HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION TECHNOLOGY AND COMMUNICATION

———————o0o———————

PROJECT I REPORT

315 Bird Species Image Classification using Convolutional Neural Netword, Transfer Learning and Image Retrieval

Supervisor: **Ph.D Nguyen Nhat Quang**

Student Name: **Nguyen The Vu**

Student ID: **20194214**

Ha Noi, January 2022

315 Bird Species Image Classification using Convolutional Neural Netword and Image Retrieval

Nguyen The Vu[1]

Hanoi University of Science and Technology, Ha Noi, Viet Nam, vu.nt194214@sis.hust.edu.vn

Abstract. Bird recognition is a hard problem for both expert researcher and non-expert people with academic purpose. The discovery and detection of bird species can contribute to many aspect such as research on climate changing, research on diversity of surrounding environment, spreading of flower pollen in agriculture. However, using classical method like illustration book or remember all the bird species can be extremely hard and exhausting. So that, I proposed to apply machine learning, deep learning and computer vision technique to build a fast, convenient and easily to access system to solve bird classification problem. In this report, I will discuss my approach in solving this problem by using Convolutional Neural Network, Transfer Learning and Image Retrieval. The model we created would be trained on a large dataset of bird image on Kaggle to make sure that my model performs properly.

Keywords: Convolutional Neural Network · Transfer Learning · Image Retrieval · Bird species classification.

1 Introduction

Now a day, diversity in bird patterns has become a significant issue for us to tell which kind of bird ones is. Bird are responsive to changes in sensitive ecosystems so that they help us to recognize different life forms on the earth effectively. Due to great species variation, it is difficult for an ordinary person to recognize the sub-category of a bird only by its appearance. However, annotating all the images by classical method such as using illustration books or by person's expert knowledge is extremely hard and exhausting. Thus, an automatic classification system for bird species will be very convenient and useful for many practical applications. For researchers working outdoors, they can easily, immediately identify and classify known bird species by using the system via device like smartphone. For public, the system could provide much fun and arouse peoples interest in birds and could benefits the protection of birds.

In this project, I have proposed an approach to tackle this problem by training an Convolutional Neural Network which has been built based on the architecture of AlexNet [3], a famous Neural Network won the ILSVRC competition in 2012. I have also used pre-trained ResNet [7] to extract feature vectors of the images and applied that for classification task by adding a fully connected layer. In addition, I have used it as an input vector for Image Retrieval method.

2 Dataset

In this project, I have used the 315 Bird Species - Classification dataset on Kaggle. This dataset consists of 45980 training images, 1575 testing images and 1575 validation images belonging to 315 bird species. All images are 224 x 224 x 3 color images in jpg format. Images were gather from internet searches by species name, cropped and resized so that the bird occupies at least 50% of the pixel in the image. Although the training set is not balanced and have a varying number of images per species, each species has at least 120 training images.

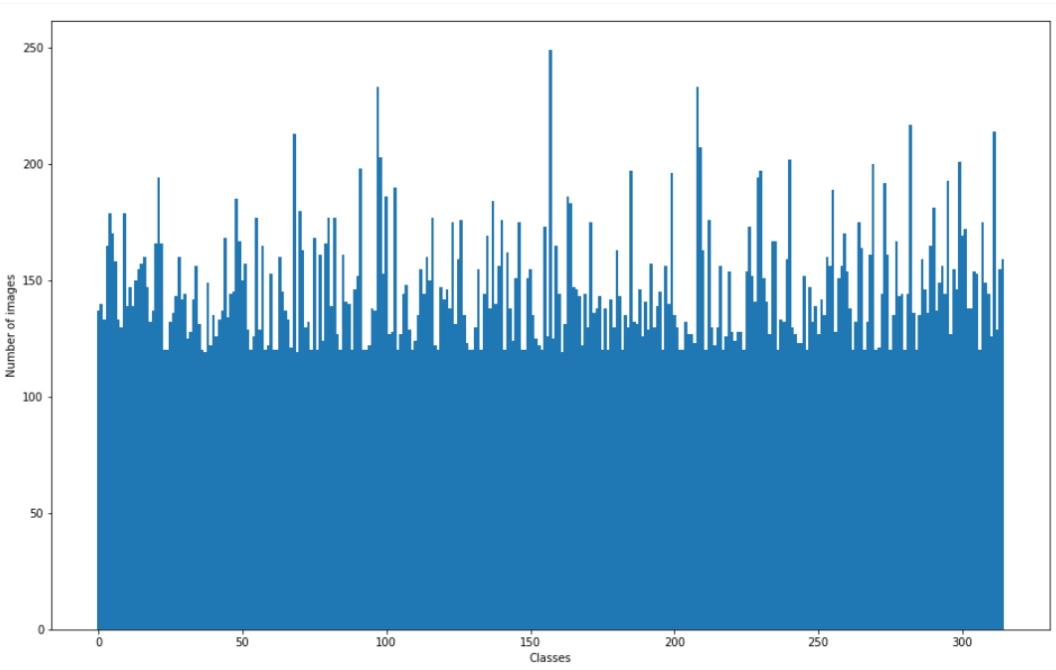


Fig. 1. Distribution of the number of images per label

3 Purposed Approach

3.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial

time series[5]. It was first proposed by Yan Lecun in 1998 for classifying handwriting number and now it is a widely applied deep neural network in the field of computer vision because of these some advantages. The CNN algorithm can avoid the explicit feature extraction, and implicitly to learn from the training data. CNN is also computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universally attractive[2].

3.1.1 Theory

All CNN models follow a similar architecture, which is shown in the figure below.

