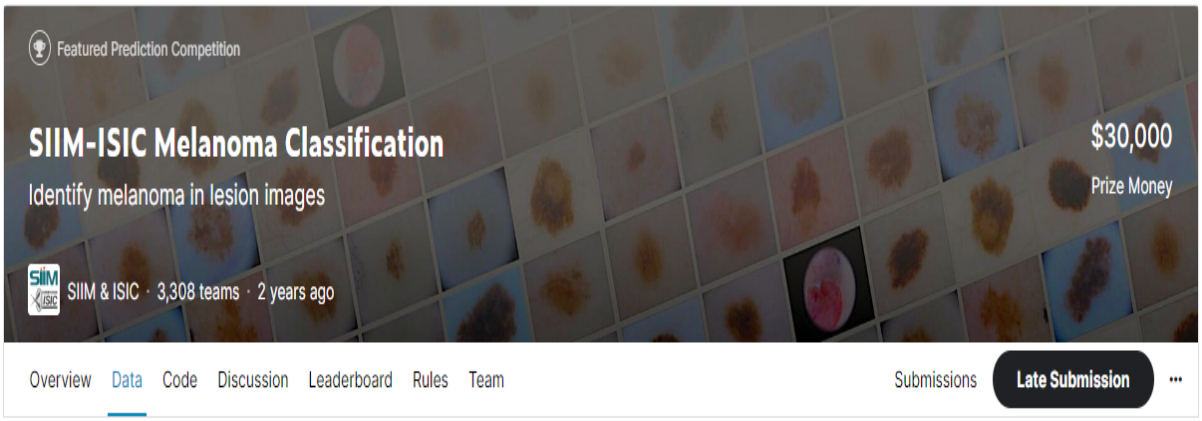

Figure 3: ISIC Dataset

The dataset was initially acquired as a zip file from Kaggle and put on a private Google Drive. You can access it by clicking the following link:
SIIM-ISIC Dataset

Below are the steps to use this dataset:

- **Step 1:** Download this dataset and upload to your Google Drive.
- **Step 2:** Now login to Google Collab and run the code file named: **21133964_Research Project_Code**
- **Step 3:** While executing the code in Figure 4 in the second code cell, the system will request authorization to access the same Google Drive. The code will automatically stage the stage to notebook and unzip it after receiving permission.

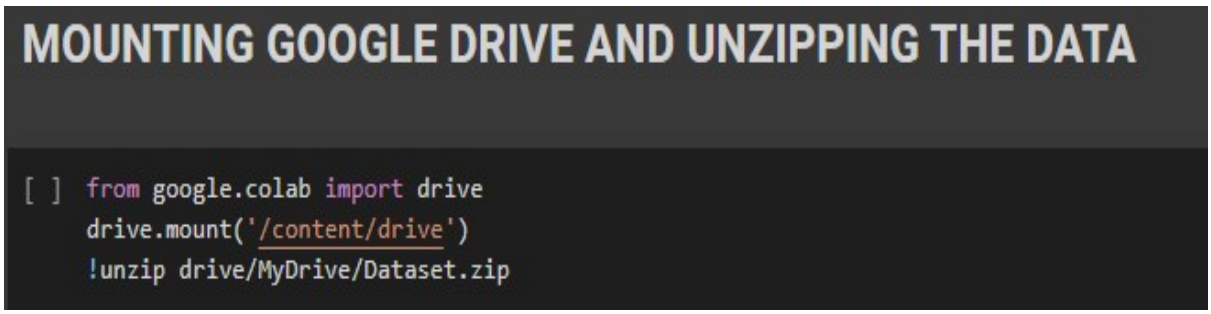

Figure 4: Google Drive Mount

[3]https://www.kaggle.com/competitions/siim-isic-melanoma-classification/data

3.2 Data Preparation

Using Numpy arrays and Pandas data frames, the skin cancer dataset was imported. The following actions were taken to pre-process the data:

3.2.1 Data Loading

The code snippets for various data extraction and modification tasks are displayed in the following figures. The several libraries that were imported as part of the development are shown in Figure 5.

