Natural Image Classification with Convolutional Neural Networks

https://github.com/gmgl21/Machine-Learning-Project-Natural-Images-with-8-classes

Gloria Lee, Sarah Santos

CECS456: Machine Learning

Dr. Wenlu Zhang

May 15. 2024

Introduction

Convolutional neural networks (CNNs) have impacted the world of artificial intelligence (AI) and computer science. They are a deep neural network class used to analyze visual imagery and are known for high accuracy in image pattern recognition. In this project, we designed multiple CNN models to identify and categorize eight types of natural images. We detail this research's data-cleaning process, model architecture, and analysis. Furthermore, the results of all four models are compared with one another to determine which CNN model has the highest accuracy in image classification. Sarah made Model A and Gloria made Model B ResNet34, Model B ResNet34 V2, and Model C ResNet50.

Dataset and Related Work

The dataset used for this research was provided through Kaggle, containing 6.899 images classifying eight categories including dog, cat, airplane, car, flower, fruit, motorbike, and person. The resolution of the images varied between 3 kB to 30 kB. The initial steps were to set up the environment and download the data to our Google Drive. Downloading the zip file from Kaggle was 359 MB, and opening the zip file and uploading the contents to Google Drive took around three hours. The main folder divided the dataset into the corresponding categories. For Model A, the dataset was divided into training, validation, and test sets. The training set was 80% of the entire data set with 20% remaining for the validation and test sets. However, the two sets vary in weight as 60% of that 20% is towards the validation data. Therefore, 12% of the entire dataset went towards the validation data and the remaining 8% was for the test data. Model B ResNet34, the dataset was from Kaggle. Still, the lowest quality of the images were removed, leaving only so in total there were 6,399 images classifying the eight categories including dog, cat, airplane,

car, flower, fruit, motorbike, and person. Model B ResNet34 V2 and Model C ResNet50 datasets were from Kaggle. Still, the lowest quality of the images of the fruit category was removed, leaving only so in total there were 6,721 images classifying the eight categories including dog, cat, airplane, car, flower, fruit, motorbike, and person. Then they were divided into training, validation, and test sets. Random images were chosen for 5% of each category and were moved to a separate folder for testing data. The remaining 95% were for training and validation data, 90% were used for training data, and 5% for validation data.

Methodology

Model A

The architecture of Model A follows a general CNN design with three convolutional layers (Conv2D), batch normalization and max pooling after each Conv2D, and the last layers as flatten, dense, and dropout. The convolutional layers have filters that increase, doubling the size of the previous filter. Following, the batch normalization is applied to improve stability and the max pooling helps reduce the computational cost while minimizing overfitting. Afterwards, the flatten layer transforms the 3D output into a 1D vector for feature classification. The dense layer contains 128 neurons for combining features learned from the previous layers into patterns used for the final classification. Then the Dropout layer drops half of the neurons in the previous layer for training to enforce the CNN to learn more robust features. Finally, the output layer is a dense layer in which softmax activation is used to produce a probability distribution between the class labels. To compile, the model uses the Adam optimizer and a categorical cross-entropy loss function. All images were resized to 224x224 pixels with three color channels. The resizing is

important for maintaining uniformity across all images in order to process the CNN effectively. Lastly, Model A is built and summarized with the data input.

Model B ResNet34

The architecture for Model B ResNet34 uses a 34-layer convolutional neural network and ResNet34 is pre-trained on the ImageNet dataset. However, if you look at the model output it looks like 50 layers and the largest filter was [512, 512, 2048]. Therefore, there may have been a miscalculation with the layers in the function. The output layer is a dense layer in which softmax activation is used. The model uses the Adam optimizer and a categorical cross-entropy loss function. All images were resized to 180x180 pixels with three color channels, since a lot of images range from 100x100 pixels or 256x256 pixels, finding the average size between them. Model B ResNet34 is built and summarized with the data input.

Model B ResNet34 V2

The architecture for Model B ResNet34 V2 uses a 34-layer convolutional neural network and ResNet34 is pre-trained on the ImageNet dataset. However, if you look at the model output it looks like 50 layers, and the largest filter was reduced [512, 512, 512] compared to Model B ResNet34. Therefore, there may still be some miscalculation with the layers in the function. The output layer is a dense layer in which softmax activation is used. The model uses the Adam optimizer and a categorical cross-entropy loss function. All images were resized to 180x180 pixels with three color channels, since a lot of images range from 100x100 pixels or 256x256 pixels, finding the average size between them. Model B ResNet34 V2 is built and summarized with the data input.

