**Pixel Play’25 Challenge Report**

**~ INDRANIL SINGHA-24112054 (9476428752)**

**Introduction**

This project focuses on classifying animal images into 50 distinct classes with the best model we can come up with. Here, I have used ResNet101 to serve the purpose. ResNet101 is a deep convolutional neural network known for its robust feature extraction capabilities by being exposed to training on vast images. The model utilizes its pre trained weights and bias to enhance accuracy while trying to minimize computational costs. This project includes data loading, preprocessing, training, and evaluation stages. Moreover, techniques such as freezing and unfreezing certain layers were used to obtain the best results.

### **Data Loading**

**Custom Dataset:** A custom dataset was created to access images easily, which returned the images in a NumPy format. That custom dataset was accessed randomly every epoch, loaded in batches of size 16. 15% of the data is kept for evaluation.

A separate dataset was created for test images in a different way, which were also transformed randomly every epoch to get the best outcome.

### **Model Development**

#### **1. Data Preprocessing:**

* All images were randomly cropped between 70% to 100% and then resized to 300x300.
* For better generalization of the model Random Horizontal flip was applied
* Images were randomly rotated between 0 to ±30 degrees
* The NumPy format is converted to tensor
* Normalized pixel values using the mean [0.485,0.456,0.406] and standard deviation [0.229,0.224,0.225]

#### **2. Model Resnet101**

ResNet101 is a deep convolutional neural network architecture. Its key innovation is the introduction of **residual connections**, which help address the vanishing gradient problem in very deep networks. These connections allow gradients to flow directly through shortcut paths, enabling the network to learn better and deeper representations.

* **Residual Blocks**: Each block includes convolutional layers, batch normalization, ReLU activation, and skip connections to bypass some layers, facilitating better gradient flow.
* **Layer Composition**:  
  ResNet101 consists of:
  + An **initial convolutional layer** followed by max pooling.
  + A **fully connected layer** at the end for classification.
* **Pretrained Weights**: Initialized with ImageNet weights for improved feature extraction

**Advantages of ResNet101**:

* Enables training of very deep networks without performance degradation.
* Performs well on image classification tasks due to its high capacity for representation learning.

#### **3. Training:**

* **Loss Function: Cross** Entropy Loss
* **Optimizer:** SGD Optimizer was used
* **Epochs and Batch Size:** 25 epochs were done initially. Batch Size was 16
* **Learning-rate scheduler:** MultiStepLR which will decrease initial learning rate after 16 and 21epochs by a factor of 0.9
* **Hardware:** Kaggle GPU P100

### **Explainability:**

The ResNet101 model was loaded with pretrained weights and biases for all layers. The network consists of four main convolutional layers and a fully connected (head) layer at the end. To achieve optimal performance, a fine-tuning strategy was adopted:

1. **Freezing Layers**:
   * All layers except the **last convolutional layer** and the **head layer** were frozen.
   * This approach allowed the lower layers to retain their pretrained knowledge, while the unfrozen layers were fine-tuned to adapt to the specific dataset.
   * Among all combinations tested, this configuration yielded the best results.
2. **Learning Rate Scheduler**:
   * A learning rate scheduler was implemented to adjust the initial learning rate dynamically.
   * scheduler decreased the learning rate by a factor of 0.9 after the 16th and 21st epochs.
   * This was decided based on the analysis of the loss graph during the initial training phases, which revealed diminishing returns with the original learning rate.
3. **Image Transformations**:
   * Random transformations like the rotation, flipping, and cropping of both training and testing images were performed.
   * These augmentations ensured that the model learned robust features by exposing it to different orientations and perspectives of images
4. **Prediction Strategy**:
   * During testing, a **voting mechanism** was employed for predictions.
   * Multiple predictions were made for each image, and the final output was determined based on the majority vote across these predictions.
   * This strategy enhanced the reliability of the classification results.
5. **Output Generation**:
   * The final predictions were compiled and saved in a **CSV file**.

### **Results**

* **Model gave a 91-93% accuracy on the 15% seen dataset**
* **Model gave a 80% accuracy on the test dataset (unseen)**

### **Conclusion**

### **Challenges**

1. **Zero-Shot Learning**:
   * There was a need to explore **zero-shot methods** to train the model more effectively without relying heavily on labeled data.
   * However, finding reliable resources and understanding these advanced techniques proved difficult.
2. **Computation Power**:
   * Despite using a **Kaggle GPU P100**, the training process often took **2-3 hours**, which significantly delayed the analysis and refinement process.
   * Kernel crashes occasionally occurred, adding to the frustration and slowing progress.
3. **Trial-and-Error Approaches**:
   * A significant amount of time was spent experimenting with various configurations to find the optimal combination of hyperparameters and techniques.
4. **Pretrained Networks**:
   * Testing different pretrained networks (e.g., Resnet18, Resnet50, EfficientNet) to identify the best fit for the task required considerable effort.
5. **Accuracy Fluctuations**:
   * Updates to the code sometimes caused the accuracy to drop, requiring careful debugging and validation of changes.
6. **Imbalanced Dataset**:
   * The dataset had fewer training images of **cows**, leading the model to mistakenly classify them as **Dalmatians** due to similar body color patterns.
7. **Batch Size Optimization**:
   * Determining the best **batch size** involved balancing accuracy with computation speed, which demanded numerous experiments.
8. **Unknown Errors**:
   * Several errors encountered during the project were difficult to diagnose and resolve, requiring significant troubleshooting efforts and external research.

### **Learning outcomes**

* + 1. Hands-on experience with Kaggle, using its computational resources and features to develop and train models.
    2. Developed to create networks from scratch, using pre-trained models, and fine-tuning to meet specific requirements. Experienced in techniques for modification of existing architectures to improve performance
    3. Obtained knowledge on learning rate schedulers and how to change the learning rate dynamically to converge a model.
    4. Recognized the role of data augmentation in increasing the accuracy and model generalization across various datasets.
    5. Increased ability to apply advanced OOP concepts for developing modular and reusable code structures. Developed skills to diagnose and resolve errors, including those encountered for the first time, which fostered a deeper understanding of model behavior and implementation challenges.