EE691 - RnD Project

SELF-SUPERVISED LEARNING FOR FINE-GRAINED IMAGE CLASSIFICATION (Dataset Used: CUB-200-2011)

GITHUB REPO:

https://github.com/Disha21-mithoo/EE691_RnD_codefiles_ssl.git

Submitted By:

DISHA PABRI (20D070027)

Under Professor:

Dr. BIPLAB BANERJEE

1 INTRODUCTION:

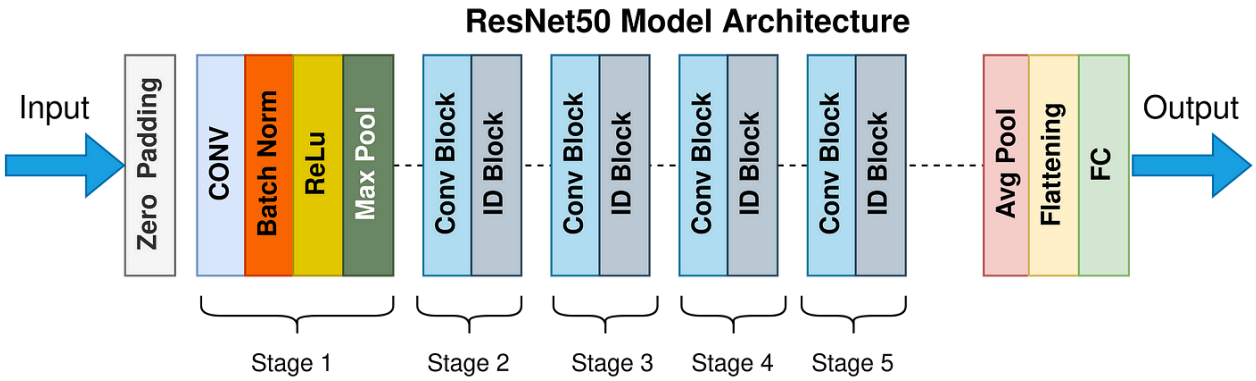
Self-supervised learning (SSL) has emerged as a promising paradigm in machine learning, allowing models to learn useful representations from unlabeled data. Unlike supervised learning, which requires labeled examples for training, SSL leverages the inherent structure or characteristics of the data itself to generate supervisory signals. This approach not only reduces the reliance on expensive labeled datasets but also enables models to learn more generalized features, thus improving performance on downstream tasks.

The **Caltech-UCSD Birds (CUB)** dataset stands as a quintessential benchmark in this domain, comprising a diverse collection of bird species with fine-grained annotations. Recognizing birds from images poses a significant classification task due to variations in pose, lighting conditions, and background clutter, making it an ideal candidate for exploring SSL techniques. It consists of a collection of **11,000 images** belonging to **200 different bird species**, with each species containing approximately 30 images.

This report delves into the realm of **SSL** for image classification, focusing specifically on the application of two prominent SSL auxiliary tasks: **Deconstruction and Construction Learning (DCL)** and **Pretext Invariant Representation Learning (PIRL)**. Through an empirical investigation on the CUB dataset, we aim to compare the efficacy of these methods in generating meaningful representations and improving classification accuracy. By harnessing the power of SSL, we endeavor to unlock novel insights and advancements in the field of computer vision research.

2 METHODS AND IMPLEMENTATIONS:

Feature Extraction:
To extract features from the images, I utilized a pretrained ResNet-50 model obtained from TensorFlow Hub. These features were then stored as numpy arrays for further processing.



Model Architecture:
I designed a custom PyTorch model called `TorchVisionSSLPIRL` for self-supervised learning. The model consisted of three main components:

- A feature head (head) for processing the extracted features.
- A jigsaw head (head_jig) for predicting representations used in jigsaw puzzle solving.
- A classification head (classification_head) for predicting class labels.

```
TorchVisionSSLPIRL(
  (head): Sequential(
    (0): Linear(in_features=2048, out_features=2048, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=2048, out_features=128, bias=True)
    (3): Normalize()
  )
  (head_jig): JigsawHead(
    (fc1): Sequential(
      (0): Linear(in_features=2048, out_features=256, bias=True)
      (1): ReLU(inplace=True)
    )
    (fc2): Linear(in_features=256, out_features=128, bias=True)
    (l2norm): Normalize()
  )
  (classification_head): Linear(in_features=128, out_features=200, bias=True)
)
```

Training Loop:
The model was trained using a stochastic gradient descent (SGD) optimizer. During each epoch, both contrastive loss and classification loss were calculated and back propagated to update the model's parameters. Training loss and accuracy were monitored and printed for each epoch.