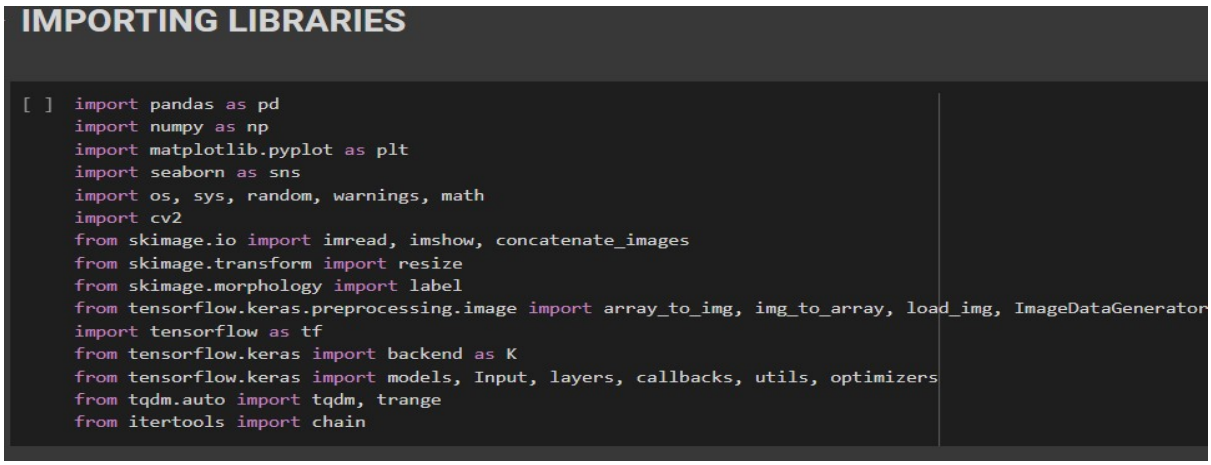

Figure 5: Importing Libraries

The data was unzipped after the libraries were imported, and the Google Drive was mounted. As shown in Figure 6, the dataset folder includes the subfolders Test Image and Train Image as well as CSV files providing depth and training data information..

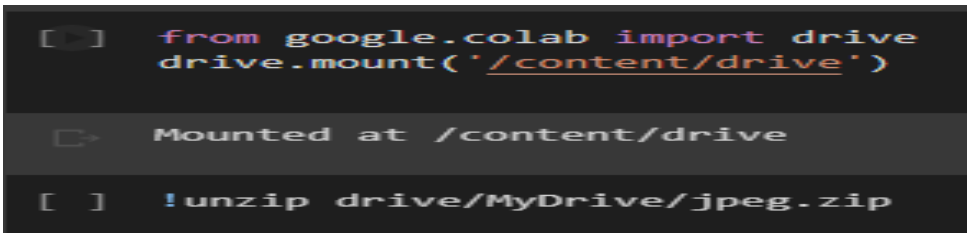

Figure 6: Dataset file Unzipped

3.2.2 Data Visualization

The code used to display the skin lesion image is shown in Figure 7.

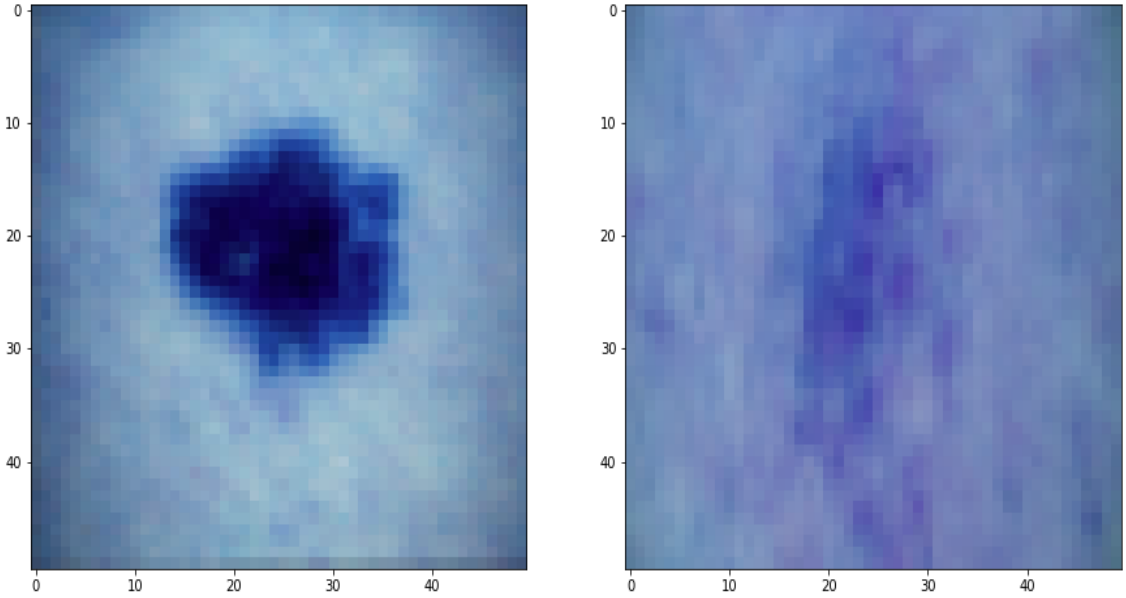

Figure 7: Skin lesion image

3.2.3 Data Augmentation and Split

The data was then augmented utilizing a variety of data augmentation techniques, as shown in the code image below.15000 images were augmented .

```
[ ]  import glob
     import cv2
     from PIL import Image
     import numpy as np
     from tqdm import tqdm
     image_array=[]
     l=[]
     i=0
     for img in tqdm(glob.glob("/content/jpeg/train/*.jpg")):
         image= cv2.imread(img)
         image_from_array = Image.fromarray(image, 'RGB')
         size_image = image_from_array.resize((50,50))
         image_array.append(np.array(size_image))
         i=i+1
         if i==15000:
             break
```

