Final Year Interim Report

Benign and Malignant skin cancer classification & detection

Author:
Aliahmed JUNEAD

Supervisor:
Dr. Nikolay NIKOLAEV

Interim report submitted in fulfilment of the requirements

for BSc Computer Science Degree

January 29, 2022

Abstract

Bsc Computer Science Degree

Benign and Malignant skin cancer classification & detection

By Aliahmed JUNEAD

Skin cancer is the most common form of cancer around the world and having over 5 sunburns doubles the risk of developing skin cancer. Skin lesions are abnormal growths on the body in comparison to the skin around it, some of which are cancerous and noncancerous. If cancerous ones are detected early and accurately, the chance of survival increases abundantly. Trained professionals and dermatologists handle the diagnosis of skin cancer. However, when misdiagnosed, it's often mistaken for a far less destructive skin condition, such as eczema, which could be a fatal mistake. With Artificial Intelligence and Deep Learning advancing at a monumental rate, it is increasingly explored within the field of medicine and diagnosis to provide an automated solution with increasingly more accuracy. This project aims to provide an “autonomous” diagnostic solution, with the help of machines. To diagnose and classify skin lesions into their respective categories, it needs enough data to learn from, and with a hefty amount of data on the internet, this won't be an issue. The selected data for this project comprises over 2000 images that belong to 9 classifications of skin lesions, 2 of which are cancerous, and the model has achieved an accuracy of 20% in a short development time frame and is likely to further improve.

Contents

Keywords

Convolutional neural networks (CNN), Deep learning (DL), neural networks, Artificial Intelligence, skin cancer, machine learning, malignant, benign.

1. Introduction

As a description of my final year project; it is a convolutional neural network (subtending from DL) that classifies different skin lesions into 9 different classifications of skin cancer. Those 9 different skin diseases are classified as either benign or malignant. The objective for this DL model is to be able to predict whether a magnified image of a particular skin lesion is either benign or malignant (in this case, the meaning of benign is that of something that is not harmful/non-cancerous, and malignant meaning that is is harmful and is cancerous) whilst also classified into the *type* it may be. Effective classification of medical images is proving to be essential in the modern era of medicine, and research on medical image classifications by CNN models have performance levels rivalling human experts (1). Hence, this project yields great importance and fulfils a need, albeit of lesser scale, but of which has great potential.

2. Background Research

2.1. Skin cancer

The skin is the largest organ of the human body, provides vital protection against injury and is a key component to regulating body temperature. With doctors in the U.S. diagnosing 3 million people each year with skin cancer each year, it is the most common type of cancer (2). Skin cancer occurs in the epidermis of the skin, which is the outermost layer, whereby the abnormal cells grow at an alarming rate (3). A tumour can be described as either *benign* or *malignant.* Benign tumours are noncancerous and do not spread whereas Malignant tumours are cancerous and grow quickly, destroying tissue as it spreads across the body (4).

2.2. Why a CNN?

DL is a branch of machine learning and by extension, CNN is a branch of DL, which is the algorithm commonly used in computer vision tasks, specifically image classifications. What makes CNN so useful is that it is an excellent feature extractor; whereby the input data is operated on by the convolutional layer, in which digital filters perform convolutional operations (5). Therefore, its powerful analysis capabilities of detecting patterns of each abstraction of images make it a vital reason I had opted for a CNN image classification model for my project (6). However, these capabilities are only facilitated by vastly large amounts of data, which can be riddled with ethically based controversy (*2.6 Controversies & ethics*), but also means that CNNs learn slowly, but this can be extremely rewarding.

2.3. Components of a CNN

CNN's are a combination of an input layer, an output layer and hidden layers. The convolutional layers neurons comprise three-dimensional output volume, number of samples & the shape of the samples (7). What makes a CNN different from a densely connected layer? As Francois Chollet had mentioned in his book: densely connected layers learn global patterns from the input given, such as patterns including *all* pixels. In contrast, convolutional layers learn local patterns of the image input given. These patterns are found in the 2D windows of the samples. To reiterate, convolutional layers are efficient in image classifications, with the capabilities of learning unique patterns on different ‘locations’ of an abstract visual problem (8).

2.3.1. Conv2d layer

As I am using Keras (library for a python interface for AI neural networks), my model will use conv2d layers, which are 2d convolutional layers. As mentioned previously, these layers look at specific portions of the image and learn patterns. I took the following diagram from Collet's book *Deep Learning with Python,* and it is a visual representation of how abstractions are formed:

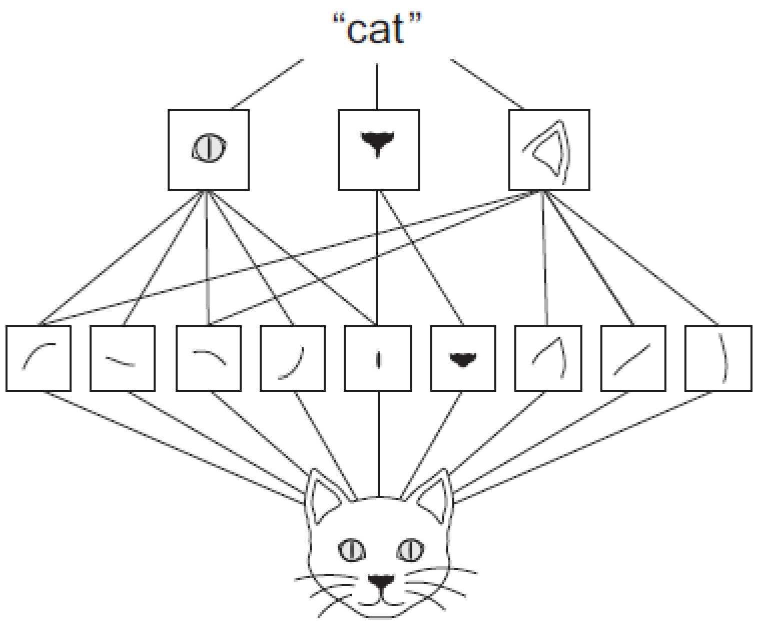


Figure 1: Shows spatial hierarchy of an image, split into various features (8)

2.3.2. Relu activation function

The conv2d & dense layers are of activation *relu* which is a type of activation function that is linear. The other activation functions are TanH and Sigmoid, however relu is cheaper computationally, with less rigorous maths (mathematically represented as: ***f(x) = max(0,x)***), making it simpler, hence resulting in reduction in time when running (9). The purpose of an activation function is to simply "transform inputs into outputs" (10).

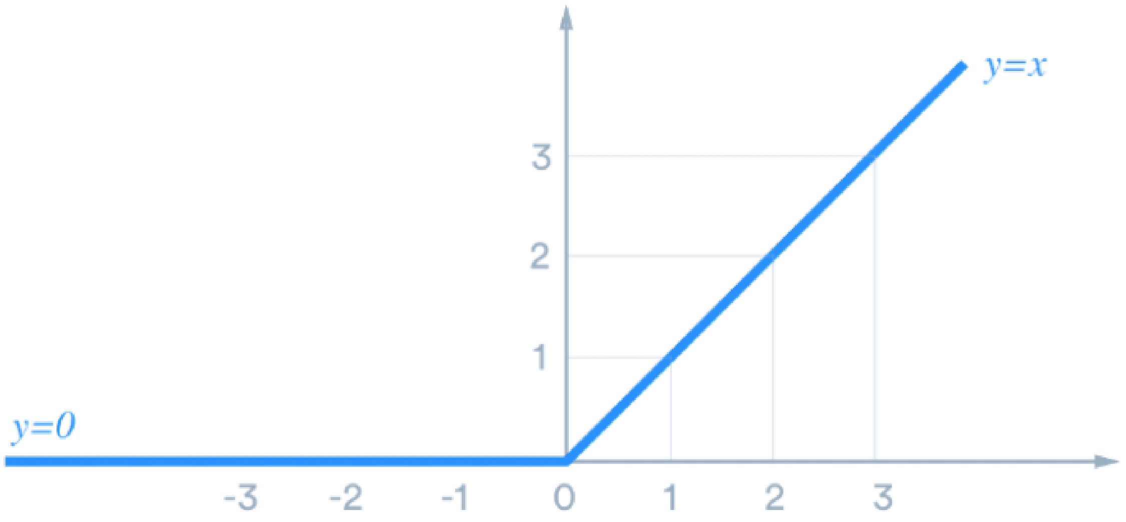


Figure 2: a linear graph of a Relu function (10)

2.3.3. Backpropagation

Backpropagation helps calculate the gradient of a loss function regarding all the weights in the network (11). Most frameworks have backpropagation already implemented, hence I don't need to code for it, but it is important to understand how backpropagation works in CNN.

$$\mathbf{dA}^{[l]} = \frac{\partial L}{\partial \mathbf{A}^{[l]}} \quad \mathbf{dZ}^{[l]} = \frac{\partial L}{\partial \mathbf{Z}^{[l]}} \quad \mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} \quad \mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}}$$

Figure 3: derivatives of backpropagation: chain rule. This is an abstraction for the process of backpropagation (12)

These derivatives are then used to update values of parameters in the form of gradient descent (12).

2.3.4. Loss function

The purpose function is to measure how far the model's estimated output values are from its actual values. For my project the loss function to be applied will be of *categorical crossentropy* which is the loss function used for multi-class classification problems such as mine.

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i$$

Figure 4: categorical crossentropy loss function calculates the loss of an example by computing the sum (13)

2.3.5. K-fold cross validation

There are 3 types of evaluation protocols that are implemented in machine learning and DL models: *k-fold cross validation, hold out validation, iterated k-fold cross validation.* Hold out validation is only operable on models that deal with large amounts of data, whereby the data is split into training, test and validation sets. K-fold cross validation is where datasets are divided into *k* numbers of sections known as *folds* (14). Each of these folds is then used as testing data at some point. For example, if k = 6, the dataset is split into 6 folds, and each fold is used as test data while the rest is used as training data.

2.4. Previous works in this field

The advancement of the field of AI has allowed for a successful smartphone-based skin disease classification model which uses MobileNet CNN. The android application classifies over 3000 images to be trained and with the use of sampling and pre-processing methods the model has achieved an accuracy of 94% (15).