AUXILIARY TASK TECHNIQUES:
The project explores two different self supervised auxiliary tasks for the fine-grained classification problem: Pretext invariant representation learning (PIRL) and Deconstruction and Construction Learning (DCL). The project uses a parallel pipeline for all the SSL implementations, where each model is trained end-end with a combination of classification loss and SSL loss.

2.1 Deconstruction and Construction Learning (DCL):

A self-supervised learning (SSL) technique that operates by systematically deconstructing images into uniform patches and rearranging them in a jigsaw puzzle-like pattern. This method employs a region confusion mechanism to shuffle the patches, thereby inducing the model to learn meaningful representations based on the spatial relationships between different regions of the image.

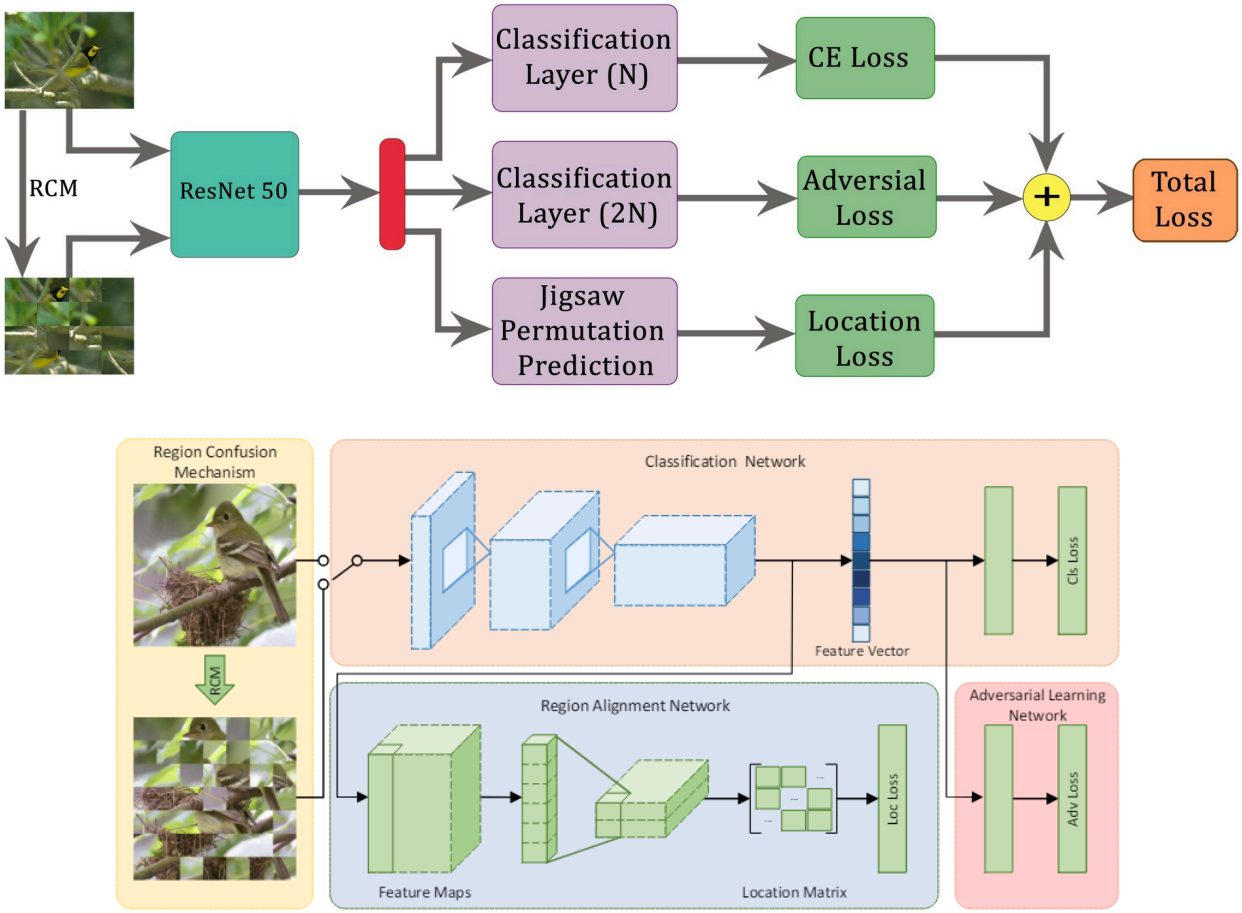


Figure 2. The framework of the proposed DCL method which consists of four parts. (1) Region Confusion Mechanism: a module to shuffle the local regions of the input image. (2) Classification Network: the backbone classification network that classifies images into fine-grained categories. (3) Adversarial Learning Network: an adversarial loss is applied to distinguish original images from destructed ones. (4) Region Alignment Network: appended after the classification network to recover the spatial layout of local regions.

2.1.1 Destruction Learning:

2.1.1.1 Region Confusion Mechanism:

Our proposed Region Confusion Mechanism (RCM) is designed to disrupt the spatial layout of local image regions. Given an input image I, we first uniformly partition the image into N × N sub-regions denoted by $R_{i,j}$, where i and j are the horizontal and vertical indices respectively and $1 \leq i, j \leq N$.

For the j th row of R, a random vector q_j of size N is generated, where the i th element $q_{j,i} = i + r$, where $r \sim U(-k, k)$ is a random variable following a uniform distribution in the range of $[-k, k]$. Here, k is a tunable parameter ($1 \leq k < N$) defining the neighborhood range. Then we can get a new permutation σ row j of regions in j th row by sorting the array q_j , verifying the condition:

$$\forall i \in \{1, ..., N\}, \left|\sigma_j^{row}(i) - i\right| < 2k. \quad (1)$$

Similarly, we apply the permutation σ_i^{col} to the regions column-wisely, verifying the condition:

$$\forall j \in \{1, ..., N\}, \left|\sigma_i^{col}(j) - j\right| < 2k. \quad (2)$$

Therefore, the region at (i, j) in original region location is placed to a new coordinate:

$$\sigma(i, j) = (\sigma_j^{row}(i), \sigma_i^{col}(j)). \quad (3)$$

Figure 3. Example images for fine-grained recognition (top) and the corresponding "destructed" images by our proposed RCM (bottom).

2.1.1.2 Adversarial Learning :

Considering the original images and the destructed ones as two domains, the adversarial loss and classification loss work in an adversarial manner to 1) keep domain-invariant patterns, and 2) reject domain-specific patterns between I and φ(I).

$$D(I, \theta_{adv}) = \text{softmax}(\theta_{adv} C(I, \theta_{cls}^{[1,m]})), \quad (5)$$