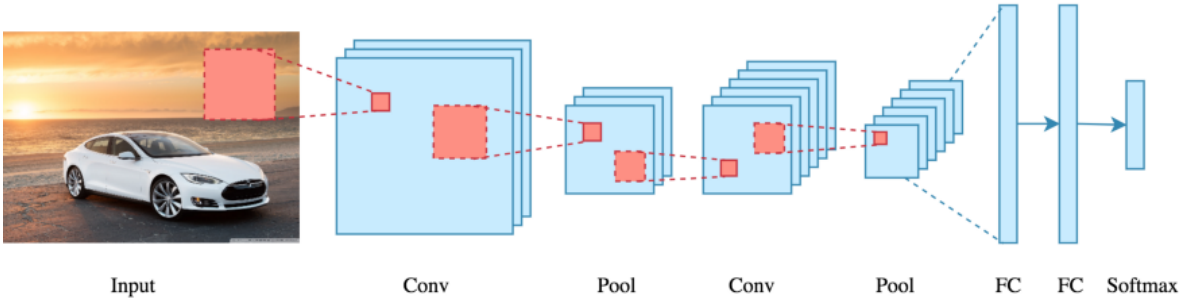


Fig. 2. CNN architecture

As we can see, the CNN involves 3 main layers : Convolutional layer, Pooling layer and Fully Connected layer. We will now dive into each component.

Convolutional layer

Convolution layer is the first layer to extract features from images. Learnable parameters in this layer is a series of filters known as convolutional kernel. Kernel can be understood as a small 2D matrix of intergers with the size of k * k (k is an odd number and usually equal to 3, 5) which is used to establish a relationship of the center pixel with respect to its neighbor pixels. By sliding the kernel horizontally, vertically through the pixels in the images and performing **Convolution operation**, we can produce feature map which contains the extracted features of input images. When sliding kernel, we also need to care about **stride** and **padding**.

Convolution operation

Convolution is a mathematical operation to merge two sets of information. Given an input image I and a filter (kernel) K of dimensions k_1, k_2, the convolution operation is given by:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{1}$$

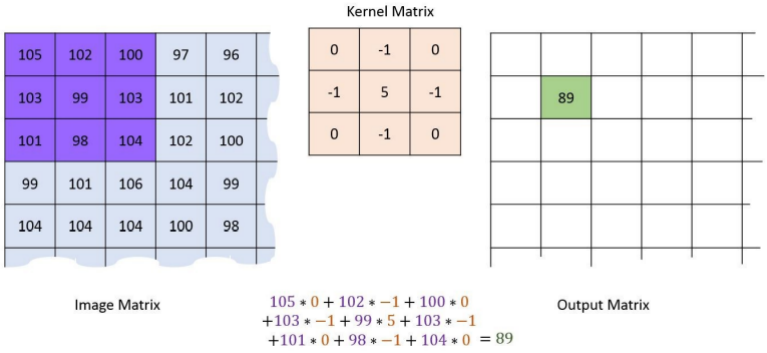


Fig. 3. Convolution operation

Stride

Strides reflect the number of positions by which the kernel matrix shifts after applying a convolution on the middle element. A stride value of 1 indicates that the kernel matrix gets shifted only by 1 pixel in the either direction, right or down.[6]

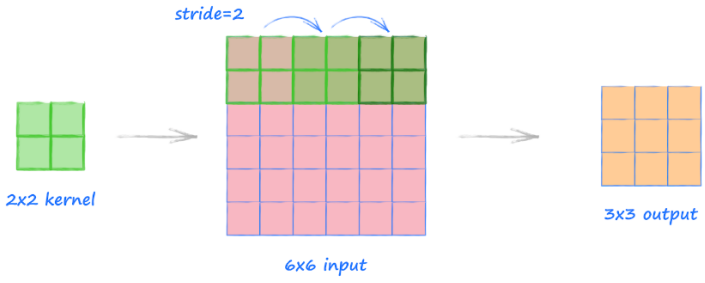


Fig. 4. Affect of stride on movement of the kernel

Padding

To conserve the information in edge pixels and keep the size of feature maps, we have to add a layer of zeros along the edges of the larger matrix/image. This addition of zeros on each side of the matrix is known as padding. If $N \times N$ is the dimension of the kernel matrix, we need to add $\left\lfloor \frac{N}{2} \right\rfloor$ zero layers to the edges. [6]

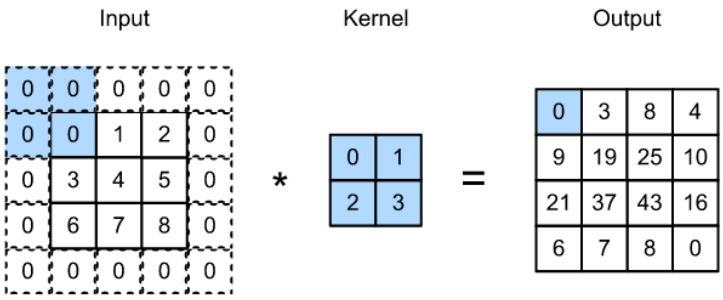


Fig. 5. Input and output when apply padding

Pooling layer

Pooling layer is usually used between convolutional layers for reducing the spatial size but maintain important properties. The dimension reduction decrease the computational power required. In this process, rules of stride and padding are kept the same as above. There are three types of pooling: Max Pooling, Average Pooling and Global Pooling. In this project, I only use Max Pooling so that the output feature map contains the most prominent features of the previous feature map. Max Pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.