Figure 8: Data Augmentation

3.3 Modelling

3.3.1 Model Building Blocks

The ResNet deep learning model's building elements were defined as three functions. The batch normalization of the input layer is first done using the BatchActivate() function, and then the relu activation layer is utilized to activate. Convolution block() applies Conv2D() and BatchActivate operations to the input after taking the input layer, filers size, and stride size as input. As seen in Figure 9, the residual block() method first activates the input before performing two convolutions on it.

```
[ ] from keras.models import Sequential,Model
    from keras.layers import Dense,Conv2D,MaxPooling2D,Dropout,Flatten,MaxPool2D
    from keras.optimizers import RMSprop,Adam
    from keras.layers import Activation, Convolution2D, Dropout, Conv2D,AveragePooling2D, BatchNormalization,Flatten,GlobalAveragePooling2D
    from keras import layers
    from keras.regularizers import l2
    from keras.callbacks import ModelCheckpoint,ReduceLROnPlateau
    from tensorflow.keras.applications.resnet50 import ResNet50
```

Figure 9: ResNet Building Blocks

3.3.2 ResNet50 Model Implementation

The mentioned model was used to construct the ResNet50 architecture.

```
base_model = ResNet50(weights='imagenet',include_top=False, input_shape=(50,50,3))
x = base_model.output
x = Flatten()(x)
x=Dense(500, activation='relu')(x)
x=Dropout(0.2)(x)
predictions = Dense(2, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers:
    layer.trainable = False
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 4s 0us/step
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 50, 50, 3)]  0           []

 conv1_pad (ZeroPadding2D)      (None, 56, 56, 3)    0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 25, 25, 64)   9472        ['conv1_pad[0][0]']

 conv1_bn (BatchNormalization)  (None, 25, 25, 64)   256         ['conv1_conv[0][0]']

 conv1_relu (Activation)        (None, 25, 25, 64)   0           ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)      (None, 27, 27, 64)   0           ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)      (None, 13, 13, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 13, 13, 64)   4160        ['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 13, 13, 64)  256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 13, 13, 64)  0           ['conv2_block1_1_bn[0][0]']
 n)
```

Figure 10: ResNet50 Implementation

3.3.3 Model Building

The categorical crossentropy loss function and Adam optimizer were used to create the model. Accuracy and the metrics mentioned above were utilized to build the model. There were also implemented several combinations of optimizers and loss functions, including the Nadam, SGD, Adamax, and weighted cross entropy loss functions..

