

Master of Technology (IS)

Continuous Assessment 2

Project Report

Team Member:

CAO LIANG A0012884E

TAN CHIN GEE A0195296M

GENG LIANGYU A0195278M

Contents

[Executive Summary 3](#_Toc17663207)

[Business Problem Background 3](#_Toc17663208)

[Project Objectives & Success Measurement 3](#_Toc17663209)

[Project Solution Design 4](#_Toc17663210)

[Project Implementation 4](#_Toc17663211)

[Project Performance & Validation 5](#_Toc17663212)

[Project Conclusions: Findings & Recommendation 5](#_Toc17663213)

# Executive Summary

This project constructs deep-learning model to determine the public domain images classification.

# Business Problem Background

The project requires to not use the pre-built image dataset from the public sources such as Kaggle, and the images should be downloaded from public domain without royalty and copyright. Also the number of images for each class should at least 1000.

The project also requires to use only deep-learning model to resolve the classification problem.

# Project Objectives & Success Measurement

We decide to select 3 classification class: cat, bird, and dog for the project, so should download the required the public domain images and keep at least 1000 images for each classification class, and build the deep-learning model to train and validate the model.

The measurement for the project will depends on the prediction accuracy on the test data, and it should show at least 10% or more improvement on classification accuracy to be success, or it should reach at least 75% classification accuracy after tuning model.

# Project Solution Design

The project should prepare the required image dataset, then filter out the unfitted images, then construct the training dateset and test dataset, and then constructs deep-learning model and perform hyper-parameter tuning, and finally perform the comparison and select the final classification model to highest accuracy.

The project follows the supervised learning process as below.

# Project Implementation

The project performs the steps below.

1. Image Data Collection

We have tried the different ways to download the required image.

* 1. <https://www.pexels.com/api/>

We first tried to download from pexel.com, and there is existing python PyPexels package that can be utilized to download images. But there are some major drawbacks in this approach, firstly, there are limited files that can be downloaded every day (<= 200), secondly, there is no enough images for our selected class i.e. cat, bird and dog, and third, and most serious, the image quality for our selected class is not good, and especially for bird and cat, many (>50%) are irrelevant images.

We have developed “pyprexels.py” application for testing and downloading images from the web site, and finally give up the solution based on reasons above.

* 1. <https://snappygoat.com>

Next, we choose “snappygoat.com” and develop “searchimg.py” to download images, but still encounter the issues. First, the images for each class are not enough (<1000), second, some of the images are irrelevant, third, and most serious, the images are not truly royalty free. And we have to pay the fee in order to use these images for free. So we also give up this solution.

* 1. <https://www.dreamstime.com>

This is the solution we are using in the project. The site “dreamstime.com” not only provides the enough images but also most of images from our search are very good to be the candidate image data for the project. And we also learn that the images from “dreamstime.com” are truly royalty free for the project without any charge.

We develop “getpubimg.py” application to download all the images for the project, and each class has 1200 images. We adopt selenium to implement the solution, since the web site html content is embedded in the script and cannot be parsed by html parser such as BeautifulSoup. Also the web site takes certain measures to prevent auto-download scrapper, and we use the timer to wait for 15 seconds after clicking next page link to bypass the restriction.

1. Pre-Processing

We have performed the following steps to filter irrelevant images.

2-1 Remove empty images

We noticed some of the image files contains only blank images, so all these images are removed.

2-2 Save image data and class labels

We develop “HDF5\_Convertor.py” to read images file, convert its resolution to 128x128, and save its content and class label to “data128.h5” hdf5 format file.

During this process, we also filter out the images which does not have low validation results based on different reasons blow. We develop “HDF5\_Convertor\_optimize\_img.py” to read all available image files, and “PSUPR\_CA2\_optimize\_img.py” to validate the image with pre-built model. The pre-built model weight parameters are loaded from “PRMLS\_CA2\_74.54.hdf5” file. All images with below 0.1 prediction probability are recorded to “unfit\_list.txt”, and thus be filtered out in “HDF5\_Convertor.py” process.