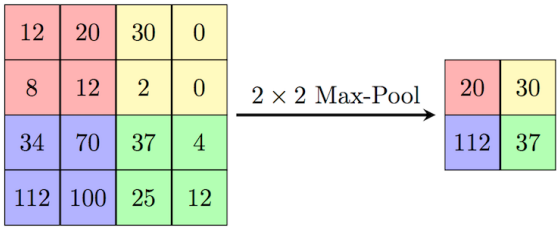


Fig. 6. Max Pooling

Fully Connected Layer

After the convolution + pooling layers we add a couple of fully connected layers to wrap up the CNN architecture. The fully connected layer has the same architecture as ANN and it can learn non-linear combinaions of the high-level features as represented by the output of the convolutional layer. Thus, it is usually used for classification task. The net input is calculated by an linear function:

$$Net = w_0 + \sum_{i=1}^{m} w_i x_i \tag{2}$$

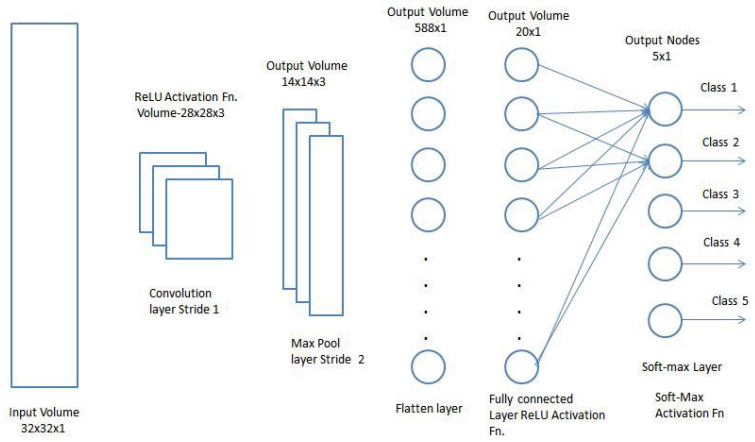
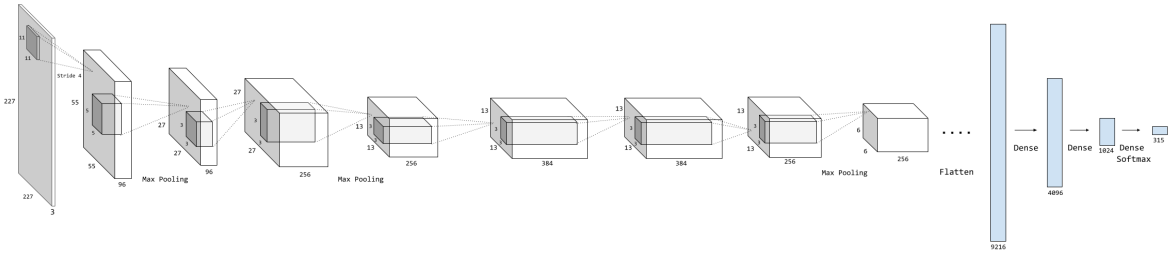


Fig. 7. Fully Connected Layer

3.1.2 Network architecture

As mention before, My Convolutional Neural Network model are built based on the architecture of AlexNet but with some adjustments. The architecture of the model are shown as below :