Therefore, with a CNN you can achieve a very successful rate of diagnosis similar to the model mentioned previously, which exceeds even human diagnosis.
Another example can be found in 2018 when researchers had developed a CNN model with the use of 100,000 images of cancerous skin samples. This has achieved accuracy in detecting skin cancer greater than experienced dermatologists (16).
Needless to say, this goal is only reached with broad tuning & engineering of the model with consistent testing.

2.5. Challenges of multi-class classification & deep learning

In general, DL is a very data-driven field of AI taking into account the learning that is required to achieve an accurate model. Hence, the requirement of adequate and/or large amounts of data is compulsory in that regard. The more data there is, the more accurate the model becomes and the less likely it is to overfit. Therefore there is often the problem of having a shortage of data (17).

Gathering data can be extremely time consuming but may often be a limited option if a usable dataset is not available, but as mentioned previously, DL neural networks require large sums of data that also need to be labelled which can prove to be a huge setback.

Notwithstanding the aforementioned reasons, I have a usable dataset, albeit a smaller one, it is still adequate for the task at hand however, having a larger one would help to develop a far more accurate model making it far easier to mitigate overfitting.

2.6. Controversies & Ethics

An article suggests AI skin cancer detection models may have the risk of being less accurate for darker complexions (18). The main reason for this is the lack of data, as most skin cancer datasets are of overwhelming Caucasian samples only. This means that AI models will not have adequate training to classify for other complexions due to limited training on certain skin types. Therefore, this limits the usage of a sensitive classification model, as it has the danger of only being exclusively used for a certain demographic. Of course, this depends on the regulations of a domain. However, the reduced accuracy would deem it to be redundant if it is substantially inaccurate. A study has found that 14 datasets of skin cancer, out of 21 available online, had registered the various skin types/complexions used, however, 11 of the 14 were from the continents of Europe, North America and Oceania, many of those datasets lacked any images from people of colour (19). The lack of could cause wider implications for the future, questioning the ethics of deploying such models.

3. Methodology

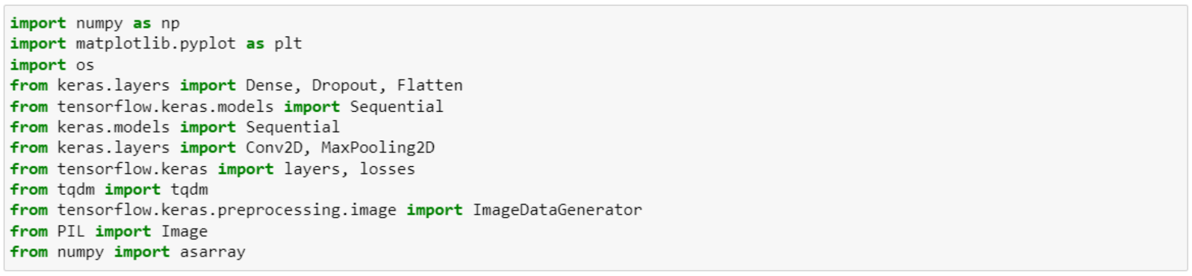
3.1. Dataset

The dataset was formed by the International Skin Imaging Collaboration (ISIC) and is distributed on Kaggle. The images are sorted & divided according to the label it belongs to and were divided into a relatively equal number of samples per classification. Extracting and producing data of my own is 1) a time-consuming task, 2) I don't have the necessary skills of taking medical-grade images that will be sufficient for my model to learn from. The particular dataset I will use has *2801* images which are labelled into 9 different skin lesions. The malignant skin lesions are *Basal cell carcinoma* and *melanoma.* The benign skin lesions are *actinic keratosis, pigmented benign keratosis, dermatofibroma, nevus, seborrheic keratosis, squamous cell carcinoma, vascular lesion.* It's important to note that many of the skin lesions may appear to be similar to the average observer and fully trained professionals alike, hence the importance & demand for AI skin detection technology arises.

3.2. Model Architecture

Before building my model I had to establish a baseline. In this case, the baseline is 1/9 or 0.11, hence to gain statistical power, my model must exceed 0.11. As mentioned previously, I will be using the Keras library which is a high-level API of Tensorflow and my model will be executed in a Jupyter notebook.

I started by importing the following dependencies:

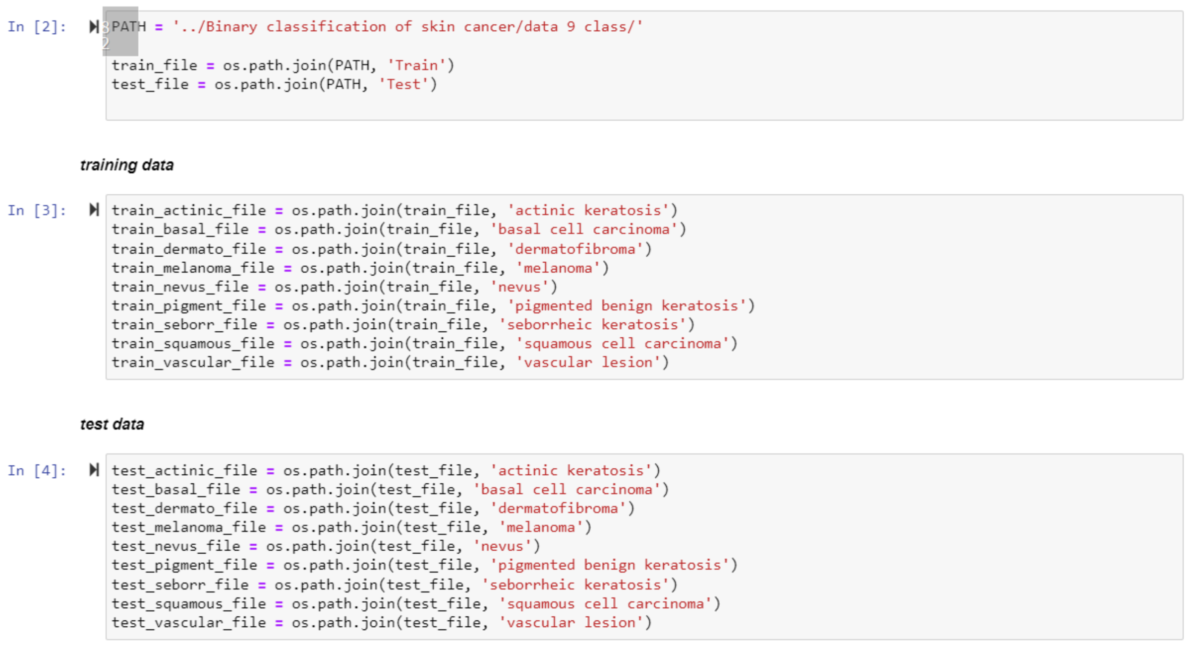


```python
import numpy as np
import matplotlib.pyplot as plt
import os
from keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.models import Sequential
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import layers, losses
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import Image
from numpy import asarray
```

Figure 5: list of dependencies

These are important for me to build a deep learning model, importing 2d convolutional layers and the hidden layers required to build a successful model. I had also imported *matplotlib* to be able to represent the results of training of both validation and training of the model.

I had then preprocessed my data, splitting the dataset into training and test data, assigning variables to each category accordingly:



```
In [2]:  PATH = '../Binary classification of skin cancer/data 9 class/'

         train_file = os.path.join(PATH, 'Train')
         test_file = os.path.join(PATH, 'Test')
```

training data

```
In [3]:  train_actinic_file = os.path.join(train_file, 'actinic keratosis')
         train_basal_file = os.path.join(train_file, 'basal cell carcinoma')
         train_dermato_file = os.path.join(train_file, 'dermatofibroma')
         train_melanoma_file = os.path.join(train_file, 'melanoma')
         train_nevus_file = os.path.join(train_file, 'nevus')
         train_pigment_file = os.path.join(train_file, 'pigmented benign keratosis')
         train_seborr_file = os.path.join(train_file, 'seborrheic keratosis')
         train_squamous_file = os.path.join(train_file, 'squamous cell carcinoma')
         train_vascular_file = os.path.join(train_file, 'vascular lesion')
```

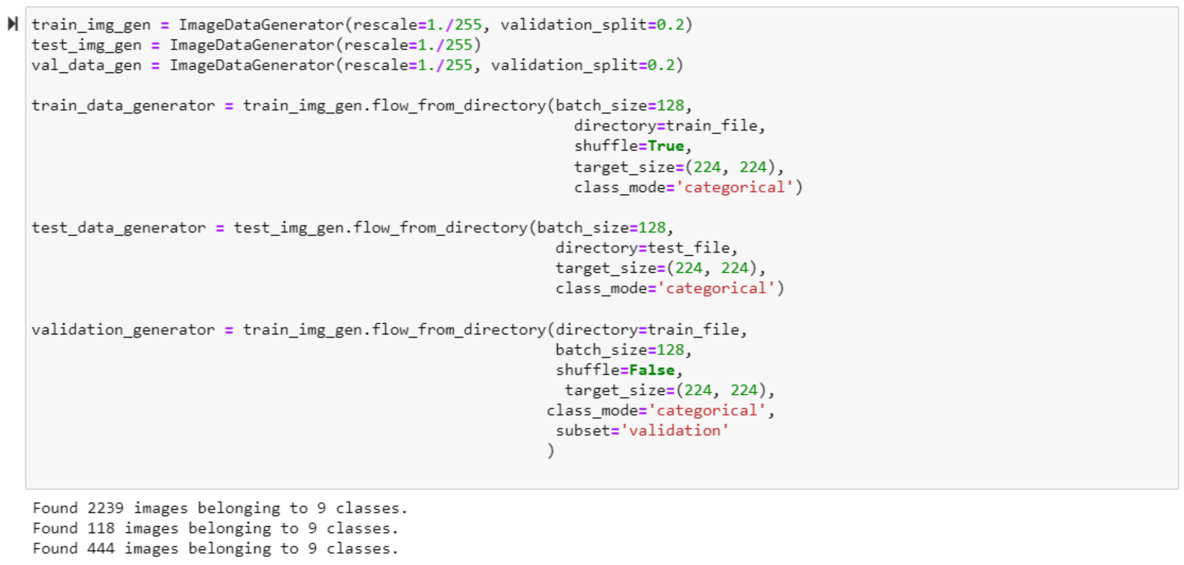
test data

```
In [4]:  test_actinic_file = os.path.join(test_file, 'actinic keratosis')
         test_basal_file = os.path.join(test_file, 'basal cell carcinoma')
         test_dermato_file = os.path.join(test_file, 'dermatofibroma')
         test_melanoma_file = os.path.join(test_file, 'melanoma')
         test_nevus_file = os.path.join(test_file, 'nevus')
         test_pigment_file = os.path.join(test_file, 'pigmented benign keratosis')
         test_seborr_file = os.path.join(test_file, 'seborrheic keratosis')
         test_squamous_file = os.path.join(test_file, 'squamous cell carcinoma')
         test_vascular_file = os.path.join(test_file, 'vascular lesion')
```

