SYDE 572 – Enhanced Image Classification with Data Augmentation for Household Images

Name: Karla Castro

Student ID#: 20745522

Introduction

This report outlines a neural network training pipeline that mainly encompasses data preparation using robust data augmentation and transfer learning using both a DenseNet architecture and a ResNet architecture with the goal of obtaining a high classification accuracy on a dataset of common household objects.

The data used to train and test the model consisted of jpg images in RGB format with dimensions of 102x530x3. The train and test datasets were given as:

- Train: 22 object categories, each category contains 5 training images.
- Test: 517 test images for the 22 categories.

Moreover, a code to determine the accuracy based on the count of images belonging to a class was done to have an estimation of how the model was performing. This was based on mismatch of a class, whenever there was a match, it assigned a 1 and 0 for any mismatch.

The code submitted is in a jupyter notebook. Collab was used to train the model given it allowed access to a GPU. The paths stated in the code are connected to my drive since I uploaded the folders there for easier access.

Data Preparation & Data Augmentation

Based on the knowledge that the accuracy of the image classification model will likely depend on the quality and structure of the utilized dataset, some preparation and transformations were performed. To prepare the dataset for the training process, the images were first resized to 224x224 pixels since both the ResNet50, and the DenseNet-121 models achieve the highest accuracies in classification using image size 224x224 pixels. Then, the mean and standard deviation of the training dataset were calculated to ensure the models' performances were evaluated against genuinely novel data during testing. By doing this, the models had a more efficient training process and were more robust.

Then, some data augmentation was done to improve the model's ability to generalize from the training data to unseen images thereby enhancing its performance. Some randomized transformations were done to increase the size of the dataset. Some augmentation techniques performed by the model were:

- Random rotations to introduce the rotational invariance.
- Random vertical flips and horizontal flips to mimic different viewing angles and perspectives.
- Gaussian blurs to introduce defocussing given blurry images in real test cases.
- Random Adjust Sharpness for adjusting the sharpness of the image randomly with a given probability.
- Normalization to standardize the pixel value distribution of the input images.

The figure below represents an example of the data after applying normalization and some transformations of the data:

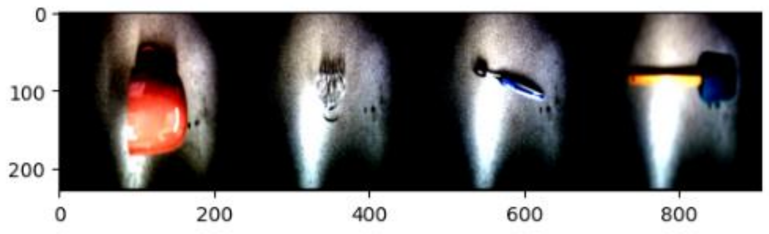

Figure 1. Images transformations using estimated mean and stdv from training set.

Custom Dataset Handling

To accommodate the dataset's structure for both the train and test datasets, two classes were created *CustomDataset* and *ImageFolderCustom*. The purpose of these classes was to load images from the folders and apply the necessary transformations so that they can be inputted to the *DataLoader*. The *ImageFolderCustom* class was specific to the train dataset since it enabled natural sorting which was key to reach a high accuracy since coherent sequence in the dataset was maintained.

Model Selection

A ResNet architecture model was chosen to handle this classification problem since it is an excellent choice for image classification problems due to its deep architecture that effectively captures complex features and patterns in images, making it well-suited for diverse and intricate datasets. Furthermore, ResNet50 comes pre-trained on a large and varied dataset (ImageNet), providing a robust starting point for feature extraction and transfer learning, which can be particularly beneficial when working with limited or specific datasets, such as household images.

Another model used was DenseNet, where each layer receives input from all preceding layers. This means that the feature maps learned in all previous layers are used as inputs to subsequent layers, creating very rich and diversified feature sets that can capture nuances and details from the images. This characteristic is known as feature reuse, and it allows DenseNet to be more parameter-efficient, eliminating the need to learn redundant features, and to potentially achieve better performance on tasks with variability of data points within the same class. This variability was observed in both the train and test datasets where images within the same class had significant differences in terms of appearance, positions, lighting, and colors.

