Animal Classification

Wei Long Wan (a1801549) Jian Yi Tai (a1836176)

1. Introduction

Our group has chosen the Animal dataset which contains 6270 images of size 224x224 of 151 different classes of animal. The objective is to improve the baseline model so that it will achieve a higher accuracy in categorizing the animals into their respective categories. Efficiency will also be considered which comprises not only the accuracy but also the computation costs of the training phase of the model.

The baseline code builds and trains a deep neural network through the PyTorch library. It loads the data and performs a set of image transformations on it. These include: halving the size of the image, flipping half of the input images horizontally and extracting the center of the image of size 112x112. The images are randomly split into either the training, validation, or test set, and are used to create the data loaders. Each data subset loads in 16 images for each batch during training. The baseline model defines an image classification convolutional neural network (CNN) that consists of 4 sampling layers and a fully connected layer. The network uses the Rectified Linear Unit (ReLU) activation function to enhance its non-linearity. MaxPooling is applied to downsample the feature maps, selecting the maximum value in each region to achieve spatial invariance. The convolutional layers have input channels of [3, 64, 128, 128], a kernel size of 3x3, and a stride of 1. To train the model, the Adam optimization function is used with a learning rate of 0.001. During training, the model utilizes the cross-entropy loss function to evaluate how well it predicts the training data. This loss function measures the discrepancy between the predicted probabilities and the actual labels.

2. Modifying the baseline model

Through our research, we have explored and tested many ways to modify the base model. Firstly, we have increased the number of max epochs from 10 to 100, and implemented an early stopper to stop model training once the validation loss hits convergence. Next, we implemented a learning rate scheduler which dynamically adjusts the learning rate during the training process. Through ExponentialLR(...,gamma=0.9), the learning rate is multiplied by gamma after each epoch, and causes the learning rate to decrease exponentially. This allows the model to take smaller steps as training progresses. We also did further testing with data augmentation, and replaced the Resize() function in the baseline with RandomResziedCrop() which randomly crops a portion of the image with a specified size. Compared to the original model, we instead get a snippet of the image instead of downsizing the image. We also added RandomRotation(15), which randomly rotates the image by 15 degrees. This helps the model become invariant to the orientation of objects, making it more robust to images captured from different angles.

2.2: Hyper-parameters

When adjusting the learning rate (LR), using a high LR like 0.1 resulted in poor accuracy, indicating that the model struggled to find an optimal solution. On the other hand, lowering the learning rate allowed the model to converge to a more optimal solution, improving the accuracy of the predictions. Using Stochastic Gradient Descent (SGD) resulted in slower learning, requiring more epochs to reach convergence. Adam optimizer combines ideas from adaptive learning rates and momentum. It had a faster convergence and improved training efficiency. An alternative variant, AdamW, incorporates weight decay regularization into the Adam optimizer. This may further enhance the model's performance by reducing overfitting and improving generalization.

2.3: Loss Functions

Used in the base model, Cross Entropy, calculates the average difference between predicted and actual values by comparing probability distributions. It penalizes models heavily for being overly confident and incorrect. This makes it suitable for binary classification tasks.

Like Cross Entropy, Hinge Embedding loss is effective for binary classification. It emphasizes the sign difference between predicted and actual class values. It motivates examples to have the correct sign, making it useful for determining dissimilarity or similarity between inputs. Mean Squared Error (MSE) is a regression-oriented loss function that punishes models for larger mistakes more severely. It computes the average of squared differences and is sensitive to outliers. In contrast, L1 (Mean Absolute Error) provides a robust measure of error in regression problems, considering the absolute differences between actual and predicted values and being less affected by outliers.

2.4: Architectural changes

Batch Normalization is a technique used in supervised learning to normalize the outputs between layers in a neural network. It standardizes the activations of each batch of images. This ensures that the subsequent layers receive inputs with a stable and standardized distribution, leading to more effective analysis of the data. Batch Normalization stabilizes and speeds up the training process of deep neural networks. By normalizing the outputs of the previous layer, it maintains consistent assumptions about the spread and distribution of inputs during weight updates. This stability facilitates faster convergence during training. Additionally, Batch Normalization reduces generalization error by preventing overfitting. By normalizing the layer, it prevents activation values from becoming too high or too low, which helps the network generalize better to unseen data. We found this model to be the most efficient and have used this in our final model. We tried Batch Normalization with dropout but it lowered the accuracy so we omitted it.

The Leaky Rectified Linear Unit (Leaky ReLU) is a modified version of ReLU. With a negative input, Leaky ReLU returns a small negative value proportional to the input instead of returning 0. This can be particularly useful in models where a large number of neurons may encounter negative inputs. We did not find any significant difference between the two so we kept with the original.

ResNet is a deep learning model designed to support multiple convolutional layers. ResNet stacks multiple layers that get skipped, and uses the activations of the previous layer instead. This compresses the network into fewer layers which speeds up the initial training of the model. Through our testing, ResNet18 has a higher accuracy (66.9) however it has a larger computational cost (0.96). Compared to Batch Normalization, it has a much lower efficiency. Having more layers compared to the base model (18 to 4), it requires more computational resources therefore being less efficient. Another explanation could be that the size of the dataset is too small relative to the complexity of the ResNet18 structure.

3. Conclusion and Future Prospects

In conclusion, by applying Batch Normalization, learning rate scheduling, and data augmentation techniques we achieved our best results. Through these techniques, the model achieved an accuracy of 63.5% with 0.69 GFLOPS and efficiency of 92.03.

For additional improvements, we could apply transfer learning to our model. It is the reuse of a pre-trained model on a new set of images. In transfer learning, the model uses the knowledge gained from a previous task to improve generalization for another hence improving efficiency.

Method	Batch Normalisation	Data Augmentation	Learning Rate Scheduling	Accuracy (%)	Computational costs (GFLOPS)	Efficiency (Acc/GFLOPS)
Baseline (1)				39.0	0.69	56.52
Lr = 0.01 (2)				1.6	0.69	2.32
Opt = SGD (3)				18.1	0.69	26.23
Opt = AdamW (4)				38.6	0.69	55.94
Baseline (5)	✓			55.7	0.69	80.72
Opt = SGD (6)	✓			43.8	0.69	63.48
Baseline (7)	✓		✓	59.5	0.69	86.23
Baseline with leaky RELU (8)	✓		✓	60.3	0.69	87.39
Baseline (9)	✓	✓	✓	63.5	0.69	92.03
Lr = 0.01 (10)	✓	✓	✓	50.7	0.69	73.48
Lr=0.0001 (11)	✓	✓	✓	61.0	0.69	88.41

Table 1. Ablation Study Table, refer to graphs in 'Graphs' folder.

Reference

Brownlee, 2017, Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, Machine Learning Mastery, viewed 15 June 2023, <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>.

Datagen, 2023, ResNet: The Basics and 3 ResNet Extensions, datagen, viewed 14 June 2023, <https://datagen.tech/guides/computer-vision/resnet/>.

Deepchecks, 2023, Batch Normalization, deepchecks, viewed 14 June 2023, <https://deepchecks.com/glossary/batch-normalization/>.

Donges, 2022, What Is Transfer Learning? Exploring the Popular Deep Learning Approach., builtin, viewed 20 June 2023, <https://builtin.com/data-science/transfer-learning>.

Parico, Ahamed, 2021, Why Were Leaky Rectified Linear Unit and Mish Used as the Activation Functions for the YOLOv4 Models?, bio-protocol, viewed 15 June 2023, <https://bio-protocol.org/exchange/minidetail?type=30&id=9907666>.