Figure 6: data preprocessing

Currently, my dataset is on the local machine, hence calling the path and the directories to divide the data appropriately, the directories are already pre-labelled. I acknowledge that this is not the best method, as it uses up storage and makes it difficult to run on different devices, however, I will be installing the dataset.

I had used ImageDataGenerator, which generates batches of tensor image data with real-time data augmentation. This forms the training, test and validation generator. The shape of the images are 224 x 224, and the training data has been split into an 8:2 ratio. This divides the training data into the training set and the validation set. Hence, In this model, is an example of hold-out validation. Since the prototype, the model did not have any form of validation, which is something that had subsequently been added after feedback. Figure 7 shows how this has been implemented:

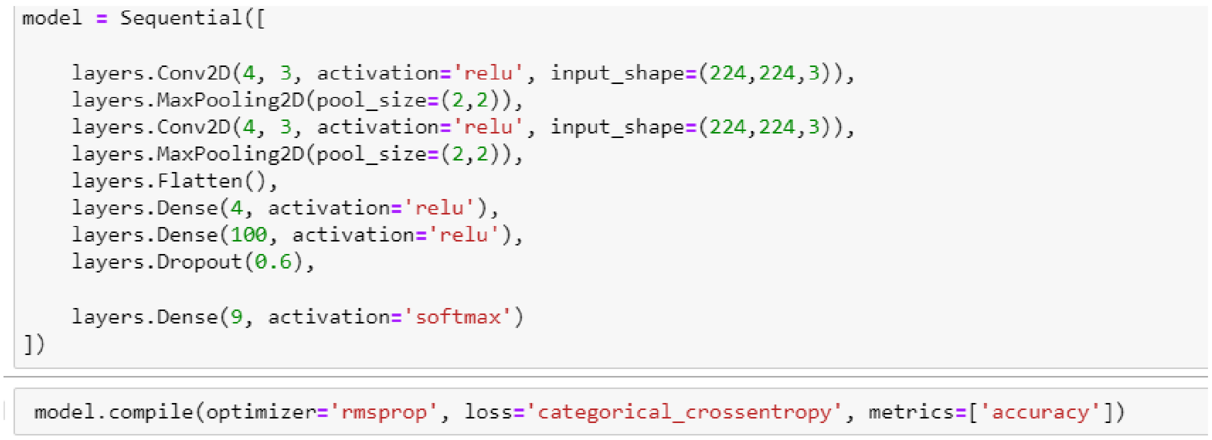


```
train_img_gen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
test_img_gen = ImageDataGenerator(rescale=1./255)
val_data_gen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_data_generator = train_img_gen.flow_from_directory(batch_size=128,
                                                          directory=train_file,
                                                          shuffle=True,
                                                          target_size=(224, 224),
                                                          class_mode='categorical')

test_data_generator = test_img_gen.flow_from_directory(batch_size=128,
                                                        directory=test_file,
                                                        target_size=(224, 224),
                                                        class_mode='categorical')

validation_generator = train_img_gen.flow_from_directory(directory=train_file,
                                                         batch_size=128,
                                                         shuffle=False,
                                                          target_size=(224, 224),
                                                        class_mode='categorical',
                                                         subset='validation'
                                                        )
```

```
Found 2239 images belonging to 9 classes.
Found 118 images belonging to 9 classes.
Found 444 images belonging to 9 classes.
```

Figure 7: data preprocessing continued, ImageDataGenerator and resizing images

In the following figure, I finally begin to build th[1]e model. I have 2 convolutional layers, and a few hidden layers, whilst also adding dropout. The final layer activation is of softmax, which is required for multi-class tasks. The optimizer is ‘rmsprop’ which is a default value of 0.001:

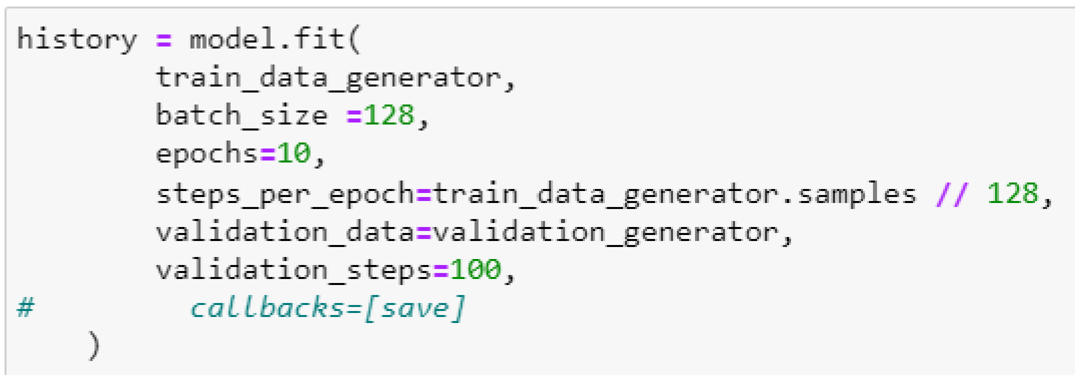


```
model = Sequential([

    layers.Conv2D(4, 3, activation='relu', input_shape=(224,224,3)),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(4, 3, activation='relu', input_shape=(224,224,3)),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dense(4, activation='relu'),
    layers.Dense(100, activation='relu'),
    layers.Dropout(0.6),

    layers.Dense(9, activation='softmax')
])
```

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

Figure 8: Model Architecture

To run the model on training data, I had used *model.fit* and the training generator. Figure 9 shows that the model will run on 10 epochs, but this will be varied throughout testing and tuning of the model:



```
history = model.fit(
        train_data_generator,
        batch_size =128,
        epochs=10,
        steps_per_epoch=train_data_generator.samples // 128,
        validation_data=validation_generator,
        validation_steps=100,
#         callbacks=[save]
    )
```

Figure 8: training the model on training data

To evaluate the model and to see how well it is doing I had used accuracy as a metric of success. The accuracy will give information on how well the model is doing to successfully classify the images into the correct categories.

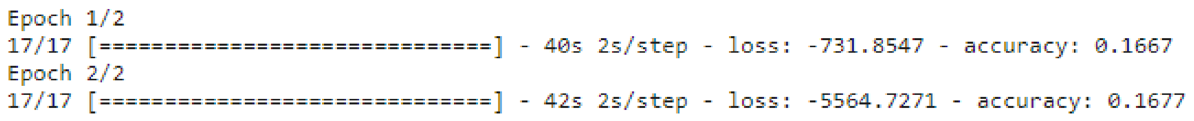
This model is not at all at an optimal level, it will require further tuning and even changes to the architecture. However, currently, the model is functioning and is outputting results.

When running experimentation and hyperparameter tuning, I will be taking guidance from the universal workflow method. Hence, after applying appropriate validation techniques, I will be forming multiple models to experiment on using different hyperparameter values. Forming a model that overfits, will allow me to develop a model that will be accurate whilst mitigating overfitting. Moreover, I understand that the model will need to be trained on much higher epochs to understand when the model overfits etc.

[1] *disclaimer: this form of validation protocol is not optimal for this dataset as the dataset is far too small. As mentioned in the background, for smaller datasets such as this, the most appropriate evaluation protocol is cross-validation such as k-fold. Please see the ‘Progress of project and plans’ section of the report for further comments.*

4. Results

I had graphically displayed the results of the previous model. However, due to validation error as mentioned previously, the final validation of the model is not visible, hence only showing the training arc of the model for loss and accuracy.

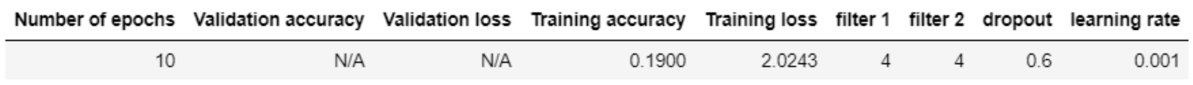


```
Epoch 1/2
17/17 [==============================] - 40s 2s/step - loss: -731.8547 - accuracy: 0.1667
Epoch 2/2
17/17 [==============================] - 42s 2s/step - loss: -5564.7271 - accuracy: 0.1677
```

Figure 9: previous training accuracy & loss

The following results are for a model which has 2 conv2d layers, each followed by max pooling layers and hidden layers. Further information can be seen in figure 9:

Figure 10: training accuracy & loss



Number of epochs	Validation accuracy	Validation loss	Training accuracy	Training loss	filter 1	filter 2	dropout	learning rate
10	N/A	N/A	0.1900	2.0243	4	4	0.6	0.001

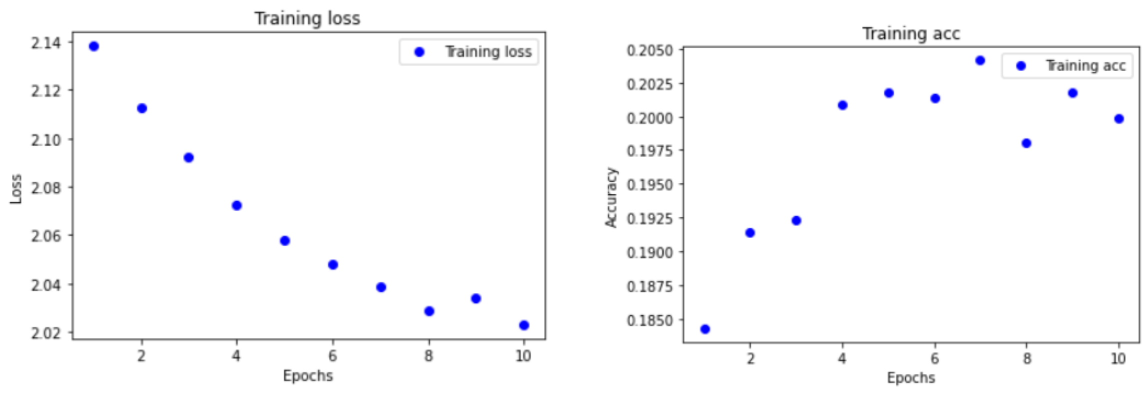
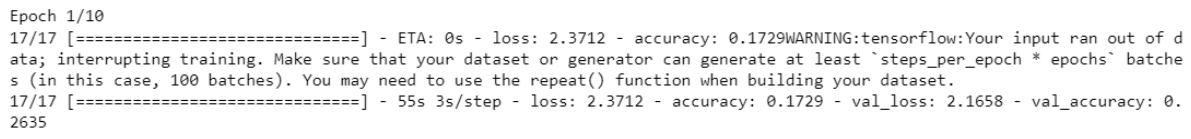


Figure 11: training accuracy and loss per epoch



```
Epoch 1/10
17/17 [=============================] - ETA: 0s - loss: 2.3712 - accuracy: 0.1729WARNING:tensorflow:Your input ran out of d
ata; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batche
s (in this case, 100 batches). You may need to use the repeat() function when building your dataset.
17/17 [=============================] - 55s 3s/step - loss: 2.3712 - accuracy: 0.1729 - val_loss: 2.1658 - val_accuracy: 0.
2635
```

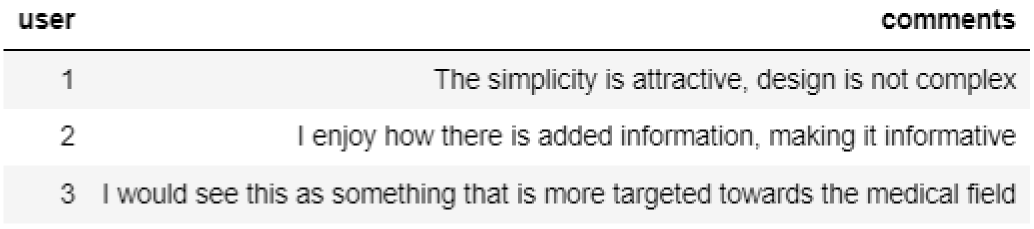
Figure 10: validation accuracy after the first epoch

We can see here that there seems to be an error after the first epoch. This is the direct reason for what had been stated in the *methodology.* Simply put, the validation being used is not the appropriate validation method. However, the first epoch had given validation accuracy and validation loss of over 200% and an accuracy of over 26%. The model is far from ready however the results can be refined with better tuning and architecture. Therefore, for more meaningful results I will need to implement k-fold cross validation or use a different dataset with a greater number of images, to ensure that I have the necessary amount of data to be able to train my model and develop a moderately high degree of accuracy.

5. Discussion

First of all these results prove to be an important milestone as they are a testament to the model being able to run, although currently with poor accuracy and loss, it still manages to execute a CNN architecture. Moreover, with these results, the true accuracy of the model can't be perceived because of the incorrect validation protocol. As I had split the training data to form data for a validation set (resulting in hold-out validation protocol), this has resulted in a deficient model. This is mainly due to my ignorance on how to treat a model with a smaller dataset, but with further investigation and analysis I will be able to fix these errors and continue to conduct experiments to mitigate overfitting and to increase the accuracy of the model. Although these erroneous results may seem like a setback of some kind, I believe that once solving them, it can prove to be a beneficial learning curb. ML developers have stated that: “data tells a story if you have enough of it” (20) which may be something for me to consider to achieve a model which is optimal.

I would like to add that the original plan I had set in place is still being followed with minimal changes to the itinerary of the project. I had gathered user feedback of my proposed web application wireframes:



user	comments
1	The simplicity is attractive, design is not complex
2	I enjoy how there is added information, making it informative
3	I would see this as something that is more targeted towards the medical field

As this is an ongoing survey and due to the pandemic I had been restricted in gathering further potential users to make comments on proposed ideas and user interaction. This was particularly difficult as I had contracted the virus which has made it difficult to continue with said itinerary however I haven't strayed off course too far that requires the planning and intended milestones to be changed and unattainable.

6. Conclusion

To conclude, in this report I had given details on how I am to implement a DL model which successfully detects and classifies a skin lesion into either a malignant or benign category. I had given details of the background of my project, in a technological context referring to a CNN and the architectural details of a CNN, whilst also identifying challenges and potential moralistic controversy that must be considered. The main purpose of this report is to display progression and comprehensive evaluation of methods and results of current work.

7. Progress of Project and Future Plans

Since the project specification and the project prototype, my model has changed quite a bit. I had not had any sort of validation protocol implemented and in this iteration, I had holdout validation. Granted, it is not a valid form of validation protocol as it has caused issues, it has proved to be a step in the right direction.