Finally, DenseNet-121 is considered a comparable model to ResNet-50 in terms of complexity. As shown in the image below comparing the parameter and computational efficiency of ResNet and DenseNet. The number of parameters of ResNet is directly proportional to *C x C* while the number of parameters of DenseNet is directly proportional to *l x k x k*. And, since k << C, the DenseNet architecture will have a smaller size than Resnet.

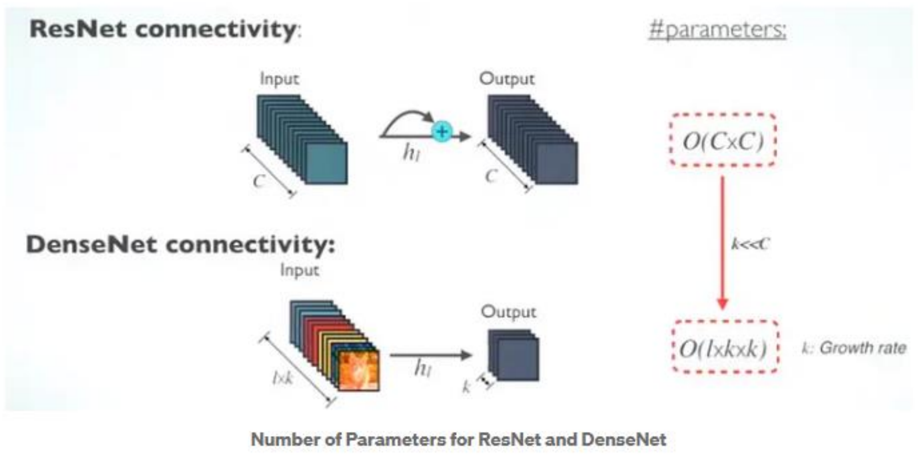

Figure 2. DenseNet vs ResNET-50.

Training Procedure

Stochastic Gradient Descent (SGD) with momentum was chosen as the optimizer for the training phase. This optimizer led to better accuracies compared to the Adam optimizer. There was also some fine-tuning of the learning rate and the momentum to strike a balance between the speed of convergence and the stability of the learning process.

The classifier for both Resnet-50 and DenseNet-121 were also fine-tuned, taking advantage of the pre-trained features from the convolutional base, which remains frozen. This strategy helped in capitalizing on the robust features learned from large-scale datasets on which the model was pre-trained.

Overall, higher accuracies were obtained when changing the classifier architecture to the one showed in figure 3, where some Batch Normalization layers were added as well as some Dropout as regularization techniques.

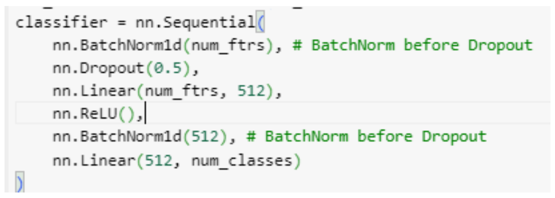
```
classifier = nn.Sequential(
    nn.BatchNorm1d(num_ftrs), # BatchNorm before Dropout
    nn.Dropout(0.5),
    nn.Linear(num_ftrs, 512),
    nn.ReLU(),
    nn.BatchNorm1d(512), # BatchNorm before Dropout
    nn.Linear(512, num_classes)
)
```

Figure 3. Classifier architecture fine-tunning

Results

Upon completion of the training process, the ResNet-50 model achieved a loss ~ 0.03 whereas de DenseNet-121 achieved a loss ~ 0.02. The epoch vs loss graphs can be found below in figure 4 for the Resnet-50 model and figure 5 for the DenseNet-121 model.

