A Comparison of Computer Vision Approaches for Food-Image Classification

By Claudia Gusti

Abstract

In this report, I present an approach to classify food images using SOTA computer vision systems. The goal of this paper is to train and fine tune a deep learning SOTA CNN model with >85% accuracy for top-1 for the test set on the Food-101 dataset.

1 Introduction and Motivation

There has been a rise in obesity cases in recent decades and thus, there has been an increased interest in tracking diets [1]. However, people often forget the many foods they have ingested throughout the day and therefore can't accurately measure the number of calories they have ingested. One possible solution to tackle this problem is to build a food recognizer system, where a model could identify different food items in an image and record it in a database containing a list of foods ingested on a particular day. This paper aims to explore different computer vision approaches using SOTA CNN-based models for an automatic food recognizer system.

2 Data

2.1 General Information About Data

I utilized Food-101 for training and testing our model. Food-101 is a public data set that contains 101 000 images from within 101 categories (multi-class). Each type of food has 1000 images that has been split into 750 training and 250 test samples. The training images are not clean and thus contain some amount of noise from intense colors and wrong labels. The labels for the test images have been manually cleaned. All images have been rescaled to have a maximum side length of 512 pixels

2.2 Data Preprocessing and Augmentation

Since training data was not cleaned, all images had to be normalized and resized according to the image size criteria of the model architecture.

2.3 Data Augmentation

Data augmentation is the process of generating new transformed versions of images from the given image dataset to increase its diversity. These image augmentation techniques not only expand the size of a dataset but also incorporate a level of variation in the dataset which allows our model to generalize better on unseen data. I applied data augmentation to the training set by randomly flipping images horizontally, slightly zooming in and out of each image and applying shear transformations at random using the ImageDataGenerator class from kKeras[8]. The main benefit of using ImageDataGenerator is that we can generate augmented images on the fly while the model is training.

3 Methods

3.1 Evaluating CNN Architectures

The Convolutional Neural Network (CNN) is a deep learning method commonly used in Computer Vision applications [4], The CNN is comprised of several building blocks and is designed to learn features at a spatial level through backpropagation. There are three layers in a single CNN structure: convolution, pooling and fully connected layers [4].

For this application, I chose 3 promising CNN-based architectures to compare –VGG16[2], InceptionV3 [3] and Xception [4] – which has three of the highest top-5 accuracy on the standard ImageNet validation set [4]. A brief summary of the respective architectures that intrigued me can be found below

1) VGG-16: A CNN that is 16 layers deep. During training, the input is a 224x224 RCB image. Preprocessing involves subtracting

the mean RGB value, computed from the training set on each pixel. The image is passed through a stack of convolutional layers, which have filters of dimension 3x3 with convolution stride of 1x1. Spatial pooling is carried out by five max pooling layers, which follow some of the of the conv. Layers (but not all the layers are followed by max pooling). Max-pooling is performed over a 2x2 pixel window [2]

2) InceptionV3: CNN architecture from the Inception family that makes several improvements using Label Smoothing, Factorized 7x7 convolutions, and the use of an auxiliary classifier to propagate label information lower down the network (along with the use of batch normalization for layers in the side head). Label Smoothing is a regularization technique that introduces noise to the labels. This accounts for the fact that datasets may have mistakes in them, so maximizing the likelihood of log p(y|x) directly can be harmful. Batch normalization aims to reduce the internal covariate shift, and in doing so aims to accelerate the training of deep neural nets. [3]

3) Xception: CNN architecture that relies solely on depth wise separable convolution layers. While standard convolution performs channel wise and spatial-wise computation in one step, Depth wise Separatable Convolution splits the computation into two steps: depth wise convolution applies a single convolution filter per each input channel and pointwise convolution is used to create a linear combination of the output of the depth wise convolution. [4]

3.2 Mini test and train data sets

Because experimenting with deep learning models on the complete Food 101 dataset would take a lot of time and computational power, I created a train and test subset of 4 food classes with 1000 images in each food class to evaluate the performance of VGG16, InceptionV3 and Xception on accuracy and categorical cross entropy loss through 10 epochs with a batch size of 16.

In order to fine-tune the architecture (loaded with pre-trained weights) to the Food-101 dataset, I had to concatenate an additional global average pooling layer, 1 fully connected layer with 128 hidden layers, and output layer having 101 softmax units.

I froze the base model architecture and then added the top layers on the mini dataset of 4 food classes and evaluated accuracy and loss across the three models. The following graphs show accuracy and loss plots across InceptionV3, VGG16 and Xception.

It turns out that InceptionV3 gives us a better accuracy rate upon convergence as compared to VGG16 and Xception on the mini validation set. So, I trained and fine-tuned InceptionV3 to the complete dataset and evaluate accuracy and loss.

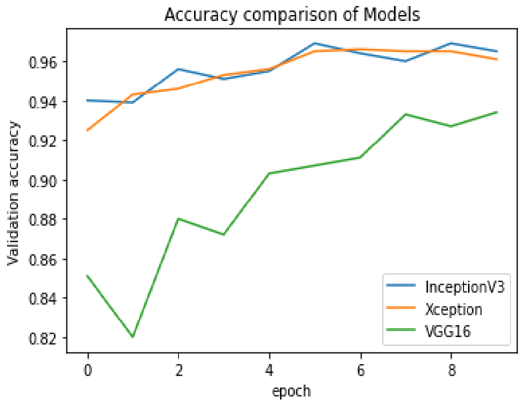

Fig. 1 Accuracy plot over 10 epochs on mini batch

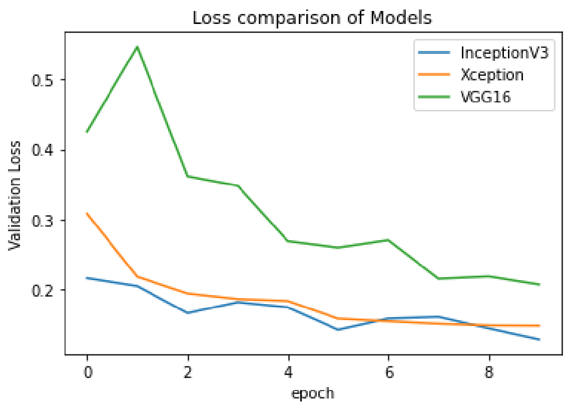

Fig. 2 Loss plot over 10 epochs on mini batch

3.3 Train and fine – tune InceptionV3 to Food 101

Because InceptionV3 gave us the highest accuracy and the lowest loss rate upon convergence when trained on the mini train and test set. I chose InceptionV3 to train with the entire Food101 training set.

I split the entire Food 101 training set into 75% training and 25% validation set. The training and validation set is fully representative of the 101 food classes present in the dataset

For fine-tuning, similar like before, I froze all the convolution layers within the base InceptionV3 architecture and trained only the top layers that I added previously (with the 128 layers with activation function relU) with a small learning rate to prevent overfitting(l=0.0001). I also utilized L2 regularization during training to prevent overfitting, as well as implemented reduced learning rate on plateau function during training. The model converged with accuracy 0.8160 accuracy.

Secondly, I attempted to fine-tune the model further on our training set by unfreezing layers 279 onwards from our base model, which resulted in an increase in model overall accuracy to 0.8538

4 Results

4.1 Training top layer only on 10 food classes

Within 10 epochs, our fine-tuned model (utilizing InceptionV3) as the base architecture achieved a validation accuracy of 0.8160. The accuracy and loss plots are shown below

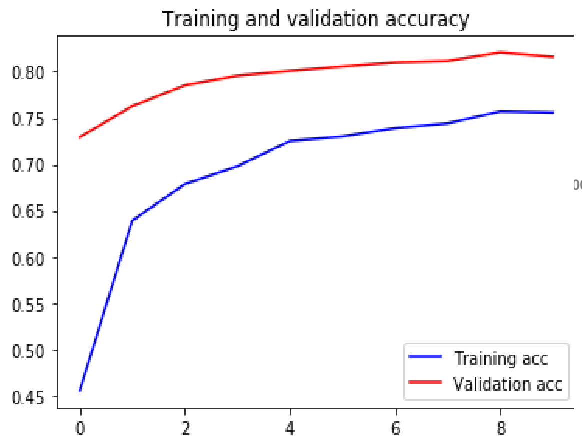

Figure 3. Training and validation accuracy for entire Food 101 dataset

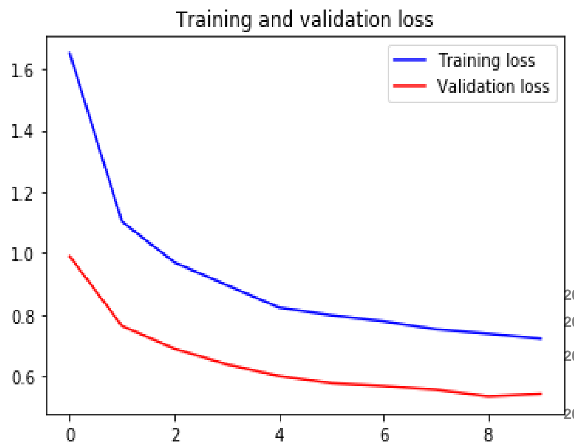

Figure 4. Training and validation accuracy for entire Food 101 dataset

When further investigating which food classes the model failed to predict correctly, I found that the model made the most mistakes on pastries like ‘Baklava’, ‘Apple pie’ and ‘beignets’ since both foods have the same color and texture and might look slightly similar.

Figure 5. Examples of misclassified images from validation set

4.2 Re-fine tuning

After evaluating the results from the last 10 epochs, I unfroze the last 279 layers and onwards from within the InceptionV3 base architecture. I trained the model over the same training subset consisting of 10 classes and evaluated the results below. After fine-tuning, the model converged with an accuracy rate of 0.8538, which is a 0.0378 improvement from the round of training.

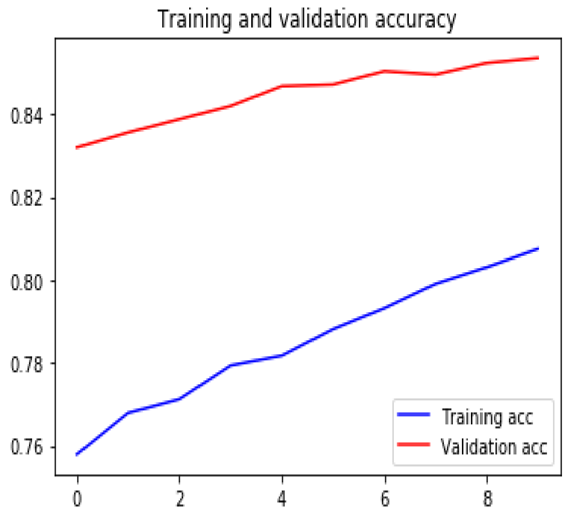

Fig 6. Training and validation plots over 10 epochs

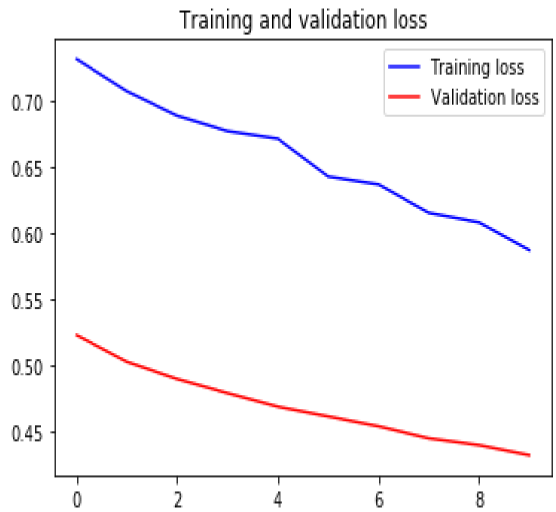

Fig 7. Training and validation plots over 10 epochs

Upon closer inspection of the misclassified examples, most of the misclassified examples are the misclassified examples from the previous training which turned out to be mostly baked pastries such as 'applie_pie' and 'bread_pudding'

Figure 8. Examples of misclassified examples from validation set

5 Conclusions

In this paper, I proposed a transfer learning approach to transfer the knowledge learned from the source dataset (ImageNet) to our target dataset (Food 101) by attaching a classifier to the InceptionV3 base architecture and fine tuning our model to Food 101 through retraining the weights of certain layers within our model to increase overall accuracy on the Food 101 validation set.

In future work, provided GPU resources are available, I want to retrain all the layers on Food 101 training set on our model to further increase accuracy. Further, I also plan to evaluate multi modal classification architectures using Vision and Language models such as ViLT and CLIP [7] that have gained promising results in computer vision applications such as image classification in the past few years.

Acknowledgements

Dr Korpusik for mentorship and guidance throughout the semester

References

[1] “Obesity and overweight,” 2021.

[2]Karen Simoyan, Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *Computer Vision and Pattern Recognition*, arXiv:1409.1556 , 2015

[3]Chrstian Szegedy, Vincent Vanhoucke, Sergey loffe, Jonathon Shlens, Zbigniew Wojna, “Rethinking the Inception Architecture for Computer Vision”, arXiv:1512.00567, 2015

[4]Francois Chollet, “Xception, Deep Learning with Depthwise Seperable Convolutions”, arXiv:1610.02357v3, 2017

[5] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, “Understanding of a convolutional neural network,” in 2017 International Conference on Engineering and Technology (ICET). Ieee, 2017, pp. 1-6

[6] Lukas Bossard, “Mining Discrimminative Components and Random Forests”, Computer Vision – EECV 2014, pp 446-461

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, “Learning Transferable Visual Models From Natural Language Supervision”, arXiv:2103.00020

[8]https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#ags