**DISCRIMINATING BETWEEN REAL AND FAKE IMAGES OF THE GALAXIES**

BY DHANUJA ARACHCHI, OMOREGBEOMWAN OLOTU, RANAWEERA RANAWEERA, OMESHA SAMARAKOON

*DEPARTMENT OF PHYSICS, ASTRONOMY AND MATHEMATICS, UNIVERSITY OF HERTFORDSHIRE*

**INTRODUCTION**

We present a neural network model built for the purpose of identifying real and fake images of the galaxies. The model was trained using convolutional neural networks due to their ability to learn hierarchical representations of image features, use shared weights to reduce the number of parameters, and their ability to handle spatial data.

We implemented the method in TensorFlow/Python programming language which can be found in our [GitHub](https://github.com/21062872/NNML-GP9/blob/main/NNML_Group9_Code_v1.ipynb) repository and [Google Colab workbook](https://colab.research.google.com/drive/1Z2QCQsJ3raDvzWNow7qiW3sBxlznbQjw?usp=sharing).

**METHODOLOGY**

The image data used for this task is from the Sloan Digital Sky Survey that imaged hundreds of thousand of galaxies and AI reproduced images of the galaxies. The (64, 64, 3) shaped images, totalling 8,000 contains 6000 images of training data which is evenly split between the real and fake images while the test data contains an evenly distributed number of real and fake images totalling 2000. This is done to avoid bias and ensure that the model is trained to recognize both real and fake images equally well.

The image data was uploaded to Google Colab for analysis and the tf.keras.utils package was used to load the data. The package automatically resizes the image shape to (256,256,3) and groups them in batches of 32 to reduce the training time and optimize the process. This means for the training images of 6000, training will be done in 32 batches with each batch containing 188 images. Then, we classified the images in a binary format where real images = 1 while fake images = 0 as seen in the image below:

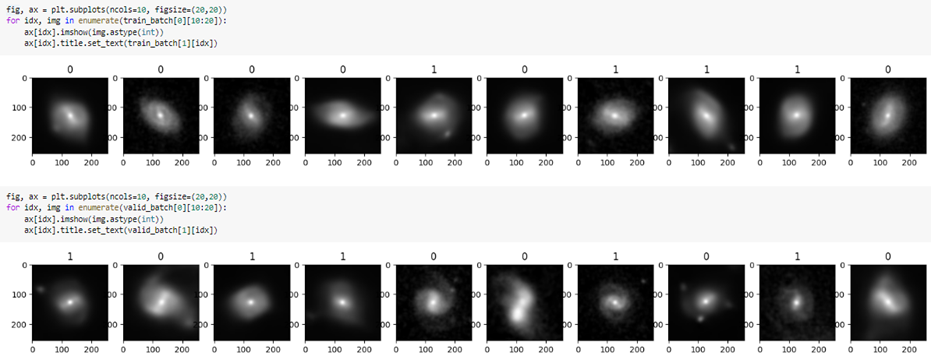


Figure showing image labelling.

The image data was normalized after augmentation was done on the images to enhance the robustness of the training data, improve the model performance and its generalization capabilities.

