CPSC 542 - Assignment 1

Spencer Au

Problem Statement

The goal of this project is to use a pre-trained model to classify images of different types of food, and then using a food images dataset to fine tune the model to classify the images of food. The relevance of being able to accurately classify food images is that it can be used to help people track their food intake, identify images on social media sites such as Yelp or Instagram, and possibly could even be used to help people with dietary restrictions or allergies. The task of classifying (food) images involves understanding complex patterns, which traditional machine learning models may struggle to handle efficiently due to their limited capacity for feature extraction, etc. Deep learning models, particularly convolutional neural networks (CNNs), excel in image classification tasks across many datasets and benchmarks. This capability to learn directly from data and improve with scale makes deep learning the de facto standard for image classification problems. The plan to solve the problem is to first start with a pre-trained model, and then fine tune the model using a dataset of food images and incrementally unfreezing the layers of the model. I will augment the images in the dataset to help the model generalize to new images of food, fine tune the model, and then evaluate the model's performance on a val and test set. This task requires a deep learning solution because the model will need to be able to classify images of food into 35 different categories, and the model will need to be able to generalize to new images of food, as well as be able to classify images of food that are not in the training set or food that may be occluded, be of different colors, have different backgrounds, or lighting conditions that cannot be effectively controlled for via image augmentation.

Data

The data used in this project was obtained from the Food Image Classification Dataset from Kaggle. The dataset contains over 24,000 images of food from 35 different categories. The images are in jpg format and are of varying sizes. The dataset is split into a training set, a validation set, and a test set. The training set contains 80% of the images, while the validation and test sets contain 10% of the images each. I used a quick Python script to re-organize the

images into 3 separate subdirectories, as originally it was formatted as a single directory with each subdirectory representing a different category of food. Each of the 3 subdirectories contains 35 subdirectories each representing a different category of food. Some issues with the data were that while a handful of categories such as Tacos, Hot Dogs, and Fries had a good amount of images (~1500), there were some categories such as various Indian dishes that contained only 250-300 images. This could potentially lead to overfitting on the categories with fewer images, and could also lead to the model not being able to generalize to new images of food in those categories. Overall it does seem that the images are of good quality, with various colors being included and the images being of food from different angles.

Methods

Preprocessing

In terms of preprocessing and augmentation, I utilized a variety of image augmentation methods such as rotation of up to 40 degrees, width and height shifts of up to 20%, shear transformations, zooming of up to 20%, horizontal flipping, and the 'nearest' fill mode for filling in new pixels that might be introduced by rotations or width/height shifts. This was done to help the model generalize to new images of food, and to help the model learn to classify images of food that may be occluded, be of different colors, have different backgrounds, or lighting conditions that cannot be effectively controlled for via image augmentation. For the validation and test datasets, augmentation was not applied in order to have more objective accuracy metrics. However, all 3 datasets underwent standardization using the `preprocess_input` function specific to the VGG16 model architecture. This function adjusts the color channels of the images to match the preprocessing applied to the original ImageNet dataset, on which VGG16 was pre-trained. This standardization involves scaling the pixel values and centering with respect to the ImageNet dataset mean, ensuring that the input data is in the most suitable format for the model. This was all done using the `ImageDataGenerator` class from the Keras library, which allowed for efficient and real-time data augmentation and standardization during model training. The images from all datasets were resized to 224x224 pixels, aligning with the input size requirement of the VGG16 architecture. Data generators were then utilized to efficiently manage the data during model training, providing a continuous stream of processed images in batches of 32.

Model

The base model used was the VGG16 model, which is a pre-trained model that has been trained on the ImageNet dataset. The VGG16 model is a deep convolutional neural network that has 16 layers, and is known for its simplicity and ease of understanding. I did not include the top layer, and instead utilized our own custom classifer, which consisted of a flatten layer, a dense

layer with 128 units using the ReLU activation function and l2 regularization, a dropout layer with a dropout rate of 0.2, and a final dense layer with 35 units using the softmax activation function. The model was compiled using the Adam optimizer, the categorical crossentropy loss function, and the accuracy metric. The model was initially trained with the base model layers frozen, and then incrementally unfrozen and fine-tuned with a lower learning rate. The intial learning rate was set to 0.001, and was reduced by a factor of 10 to 0.0001 for the fine-tuning step. For the first fine tuning step, I unfroze 3 layers, and then incrementing by 2 layers until I achieve suffient test, train, and val performance. The model was first trained for 200 epochs, and then fine-tuned for 100 epochs. I utilized early stopping with a patience of 15 epochs, and a model checkpoint to save the best model weights. The model was trained on the Chapman University DGX Cluster, which is a high performance computing cluster with 8 NVIDIA A100 GPUs.

Results and Discussion

Metric	Train	Val	Test
Loss	0.9085	0.5500	0.4397
Accuracy	0.7301	0.8833	0.9156

I stopped training after training the model at 50 epochs and unfreezing 5 layers, as the model seemed to be performing sufficiently on the validation dataset. The model generally performed well, and it's not very surprising that it performed especially well on the validation and testing set compared to the training set, as the testing set had undergone image augmentation. From the metrics above, the model generalizes well to underseen data, and is overall a pretty solid model with respect to classifying food. Some of the parameters I had messed with were changing the optimizer, primarily between Adam and SGD, with the latter seemingly have better performance. In addition, I had also initially tried to train my own model from the ground up, but the metrics I was getting indicated there was extreme overfitting. As a result, I settled with transfer learning with the popular VGG16 model, and the results were much better. In the future, I would like to experiment with other pre-trained models such as ResNet or Inception3, as I had seen some metrics that indicate that they might perform better than VGG16, especially with food image classification. I would also like to experiment with more fine-tuning, and potentially using a learning rate scheduler to help with the fine-tuning process. Overall, I learned a lot from this assignment, and I got a lot of hands on experience with deep learning and convolutional neural networks. I had to learn a lot from the start since I'm currently also taking CPSC 393, and had to rely on online resources such as TowardsDataScience to help explain some of the concepts as well as how to use and incorporate the different aspects of the Keras library.