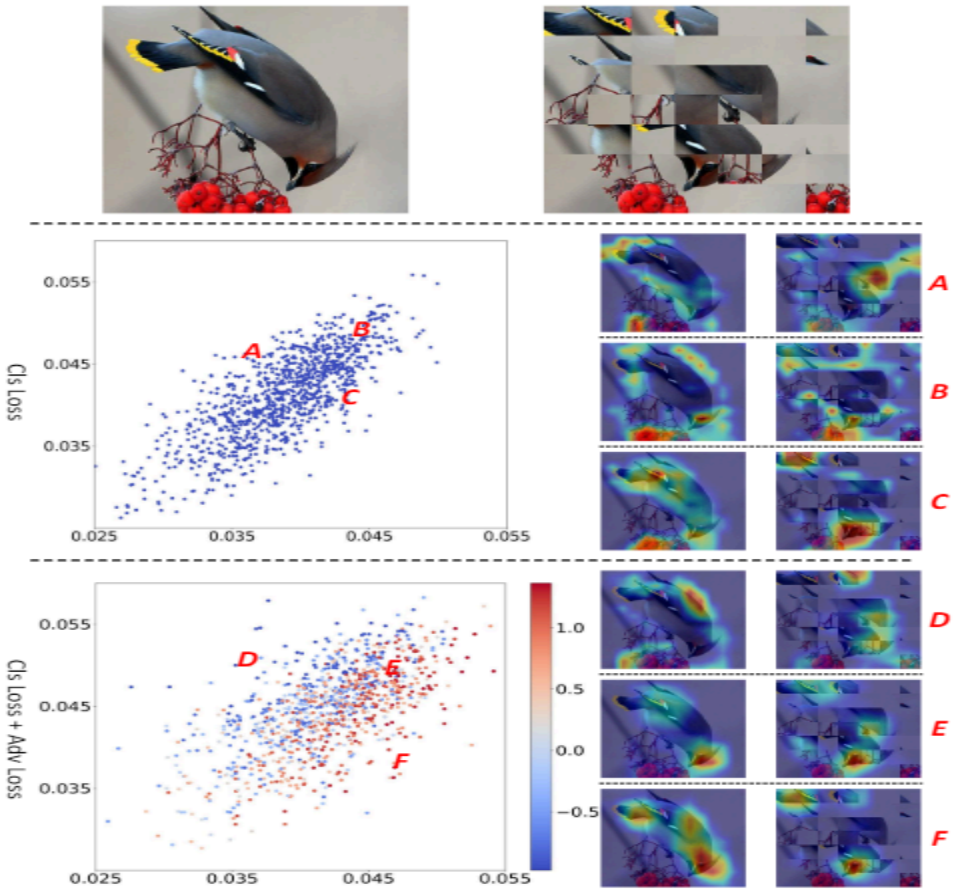
where $C(I, \theta_{cls}^{[1,m]})$ is the feature vector extract from the outputs of the m^{th} layer in backbone classification network, $\theta_{cls}^{[1,m]}$ is the learnable parameters from the 1^{st} layer to m^{th} layer in the classification network, and $\theta_{adv} \in \mathbb{R}^{d \times 2}$ is a linear mapping. The loss of the discriminator network $\mathcal{L}_{adv}$ can be computed as:

$$\mathcal{L}_{adv} = -\sum_{I \in \mathcal{I}} \boldsymbol{d} \cdot \log [D(I)] + (\mathbf{1} - \boldsymbol{d}) \cdot \log [D(\phi(I))] . \quad (6)$$

2.1.2 Construction Learning:

Given an image I and its corresponding destructed version φ(I), the region Ri,j located at (i, j) in I is consistent with the region Rσ(i,j) in φ(I). Region alignment network works on the output features of one convolution layer of the classification network C(·, θ[1,n] cls), where the n th layer is a convolutional layer. The features are processed by a 1 × 1 convolution to obtain outputs with two channels. Then the outputs are handled by an ReLU and an average pooling to get a map with the size of 2×N ×N. The outputs of region alignment network can be written as:

$$M(I) = h\left(C(I, \theta_{cls}^{[1,n]}), \theta_{loc}\right), \quad (8)$$



Visualization of filters learned by using Lcls and Lcls + Ladv respectively

Pretext invariant representation learning (PIRL):
A self-supervised learning (SSL) technique designed to learn transformation-invariant representations from unlabeled image data. PIRL operates by leveraging both the original image and jigsaw patches derived from the input image to train a shared backbone neural network.

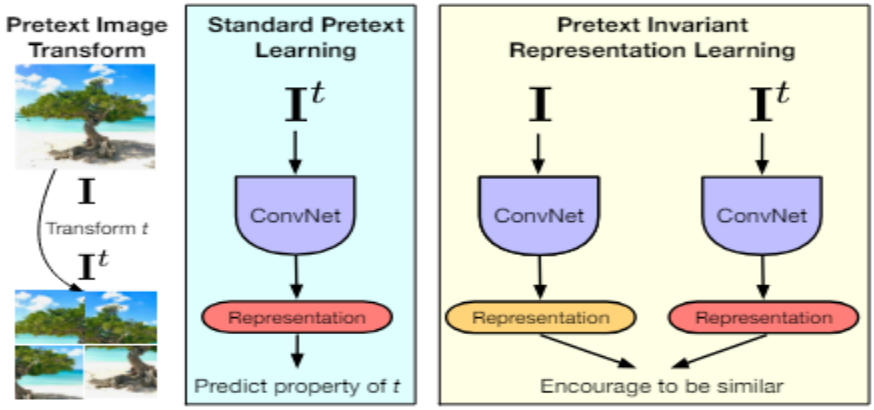


Figure 1: Pretext-Invariant Representation Learning (PIRL). Many pretext tasks for self-supervised learning [20, 54, 85] involve transforming an image $\mathbf{I}$, computing a representation of the transformed image, and predicting properties of transformation t from that representation. As a result, the representation must *covary* with the transformation t and may not contain much semantic information. By contrast, PIRL learns representations that are *invariant* to the transformation t and retain semantic information.