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 11: Model Compile

3.4 Evaluation

3.4.1 Model Training

TBefore training the model, Keras callbacks were implemented. and when the accuracy did not increase after five epochs, the learning rate dropped to 0.1. The learning rate's minimum value is set to 1e-12. The model was ran with these parameters for 20 epochs with a batch size of 63.

```
filepath="weights.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save_best_only=True, mode='min')
history=model.fit(x_train,y_train,batch_size=128,epochs=20,verbose=1,validation_split=0.33,callbacks=[checkpoint])

Epoch 1/20
63/63 [==============================] - ETA: 0s - loss: 0.2178 - accuracy: 0.9705
Epoch 1: loss improved from inf to 0.21780, saving model to weights.hdf5
63/63 [==============================] - 162s 3s/step - loss: 0.2178 - accuracy: 0.9705 - val_loss: 0.0951 - val_accuracy: 0.9813
Epoch 2/20
63/63 [==============================] - ETA: 0s - loss: 0.0912 - accuracy: 0.9821
Epoch 2: loss improved from 0.21780 to 0.09116, saving model to weights.hdf5
63/63 [==============================] - 150s 2s/step - loss: 0.0912 - accuracy: 0.9821 - val_loss: 0.0931 - val_accuracy: 0.9813
Epoch 3/20
63/63 [==============================] - ETA: 0s - loss: 0.0919 - accuracy: 0.9821
Epoch 3: loss did not improve from 0.09116
63/63 [==============================] - 149s 2s/step - loss: 0.0919 - accuracy: 0.9821 - val_loss: 0.0933 - val_accuracy: 0.9813
Epoch 4/20
63/63 [==============================] - ETA: 0s - loss: 0.0922 - accuracy: 0.9821
Epoch 4: loss did not improve from 0.09116
63/63 [==============================] - 149s 2s/step - loss: 0.0922 - accuracy: 0.9821 - val_loss: 0.0931 - val_accuracy: 0.9813
Epoch 5/20
63/63 [==============================] - ETA: 0s - loss: 0.0907 - accuracy: 0.9821
Epoch 5: loss improved from 0.09116 to 0.09069, saving model to weights.hdf5
63/63 [==============================] - 155s 2s/step - loss: 0.0907 - accuracy: 0.9821 - val_loss: 0.0931 - val_accuracy: 0.9813
Epoch 6/20
63/63 [==============================] - ETA: 0s - loss: 0.0923 - accuracy: 0.9821
Epoch 6: loss did not improve from 0.09069
63/63 [==============================] - 149s 2s/step - loss: 0.0923 - accuracy: 0.9821 - val_loss: 0.0931 - val_accuracy: 0.9813
Epoch 7/20
63/63 [==============================] - ETA: 0s - loss: 0.0930 - accuracy: 0.9821
Epoch 7: loss did not improve from 0.09069
63/63 [==============================] - 147s 2s/step - loss: 0.0930 - accuracy: 0.9821 - val_loss: 0.0982 - val_accuracy: 0.9813
Epoch 8/20
63/63 [==============================] - ETA: 0s - loss: 0.0920 - accuracy: 0.9821
Epoch 8: loss did not improve from 0.09069
63/63 [==============================] - 149s 2s/step - loss: 0.0920 - accuracy: 0.9821 - val_loss: 0.0936 - val_accuracy: 0.9813
Epoch 9/20
63/63 [==============================] - 154s 2s/step - loss: 0.0915 - accuracy: 0.9821 - val_loss: 0.0930 - val_accuracy: 0.9813
Epoch 10/20
63/63 [==============================] - ETA: 0s - loss: 0.0923 - accuracy: 0.9821
Epoch 10: loss did not improve from 0.09069
63/63 [==============================] - 149s 2s/step - loss: 0.0923 - accuracy: 0.9821 - val_loss: 0.0982 - val_accuracy: 0.9813
Epoch 11/20
63/63 [==============================] - ETA: 0s - loss: 0.0934 - accuracy: 0.9821
Epoch 11: loss did not improve from 0.09069
63/63 [==============================] - 149s 2s/step - loss: 0.0934 - accuracy: 0.9821 - val_loss: 0.0939 - val_accuracy: 0.9813
```

Figure 12: Model Training output

The ResNet50 model's accuracy and loss function charts are displayed in Figure. The model output's history() function was used to plot the graphs, and the subplots() method was used to create the graphs.

```
figure=plt.figure(figsize=(15,15))
ax=figure.add_subplot(121)
ax.plot(history.history['accuracy'])
ax.plot(history.history['val_accuracy'])
ax.legend(['Training Accuracy','Val Accuracy'])
bx=figure.add_subplot(122)
bx.plot(history.history['loss'])
bx.plot(history.history['val_loss'])
bx.legend(['Training Loss','Val Loss'])
```

Figure 13: Evaluation Plots

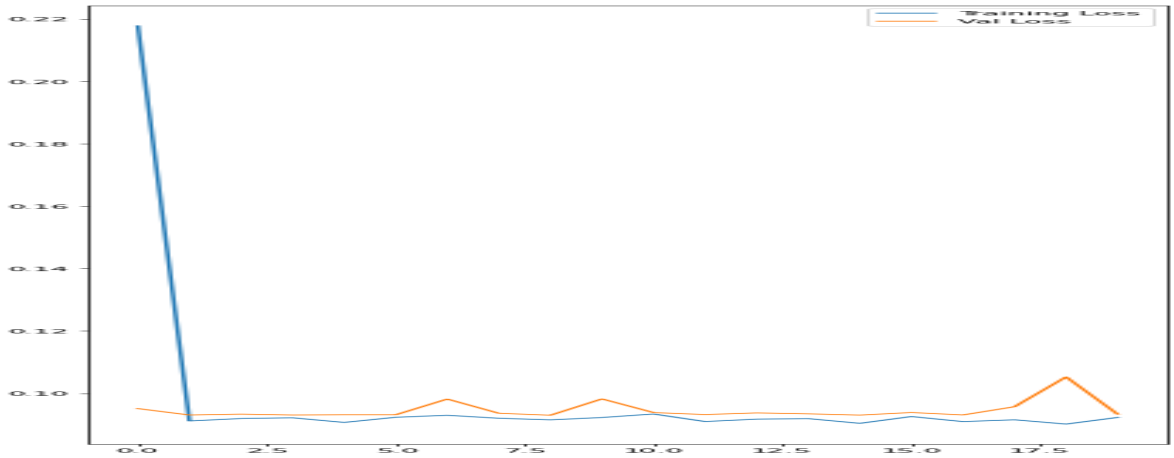

Figure 14: Loss vs Epochs

With ResNet50, the model does well at accurately identifying skin cancer.