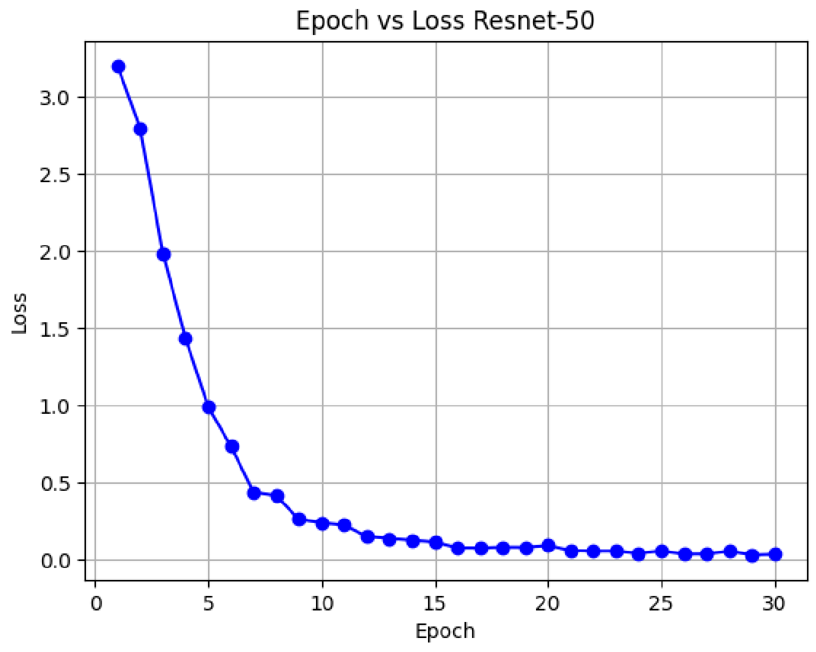

Figure 4. ResNet-50 Model Epoch vs Loss.

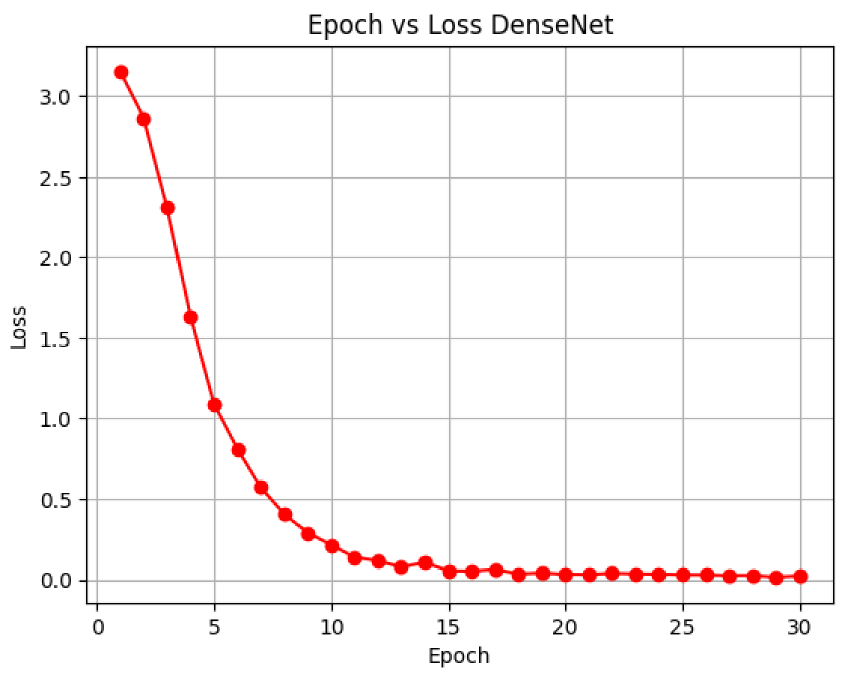

Figure 5. DenseNet Model Epoch vs Loss.

For the testing phase, the DenseNet-121 model led to a higher accuracy rate of approximately 83.94% whereas the ResNet-50 model led to an accuracy of 82.97%, meaning that both models led to similar accuracies, being the DenseNet-121 the best. This level of performance is indicative of the efficacy of the selected data augmentation strategies, before doing data augmentation, the accuracy was around 73-76%. It also demonstrated the suitability of the DenseNet architecture for the image classification task at hand.

Conclusion

The integration of targeted data augmentation techniques, normalization based on training data statistics, and the employment of the already trained ResNet-50 architecture has demonstrated effectiveness in improving the accuracy of the classifier. For the future, more investigation towards image processing techniques can be done to improve the model accuracy based on the statistics and characteristics of the data. Also, other architectures more complex than ResNet-50 can be considered.