Kidney Cortex Cells Training Using CNN

Seoeun Lee

Introduction

This research is about training a dataset about human kidney cortex cell images with 8 different types of categories of cells with gray-scale images of size 28 x 28. The objective of this project is to build a training model that can give the value accuracy of 65 and higher using 200, 000 different images without overfitting (good performance on the training data, poor generalization to other data [1]) and underfitting (bad Performance on the training data and bad generalization to other data [1]) the unseen testing dataset when compared against training dataset. There are 3 types of Neural Networks that are used in training datasets: Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). For this project, CNN architecture has been used to train the given dataset as it's known to be designed to map image data to an output variable according to Brownlee, Jason (2018) [2], where as MLP and RNN are more suitable for tabular datasets and sequence prediction problems.

Problem Analysis

According to Stewart, Matthew (2019) [3], CNN layers learn: filters of increasing complexity, 1st layers learn basic feature detection filters such as edges, corners, and more, middle layers learn filters that detect parts of objects, and last layers learn to recognize full objects. Furthermore, a project conducted to research about data augmentation in CNN for gray-scale images conducted by Wang, Jinyeong et al. (2021) [4] suggests to augment one-channel grayscale images with original images, and convert the grayscale image into three-channel colour (RGB) images.

However, due to the limitation of knowledge and time constraint, the implementation of a function that transforms the grayscale images into RGB was unsuccessful. Despite having a method that transforms grayscale images to RGB named ''tf.image.grayscale_to_rgb" [5] in tensorflow, the image will remain as grayscale image since the converted 3 channels of colour will have the same intensity [6]. Therefore, instead of converting grayscale image to RGB on top of using "RandomFlip" and "RandomRotation" for data augmentation, 'adjust_contrast' has been added for augmentation of grayscale images, to make the kidney cells more obvious from the background. There were two approaches to make the kidney cortex cells standout- using 'adjust_contrast' (Image 1.1) method or 'adjust_gamma' (Image 2.1) method. I trained and ran tests on both methods, and 'adjust_contrast' method gave higher value accuracy of 0.64230 (Image 1.2) and faster learning rate where as 'adjust_gamma' gave value accuracy of 0.61585 (Image 2.2) with the same batch and similar epoch size of the same model. Moreover, 'adjust_contrast' could detect the cell feature and the background, and 'adjust_gamma' could detect the cell shapes.

Using the information obtained from the research above, this train model was built utilizing Convolutional Neural Network (CNN) architecture with:

- Rescaling for the best network performance preventing enhancement of non-optimal paths.
- 2D convolution layers
- Batch normalization for independent learning, learning efficiency, and overfitting prevention [7]
- Max pooling for down sampling input shape dimensions (height and width) [11]
- Dense layer for classification and improving accuracy caused by disappearing gradient in neural networks [10]
- L2 regularization method for regularization to apply "squared magnitude" of coefficient as penalty on the layer's output [8][9]
- Dropouts for regularization to prevent overfitting and to enhance learning of the model by switching percentage of neurons in the network. [7]

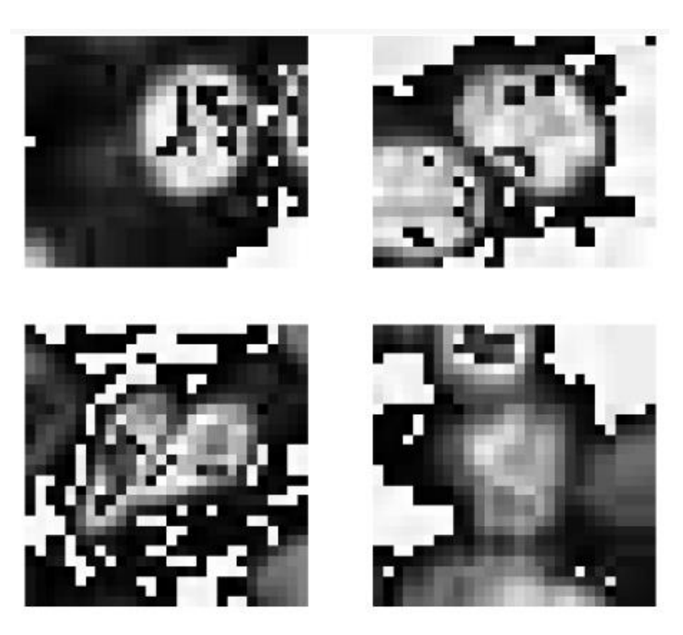
Image 1.1 Kidney Cortex Cell using Adjust Contrast

