# COMP4423 – Assignment 2

**LEUNG Kit Chuen**

## A. Task Requirement

It is a classification task because the goal is to classify images into two categories: real-world pictures captured on the PolyU campus and AI-generated images of corresponding architectures. In other words, the task involves determining whether an image is genuine or artificially generated. Aiming to assign a label or category to input data based on its features is considered a classification task.

In this scenario, a Convolutional Neural Network (CNN) model can be trained to learn the distinguishing features between real-world and AI-generated images. The model will analyze the pictures' patterns, textures, and structures to predict their authenticity. Studies have shown that AI-generated images have explainable features that vary from real photos [1]. Therefore, it is confident to use CNN to complete the tasks.

In addition, we are encouraged to classify the false images into different art styles. Using a classification model remains an effective solution to solve the problem. Therefore, the required CV task shall be constructing a model that simultaneously classifies the real and false pictures with different visuals.

## B. Data Collection and Labeling

It is assigned that data collection shall be performed manually. During collection, a total of 174 real images were collected. These images are taken in various venues, such as the platform lawn, the Innovation Tower, and the Z core platform. In addition, a total number of 94 AI-generated pictures are created with getimg.ai [2]. The A.I. tool provides three art styles: photorealism, artistic, and anime. There are 33, 33, and 28 fake images for each corresponding style.

For validating the model, 70% of the images are used as a training set, while 30% are used as a validation set. After reaching the satisfying score, we use a separate test dataset to validate the accuracy.

Initially, we attempted to fill the class with string labels. However, it raised errors, and we were forced to change the labels into numeric ones and use One Hot Encoder to modify them further.

## C. Image Preprocessing

Image preprocessing is a crucial step in computer vision projects. In the context of classifying real and AI-generated images of the PolyU campus, the following preprocessing steps are recommended:

> **Resizing**: Images in the dataset may vary in size, which can introduce inconsistency and computational overhead during training. Resizing all images to a uniform resolution ensures consistency and reduces computational complexity. The resizing operation should maintain the aspect ratio of the original images to prevent distortion. The size shall depend on the input of each model.
>
> **RGB Standardization**: To ensure consistency across the dataset, it is essential to standardizing the color channels of the images. The algorithm divides each colour value by 255 to stabilize the learning process and improve convergence during training.

In addition, data augmentation is applied to datasets to enhance the models' robustness. The provided function *augmentImage* is an image augmentation function designed to achieve the goal, particularly those trained on image data. Image augmentation is a common technique used in computer vision tasks to increase the diversity of the training dataset by applying random transformations to the input images. It helps the model generalize better to real scenarios and reduces overfitting.

Let's break down the different types of image augmentation techniques applied in the function:

> **Rotation**: Randomly rotates the image between -45 and 45 degrees. It helps the model learn to recognize objects from different viewpoints and orientations.
>
> **Flip**: Randomly flips the image horizontally or vertically. It introduces variations in the orientation of objects within the image.
>
> **Scaling**: Randomly scales the image between 0.8 and 1.2 times its original size. It simulates changes in the distance between the camera and the object and in object size.
>
> **Translation**: Randomly translates the image by up to 20 pixels in horizontal and vertical directions. It simulates shifts in the position of objects within the image.

Each augmentation operation is applied based on the value of random_seed. By using the modulo operation with different prime numbers (2, 3, 5, 7), the function ensures that each augmentation operation has a different probability of being applied. It helps introduce diverse transformations to the images while avoiding over-augmentation.

By default, there are four augmented images for each original one.

## D. Model Selection

Firstly, we shall focus on the real vs fake classification task. We will test various CNN classification models, including AlexNet, VGGNet, and ResNet. Adam Optimizer is used on all models to control the comparison.

### 1. AlexNet

AlexNet is one of the earliest CNN models proposed in 2012 [3]. Its architectural innovations paved the way for the development of more profound and powerful CNN models, revolutionizing the field of computer vision and contributing to significant advancements in image classification tasks.

To begin with, we need to reconstruct the AlexNet with Tensorflow Karas. It was relatively easy to reconstruct as the model's detail was already given. The following table shows the structure of AlexNet.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 55, 55, 96) | 34,944 |
| max_pooling2d (MaxPooling2D) | (None, 27, 27, 96) | 0 |
| conv2d_1 (Conv2D) | (None, 27, 27, 256) | 614,656 |
| max_pooling2d_1 (MaxPooling2D) | (None, 13, 13, 256) | 0 |
| conv2d_2 (Conv2D) | (None, 13, 13, 384) | 885,120 |
| conv2d_3 (Conv2D) | (None, 13, 13, 384) | 1,327,488 |
| conv2d_4 (Conv2D) | (None, 13, 13, 256) | 884,992 |
| max_pooling2d_2 (MaxPooling2D) | (None, 6, 6, 256) | 0 |
| flatten (Flatten) | (None, 9216) | 0 |
| dense (Dense) | (None, 4096) | 37,752,832 |
| dropout (Dropout) | (None, 4096) | 0 |
| dense_1 (Dense) | (None, 4096) | 16,781,312 |
| dropout_1 (Dropout) | (None, 4096) | 0 |
| dense_2 (Dense) | (None, 4) | 16,388 |