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 96, 55, 55]          34,944
              ReLU-2          [-1, 96, 55, 55]               0
         MaxPool2d-3          [-1, 96, 27, 27]               0
       BatchNorm2d-4          [-1, 96, 27, 27]             192
            Conv2d-5         [-1, 256, 27, 27]         614,656
              ReLU-6         [-1, 256, 27, 27]               0
         MaxPool2d-7         [-1, 256, 13, 13]               0
       BatchNorm2d-8         [-1, 256, 13, 13]             512
            Conv2d-9         [-1, 384, 13, 13]         885,120
             ReLU-10         [-1, 384, 13, 13]               0
      BatchNorm2d-11         [-1, 384, 13, 13]             768
           Conv2d-12         [-1, 384, 13, 13]       1,327,488
             ReLU-13         [-1, 384, 13, 13]               0
      BatchNorm2d-14         [-1, 384, 13, 13]             768
           Conv2d-15         [-1, 256, 13, 13]         884,992
             ReLU-16         [-1, 256, 13, 13]               0
        MaxPool2d-17           [-1, 256, 6, 6]               0
      BatchNorm2d-18           [-1, 256, 6, 6]             512
          Dropout-19                [-1, 9216]               0
           Linear-20                [-1, 4096]      37,752,832
             ReLU-21                [-1, 4096]               0
          Dropout-22                [-1, 4096]               0
           Linear-23                [-1, 1024]       4,195,328
             ReLU-24                [-1, 1024]               0
           Linear-25                 [-1, 315]         322,875
       LogSoftmax-26                 [-1, 315]               0
================================================================
Total params: 46,020,987
Trainable params: 46,020,987
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 12.96
Params size (MB): 175.56
Estimated Total Size (MB): 189.09
----------------------------------------------------------------
```

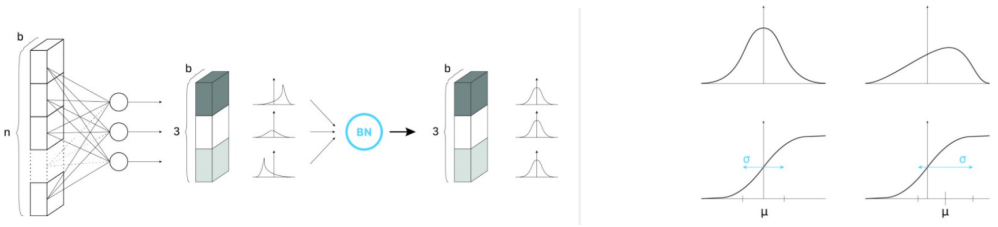
Fig. 8. Network architecture and parameters

To decrease learning time and computational power, I added **Batch Normalization** layers in place of Local Response Normalization in AlexNet. I also add **Dropout** between fully connected layers in order to avoid overfitting. I used **ReLu function** as an activation function in all layers expect the last layer where I used **Softmax function** instead.

Batch Normalization

Batch normalization is a technique for training deep neural network that standardizes the inputs to a layer for each mini-batch. Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. The input of batch norm layer are values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$. The Batch Normalization layer first determines the mean $\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$ and the

variance $\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$ of the activation values across the batch. It then normalizes the activation vector $\widehat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$. That way, each neuron's output follows a standard normal distribution across the batch. It finally calculates the layer's output $\hat{Z}_i$ by applying a linear transformation with this equation $y_i = \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$ where γ allows to adjust the standard deviation and β allows to adjust the bias, shifting the curve on the right or on the left side, these two are learnable parameters[4].

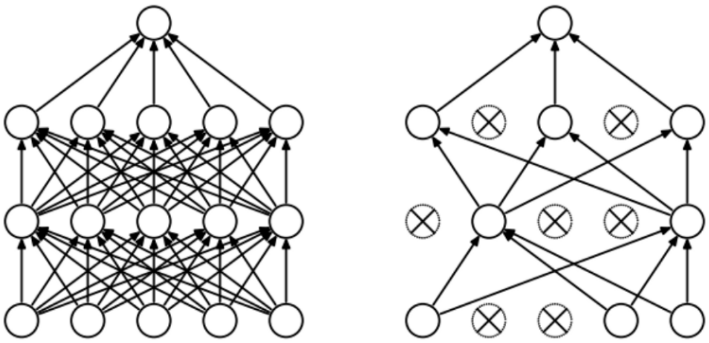


(a) (b)

Fig. 9. (a) Batch Normalization (b) Benefits of γ and β

Dropout

Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.[6]



(a) Standard Neural Net (b) After applying dropout.

Activation function

Relu

The ReLU is the most used activation function in the world right now. The ReLU function and its derivative are both monotonic, which makes it efficient and easy for computation. The ReLU function is shown below:

$$\begin{aligned} &Equation : f(x) = max(0, x) \\ &Range : 0 \rightarrow \inf \end{aligned} \tag{3}$$

Softmax

The softmax function is also a type of sigmoid function but it is very useful to handle classification problems having multiple classes.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4}$$

The softmax function is shown above, where z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z). And again, j indexes the output units, so $j = 1, 2, ..., K$. The softmax function is ideally used in the output layer of the classifier where we are trying to attain the probabilities to define the class of each input[6]. So I use that in output layer for classification task.

3.1.3 Training

I trained the model on my training dataset. I measured the error of my model by calculating **Cross Entropy Loss** between predicted value and actural value. To get predicted value, I forwarded the data through the network and then calculated loss at each layer by using a technique called **backward propagation**. And finally, I use **Adam Optimization algorithm** to update the parameters of the model.

Cross Entropy Loss

Cross Entropy Loss evaluates the performance of a classification model whose output represents the probability which label are true label, its value is between 0 and 1. Cross Entropy Loss increases when the predicted probability diverges from the actural label. In binary classification, Cross Entropy can be calculated as :

$$-\left(y \log(p) + (1 - y) \log(1 - p)\right) \tag{5}$$

In multiclass classification, we calculate the loss separately for each label and sum the result:

$$E = -\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right) \tag{6}$$

Where :

M - number of classes

y - binary indicator if class label c is correct for observation o

p - predicted probability observation o is of class c

Adam Optimization algorithm

Adam Optimization algorithm can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum[1].

Backward Propagation