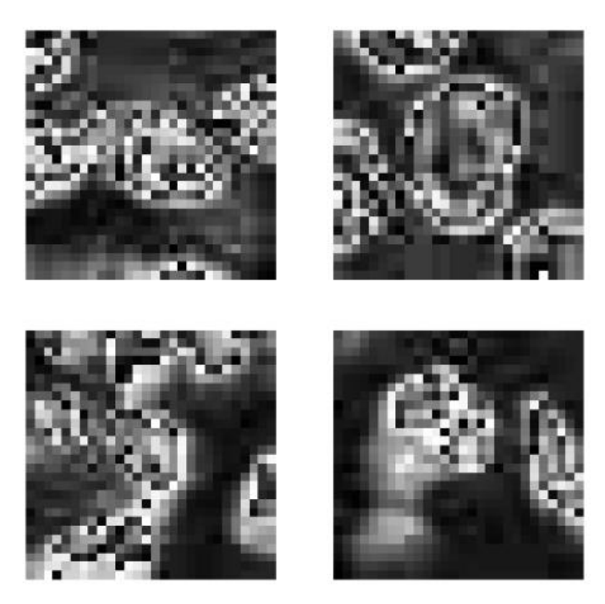
Image 2.1 Kidney Cortex Cell using Adjust Contrast

Augmentation with contrast (adjust contrast = 1.5, smooth- train, rough- validation;) vs Augmentation with gamma (adjust gamma=1.2, smooth- train, rough- validation;)