Suppose we are given an image dataset, D = {I1, . . . , I|D|} with In ∈ R H×W×3 , and a set of image transformations, T . The set T may contain transformations such as a re-shuffling of patches in the image, image rotations, etc. We aim to train a convolutional network, φθ(·), with parameters θ that construct image representations vI = φθ(I) that are invariant to image transformations t ∈ T . We adopt an empirical risk minimization approach to learning the network parameters θ. Specifically, we train the network by minimizing the empirical risk:

$$\ell_{inv}(\theta;\mathcal{D}) = \mathbb{E}_{t\sim p(\mathcal{T})}\left[\frac{1}{|\mathcal{D}|}\sum_{\mathbf{I}\in\mathcal{D}} L\left(\mathbf{v}_{\mathbf{I}}, \mathbf{v}_{\mathbf{I}^t}\right)\right], \qquad (1)$$

We contrast our loss function to losses that learn image representations vI = φθ(I) that are covariant to image transformations t ∈ T by minimizing:

$$\ell_{co}(\theta;\mathcal{D}) = \mathbb{E}_{t\sim p(\mathcal{T})}\left[\frac{1}{|\mathcal{D}|}\sum_{\mathbf{I}\in\mathcal{D}} L_{co}\left(\mathbf{v}_{\mathbf{I}}, z(t)\right)\right], \qquad (2)$$

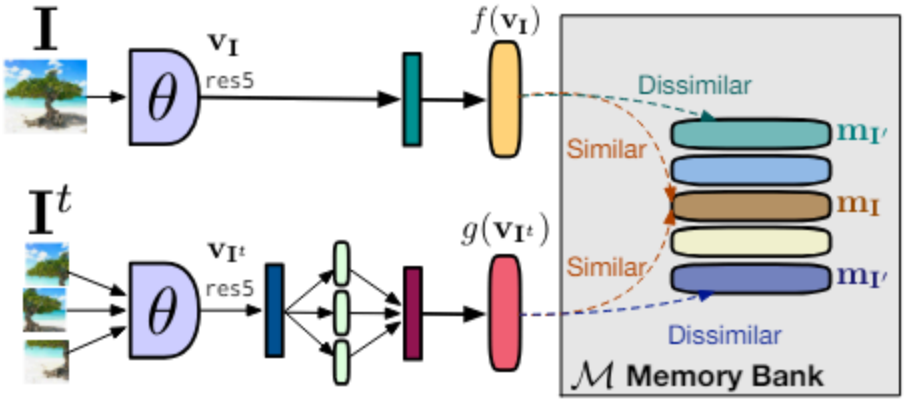


Figure 3: Overview of PIRL. Pretext-Invariant Representation Learning (PIRL) aims to construct image representations that are invariant to the image transformations $t \in \mathcal{T}$. PIRL encourages the representations of the image, $\mathbf{I}$, and its transformed counterpart, $\mathbf{I}^t$, to be similar. It achieves this by minimizing a contrastive loss (see Section 3.1). Following [81], PIRL uses a memory bank, $\mathcal{M}$, of negative samples to be used in the contrastive learning. The memory bank contains a moving average of representations, $\mathbf{m_I} \in \mathcal{M}$, for all images in the dataset (see Section 3.2).

Convolutional network: We use a ResNet-50 (R-50) network architecture in our experiments . The network is used to compute image representations for both I and I t . These representations are obtained by applying function f(·) or g(·) on features extracted from the network. Specifically, we compute the representation of I, f(vI), by extracting res5 features, average pooling, and a linear projection to obtain a 128-dimensional representation.