In order to update the parameters, we need to calculate gradients of the error. In convolutional layer, we have to compute 2 things : the derivative of error with respect to the trainable parameters of the kernels, biases and the derivative of error with respect to the input of the layer so that the previous layer can take it as the derivative of error with respect to its own output and perform the same operations. It is calculated by apply these 3 equation :

$$\begin{aligned} \frac{\partial E}{\partial K_{ij}} &= X_j \star \frac{\partial E}{\partial Y_i} \\ \frac{\partial E}{\partial B_i} &= \frac{\partial E}{\partial Y_i} \\ \frac{\partial E}{\partial X_j} &= \sum_{i=1}^{n} \frac{\partial E}{\partial Y_i} *^{*}_{full} K_{ij} \end{aligned} \tag{7}$$

Where :

E - Error signal

K_{ij} - Parameters k in row i and column j of the kernel K

X_j - Column j of the input matrix X

Y_i - Row i of the output matrix Y

B_i - Row i of the bias matrix B

3.2 Transfer Learning

Transfer learning is a machine learning technique of reusing a developed model for a task for a model on an other task. It is a popular approach in deep learning where pre-trained models which were trained with a big dataset, given a vast compute power and time resources are used as the starting point on related problems so that it can learn more quickly and perform properly.
In this project, I used ResNet18 as the backbone and I only change the last fully connected layer for classification task.

3.3 Image Retrieval

An image retrieval system is a computer system used for browsing, searching and retrieving images from a large database of digital images.[8] To solve the bird classification problem, first I need a method to vectorize the input images and build a gallery of vertorized image. Then to predict the label of query image, I vectorize it and find the k most similar vector in the dataset. In this project, I used pre-trained ResNet18 with the absence of last fully connected layer as a vectorization model and I calculated the **Euclidean distance** between query vector and every vectors in the gallery to get the similarity between the vectors. Finally, I take the k closest vector to the query vector and get the dominant label as the predicted label for query image. The procedure of the method is shown in the picture:

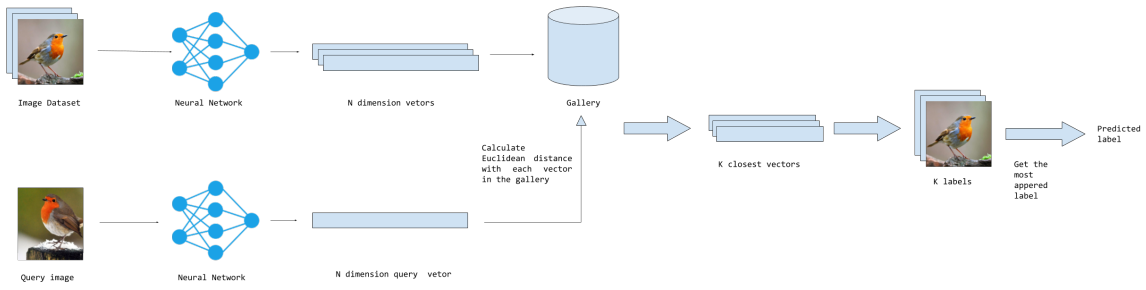


Fig. 10. Image Retrieval method procedure

Euclidean Distance

The distance between 2 n-dimension vectors is calculated by this equation :

$$EuclideanDistance = d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \tag{8}$$

4 Training

Technology and Frameworks used

Progamming Language : Python 3.9

Used Frameworks and Python packages : numpy, matplotlib, pandas, pytorch, torchvision, tensorboard, tqdm, jupyter, PIL, pickle

Environment: Google Colaboratory vitural machine

Hardware

I trained my model on Google Colaboratory vitural machine with Tesla K80 GPU for free user.

Training technique

To avoid overfitting, I reduce learning rate if the evaluation results on validation set weren't better after 2 epochs. In addition, I trained my network in batch with the batch size of 1024 to accelerate training process.

Training process

Scratch Model With my scratch CNN model, I trained it for 30 epochs. It cost about 3600s to train some first epochs and then decreased to about 2700s per epoch. Loss in training and validation accuracy are shown in the graphs below :

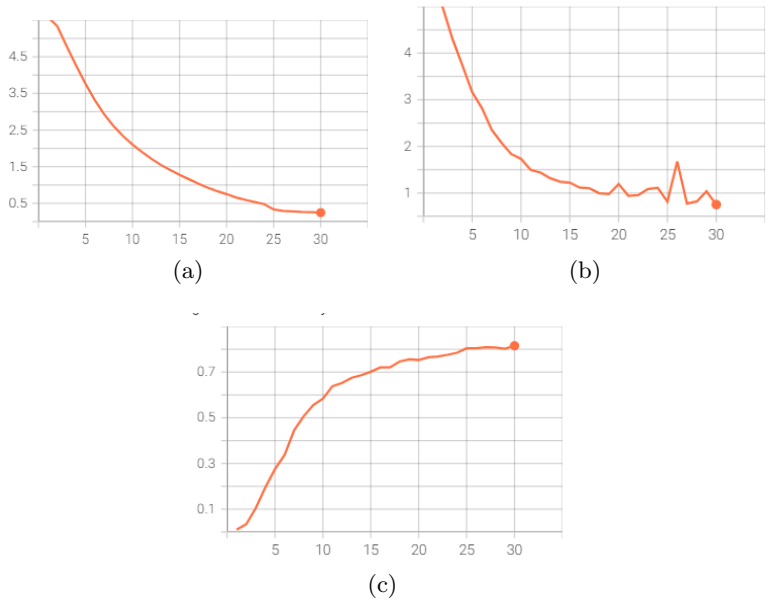


Fig. 11. (a) Training loss (b) Validation loss (c) Validation accuracy

Transfer Learning By using transfer learning, I retrained the ResNet18 model for 2 versions: for version 1, I freezed all convolutional layer and only trained the last fully connected layer; for version 2, I retrained all the network. The time consumption for training these 2 version is 3600s for the first epoch and about 360s for others. The loss of the network converged after 10 epochs of training but the loss of version 2 wasn't stable. Loss in training and validation accuracy are shown in the graphs below :

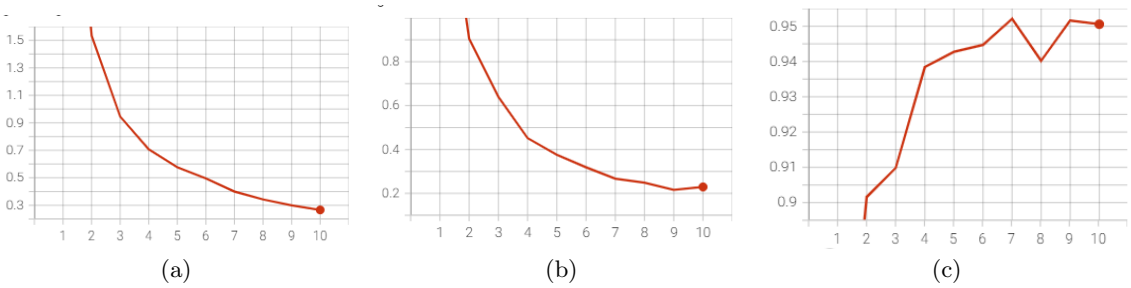


Fig. 12. ResNet Transfer Learning version 1 (a) Training loss (b) Validation loss (c) Validation accuracy

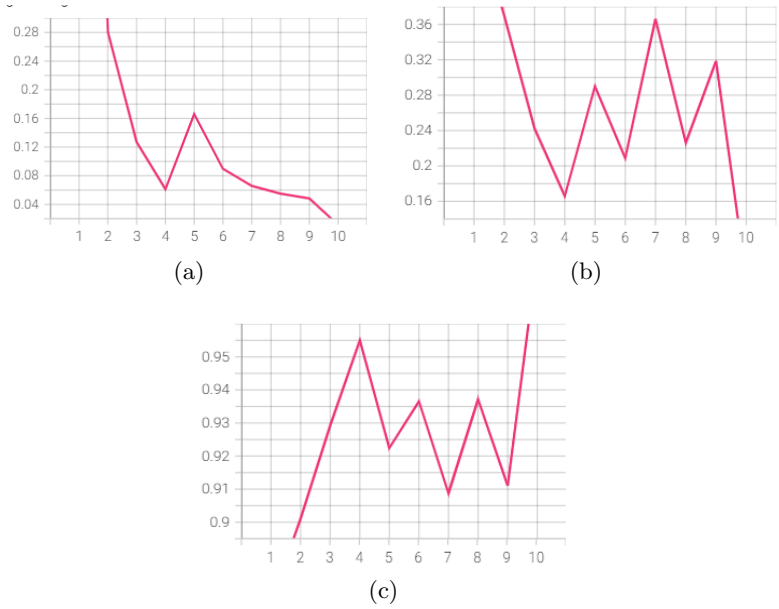


Fig. 13. ResNet Transfer Learning version 2 (a) Training loss (b) Validation loss (c) Validation accuracy

Image Retrieval To build the gallery for image retrieval, I used the second version of ResNet18 trained above. For saving memory, reducing predict time and solving unbalance data problem, I limited the number of images in my gallery to 50 images per label for 315 bird species. In total, there are 15750 vectors of images in my gallery. The time cost to build gallery is about 22 minutes.

5 Results and evaluation

In this project, I used 3 evaluation method to measure my model performance.

Exact Match Ratio

The Exact Match Ratio is defined as :

$$Exact\ Match\ Ratio = \frac{Number\ of\ examples\ with\ exact\ label\ match}{Total\ number\ of\ examples} \tag{9}$$

Classification Report

In classification report, there are 3 evaluation metrics we need to care:

1. Precision: It is the ratio of correctly predicted positive observations to the total observations. It is defined as:

$$\frac{TruePositives}{TruePositives + FalsePositives} \tag{10}$$

2. Recall: It is the ratio of correctly predicted positive observations to all the observations in the relevant class.

$$\frac{TruePositives}{TruePositives + FalseNegatives} \tag{11}$$

3. F1-score: It is the harmonic mean of precision and recall.[6]

$$2 * \frac{precision * recall}{precision + recall} \tag{12}$$

Because of large number of labels in this dataset, I only cared about the average precision, recall and f-1 score.

Confusion Matrix

The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made.

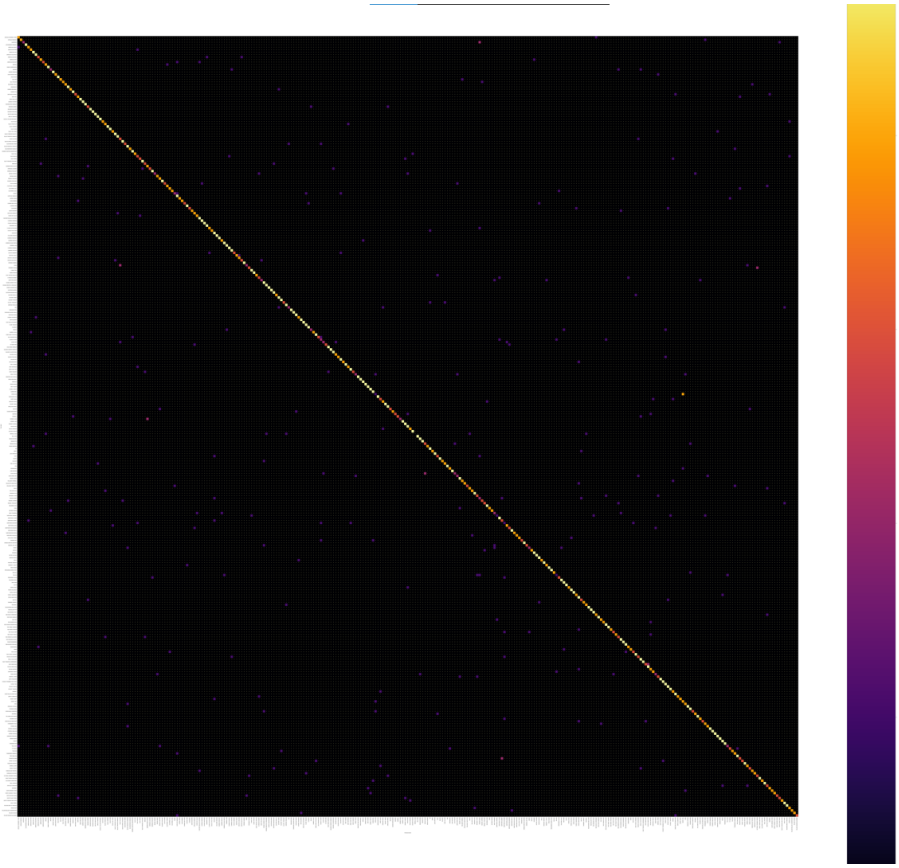
5.1 Scratch Model

With my scratch model, it can guess 82% correct images in testing dataset. The testing result and confusion matrix are shown in the pictures below.

