MIDDLE TENNESSEE STATE UNIVERSITY
DATA SCIENCE

AI Memo 777 December 11, 2023

DATA 6900:
Data Dive - Image Classification of Flowers

by

Antonio Levarity

Abstract:

This report presents a comprehensive study on image classification using various deep learning models, including a CNN, Inception V3, and Xception. The models were pre-trained on the ImageNet dataset and fine-tuned for specific classification tasks. The report discusses the process of loading these models using Keras, preprocessing input images to match the requirements of each model, and making predictions. The results highlight the accuracy of each model in classifying images and provide a comparative analysis of their performance. The report also delves into the use of Keras' ImageDataGenerator for handling multiple image files and directories. The findings of this study provide valuable insights into the practical application of deep learning models for image classification tasks, offering a foundation for future research and development in this field.

1 Introduction

This report delves into the process of training deep learning models for the task of image classification. The focus is on the use of Keras, a popular Python library for deep learning, to train these models on a custom dataset.

The training process involves feeding the model with batches of images and their corresponding labels, which is accomplished using a generator. This generator is capable of loading images from a directory, preprocessing them, and feeding them to the model in batches. This approach is particularly useful when dealing with large datasets that cannot fit into memory.

The report will discuss the 'fit' method, which is used to train the model for a specified number of epochs. During each epoch, the model learns to adjust its weights to minimize the loss function. The validation data, passed to the 'fit' method, allows us to evaluate the model's performance on unseen data at the end of each epoch.

The report will provide a detailed analysis of the training process, the challenges encountered, and the solutions implemented to overcome these challenges. It will also present the results of the training process, including the accuracy of the model on the validation data.

2 Methodology

Model Description

The models used in this study are a Convolution Neural Network, and two pre-trained deep learning model provided by the Keras library. The specific type of models are the Inception V3 and the Xception, these models are commonly used for image classification tasks. These models have been pre-trained on the ImageNet dataset, a large dataset of images covering many different categories. The final layers of the model are typically replaced and retrained to adapt the model to the specific classification task at hand.

Data Collection and Preparation

Data preparation involves loading the images, preprocessing them to match the input requirements of the model, and organizing them into batches. This is accomplished using the ‘ImageDataGenerator‘ class provided by Keras. This class can load images from a directory, preprocess them, and feed them to the model in batches. The ‘flow from directory‘ method is used to generate batches of images and labels from the specified directory. The target size of the images is set to match the input size that the model expects.

Model Training

The models are trained using the ‘fit‘ method, which adjusts the model’s weights to minimize the loss function over a specified number of epochs. The number of epochs is a hyper-parameter that determines how many times the learning algorithm will pass through the entire training dataset. Each epoch consists of one forward pass and one backward pass for all training examples. The ‘fit‘ method takes as input the training data, the number of epochs, and the validation data.

Model Evaluation

Model evaluation is performed during the training process. At the end of each epoch, the model’s performance is evaluated on the validation data. This provides a measure of how well the model is likely to perform on unseen data. The performance of the model is evaluated using metrics such as accuracy, precision, recall, and F1 score. We focused more on the accuracy metric and validation loss for the pre-trained models.

3 Experiments and Results

Hyper-Parameters

The primary hyper-parameter in this study is the number of epochs, denoted as ‘num epochs‘ in the code. The number of epochs is a critical factor that influences the performance of the model. It represents the number of complete passes through the entire training dataset. Too few epochs can result in under-fitting of the model, while too many epochs can lead to over-fitting. The optimal number of epochs was determined through experimentation i.e grid search. The grid search also allowed us to find the optimal batch size and learning rate for the convolution neural network.

Dataset Usage

The dataset used in this study is a flower image dataset which consisted of numerous samples of 14 different types of flowers. The dataset was divided into a training set and a validation set. The training set was used to train the model, while the validation set was used to evaluate the model's performance at the end of each epoch. The data was loaded and preprocessed using an ‘ImageDataGenerator‘ and the ‘flow from directory‘ method, which allowed for efficient handling of large datasets and automatic preprocessing of images.

Results

Model	**Scoring Method**	**Score**
CNN	Accuracy	0.74
CNN	Precision	0.78
CNN	Recall	0.74
CNN	F1	0.72
Inception V3	Accuracy	0.96
Inception V3	Validation Loss	0.2
Xception	Accuracy	0.96
Xception	Validation Loss	0.22

Table 1: Model Scores

The model was trained using the ‘fit‘ method of the Keras model. This method adjusts the model's weights to minimize the loss function over the specified number of epochs. At the end of each epoch, the model's performance was evaluated on the validation data. This provided a measure of how well the model was likely to perform on unseen data.

The results of the training process were stored in the ‘history‘ object. This object contains the loss and accuracy values for the training and validation set at the end of each epoch. These values can be used to plot learning curves, which can provide valuable insights into the training process, such as whether the model is over-fitting or under-fitting.

As we can see from the results from table 1 above, we note the the traditional CNN did not perform nearly as well as the other models did, producing scores all at around .7. Whereas, both pre-trained models produced very high accuracy with low validation loss.

4 Discussion and Conclusions

In this study, we have explored the process of training deep learning models for image classification tasks using Keras. The models used were pre-trained on the ImageNet dataset and fine-tuned on our specific task, along with an optimized CNN. The training process involved loading and preprocessing images using the ‘ImageDataGenerator‘ class, and training the model using the ‘fit‘ method.

The results of the training process were evaluated at the end of each epoch using a validation set. This provided a measure of how well the model was likely to perform on unseen data. It is evident to see that by far the pre-trained models provide an edge when it cam to this task, as they were already optimized and engineered to handle advance image classification.

One of the key takeaways from this study is the importance of data preparation in training deep learning models. The ‘ImageDataGenerator‘ class proved to be a valuable tool for efficiently loading and preprocessing images. Another key takeaway is the importance of monitoring the model’s performance on a validation set during the training process, which can provide early indications of over-fitting or under-fitting.

Potential improvements for future work could include experimenting with different pre-processing techniques, using data augmentation to increase the size of the training set, and evaluating the possibility of tuning hyper-parameters such as the learning rate and the number of epochs for the pre-trained models, although not necessary.

One limitation of the study is that it did not asses the images on the same sizes. The CNN had images crop to 128x128, whereas the Inception V3 and Xception have a larger image size requirement, possibly providing an edge to those models.

Overall, the study was concise and highlighted the importance of using a pre-trained model as opposed to others.

———- Bibliography ———-

https://keras.io/api/applications/inceptionv3/

https://keras.io/api/applications/xception/