Total params: 58,297,732 (222.39 MB)
Trainable params: 58,297,732 (222.39 MB)
Non-trainable params: 0 (0.00 B)

Fig 1. Model structure of AlexNet

## 2. VGGNet

The Visual Geometry Group at the University of Oxford had proposed another convolutional neural network architecture, namely VGGNet. It is known for its simplicity and uniform architecture. VGGNet achieved second place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 [4].

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 224, 224, 64) | 1,792 |
| conv2d_1 (Conv2D) | (None, 224, 224, 64) | 36,928 |
| max_pooling2d (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 112, 112, 128) | 73,856 |
| conv2d_3 (Conv2D) | (None, 112, 112, 128) | 147,584 |
| max_pooling2d_1 (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| conv2d_4 (Conv2D) | (None, 56, 56, 256) | 295,168 |
| conv2d_5 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| conv2d_6 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| max_pooling2d_2 (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| conv2d_7 (Conv2D) | (None, 28, 28, 512) | 1,180,160 |
| conv2d_8 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| conv2d_9 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| max_pooling2d_3 (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| conv2d_10 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| conv2d_11 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| conv2d_12 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| max_pooling2d_4 (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| flatten (Flatten) | (None, 25088) | 0 |
| dense (Dense) | (None, 4096) | 102,764,544 |
| dense_1 (Dense) | (None, 4096) | 16,781,312 |
| dense_2 (Dense) | (None, 4) | 16,388 |

**Total params:** 134,276,932 (512.23 MB)
**Trainable params:** 134,276,932 (512.23 MB)
**Non-trainable params:** 0 (0.00 B)

Fig 2. Model structure of VGGNet

## 3. RegNet

RegNet is a deep convolutional neural network architecture introduced by Kaiming He et al. from Mircosoft Research in 2015. It was designed to address the problem of vanishing gradients in intense neural networks by introducing skip connections or shortcuts that allow the gradient to flow more easily during training [5].

Besides introducing residual learning and skip connections, its extreme depth is well-known to the public. It contains 152 layers in its structure with 23.7M learnable parameters. Although its number of parameters is less than VGGNet's, its performance overthrew the rest of the competitors during the ILSVRC & COCO 2015 Competitions.

## E. Model Training and Testing

### About the Codebase

For convenience, we separate each model into different classes, namely AlexNet, VGGNet, and ResNet. Also, to construct a consistent dataflow, we also establish class DataUtilizer to handle general tasks, such as reading images, writing model files, and augmenting images. With an organized codebase, the workload is likely reduced, and the efficiency of development increases.

### Performance Comparison

After training each model, the following results are given:

| **Model** | **Average Validation Score** | **Final Validation Score** | **Test Score** |
|---|---|---|---|
| AlexNet | 0.5407 | 0.6296 | 0.5714 |
| VGGNet | 0.2642 | 0.6296 | 0.5714 |
| ResNet | 0.4728 | 0.0370 | 0.3571 |

Fig 3. Validation and test accuracy of the 3 models

As observed, both AlexNet and VGGNet have similar results, and ResNet has the lowest test scores. The accuracy would remain around 35% to 57% if deployed directly on campus.

## F. Fine Tuning

Given the basic models implemented, some methodologies are proposed to enhance the performance of each model.

### 1. Dropout

A dropout function is used to tackle the overfitting issue. Srivastava et al. proved that dropout can significantly reduce overfitting and improve model performance compared to alternative regularization methods [6]. We will conduct a dropout with a 0.5 rate between each complete connect layer, reducing the difference between validation and test scores.The result is as follows:

| Model | Average Validation Score | Final Validation Score | Test Score |
|---|---|---|---|
| AlexNet | 0.5840 | 0.6296 | 0.5714 |
| VGGNet | 0.4111 | 0.6296 | 0.5714 |
| ResNet | 0.6296 | 0.6296 | 0.5714 |

Fig 4. Validation and Test Accuracy after implementing dropouts

Through dropouts, the efficiency of improvement is faster. However, the performance improvement has reached its bottleneck. This enhancement remains for further experiments.

### 2. Adam Learning Rate

Optimizers play a significant role in CNN model construction. Adam optimizers are among the most common optimizers used by researchers and engineers, and their learning rate is decisive in model performance. Initially, the learning rate of Adam optimizers was set to 0.01, which is relatively high for a small dataset. Therefore, we will provide options of rate (0.0002~0.001, increase by 0.0002 each iteration) for Adam and experience which rate is the most optimal for each model.