```
******* TESTING RESULT *******
Test loss: 0.789570
Test accuracy: 0.822888 (1291 / 1575)
******************** CLASSIFICATION REPORT *********
              precision    recall  f1-score   support

    accuracy                           0.82      1575
   macro avg       0.84      0.82      0.81      1575
weighted avg       0.84      0.82      0.81      1575
```

(a)



(b)

Fig. 14. (a) Testing result (b) Confusion matrix

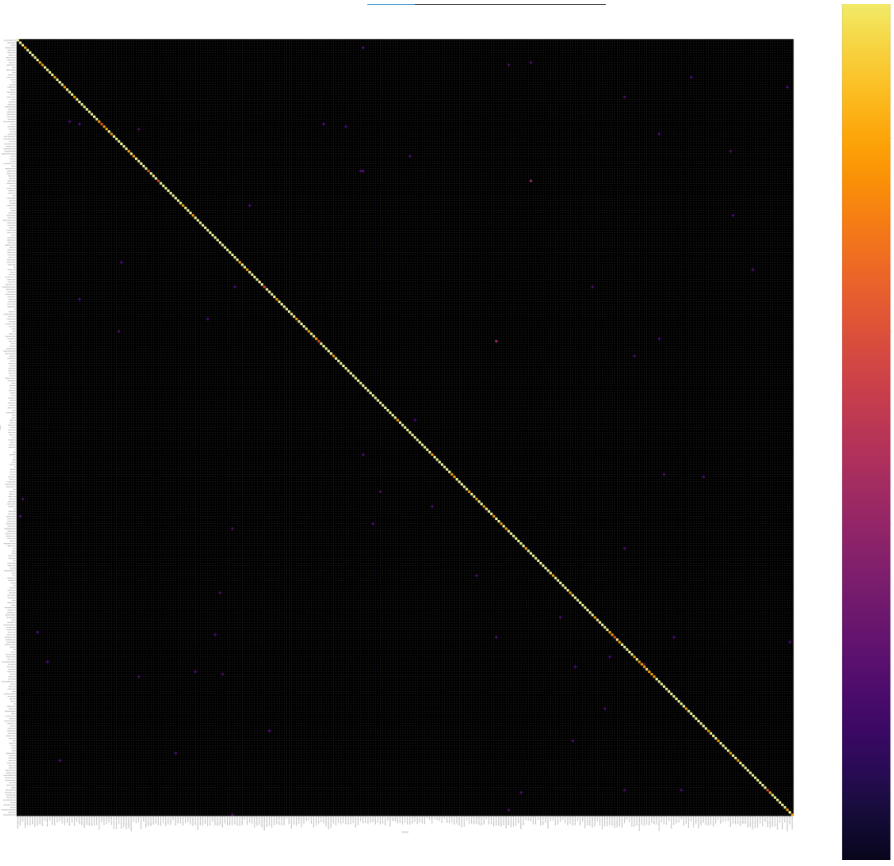
5.2 Transfer Learning model

After testing, I can see that the ResNet that I retrained all the convolutional layer performs better than the ResNet version that I only trained the last fully connected layer. The testing result and confusion matrix are shown in the pictures below.