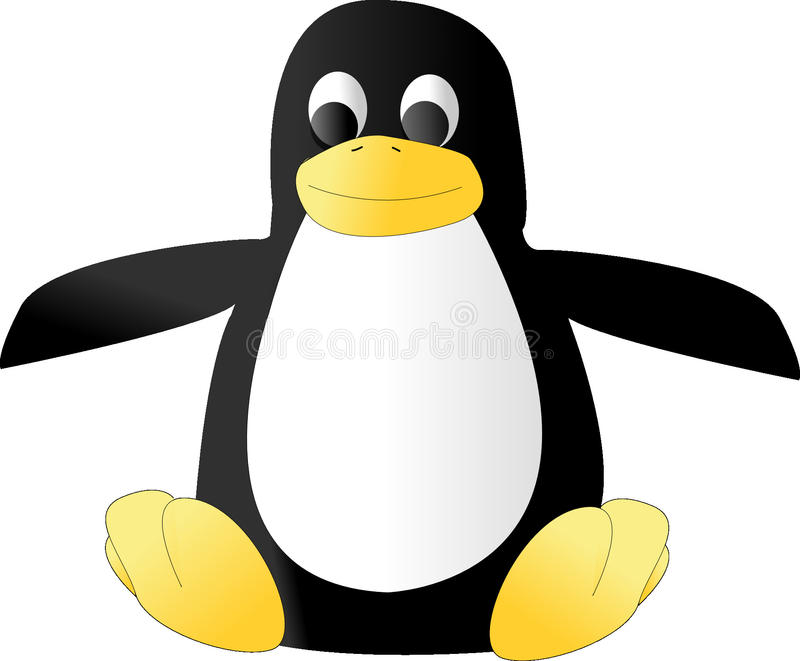
1. Targets blended into their environment because of their camouflaged coat colour



1. Targets dressed in human attire



1. Target images are either drawings or cartoons



1. Images are not of the targets



1. Targets are obscured by objects (e.g., tree branches)



1. Target images contained other objects (e.g., human)



1. Targets are small inside their images



1. Sampling

We develop “HDF5\_Dataset.py” to read image data and class labels from “data123.h5” file, and then divide them into training dataset and testing dataset, next save to “ca2data.h5” hdf5 file.

1. Deep-Learning Model Training

We develop “PSUPR\_CA2.py” and construct deep-learning ResNet model as the base model. The model is shown at “PRMLS\_CA2\_model\_final.pdf” below.



It contains total 2,674,819 parameters, and inside them there are 2,668,067 trainable parameters and 6,752 non-trainable parameters.

1. Hyper-parameter Optimization

We develop the base deep-learning model with all known best practice parameters, and the model achieves 73.36% accuracy. Then we adjust the parameters in the ways described in below sections, and keep the adjustment if the model accuracy increases.

5-1 Select activation function

We choose Relu activation function so it will be better to update and prevent gradient vanish comparing to other activation function such as SigMoid.

5-2 Set weights initialization value

We choose He initialization to set weight initialization values, so to avoid to assign value to be too small or too large.

5-3 Restrict weights

We select L2 regularizatio to control the magnitude of the weights so to keep the smaller weights value.

5-4 Add dropout

We have tested to add dropout between layers, but the accuracy results are not worse, so we have removed the dropout.

5-5 Perform batch normalization

We use the batch normalization to normalize the output for the convolutional layer.

5-6 Design residual network

We follow the ResNet design rules to design the residual network to improve the deep-learning network performance. We designed the residual layers by: (1) specifying the number of residual blocks from a range between 5 to 11, and (2) specifying the number of stages from between 3 to 4.

We kept the values of certain parameters constant in specifying the residual layers. Some of these parameters and their values are:

* Kernel size = 3
* Strides = 1
* Kernel initializer = he normal

The first stage of the residual layers does not contain a downsample block. But each of the subsequent stage comprises an initial downsample block with the reminding blocks being simple blocks.

We increased the number of filters from 16 for the initial stage of the residual layer to 32, 64, and 128 for each of the subsequent stages.

5-7 Set learning scheduler

We create and utilize the learning scheduler to vary the learning rate, so learning rate is reduced when the learning epochs increase.