Model C ResNet50

The architecture for Model C ResNet50 uses a 50-layer convolutional neural network and ResNet50 is pre-trained on the ImageNet dataset. The output layer is a dense layer in which softmax activation is used. The model uses the Adam optimizer and a categorical cross-entropy loss function. All images were resized to 180x180 pixels with three color channels, since a lot of images range from 100x100 pixels or 256x256 pixels, finding the average size between them. Model C ResNet50 is built and summarized with the data input.

Experimental Setup

Model A

Model A uses deep learning libraries including TensorFlow and Keras. For data handling, OpenCV, NumPy, matplotlib, and Scikit-learn were imported. The entire project was created in Google Colab, with access to the dataset from Google Drive. For the training, the set is trained over 20 epochs and the images are processed in batches of 16. The training set is shuffled to maintain consistent evaluation performance between training epochs while the test set is not shuffled to ensure stable performance metrics. This experimental setup was designed to effectively handle large datasets while maintaining memory usage and computations.

Model B ResNet34, Model ReNet34 V2, and Model C ResNet50

Model B ResNet34, Model ReNet34 V2, and Model C ResNet50 use the same deep learning libraries as Model A. All of them were created in Google Colab, with access to the dataset from Google Drive. The shuffled training set is trained over 20 epochs and the images are processed in batches of 32. The validation and test sets were not shuffled.

Measurement

The models measure their performances in loss and accuracy for each set (training, validation, test) and save the best epoch, in order to use the best model to evaluate the test set. In addition, the classification report evaluates the model on precision, recall, and F1-score. These evaluations provide information on the model's performance of what classes it identified most to least accurately.

Analysis, Intuitions, and Comparison

The generic CNN structure of Model A provided an accurate performance in which all classes were identified. The worst-performing set was the validation set with a 3.97% loss and 74.31% accuracy. The best performance was from the training set with a 1.11% loss and 86.11% accuracy. As for the test set its performance was similar to the validation set with a 2.53% loss and 75.69% accuracy. The model had the best precision with the fruit category at 100% and performed the worst with the flowers at 42%. The recall performance did well with cars, flowers, and motorbikes at 100% and poorly with cats at 23%. The F1 score for Model A did well with motorbikes at 99% and lacked with cats at 36%. Model B ResNet34, the best performance from the test set was 8.93% loss and 11.81% accuracy. The model had the best precision with airplane, car, flower, and fruit categories at 100% and performed the worst with cat category at 70%. The recall performance did well with fruit, motorbike, and person categories at 100% and worst with dog category at 55%. The F1 score was the best with the fruit category at 100% and worst with the dog category at 67%. Model B ResNet34 V2, the best performance from the test set was 6.49% loss and 16.97% accuracy. The model had the best precision with the fruit category at 100% and performed the worst with the cat category at 79%. The recall performance did well

with fruit, motorbike, and person categories at 100% and worst with the dog category at 58%. The F1 score was the best with the fruit category at 100% and worst with the dog category at 69%. Model C ResNet50, the best performance from the test set was 7.77% loss and 9.22% accuracy. The model had the best precision with seven out of the eight categories at 100% and performed the worst with the dog category at 94%. The recall performance did well in seven out of the eight categories at 100% and worst with the cat category at 95%. The F1 score was the best with six out of the eight categories at 100% and worst with the dog and cat categories at 97%.

Conclusion

In conclusion, among the four models evaluated, Model A displayed the best overall performance. Although Model C ResNet50 did well in most categories including precision, recall, and the F1 score, it demonstrated the model was overfitting, which is common for small-sized datasets. One way of improvement is using a simpler model architecture as used by Model A. This approach helps in achieving generalized results and maintaining a high performance.

Works Cited

Kaggle: Natural Images

ChatGPT

George, Retty. “Final Project: Natural Image classifier”. *Medium*. 30 April 2022, https://medium.com/@retty.george/natural-image-classifier-160415b40c5a.

Grigore, Mihaela. “ResNet + Keras: code from scratch & train on GPU”. *Kaggle*.https://www.kaggle.com/code/mishki/resnet-keras-code-from-scratch-train-on-gpu.