```
******* TESTING RESULT *******
Test loss: 0.223549
Test accuracy: 0.959507 (1508 / 1575)
******* CLASSIFICATION REPORT *******
              precision    recall  f1-score   support

    accuracy                           0.96      1575
   macro avg       0.97      0.96      0.96      1575
weighted avg       0.97      0.96      0.96      1575
```

(a)



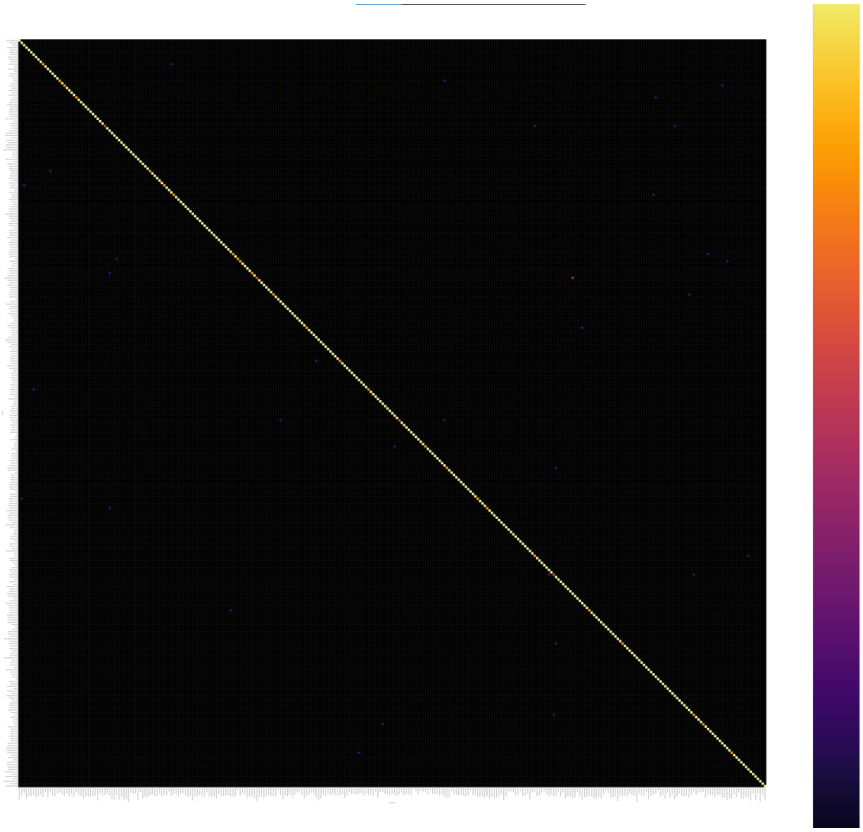
(b)

Fig. 15. (a) Testing result (b) Confusion matrix of Transfer ResNet only trained last fully connected layer

