VLG Recruitment Challenge ‘24 Report

Sanatan Kumar Gupta 24323035 8866113558

Introduction

This project focused on creating a neural network to categorize images of animals into 40 different classes. The task presented several challenges, such as handling large datasets and **attaining high accuracy** with a limited amount of training data. Initially, a custom **ResNet block model** was developed but due to low accuracy it was later replaced with **transfer learning** using **ResNet50** (on IMAGENET dataset) to improve accuracy.

Model Development

PS: (I apologize for inserting the code till training in one block only)

1. Data Preprocessing

- **Image Preprocessing Techniques:**
  - Images were **resized** to 128x128 pixels to maintain consistency.
  - Data augmentation techniques such as random horizontal flips, random rotations (up to 10 degrees), and color jittering (brightness, contrast, and saturation) were employed to increase dataset diversity.
  - **Normalization** was applied with a **mean of 0 and standard deviation of 1**.
- **Data Restructuring and Handling:**
  - The dataset was explicitly split into **80% for training and 20% for validation** using `torch.utils.data.random_split`.
  - Separate DataLoaders were created for training and validation subsets to ensure proper batching and shuffling.

2. Model Architecture

- **Initial Model:**
  - A **ResNet block model** was created using **MSRA/He** weight initialization(as this is better than xavier for cross entropy loss function) and **batch normalization** for stabilization.
  - **Adam optimizer** was used with a gradient descent approach. However, this model showed poor accuracy on the validation set.
- **Transfer Learning with ResNet50:**
  - A pretrained **ResNet50** model was loaded with weights trained on ImageNet.
  - The fully connected (FC) layer of ResNet50 was replaced with a custom layer to classify 40 animal classes:
    resnet50.fc = nn.Linear(in_features=2048, out_features=40)

- Early layers of ResNet50 were frozen to retain the learned weights for general features.
- The **later layers** and the **new FC layer** were **fine-tuned** to adapt to the specific dataset.
- A **learning rate scheduler** was used to reduce the learning rate dynamically during training, improving convergence.

3. Training

- **Loss Function:** CrossEntropyLoss was employed to handle multi-class classification.
- **Optimizer:** Adam optimizer was used with a learning rate of 0.001.
- **Scheduler:** A learning rate scheduler with step size 5 and gamma 0.5 was applied to adapt learning rates.
- **Epochs and Batch Size:** Training was conducted for 20 epochs with a batch size of 32.
- **Hardware:** Used NVIDIA P100 cloud GPU from kaggle.

Results / Detailed Explanation of the Model

The transition from a custom ResNet block model to a transfer learning approach using ResNet50 led to a significant improvement in performance. This section explores the reasons behind these improvements, the model's advantages, and the lessons learned throughout the development process.

Model Choice and Justification:

- **Initial Model (Custom ResNet Block):** The custom ResNet block model was designed from scratch, utilizing convolutional layers with batch normalization and MSRA/He weight initialization. While this approach allowed flexibility, it struggled to extract meaningful features from the dataset. The limited dataset size and insufficient training data led to poor performance, with the model achieving only around 18% testing accuracy. This outcome highlighted the challenge of training deep neural networks without sufficient data, as the model failed to generalize well to unseen images.
- **Switch to Transfer Learning (ResNet50):** ResNet50 was chosen as the transfer learning model due to its proven effectiveness in image classification tasks. ResNet50, with its 50 layers, is known for its ability to learn deep hierarchical features and overcome the vanishing gradient problem through residual connections. This network, pre-trained on ImageNet, had already learned to identify a wide range of features from a large and diverse dataset, making it perfect fit for fine-tuning on the animal classification task.

  Why ResNet50 was used:
  - **Pre-trained Weights:** The pretrained ResNet50 model had already learned to recognize basic features like edges, textures, and patterns from ImageNet, which

could be transferred to the animal classification task. This drastically reduced the need for training from scratch, especially with limited data.
    - **Deep Architecture with Residual Connections:** The 50-layer depth allowed ResNet50 to learn complex and abstract features that are essential for distinguishing between the various animal classes in the dataset.
    - **Generalization:** Transfer learning using ResNet50 provided the model with robust feature extraction capabilities that helped it generalize well, even with limited dataset size.