Hyperparameters: We implement the memory bank as described in and use the same hyperparameters for the memory bank. Specifically, we set the temperature in Equation 3 to τ = 0.07, and use a weight of 0.5 to compute the exponential moving averages in the memory bank. Unless stated otherwise, we use λ= 0.5.

3 RESULTS AND OUTPUTS:

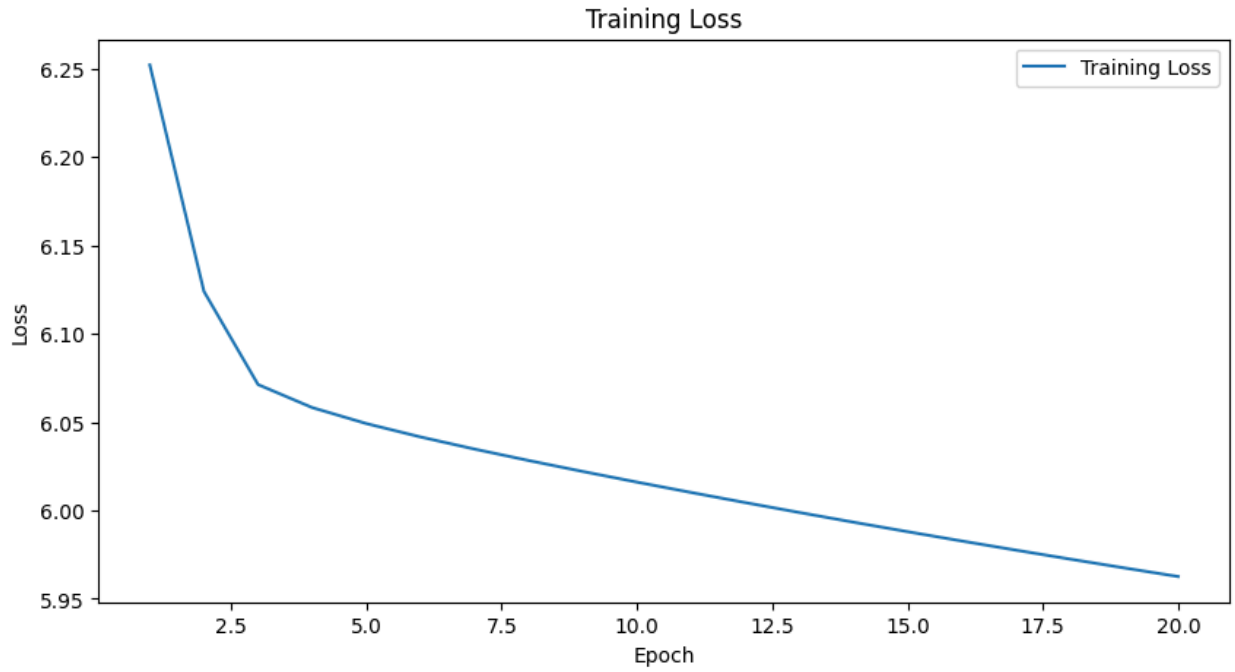
Here are the losses on each epoch while training and the accuracy journey throughout.