```
******* TESTING RESULT *******
Test loss: 0.057259
Test accuracy: 0.981585 (1542 / 1575)
******* CLASSIFICATION REPORT *******
              precision    recall  f1-score   support

    accuracy                         0.98      1575
   macro avg       0.98      0.98      0.98      1575
weighted avg       0.98      0.98      0.98      1575
```

(a)

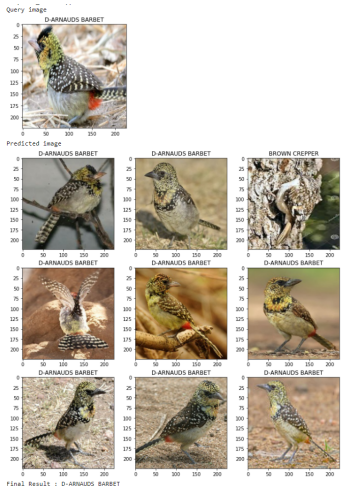


(b)

Fig. 16. (a) Testing result (b) Confusion matrix of Transfer ResNet retrained all the network

5.3 Image Retrieval

In this method, I set my k which is the hyper parameter of selecting k most similar vectors to query vector for getting prediction equal to 5. The pictures below displays how my system shows 9 most similar images to query image and evaluation result.



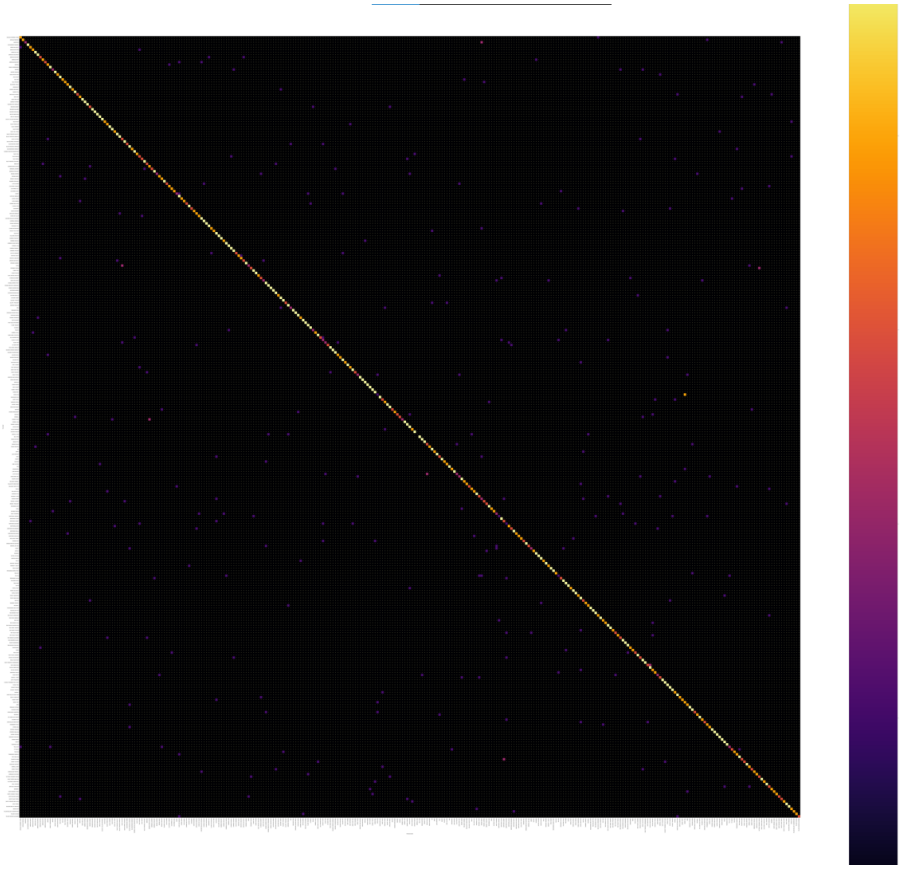
(a)

```
********** TESTING RESULT ***********
Test accuracy: 0.945397 (1489 / 1575)

******************** CLASSIFICATION REPORT *********
```

	precision	recall	f1-score	support
accuracy			0.95	1575
macro avg	0.95	0.95	0.94	1575
weighted avg	0.95	0.95	0.94	1575

(b)



(c)

Fig. 17. (a) Get retrieve images (b) Testing result (c) Confusion matrix

6 Conclusion

In this project, I have proposed 3 method to classify the species of the bird from high-definition camera photos by using CNN, transfer learning and image retrieval. The testing result shows that the model using transfer learning achieved the highest overall F1 score equal to 98%. In the end, this project can help researcher easily identify known birds and contribute to bird preservation by arousing people interest of bird. However, my system not yet have a user interface, which can be hard to upload a photo of bird and get the name of the species. Besides, this model have only trained for 315 of bird species whereas there are thousand of bird types out there. I need to update frequently to get in keep to the various change in bird categories.
Through this project, I have learned how to solve a basic computer vision problem which is image classification problem. I know how to use CNN, transfer learning and the method to find an image in a big dataset.

7 Future work

1) Find a way to solve the unbalance data problem.
2) Apply basic Computer Vision technique such as filter, Object Detection, Segmentation for the model to work well on a hard or bad photographs.
3) Build UX/UI so user can easily access and use the system.

References

[1] Vitaly Bushaev. “Adam — latest trends in deep learning optimization”. In: ().
[2] Arden Dertat. “Applied Deep Learning - Part 4: Convolutional Neural Networks”. In: (2017).
[3] Alex Krizhevsky & Ilya Sutskever & Geoffrey E. Hinton. “ImageNet Classific, n with Deep Convolutional Neural Networks”. In: (2012).
[4] Johann Huber. “Batch normalization in 3 levels of understanding”. In: ().
[5] Diego Junco. “Introduction to Convolutional Neural Networks”. In: (2021).
[6] B V SATYA SAI & S APUROOP & S MOUNIKA & D SAI KUMAR. “Identification of Bird Species using Convolution Neural Networks”. In: (2020).
[7] Kaiming He & Xiangyu Zhang & Shaoqing Ren & Jian Sun. “Deep Residual Learning for Image Recognition”. In: (2015).
[8] Wikipedia. “Image retrieval”. In: ().