5-8 Use image augmentation

We use the image augmentation to generate the randomly varied images in the beginning of each epoch, so to force the deep-learning model to learn the features that are relevant to the classification.

5-9 Adjust batch size

We choose small batch size (32) to improve the accuracy, and also it is still allowed the laptop GPU used for training to work as normal.

5-11 Ensemble models

We ensemble 2 models using the integrated stacking method, which combines the 2 models output and constructs common dense layer to finally output the results. The final accuracy achieves 79.61% and shows big improvement on the individual model 76.31% accuracy.

5-10 Problems encountered and challenge

1. Model Training Result Different

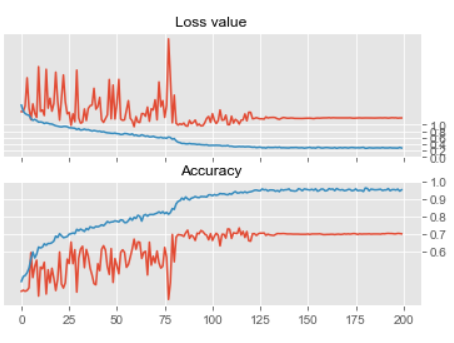
We notice that even for the same model design, the model accuracy after each training is very different, sometimes even reaches 4% to 5% accuracy difference. So we save the individual model weights in the separate file, then will be able to replicate the same accuracy with the model design, and avoid to lose the best model.

1. Determination on good parameters

We observe that the model performance differs for each separate model training process. In order to determine the good parameters to be changed, we only choose single parameter to change at each time, for example, learning rate. Then perform several trainings. We will consider the model best accuracy across the several trainings, and analyse the loss value and accuracy training diagram, then determine whether to keep the changes to the model. After that, we continue to pick the next available parameter and perform the same evaluation process again.

1. Unstable model

We notice that the base model is unstable from training loss value and accuracy chart below.



So in order to fix the issue, we have tried many different ways, e.g. change learning rate, add drop-out layers, add more layers, but the improvement is trivial. We finally find out the way to stabilize the model is to use the integrated stacking models. After integrating 2 same models, the training loss value and accuracy chart shows as below.



1. Challenge

The biggest challenge we are facing is the computer resource constrains. The normal training time for our deep-learning model takes 2 hours to 4 hours. And we do not have enough time to improve the model. So the best model and best performance is the one we achieve by good luck.

1. Post-Processing
2. Final Deep-Learning Model

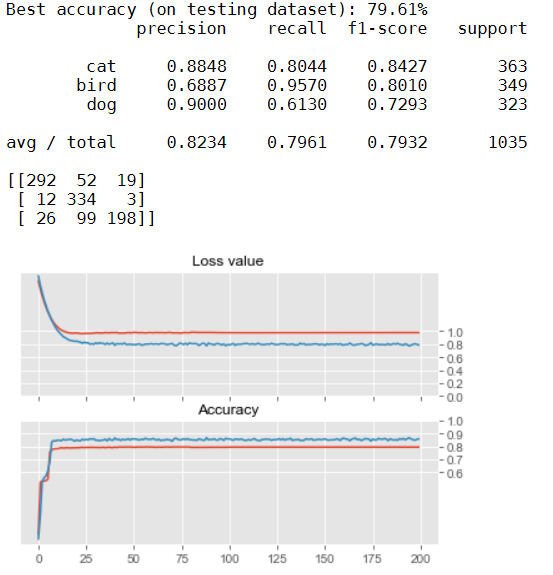
We choose the ensemble deep-learning ResNet model as the final dee-learning model, and develop “PSUPR\_CA2\_Ensemble\_NN.py” to implement the model. The 2 models inside the ensemble model are pre-trained deep-learning ResNet models. The model diagram is shown at “PRMLS\_CA2\_Ensemble\_NN\_model\_final.pdf”.



It contains total 5,349,701 parameters, and inside them there are 63 trainable parameters and 5,349,638 non-trainable parameters.

# Project Performance & Validation

The project final deep-learning model archives 79.16% accuracy validated with test dataset, so it meets the project performance requirement. The validation result is shown below.



# Project Conclusions: Findings & Recommendation