Fine Tuning and Ensembling Deep Architectures on Large Datasets in Resource Constrained Environment

Amama Mahmood
Department of Robotics
Johns Hopkins University
Baltimore, USA
amahmo11@jhu.edu

Heeyeon Kwon
Department of Robotics
Johns Hopkins University
Baltimore, USA
hkwon28@jhu.edu

Nafisa Ali Amir
Department of Computer Science
Johns Hopkins University
Baltimore, USA
namir2@jhu.edu

***Abstract*—The trend for health consciousness is prompting more people to count their calorie intake. Certain apps like MyFitnessPal, Lose it, Samsung health and Cron-o-meter give calorie count and nutrition value of various food items. Most of the existing solutions make use of neural network based image classification models. This paper aims to study the efficacy of two state of the art architectures namely Resnet-152[1] and Inception-v3[2] for food image classification and attempt to ensemble the models. We propose a naive approach of ensembling the output of the models for better accuracy. We culminate our work by evaluating the models and their ensemble on a twin dataset.**

Keywords—calorie intake, image classification, Resnet-152, Inception-v3, ensemble, food-101

I. Introduction

Image classification has applications in various spheres of life and is therefore a well researched area. Some of the work[3][4][5] already done in the image recognition of food dishes have focused on identifying food ingredients, extending large scale image dataset, increasing classification categories and experimenting with other approaches on the standard datasets for improving the classification accuracy. Yet, the efforts are restricted by the computational resources and large scale image data to train the model. For the specific case of food image classification, there exist at least 91 kinds of cuisines from around the world, according to wikipedia[6], with each cuisine offering 10s and 100s of dishes. This puts forth a larger problem of catering to a wider section of the globe without compromising accuracy. This requires a careful study of existing architectures best suited for such image classification task, and which can be tweaked and tuned for improving accuracy on a generalised dataset. It is also essential to understand how performance on these architectures scale with respect to the number of categories to classify into.

Neural Networks get deeper because they generalise well. The number of layers kept on increasing until the problem of degradation came up and this was solved using Residual blocks of Resnet. While Resnet attempts to improve performance using skip connections, Inception module concatenates multiple convolution layers and stacks them together. We study these two excellent architectures for food image classification and try to ensemble them. In the following sections, we give details about our approaches, dataset, preprocessing on the dataset, models and ensembling, and eventually results with some discussion and scope of further exploration.

II. Dataset

A. Training and Validation

The dataset selected for offline classification is ETHZ Food-101 [7]. Food-101 provides a diverse food dataset containing 101 food categories with 101,000 images. For each class there are 250 test images and 750 noisy training images. A subset of food images are shown in figure 1.

Fig. 1. Samples of images from ETHZ Food-101 dataset [7]

B. Test

UPMC Food-101[8] is considered a twin dataset of ETHZ Food-101. It has about 100,000 images of food recipes classified into the same 101 categories as the ETHZ and have approximately the same size. This dataset is selected to evaluate the model, once it is trained and validated on the former dataset. Only the test images of UPMC are used for evaluation of models. Test dataset consists of about 225 images from each class with a total of 22750 images.

III. Pre-Processing

To make sure, each input parameter has same range we normalise the data. Images are scaled appropriately for each case since Resnet-152 expects input dimension 224x224x3 and Inception-v3 expects input dimension 299x299x3.

Data augmentation is a technique for extending the existing dataset by applying image processing methods. We did some data augmentation by randomly flipping the images along horizontal axis and cropping to a random size and aspect ratio before resizing to expected dimension using torchvision's transforms.RandomHorizontalFlip() and transforms.RandomResizedCrop. The preprocessing step is done dynamically to save memory. Apart from train and test loaders for Food101, a separate test loader is created for evaluation in final phase for UPMC-Food101 test images.

IV. MODELS

A. Residual Neural Network

Residual Neural Network (Resnet) is a deep convolutional neural network targeted to solve issues, such as:

1. Deeper networks are hard to train due to increased complexity.
2. Degradation problem: output is hard to optimize since, with increased depth, error rate of classification drastically increases after a certain point.
3. The so-called "vanishing gradient" problem that arises due to increased depth of network.

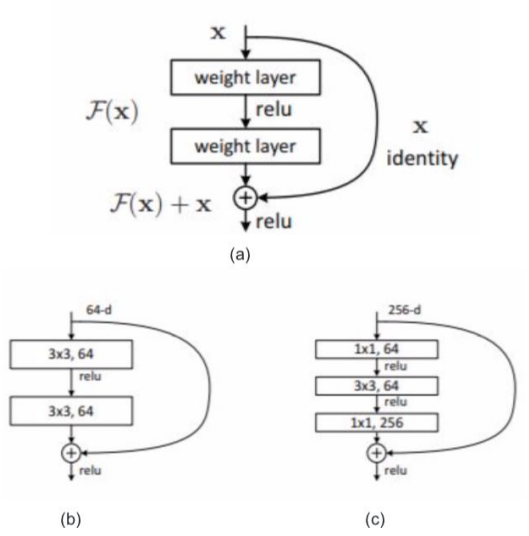


Fig. 2. (a) Residual learning block (b) Basic implementation (c) Bottleneck implementation[1]

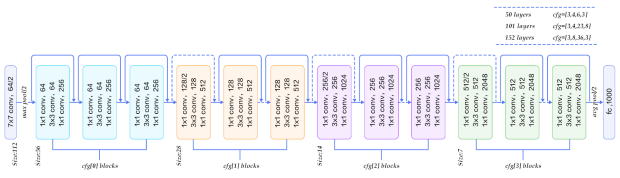


Fig. 3. Architecture of Resnet for Imagenet [10]

