# GNR638 Machine Learning for Remote Sensing II

## Mini Project 1


Group Members:

- Aryan Adinath Popalghat (210020088)
- Divyam Gupta (210020044)
- Durgesh Sahane (210110096)



March 6, 2024

Indian Institute of Technology, Bombay

# 1. Introduction

In this project, we aim to develop a model for image classification on the CUB-200-2011 dataset. The CUB-200-2011 dataset is a widely used benchmark dataset containing images of 200 bird species, with a total of 11,788 images. Each image is annotated with the corresponding bird species label.

One of the key challenges in building deep learning models is balancing model complexity and performance. While deep neural networks are capable of achieving state-of-the-art results, they often come with a large number of parameters, leading to increased computational and memory requirements. To address this challenge, we impose a constraint on our model to have **less than 10 million parameters** while still achieving competitive performance on the classification task.

Our approach involves designing and training a convolutional neural network (CNN) architecture tailored to the characteristics of the CUB-200-2011 dataset. We explore various architectural choices, including the number of layers, filter sizes, and pooling strategies, to strike a balance between model complexity and performance. Additionally, we employ techniques such as regularization and optimization to enhance the generalization ability of our model and prevent overfitting.

The success of our project has implications beyond the specific task of bird classification. By demonstrating the feasibility of building a high-performing model with limited parameters, we contribute to the broader goal of developing efficient and scalable deep learning solutions for real-world applications.

In this report, we provide a detailed overview of our methodology, experimental setup, results, and analysis. We also discuss the implications of our findings and suggest directions for future research in the field of efficient deep learning models for image classification.

# 2. Data Preprocessing

The success of any machine learning model heavily relies on the quality and suitability of the dataset used for training. In this section, we outline the preprocessing steps applied to the CUB-200-2011 dataset to ensure its compatibility with our classification task and to enhance the performance and robustness of our model.

1. **Dataset Overview:**
   The CUB-200-2011 dataset consists of a collection of bird images, each associated with a specific species label. Before proceeding with preprocessing, we performed an initial exploration of the dataset to understand its structure, distribution of classes, and image characteristics.
2. **Data Augmentation:**
   To increase the diversity and variability of our training data and to prevent overfitting, we applied data augmentation techniques. These techniques include random rotation, horizontal flipping, and random cropping. Data augmentation helps the model learn invariant features and improves its generalization ability.
3. **Image Rescaling and Normalization:**
   All images in the dataset were rescaled to a fixed size to ensure uniformity across the dataset and to facilitate efficient training. Additionally, we performed pixel normalization to standardize the pixel values of the images, typically by subtracting the mean and dividing by the standard deviation. Normalization helps stabilize the training process and accelerates convergence.
4. **Data Splitting:**
   We partitioned the dataset into training, validation, and test sets. The training set is used to train the model, and the test set is used to evaluate the final model performance. We ensured that the distribution of classes is balanced across all sets to avoid biased model evaluation.

5. **Data loading and Batching:**
   We utilized PyTorch's data loading utilities to efficiently load and preprocess the dataset. Data loaders were employed to generate mini-batches of data during training, allowing for parallel processing and memory optimization.

# 3. Model

We employed the ResNet-18 architecture as our baseline model and customized it to meet our project's constraints of less than 10 million parameters. The ResNet-18 architecture consists of a series of convolutional layers with residual connections, facilitating deeper networks while mitigating the vanishing gradient problem. To meet our parameter constraint, we reduced the number of filters and depth of the network, optimized regularization techniques, and carefully selected hyperparameters. This modified architecture retains the essence of ResNet-18 while ensuring computational efficiency and competitive performance on the CUB-200-2011 dataset.

1. **Architecture:**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| **Conv2d-1** | **[-1, 32, 112, 112]** | **4,704** |
| **BatchNorm2d-2** | **[-1, 32, 112, 112]** | **64** |
| **ReLU-3** | **[-1, 32, 112, 112]** | **0** |
| **MaxPool2d-4** | **[-1, 32, 56, 56]** | **0** |
| **Conv2d-5** | **[-1, 32, 56, 56]** | **9,216** |
| **BatchNorm2d-6** | **[-1, 32, 56, 56]** | **64** |
| **ReLU-7** | **[-1, 32, 56, 56]** | **0** |
| **Conv2d-8** | **[-1, 32, 56, 56]** | **9,216** |
| **BatchNorm2d-9** | **[-1, 32, 56, 56]** | **64** |
| **ReLU-10** | **[-1, 32, 56, 56]** | **0** |