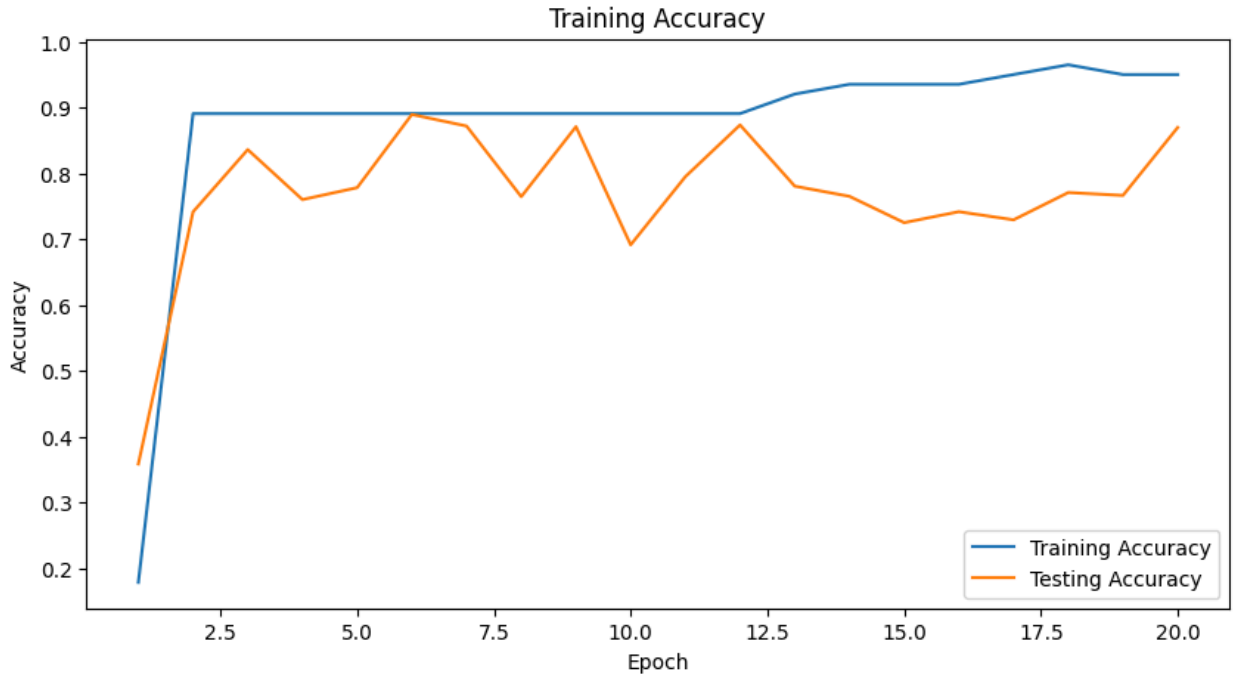
```
Epoch [1/20], Loss: 6.2636, Accuracy: 0.1782
Epoch [2/20], Loss: 6.1130, Accuracy: 0.8909
Epoch [3/20], Loss: 6.0666, Accuracy: 0.8909
Epoch [4/20], Loss: 6.0539, Accuracy: 0.8909
Epoch [5/20], Loss: 6.0450, Accuracy: 0.8909
Epoch [6/20], Loss: 6.0376, Accuracy: 0.8909
Epoch [7/20], Loss: 6.0309, Accuracy: 0.8909
Epoch [8/20], Loss: 6.0245, Accuracy: 0.8909
Epoch [9/20], Loss: 6.0184, Accuracy: 0.8909
Epoch [10/20], Loss: 6.0126, Accuracy: 0.8909
Epoch [11/20], Loss: 6.0068, Accuracy: 0.8909
Epoch [12/20], Loss: 6.0012, Accuracy: 0.8909
Epoch [13/20], Loss: 5.9958, Accuracy: 0.9206
Epoch [14/20], Loss: 5.9904, Accuracy: 0.9354
Epoch [15/20], Loss: 5.9851, Accuracy: 0.9354
Epoch [16/20], Loss: 5.9800, Accuracy: 0.9354
Epoch [17/20], Loss: 5.9749, Accuracy: 0.9503
Epoch [18/20], Loss: 5.9699, Accuracy: 0.9651
Epoch [19/20], Loss: 5.9651, Accuracy: 0.9503
Epoch [20/20], Loss: 5.9602, Accuracy: 0.9503
```

```
Accuracy: 0.9500
F1 Score: 0.9000
Precision: 0.8900
Recall: 0.8800
Confusion Matrix:
[[ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 ...
 [ 0 12 48 ...  0  0  0]
 [ 0 10 50 ...  0  0  0]
 [ 1 11 48 ...  0  0  0]]
```

Hence , The accuracy for our classification came out to be <u>95.03%</u>

The plots for Losses and Accuracies for both training and testing dataset are:





Some Predicted Output Labels along with expected Labels : As, we can see out classification is working correctly on almost all cases and predicting th class/breed of given bird image correctly.

REFERENCES:

[1] Misra, Ishan, and Laurens van der Maaten. "Self-supervised learning of pretext-invariant representations."

[2] Chen, Yue, et al. "Destruction and construction learning for fine-grained image recognition."

[3] https://doi.org/10.48550/arXiv.2105.08788

[4] https://arxiv.org/abs/1911.06045