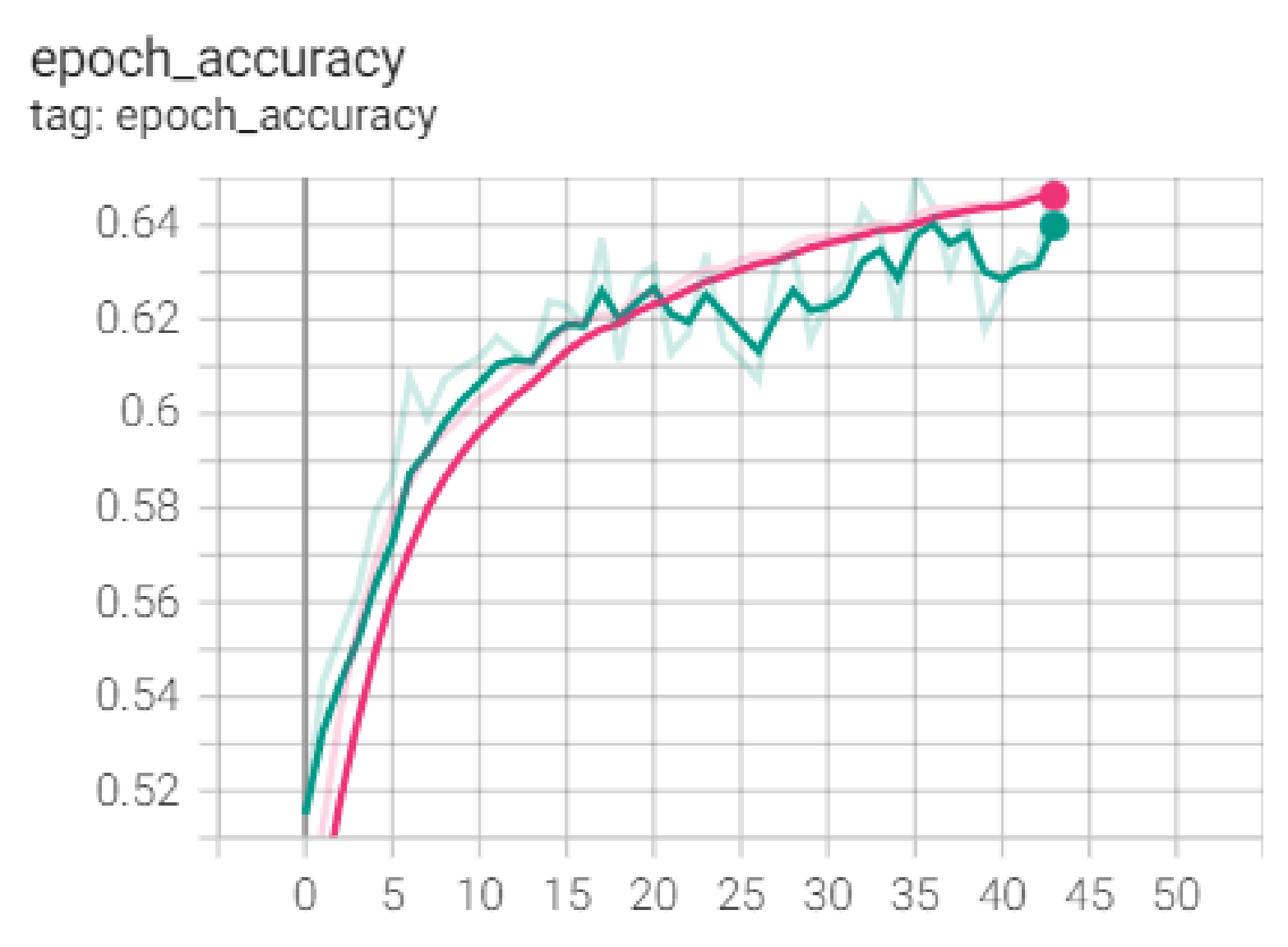

Image 1.2 Accuracy using Adjust Contrast

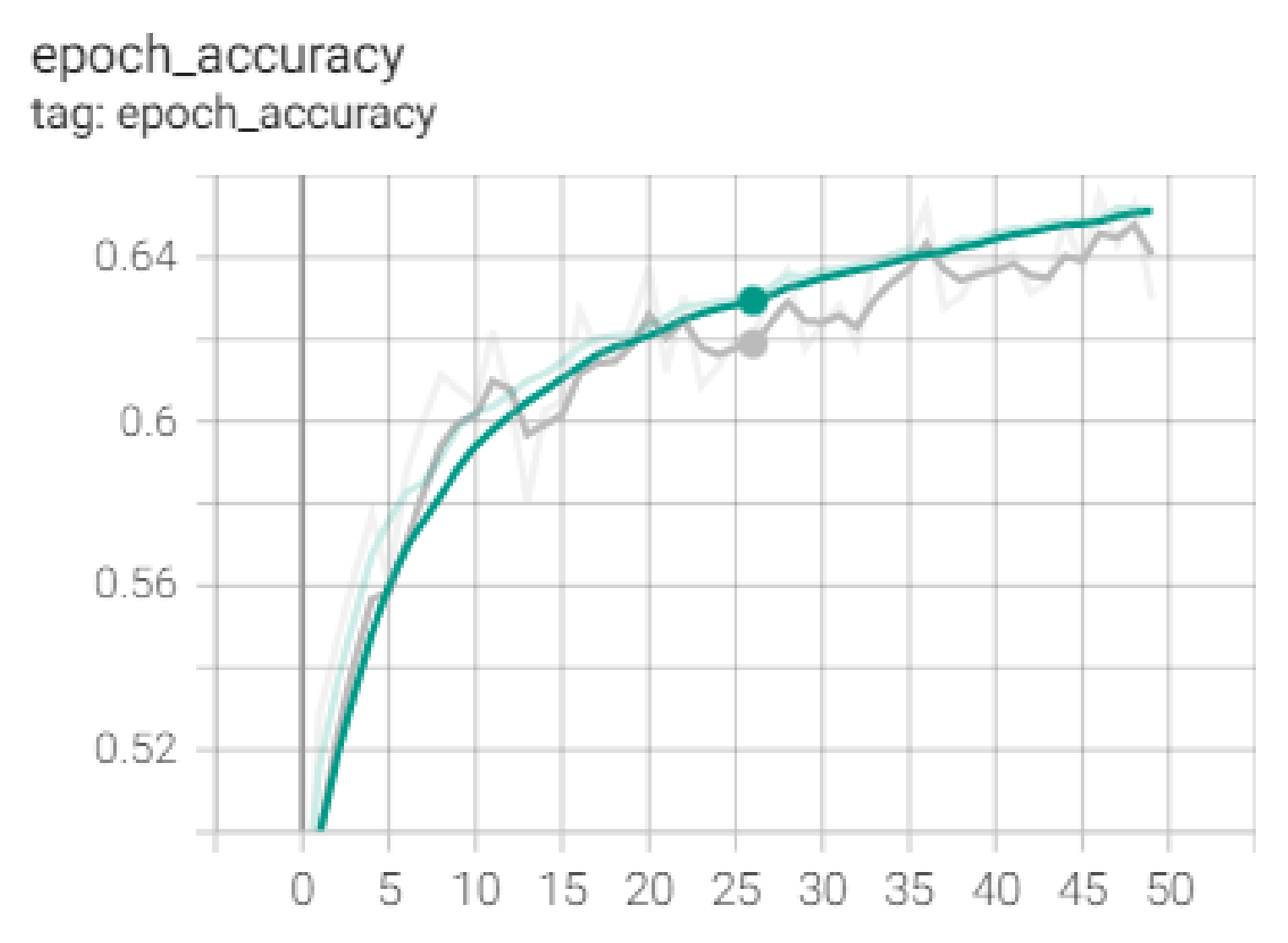

Image 2.2 Accuracy using Adjust Contrast

Implementation & Evaluation Method

The dataset training model with data augmentation using "adjust_contrast" function, has produced the result of 0.64230 with batch size of 128 and 44 epochs. As recommended in the computer labs in the class and by Bronshtein, Adi (2017) validation split of 0.2 was used in splitting the training and test split. 120 000 images were used for training, 30 000 pictures were used for validation, and the rest 50 000 were used for testing the model. (Image 3)

```
Found 120000 validated image filenames belonging to 8 classes.
Found 30000 validated image filenames belonging to 8 classes.
Found 50000 files belonging to 1 classes.
Test label names (empty): ['']
```

Image 3. Number of images for training, validation, and testing

Firstly, prior to building a model, I've written out 2 functions for data augmentations. First, the data augmentation was made using Keras RandomFlip and Random Rotation. Next, a class was written for adjusting contrast.

After building functions for data augmentation, I built the model. In the beginning, the model rescales the input images to improve the neural network performance and prevents non-optimal routes performing too well. Then, the previously implemented data augmentation functions are added to the model before adding any other layers, so that the images are fed to the model giving the computer time to learn the variations of the images without any confusion. From here, I implemented 4 similar blocks of code with a same structure but different convolutional layer parameters. The block starts with 2 identical convolutional layers with the padding set to 'same', as I've learned that it produces better accuracy than putting other layers in between doing the lab challenges. Furthermore, Brownlee, Jason (2020) states that stacking of convolutional layers allows a hierarchical feature extraction of the input [13]. The two Conv2D layers are then followed by BatchNormalization, MaxPooling2D, and Dropouts. I've implemented the block in this structure as we want to stabilize learning efficiency, train different variations of images in the next block and also assume that not all trained images are necessarily correctly classified in layers with same hyperparameters. Lastly, the model gets flattened in 1-D array to compare with the neighbouring classified data, use Dense for diverse number of classes to get more accurate identification along with penalty application on the output layer regarding the accuracy. After this there's final Dropout function with a rate of 0.5 for a good regularization before testing. Total of 1.3+ millions of parameters were used in my model. (Image 4)

According to predictions, Cell Type 0 and 6 compose a bit more than half of the dataset. (Image 5) I used checkpoint to save each model and compare it with other models, and the best checkpoint model result was 0.64230.

```
=================================================================
Total params: 1,375,080
Trainable params: 1,374,120
Non-trainable params: 960
_________________________________________________________________
```

Image 4. *Information about number of parameters in model from model summary.*

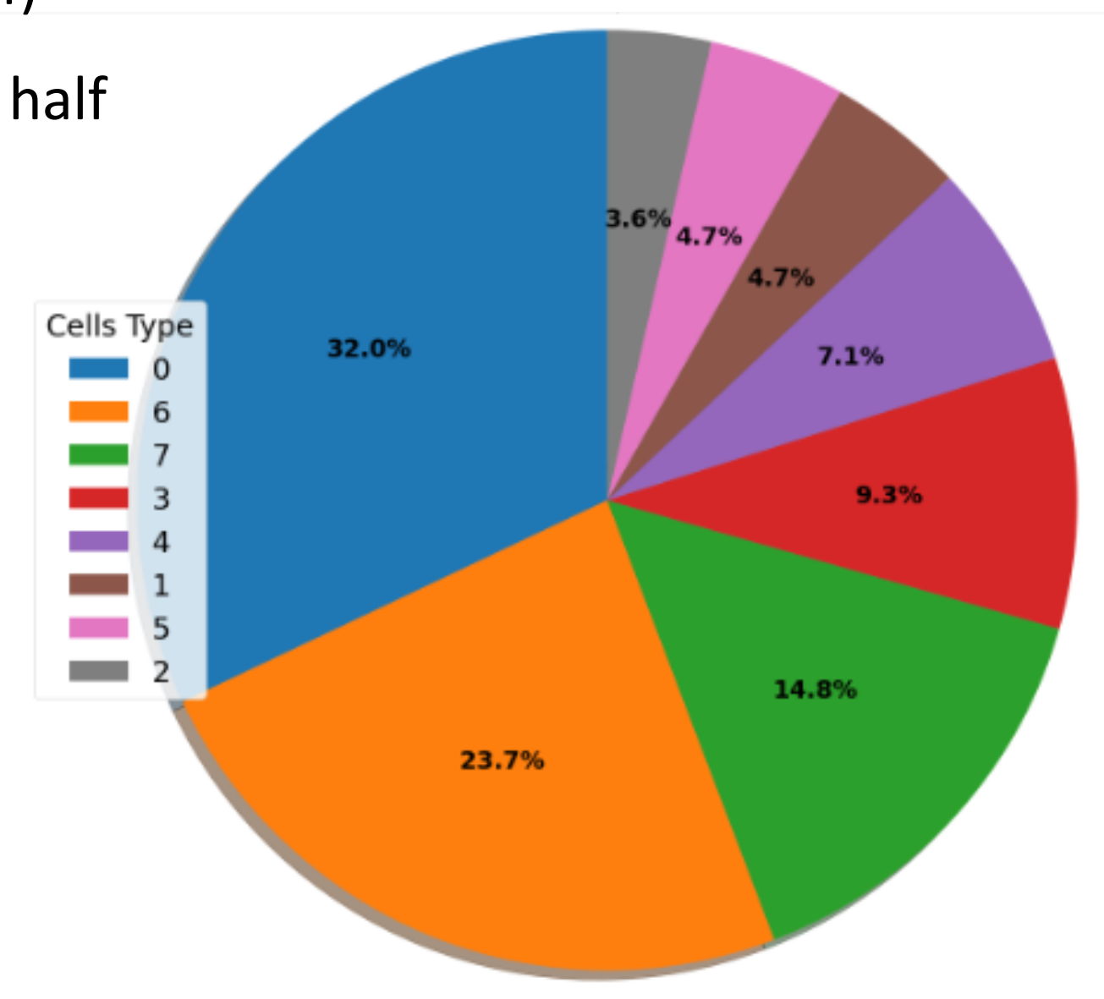

Image 5. *Pie chart of Cell Types*

Reference List

[1] Brownlee, Jason. 2019. *Overfitting and Underfitting With Machine Learning Algorithms.* Online. Available from: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/#:~:text=Overfitting%3A%20Good%20performance%20on%20the,poor%20generalization%20to%20other%20data [Last accessed: 10/05/22]
[2] Brownlee, Jason. 2018. *When to Use MLP, CNN, and RNN Neural Networks. Online.* Available from: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/ [Last Accessed: 11/05/22]
[3] Stewart, Matthew. 2019. *Simple Introduction to Convolutional Neural Networks.* Online. Available from: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac [Last Accessed: 11/05/22]
[4] Wang, Jinyeong, et al. 2021. *Data Augmentation Methods Applying Grayscale Images for Convolutional Neural Networks in Machine Vision.* Online. Available from: https://www.researchgate.net/publication/353406867_Data_Augmentation_Methods_Applying_Grayscale_Images_for_Convolutional_Neural_Networks_in_Machine_Vision [Last Accessed: 11/05/22]
[5] TensorFlow. 2022. *tf.image.grayscale_to_rgb.* Online. Available from: https://www.tensorflow.org/api_docs/python/tf/image/grayscale_to_rgb [Last Accessed: 11/05/22]
[6] edkeveked. 2021. *how to convert grayscale image to rgb RGB image?* Online. Available from: https://stackoverflow.com/questions/66084126/how-to-convert-grayscale-image-to-rgb-rgb-image [Last Accessed: 11/05/22]
[7] Dwivedi, Rohit. 2020. *Everything You Should Know About Dropouts And BatchNormalization In CNN.* Online. Available from: https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/ [Last Accessed: 11/05/22]
[8] Keras. 2022. *Layer weight regularizers.* Online. Available from: https://keras.io/api/layers/regularizers/ [Last Accessed: 11/05/22]
[9] Nagpal, Anuja. 2017. *L1 and L2 Regularization Methods.* Online. Available from: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c [Last Accessed: 11/05/22]
[10] Ruiz, Pablo. 2018. *Understanding and visualizing DenseNets.* 2018. Available from: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a [Last Accessed: 11/05/22]
[11] Keras. 2022. *MaxPooling2D layer.* Online. Available from: https://keras.io/api/layers/pooling_layers/max_pooling2d/ [Last Accessed: 11/05/22]
[12] Bronshtein, Adi. 2017. *Train/Test Split and Cross Validation in Python.* Online. Available from: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6 [Last Accessed: 11/05/22]
[13] Brownlee, Jason. 2020. *How Do Convolutional Layers Work in Deep Learning Neural Networks?* Online. Available from: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/ [Last Accessed: 11/05/22]