The following graphs indicate the relationship between the learning rate and model test score. The statistic reveals that, with a decrement in the learning rate, AlexNet benefits the most, and **the test accuracy of 0.6296** reaches a new height during the process. On the other hand, ResNet has the lowest score among the models. However, the increment of ResNet is noticeable, meaning it performs better with a higher learning rate. Overall, the effect of this fine-tuning depends on the models.

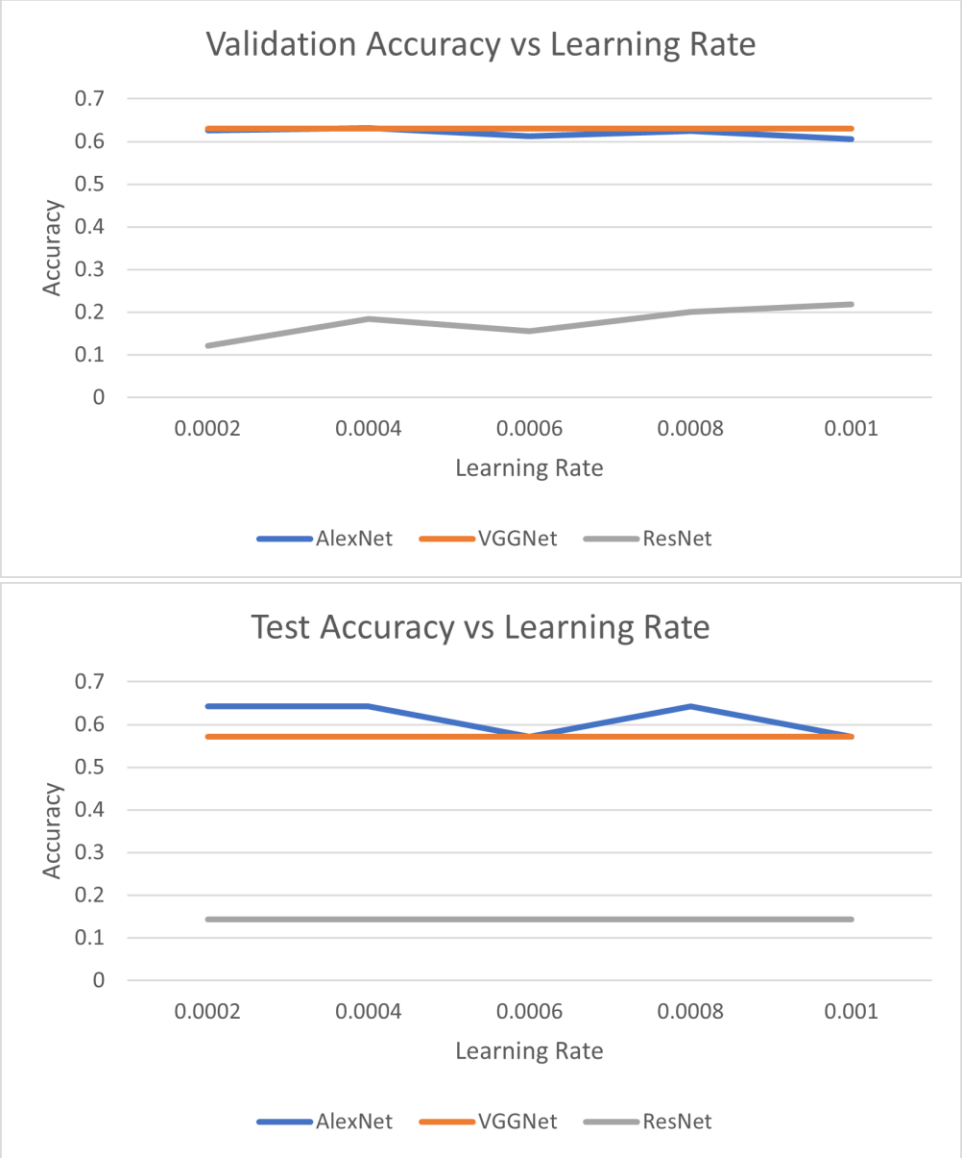


Fig 5. Accuracy vs Adam learning rate

## G. Summary

We had performed many procedures during this assignment, including task definition, data collection, image labelling and preprocessing, model selection, and fine tuning. Eventually, we obtained a classification model which identifies real pictures and style of AI-generated pictures, with a 62.96% confidence.

## References

[1] J.J. Bird, A. Lotfi, "CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images," arXiv, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.14126

[2] Webrockets, 2024. *getimg.ai* (Version 1.0) [Image Generator]. [Online]. Available: https://getimg.ai/features/image-to-image

[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," presented at Advances in Neural Information Processing Systems 25. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[4] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1409.1556

[5] K. He, X. Zhang, S. Ren, "Deep Residual Learning for Image Recognition," presented at 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7780459

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol.15, pp.1929-1958, 2014. [Online]. Available: https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf