Homework 1

Visual Recognition using Deep Learning Bird Dataset Classification

Department : IMM, Yi-Cheng Hung, ID : 310653005

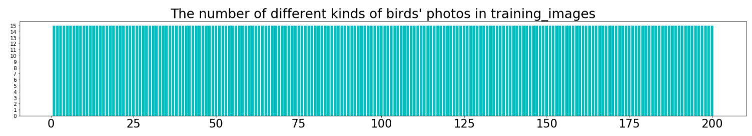
September 18, 2022

Github

Github : https://github.com/YCHung1998/Birds_Classification

Introduction
This homework is part of a series of projects for the course Selected Topics in Visual Recognition using Deep Learning. This Repository gathers the code for **Bird images classification** on the CodaLab platform . (2021 VRDL HW1)
In this classifcation task, we apply the basic data preprocessing, data augmentation and ensembling method to conquer this task. resnet50 is the main model architecture.
In the following I will introduce in detail the methods I used and executed.

0. Data preview In the beginning, take a look and check the different kind of bird data quantity is imbalance or not. It's clear that each categories has 15 images in training_images fold, so I split it into training : validation = 12 : 3 uniformly.



1. Data preprocessing

(a) Transform In the past, I had tried various transform method (be written in PyTorch package) before put them into the model training, but the performance not very well. So, I research the **Grid Mask** method [1]. This method motivation is to avoid excessive deletion and retention. In order to strike a balance between deleting and preserving the area information on the image, we need to choose suitable parameters such that conseutive information not loss too much and prevent the training model focus on the specific region. Let's show the effect below the figure.

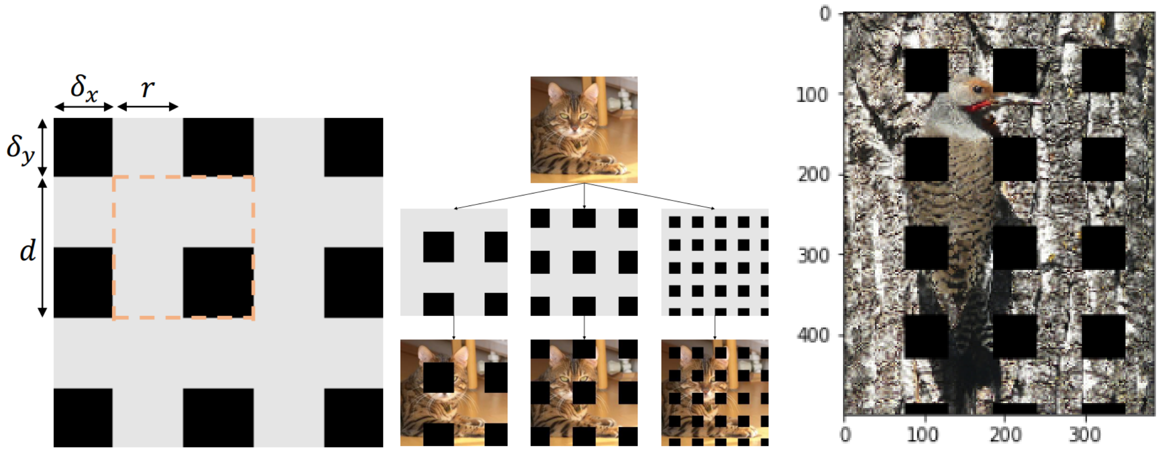


Figure 1: Left image produce a mask according to the given parameters $(r, d, \delta x, \delta y)$, the center shows that in the randomsize, the grid mask will cut into different numbers and size of black lattices to erase the information, and the right hand image that is an example image took from training image 0012.jpg.

The following list is transform on image in order:

- Rescale the pixel value from $[0, 255]$ to $[0, 1]$.
- Resize the image from uncertain to shape $(500, 500)$.
- Random apply the Gaussian noise with probability $p = 0.5$ and $\sigma = 0.01$.
- Random ordered the random transform [Horizontal flip, Rotation(degree=10°), Affine].
- GridMask transform $p = 0.2$, $r = 0.7$, randomsize $d \in (160, 300)$.
- Normalize each channel with $\mu = (0.485, 0.456, 0.406)$, $\sigma = (0.229, 0.224, 0.225)$.

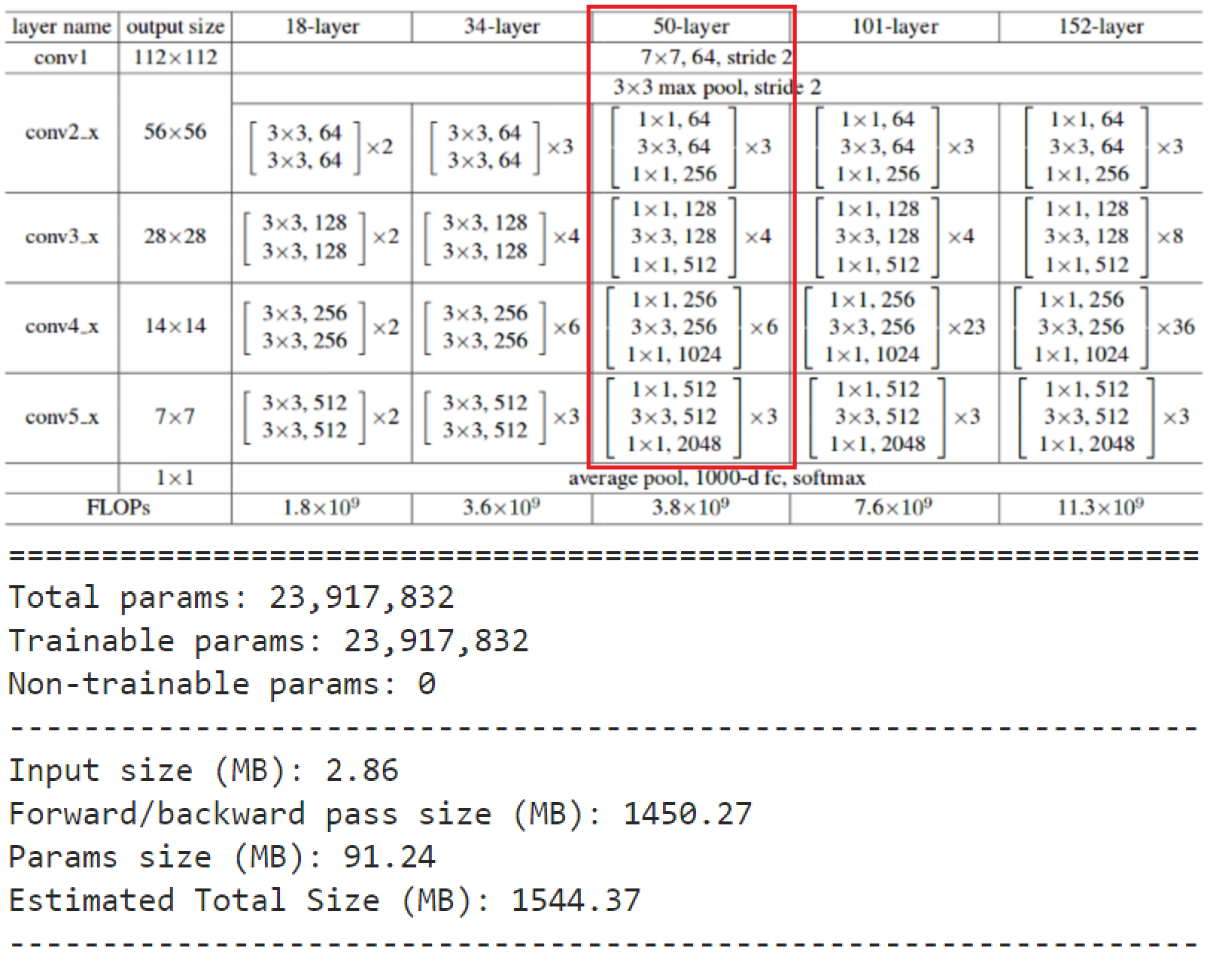
The dictionary version of transform is based on the form of the monai suite transform.(MONAI)

(b) Transform(Increasing image quantity)

In order to decreasing the computing time cost, we prepare three fold which be applied gridmask, and one folder for horizontal flipping. A total of raw data and transformed data are read in. The number of data in each folder is used as training data (a total of 9000 records). But in the last result, I found that take original image and horizontal flip into dataset is more suitable (no matter in the accuracy or in time costing.)

2. Model select

Based on the benchmark requirements, the resnet50 model was used. This model was selected to achieve the baseline standard first. During the process, other models such as Densenet121 were tried, but under the same transform MBS, the performance was not as good as the former model, so in the end it was still chose to maintain the original structure.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

```
================================================================
Total params: 23,917,832
Trainable params: 23,917,832
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 2.86
Forward/backward pass size (MB): 1450.27
Params size (MB): 91.24
Estimated Total Size (MB): 1544.37
----------------------------------------------------------------
```

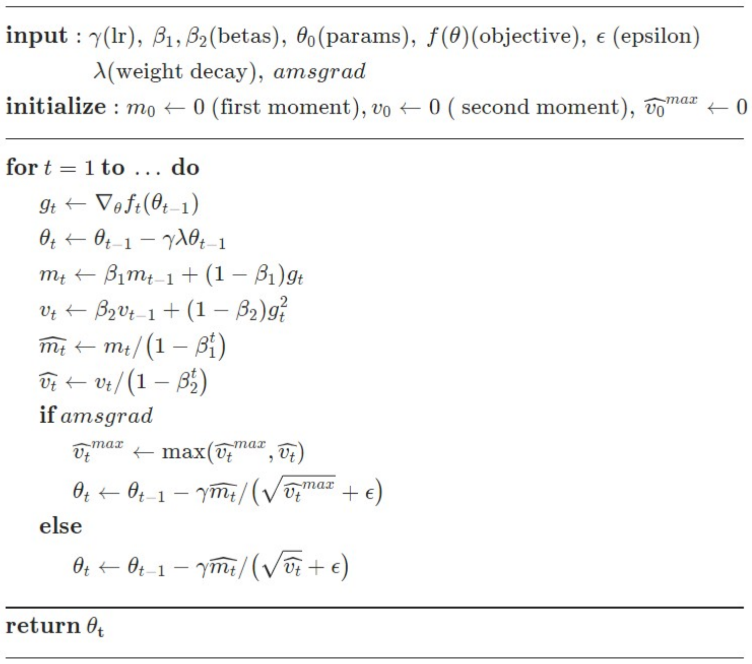
Figure 2: Listing the model information and the necessary train parameter number

3. Training

Use the resnet50 pre-training parameters to switch the tail *fc* classifier into two hundred types of output. Set the Mini-batch size is 20.

(a) optimizer

- **AdamW**
  $lr = 10^{-4}, weight\_decay = 10^{-4}$ (AdamW PyTorch)



```
input : $\gamma$(lr), $\beta_1, \beta_2$(betas), $\theta_0$(params), $f(\theta)$(objective), $\epsilon$ (epsilon)
        $\lambda$(weight decay), $amsgrad$
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ ( second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ to $\ldots$ do
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
    $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)g_t$
    $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)g_t^2$
    $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$
    $\widehat{v_t} \leftarrow v_t / (1-\beta_2^t)$
    if $amsgrad$
        $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
        $\theta_t \leftarrow \theta_{t-1} - \gamma\widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big)$
    else
        $\theta_t \leftarrow \theta_{t-1} - \gamma\widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)$

return $\theta_t$
```

Figure 3: AdamW Algorithm

- **Learning rate schedule**
  weight decay multiply by 0.8 in each 4 step.

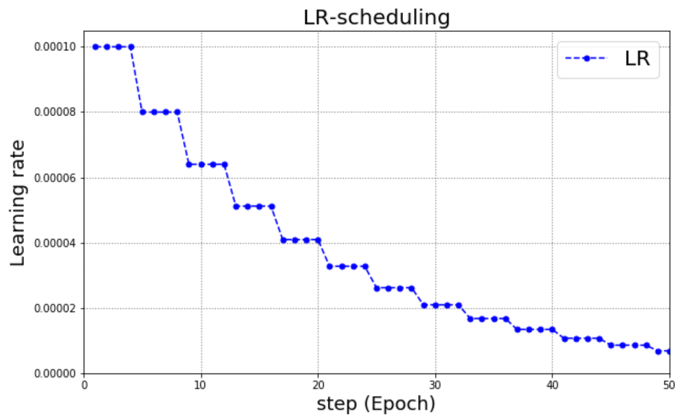


Figure 4: Learning rate schedule

(b) loss function

- **Cross entropy**

$$H(p,q) = -\sum_{x} p(x)\,\log q(x).$$

(c) k-fold and voting

In order to use the training data more effectively, we use the k fold method to divide all training data into k folders evenly and evenly distributed. Each time one folder is selected as the test set, the rest are used as the training set, and all the data are run. It can not only diversify the presentation of loss surfaces, but also make full use of all data.

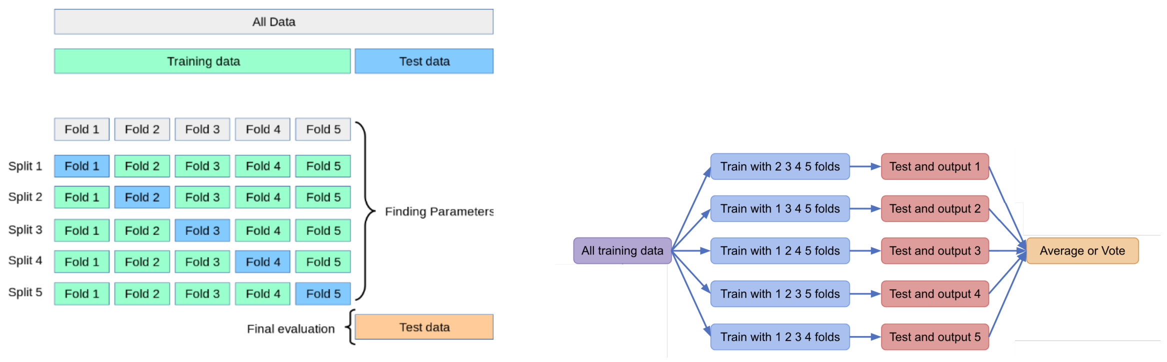


Figure 5: k-fold and voting workflow(the right figure consulted monai work flow)

4. Results In the result, I only show the best method to get the high accuracy in the testing data. Using the k fold cross-validation,

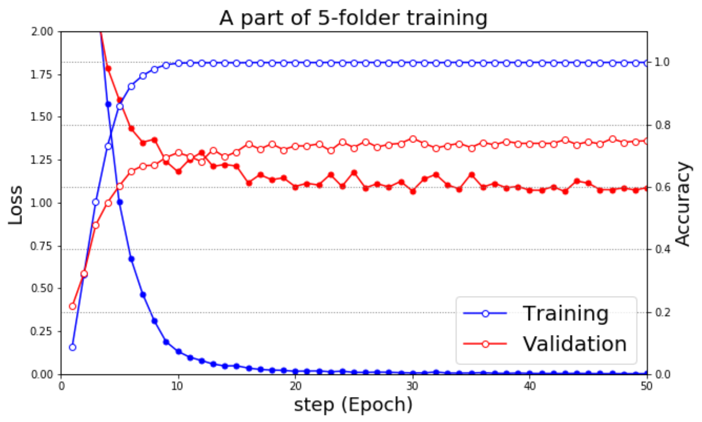
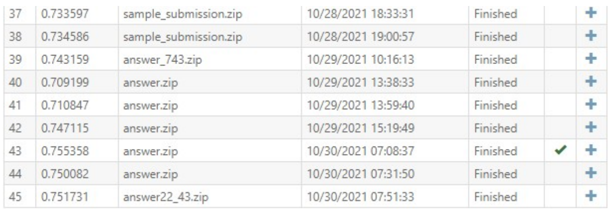


Figure 6: fold5 result
The vertical axis on the left represents loss, and the corresponding line graph is a solid polyline; the vertical axis on the right represents accuracy, and the corresponding line graph is a hollow polyline.

Implement 5-fold, put forward the best results and corresponding epochs, and list them in the bottom table. These models will vote(in the same weight) and submit the results.

k	1	2	3	4	5	mean
epoch	19	17	18	19	30	x
mAP	0.7700	0.7667	0.7467	0.7083	0.7517	0.7487

Voting Result in testing data is 0.75536.



37	0.733597	sample_submission.zip	10/28/2021 18:33:31	Finished		+
38	0.734586	sample_submission.zip	10/28/2021 19:00:57	Finished		+
39	0.743159	answer_743.zip	10/29/2021 10:16:13	Finished		+
40	0.709199	answer.zip	10/29/2021 13:38:33	Finished		+
41	0.710847	answer.zip	10/29/2021 13:59:40	Finished		+
42	0.747115	answer.zip	10/29/2021 15:19:49	Finished		+
43	0.755358	answer.zip	10/30/2021 07:08:37	Finished	✓	+
44	0.750082	answer.zip	10/30/2021 07:31:50	Finished		+
45	0.751731	answer22_43.zip	10/30/2021 07:51:33	Finished		+

Figure 7: Submission

5. Conclusion

The whole process I used stayed in the old method, and found the following problems in the process of implementation. First, the rationality of the use of transform (at the beginning a lot of random additions, but it may not make the results better , Or even worse). Secondly, the choice of optimizer, SGD and Adam are commonly used, but they cannot achieve better results. Try AdamW as much as possible. The loss can continue to decline, and the accuracy of the test set increases. The most important thing is MBS. In the past, I thought it would be as big as it can be. Through the experience of this implementation, it is better to find the most suitable than to use the maximum effect, and this value must be tried many times. Finally, use the power of k-folder, combine all the model capabilities, and use voting technology Correct the deficiencies of the model predictions each other to improve the overall accuracy. In the future, I hope to make adjustments to the loss function in one step, and the model architecture will survey more papers to try.

References

[1] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020.