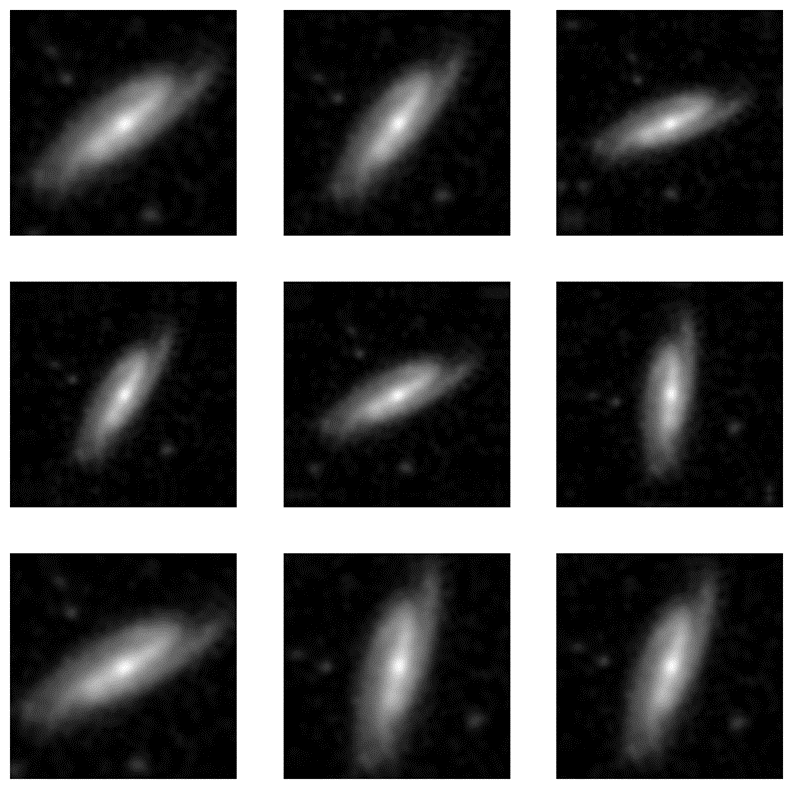


Figure 2 showing the augmented image data with horizontal flip, 0.1 rotation and 0.2 random zoom.

**MODEL ARCHITECTURE**

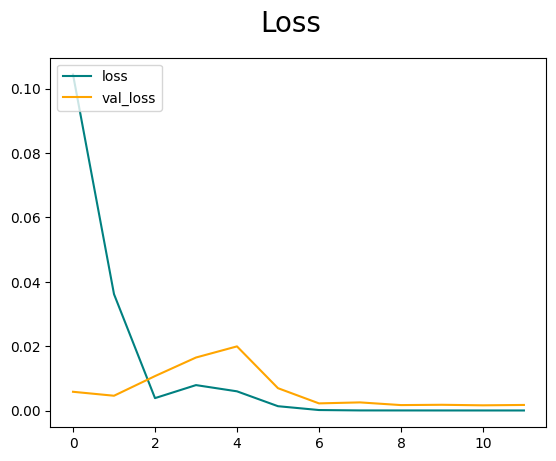
We incorporated a convolutional neural network into a sequential model for this task. The first layer is a 2D convolutional layer with 16 filters, each with a size of 3x3 pixels, The activation function RELU was used to introduce nonlinearity to the output. This layer is followed by a max pooling layer, which reduces the spatial size of the output feature maps by a factor of 2. The second layer is another 2D convolutional layer with 32 filters, also with a size of 3x3 pixels. This is followed by another max pooling layer. The third layer is a third 2D convolutional layer with 16 filters and a size of 3x3 pixels, followed by a max pooling layer. The flattened layer converts the output of the previous layer into a one-dimensional vector, which can be fed into a fully connected neural network. The first dense layer is a fully connected layer with 256 neurons and uses a RELU activation function, which applies a nonlinear transformation to the input data. The output of this layer is fed into the final dense layer, which has a single neuron and uses a sigmoid activation function to output a probability value between 0 and 1, representing the likelihood that the input image is real.

**TRAINING AND VALIDATION**

The model was trained using the Adam optimizer with the default learning rate of 0.001. We trained the model for 12 epochs, with a batch size of 32 while tf.keras.callbacks.TensorBoard callback function was used to log events . We used the BinaryCrossentropy loss function and accuracy metric to optimize the model.

The training accuracy starts at 95.13% and reaches 100% by the end of the training, indicating that the model can classify the training images with high accuracy. The validation accuracy starts at 99.85% and remains consistently high throughout the training process, indicating that the model can generalize well to new, unseen data.

The training loss starts at 0.1044 and decreases rapidly during the early epochs, indicating that the model is learning to fit the training data well. The validation loss is consistently lower than the training loss, which is a good sign that the model is not overfitting to the training data. The validation loss which started at 0.0058 fluctuates slightly during the later epochs, but remains relatively low overall, indicating that the model can generalize well to new data.

 Chart

Description automatically generated

*Figure 3 showing the loss and validation loss function both decreasing while the accuracy and validation accuracy both increases which makes the model ideal.*

**RESULTS AND CONCLUSIONS**

On testing the prediction capabilities of the model, we tested it on 10 randomly selected real and fake images each. The model was able to correctly identify both the fake and real images 100% of the time. These results demonstrate the high performance of our deep learning model.

 Graphical user interface

Description automatically generated

**REFERENCES**

1. Renotte, N. (2022). Build a Deep CNN Image Classifier with ANY Images [Video]. YouTube. <https://www.youtube.com/watch?v=jztwpsIzEGc&t=4171s>
2. Barchi, P.H., de Carvalho, R.R., Rosa, R.R., Sautter, R.A., Soares-Santos, M., Marques, B.A.D., Clua, E., Gonçalves, T.S., de Sá-Freitas, C., & Moura, T.C. (2019). Machine and Deep Learning applied to galaxy morphology - A comparative study. Astronomy and Computing, 28, 100315. <https://doi.org/10.1016/j.ascom.2019.100334>