Accuracy Improvement:

- **Validation Accuracy:** The shift to transfer learning resulted in an accuracy boost of approximately 40%. The final testing accuracy reached around 58%.
- **Comparison to Initial Model:** The custom ResNet block model struggled due to its inability to learn features effectively on a small dataset. In contrast, the ResNet50 model's transfer learning approach allowed the network to focus on relevant features that were transferable from the **ImageNet** dataset, significantly improving accuracy. The model's higher accuracy indicated **better feature extraction and classification**, ensuring more accurate predictions on new, unseen images.

Generalization Across All 40 Classes:

One of the key things in the mode was generalization. This was particularly important because animal species can have significant visual variation, and the model had to differentiate between very similar-looking species.

- **Reduced Overfitting:** The custom ResNet block model showed signs of overfitting due to its inability to effectively extract and generalize features from the small dataset. By using ResNet50's pretrained weights, the model reduced overfitting significantly, demonstrating a much more robust generalization across the diverse animal classes.
- **Class-wise Accuracy:** The final model showed consistent performance across the 40 classes, making it well-suited for any other classification task as well.

Transfer Learning Advantages:

- **Faster Convergence:** The pretrained weights provided a better starting point, reducing the time required for training. The model **did not need to start from scratch**, which led to **faster convergence(100 epochs to 20 epochs)** during training and a reduction in the number of epochs needed to achieve high accuracy.
- **Effective Fine-tuning:** By freezing the early layers of ResNet50 and only **fine-tuning the later layers** and the fully connected (FC) layer, the model effectively adapted to the new task without overfitting. Fine-tuning allowed the network to adjust its learned features to the specific characteristics of animal images while retaining valuable general features learned from ImageNet.

Model Optimization Techniques:

- **Data Augmentation:** Data augmentation techniques, such as **random horizontal flips, rotations, and color jittering**, helped to artificially increase the size of the dataset, which mitigated overfitting and improved the model's ability to generalize.
- **Learning Rate Scheduling:** The use of a dynamic learning rate scheduler further enhanced the model's convergence. By reducing the learning rate during training, the model was able to fine-tune its weights more effectively and reach an optimal solution.

Explainability

We can explain the how and why of the models using various methods

1. **Intuitive** Grad-CAM Explanation: I didnt actually ran Grad-CAM. But intuitively "educated assumptions" about which features the model is likely focusing on, based on dataset and CNN

One can argue how can i am assuming but well, implementing a Grad-CAM was a later idea and I couldn't try it actually.

2. If you are not satisfied with "Intuitive Grad-CAM" thing

The model's decision-making process can be intuitively explained based on how Convolutional Neural Networks (CNNs) typically identify features in images. CNNs learn to detect **spatial hierarchies** in images, beginning with **simple edges in early layers** and progressing to **complex patterns in deeper layers.**

These assumptions are based on the training data and how CNNs process images. The network's ability to focus on such features is consistent with its design to recognize patterns and textures relevant to each class.

Conclusion

Challenges

- **Initial Model Performance:** The custom ResNet block model exhibited low accuracy, likely due to insufficient feature extraction.
- **Training Complexity:** Managing overfitting and optimizing hyperparameters required iterative experimentation.
- Using transfer learning somehow managed to take the accuracy higher but still I expected it go around 90% but I think I lack in implementation of the model and thus leading to still a mid performance of 58%

Solutions

- Switching to transfer learning leveraged pre-trained ResNet50 weights, significantly enhancing performance.
- Explicitly splitting the dataset into training and validation sets ensured proper evaluation of generalization.

- Data augmentation mitigated overfitting and improved generalization.
- Improving the implementation of the model even more for better performance.

Learning Outcomes

- Gained hands-on experience with transfer learning and model fine-tuning.
- Developed insights into the importance of pre-trained models for complex classification tasks.
- Gained knowledge on the implementation part of deep learning.
- Improved understanding of CNN architectures and their practical implementation.