What might be left to do? As demonstrated earlier, I would need to implement k-fold cross validation for my model to show validation data. The dataset I am using is saved into the local machine which takes up excess storage, hence to remedy this, I will aim to pip install the dataset into my notebook. Once these solutions have been accomplished I can then go on to focusing on hyperparameter tuning of the model, to try to achieve a higher degree of accuracy. I also aim to modularise my code, so that I can run many tests without repeating redundant amounts of code. Once I have achieved relatively high accuracy on training data, I can then go on to test my model on unseen test data. This is achievable because I have already methodologically implemented a CNN algorithm architecture.

8. Bibliography

Link to github repository: https://github.com/7unayd/skin-cancer-classification-model

1. Yadav SS, Jadhav SM. Deep convolutional neural network based medical image classification for disease diagnosis. Journal of Big Data. 2019 Dec;6

2. Skin Cancer (Non-Melanoma) - Introduction [Internet]. Cancer.net. 2012 [cited 2022 Jan 28]. Available from: https://www.cancer.net/cancer-types/skin-cancer-non-melanoma/introduction#:~:text=A%20tumor%20can%20be%20cancerous

3. Sarnoff D. Skin Cancer Information - The Skin Cancer Foundation [Internet]. The Skin Cancer Foundation. 2018 [cited 2022 Jan 28]. Available from: https://www.skincancer.org/skin-cancer-information/

4. Cancer [Internet]. stanfordhealthcare.org. Available from: https://stanfordhealthcare.org/medical-conditions/cancer/cancer.html#:~:text=What%20is%20the%20difference%20between

5. Yadav SS, Jadhav SM. Deep convolutional neural network based medical image classification for disease diagnosis. Journal of Big Data. 2019 Dec;6

6. Huang S-C, Le T-H. Convolutional Neural Networks - an overview | ScienceDirect Topics [Internet]. www.sciencedirect.com. 2020 [cited 2022 Jan 26]. Available from: https://www.sciencedirect.com/topics/engineering/convolutional-neural-networks#:~:text=A%20CNN%20consists%20of%20neurons

7. Convolutional Neural Network (CNN) [Internet]. NVIDIA Developer. 2018 [cited 2022 Jan 26]. Available from: https://developer.nvidia.com/discover/convolutional-neural-network

8. Chollet F. Deep Learning with Python. Shelter Island (New York, Estados Unidos): Manning, Cop; 2018.

9. Adan Y. What is the ReLU function formula? When would we use this? Why? [Internet]. Quora. 2020 [cited 2022 Jan 26]. Available from: https://www.quora.com/What-is-the-ReLU-function-formula-When-would-we-use-this-Why

10. Mujtaba H. An Introduction to Rectified Linear Unit (ReLU) | What is RelU? [Internet]. GreatLearning. 2020. Available from: https://www.mygreatlearning.com/blog/relu-activation-function/

11. Johnson D. Back Propagation Neural Network: Explained With Simple Example [Internet]. www.guru99.com. 2022. Available from: https://www.guru99.com/backpropogation-neural-network.html

12. Piotr Skalski. Gentle Dive into Math Behind Convolutional Neural Networks [Internet]. Medium. Towards Data Science; 2019 [cited 2022 Jan 27]. Available from:

https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

13. Categorical crossentropy loss function | Peltarion Platform [Internet]. Peltarion.com. 2022 [cited 2022 Jan 27]. Available from: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

14. Krishni. K-Fold Cross Validation [Internet]. Medium. 2018 [cited 2022 Jan 27]. Available from: https://medium.datadriveninvestor.com/k-fold-cross-validation-6b8518070833

15. Velasco J. A Smartphone-Based Skin Disease Classification Using MobileNet CNN. International Journal of Advanced Trends in Computer Science and Engineering. 2019 Oct 15;2632–7.

16. Hofl P, Ph.D. Artificial Intelligence Better than Dermatologists in Diagnosing Skin Cancer [Internet]. Onco'Zine. 2018 [cited 2022 Jan 26]. Available from: https://www.oncozine.com/artificial-intelligence-better-dermatologists-diagnosing-skin-cancer/#:~:text=Researchers%20have%20shown%20for%20the

17. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data. 2021 Mar 31;8.

18. Davis N. AI skin cancer diagnoses risk being less accurate for dark skin – study [Internet]. the Guardian. 2021. Available from: https://www.theguardian.com/society/2021/nov/09/ai-skin-cancer-diagnoses-risk-being-less-accurate-for-dark-skin-study

19. Wen D, Khan SM, Xu AJ, Ibrahim H, Smith L, Caballero J, et al. Characteristics of publicly available skin cancer image datasets: a systematic review. The Lancet Digital Health [Internet]. 2021 Nov 9;0(0). Available from: https://www.thelancet.com/journals/landig/article/PIIS2589-7500(21)00252-1/fulltext

20. Metwalli SA. 6 Ways to Improve Your ML Model Accuracy [Internet]. Medium. 2021 [cited 2022 Jan 28]. Available from: https://towardsdatascience.com/6-ways-to-improve-your-ml-model-accuracy-ec5c9599c436