| | | |
|---|---|---|
| ResidualBlock-11 | [-1, 32, 56, 56] | 0 |
| Conv2d-12 | [-1, 32, 56, 56] | 9,216 |
| BatchNorm2d-13 | [-1, 32, 56, 56] | 64 |
| ReLU-14 | [-1, 32, 56, 56] | 0 |
| Conv2d-15 | [-1, 32, 56, 56] | 9,216 |
| BatchNorm2d-16 | [-1, 32, 56, 56] | 64 |
| ReLU-17 | [-1, 32, 56, 56] | 0 |
| ResidualBlock-18 | [-1, 32, 56, 56] | 0 |
| Conv2d-19 | [-1, 64, 28, 28] | 18,432 |
| BatchNorm2d-20 | [-1, 64, 28, 28] | 128 |
| ReLU-21 | [-1, 64, 28, 28] | 0 |
| Conv2d-22 | [-1, 64, 28, 28] | 36,864 |
| BatchNorm2d-23 | [-1, 64, 28, 28] | 128 |
| Conv2d-24 | [-1, 64, 28, 28] | 2,048 |
| BatchNorm2d-25 | [-1, 64, 28, 28] | 128 |
| ReLU-26 | [-1, 64, 28, 28] | 0 |
| ResidualBlock-27 | [-1, 64, 28, 28] | 0 |
| Conv2d-28 | [-1, 64, 28, 28] | 36,864 |
| BatchNorm2d-29 | [-1, 64, 28, 28] | 128 |
| ReLU-30 | [-1, 64, 28, 28] | 0 |
| Conv2d-31 | [-1, 64, 28, 28] | 36,864 |
| BatchNorm2d-32 | [-1, 64, 28, 28] | 128 |
| ReLU-33 | [-1, 64, 28, 28] | 0 |
| ResidualBlock-34 | [-1, 64, 28, 28] | 0 |
| Conv2d-35 | [-1, 128, 14, 14] | 73,728 |
| BatchNorm2d-36 | [-1, 128, 14, 14] | 256 |
| ReLU-37 | [-1, 128, 14, 14] | 0 |
| Conv2d-38 | [-1, 128, 14, 14] | 147,456 |
| BatchNorm2d-39 | [-1, 128, 14, 14] | 256 |
| Conv2d-40 | [-1, 128, 14, 14] | 8,192 |
| BatchNorm2d-41 | [-1, 128, 14, 14] | 256 |
| ReLU-42 | [-1, 128, 14, 14] | 0 |
| ResidualBlock-43 | [-1, 128, 14, 14] | 0 |

| | | |
|---|---|---|
| Conv2d-44 | [-1, 128, 14, 14] | 147,456 |
| BatchNorm2d-45 | [-1, 128, 14, 14] | 256 |
| ReLU-46 | [-1, 128, 14, 14] | 0 |
| Conv2d-47 | [-1, 128, 14, 14] | 147,456 |
| BatchNorm2d-48 | [-1, 128, 14, 14] | 256 |
| ReLU-49 | [-1, 128, 14, 14] | 0 |
| ResidualBlock-50 | [-1, 128, 14, 14] | 0 |
| Conv2d-51 | [-1, 256, 7, 7] | 294,912 |
| BatchNorm2d-52 | [-1, 256, 7, 7] | 512 |
| ReLU-53 | [-1, 256, 7, 7] | 0 |
| Conv2d-54 | [-1, 256, 7, 7] | 589,824 |
| BatchNorm2d-55 | [-1, 256, 7, 7] | 512 |
| Conv2d-56 | [-1, 256, 7, 7] | 32,768 |
| BatchNorm2d-57 | [-1, 256, 7, 7] | 512 |
| ReLU-58 | [-1, 256, 7, 7] | 0 |
| ResidualBlock-59 | [-1, 256, 7, 7] | 0 |
| Conv2d-60 | [-1, 256, 7, 7] | 589,824 |
| BatchNorm2d-61 | [-1, 256, 7, 7] | 512 |
| ReLU-62 | [-1, 256, 7, 7] | 0 |
| Conv2d-63 | [-1, 256, 7, 7] | 589,824 |
| BatchNorm2d-64 | [-1, 256, 7, 7] | 512 |
| ReLU-65 | [-1, 256, 7, 7] | 0 |
| ResidualBlock-66 | [-1, 256, 7, 7] | 0 |
| AdaptiveAvgPool2d-67 | [-1, 256, 1, 1] | 0 |
| Linear-68 | [-1, 200] | 51,400 |

================================================================

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 112, 112]           4,704
       BatchNorm2d-2         [-1, 32, 112, 112]              64
              ReLU-3         [-1, 32, 112, 112]               0
         MaxPool2d-4           [-1, 32, 56, 56]               0
            Conv2d-5           [-1, 32, 56, 56]           9,216
       BatchNorm2d-6           [-1, 32, 56, 56]              64
              ReLU-7           [-1, 32, 56, 56]               0
            Conv2d-8           [-1, 32, 56, 56]           9,216
       BatchNorm2d-9           [-1, 32, 56, 56]              64
             ReLU-10           [-1, 32, 56, 56]               0
    ResidualBlock-11           [-1, 32, 56, 56]               0
           Conv2d-12           [-1, 32, 56, 56]           9,216
      BatchNorm2d-13           [-1, 32, 56, 56]              64
             ReLU-14           [-1, 32, 56, 56]               0
           Conv2d-15           [-1, 32, 56, 56]           9,216
      BatchNorm2d-16           [-1, 32, 56, 56]              64
             ReLU-17           [-1, 32, 56, 56]               0
    ResidualBlock-18           [-1, 32, 56, 56]               0
           Conv2d-19           [-1, 64, 28, 28]          18,432
      BatchNorm2d-20           [-1, 64, 28, 28]             128
             ReLU-21           [-1, 64, 28, 28]               0
           Conv2d-22           [-1, 64, 28, 28]          36,864
      BatchNorm2d-23           [-1, 64, 28, 28]             128
           Conv2d-24           [-1, 64, 28, 28]           2,048
      BatchNorm2d-25           [-1, 64, 28, 28]             128
             ReLU-26           [-1, 64, 28, 28]               0
    ResidualBlock-27           [-1, 64, 28, 28]               0
           Conv2d-28           [-1, 64, 28, 28]          36,864
      BatchNorm2d-29           [-1, 64, 28, 28]             128
             ReLU-30           [-1, 64, 28, 28]               0
           Conv2d-31           [-1, 64, 28, 28]          36,864
      BatchNorm2d-32           [-1, 64, 28, 28]             128
             ReLU-33           [-1, 64, 28, 28]               0
    ResidualBlock-34           [-1, 64, 28, 28]               0
           Conv2d-35          [-1, 128, 14, 14]          73,728
      BatchNorm2d-36          [-1, 128, 14, 14]             256
             ReLU-37          [-1, 128, 14, 14]               0
```

```
            Conv2d-38         [-1, 128, 14, 14]         147,456
       BatchNorm2d-39         [-1, 128, 14, 14]             256
            Conv2d-40         [-1, 128, 14, 14]           8,192
       BatchNorm2d-41         [-1, 128, 14, 14]             256
              ReLU-42         [-1, 128, 14, 14]               0
     ResidualBlock-43         [-1, 128, 14, 14]               0
            Conv2d-44         [-1, 128, 14, 14]         147,456
       BatchNorm2d-45         [-1, 128, 14, 14]             256
              ReLU-46         [-1, 128, 14, 14]               0
            Conv2d-47         [-1, 128, 14, 14]         147,456
       BatchNorm2d-48         [-1, 128, 14, 14]             256
              ReLU-49         [-1, 128, 14, 14]               0
     ResidualBlock-50         [-1, 128, 14, 14]               0
            Conv2d-51           [-1, 256, 7, 7]         294,912
       BatchNorm2d-52           [-1, 256, 7, 7]             512
              ReLU-53           [-1, 256, 7, 7]               0
            Conv2d-54           [-1, 256, 7, 7]         589,824
       BatchNorm2d-55           [-1, 256, 7, 7]             512
            Conv2d-56           [-1, 256, 7, 7]          32,768
       BatchNorm2d-57           [-1, 256, 7, 7]             512
              ReLU-58           [-1, 256, 7, 7]               0
     ResidualBlock-59           [-1, 256, 7, 7]               0
            Conv2d-60           [-1, 256, 7, 7]         589,824
       BatchNorm2d-61           [-1, 256, 7, 7]             512
              ReLU-62           [-1, 256, 7, 7]               0
            Conv2d-63           [-1, 256, 7, 7]         589,824
       BatchNorm2d-64           [-1, 256, 7, 7]             512
              ReLU-65           [-1, 256, 7, 7]               0
     ResidualBlock-66           [-1, 256, 7, 7]               0
 AdaptiveAvgPool2d-67           [-1, 256, 1, 1]               0
            Linear-68                  [-1, 200]          51,400
================================================================
```

2. **Parameters:**

================================================================

Total params: 2,850,280
Trainable params: 2,850,280
Non-trainable params: 0

----------------------------------------------------------------

Input size (MB): 0.57
Forward/backward pass size (MB): 31.39

**Params size (MB): 10.87**
**Estimated Total Size (MB): 42.84**

**----------------------------------------------------------------**

```
================================================================
Total params: 2,850,280
Trainable params: 2,850,280
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 31.39
Params size (MB): 10.87
Estimated Total Size (MB): 42.84
----------------------------------------------------------------
```

# 4. Training

## 1. Hyperparameters:

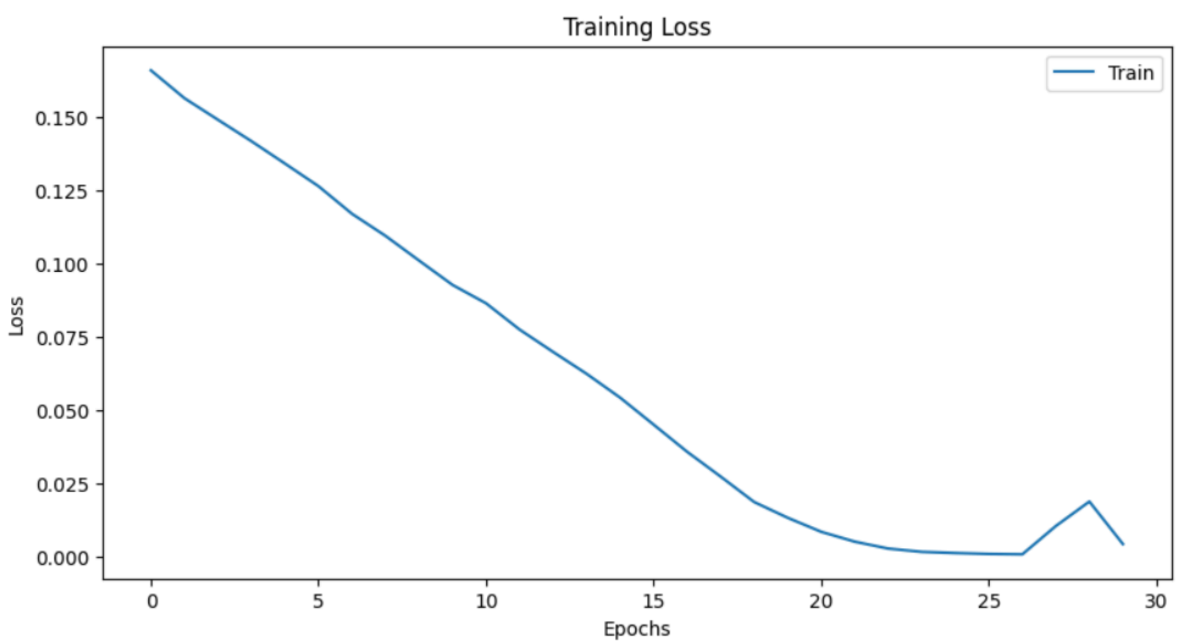
- Number of classes: 200
- Batch size: 32
- Number of epochs: 30

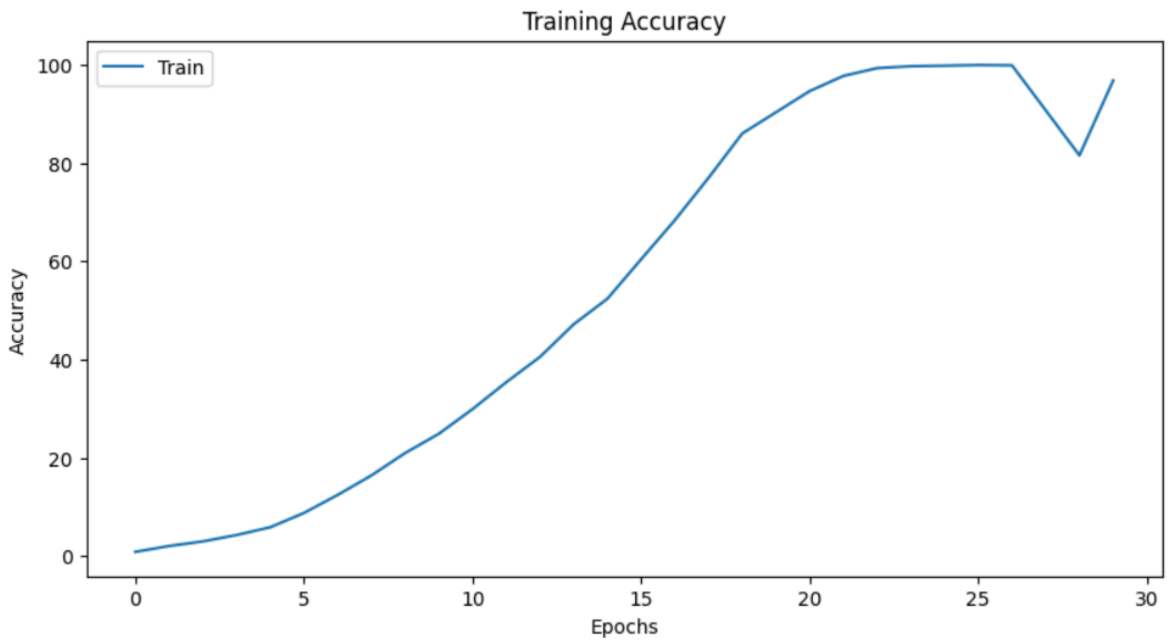
## 2. Training Cost and Accuracy:

**`Performance Summary`**

| Epoch | Training Loss | Training Accuracy |
|---|---|---|
| 1 | 0.16568245897195996 | 0.8908371040723982 |
| 2 | 0.15619492092553308 | 2.0786199095022626 |
| 3 | 0.14885187526633836 | 3.0118778280542986 |
| 4 | 0.1415379787057773 | 4.3127828054298645 |
| 5 | 0.1339182592422714 | 5.896493212669683 |
| 6 | 0.12621298476177104 | 8.79524886877828 |
| 7 | 0.1168067765114534 | 12.471719457013574 |
| 8 | 0.10929018815313529 | 16.43099547511312 |
| 9 | 0.10088828418454433 | 20.9841628959276 |
| 10 | 0.09257651808170173 | 24.90101809954751 |
| 11 | 0.08631974596917899 | 29.920814479638008 |
| 12 | 0.07737295437452481 | 35.39309954751131 |

| | | |
|---|---|---|
| 13 | 0.06975575612343814 | 40.540158371040725 |
| 14 | 0.06229199597199039 | 47.171945701357465 |
| 15 | 0.054177477770396484 | 52.37556561085973 |
| 16 | 0.04498270105958255 | 60.39309954751131 |
| 17 | 0.03572620590516615 | 68.35407239819004 |
| 18 | 0.027278319380469453 | 76.96549773755656 |
| 19 | 0.018614862843235425 | 85.98699095022624 |
| 20 | 0.013246690892357362 | 90.2997737556561 |
| 21 | 0.008470857512770533 | 94.61255656108597 |
| 22 | 0.0051170337799835396 | 97.69513574660634 |
| 23 | 0.0027559062232141176 | 99.27884615384616 |
| 24 | 0.0016490024621429857 | 99.6747737556561 |
| 25 | 0.0012457637683640137 | 99.77375565610859 |
| 26 | 0.0009343491161844748 | 99.90101809954751 |
| 27 | 0.0008055211423592836 | 99.84445701357465 |
| 28 | 0.010445750839702735 | 90.69570135746606 |
| 29 | 0.01878154013154194 | 81.53280542986425 |
| 30 | 0.004308475319473115 | 96.74773755656109 |






# 5. Results

1. **Training Accuracy:** Accuracy of the network on the train images: 99.43438914027149%
2. **Test Accuracy:** Accuracy of the network on the test images: 26.950805767599966%
3. Got 7066 correct images out of 7072 total images (99.92% accuracy) (For training data)
4. Got 1351 correct images out of 4716 total images (28.65% accuracy) (For testing data)

Note: The NewModel12.ipynb submitted in the folder is before addition of checkpoints in it. Because checkpoint calculation is later done on a different device.