The hypothesis that makes it all possible is that it is better to optimize the residual function F(x) (difference between output H(x) and input x of a network. rather than the actual output of a network:

$$F(x) = H(x) - x$$

The magical residual learning block, shown in figure 2(a), makes training of deeper models. Residual learning makes it easier to train the network output and the vanishing gradient problem is handled by "identity shortcut connection" which skips one or more layers in backpropagation by returning identity gradient. Resnet have following popular architectures, which basically vary in depth:

1. Resnet-18
2. Resnet-34
3. Resnet-50
4. Resnet-101
5. Resnet-152

Figure 2(b) shows basic implementation of residual block which is used in shallower networks such as Resnet-18 and Resnet-34 while Resnet-50, Resnet-101 and Resnet-152 used bottleneck implementation since it is computationally efficient for these deeper resnets. These residual blocks are repeated to get architectures of varied depth. Architecture of these deeper versions is shown in figure 3.

Training deeper networks is a strenuous task and requires large amounts of varied data to generalize well. That is where concept of "fine-tuning" helps. Therefore, in practice, it is preferred to fine-tune existing networks that are trained on a large dataset like the ImageNet (1.2M labeled images:http://image-net.org/index) by further training it using the dataset regarding classification problem at hand.

B. Inception

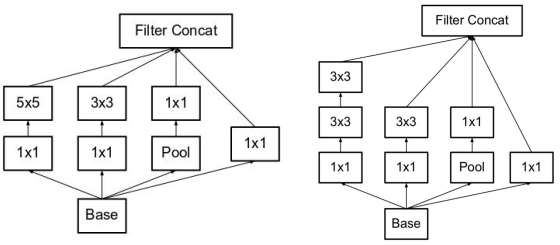
The inception architecture is a complex network formed of repeated inception modules. A general inception module is a composition of average pooling followed by 1x1 convolution, a 1x1 convolution followed by a 3x3 convolution and 1x1 convolution followed by a 5x5 with all the outputs concatenated as shown in figure 4a.

As the neural networks are getting deeper and deeper, it is a challenging task to figure out the appropriate architecture. Inception networks attempt to solve this problem by concatenating multiple convolutions of varying filter sizes. This way the network itself figures out the convolution layers best suited for the dataset. It also allows the network to extract the local and high abstract features. This behaviour is replicated at every layer by repeating the inception modules.

The inception modules are devised using a bottleneck layer which very smartly helps to keep the computational costs down. Thus, this architecture, unlike previous convolutional models, can perform well under constraints on memory and computational budget. It also solves the problem of overfitting faced by very deep networks.

Currently, there are several different versions of the architecture. The three most popular ones are:

1. Inception V1
2. Inception V2 and Inception V3
3. Inception V4 and Inception-Resnet



(a) (b)

Fig. 4. Inception Modules [4] (a) Inception-v1 (b) Inception-v2

Inception-v1 also known as GoogLeNet has 9 such inception modules stacked linearly. It is 22 layers deep. Inception-v2 is an improvement of its predecessor which factorizes 5x5 convolution to two 3x3 convolution operations as shown in figure 4b, to improve computational speed. The Inception-v3 was proposed in the same paper as v2 and is similar in architecture to v2 with some modifications. The Inception-v3 uses RMSProp optimizer, factorized 7x7 convolutions, label smoothing and batch normalisation in its auxiliary classifier. It therefore generalises well and prevents overfitting at lower computational cost compared to other dense versions.

Inception-v4 and Inception-Resnet although not discussed in this paper, are mentioned here for the sake of completeness. The former is a uniform and simplified version of Inception-v3 while the latter as its name suggests introduces a residual inception block which incorporates features from the two architectures.

C. Our Ensemble Model:

For ensembling two networks, we used the individually trained models on all 101 categories for both Resnet and Inception architectures and saved the parameters in each case for the best validation accuracy obtained within a set number of epochs. In the evaluation phase, each model outputs a posterior probability for each class. This probability is then combined to give the final prediction using the equation 1.

$$p(y_i \mid x_i) = \mu\, p_{inception}(y_i \mid x_i) + \nu\, p_{resnet}(y_i \mid x_i)$$

In the above equation, μ and ν are hyperparameters chosen to maximise validation accuracy. Best results are observed at μ = 0.5 and ν = 0.5 (simple averaging). This is because both Resnet-152 and inception-v3 have almost same validation accuracy.

V. Approach

Initially we started with 10 number of classes to setup and test the two models with different learning rates, batch size, and epochs. The brief results (given in detail in progress report) are:

TABLE I. Top-1 Error Rate

Model	Error Rate
Resnet-18	15.16%
Resnet-34	14.44%
Resnet-50	13.64%
Resnet-101	12.56%
Resnet-152	15.20%
Inception-v3	12.96%

Fig. 5. Results with 10 categories classification by fine tuning state-of-the-art architectures

When comparing the results across models, as was expected Resnet-50, Resnet-101, and Resnet-152 outperformed Resnet-18 and Resnet 34, the former being made of 3 layer deep residual blocks compared to the later which are made of 2 layer deep residual blocks. It is interesting to note that Resnet-18 has lower error rate than Resnet-34 and Resnet-50 has lower error rate than Resnet-101. Inception-v3 and Resnet-101 have almost similar error rates, however is much slower to train than Resnet, which turned out to be a bottleneck when we moved on to more number of categories and larger datasets. All the models

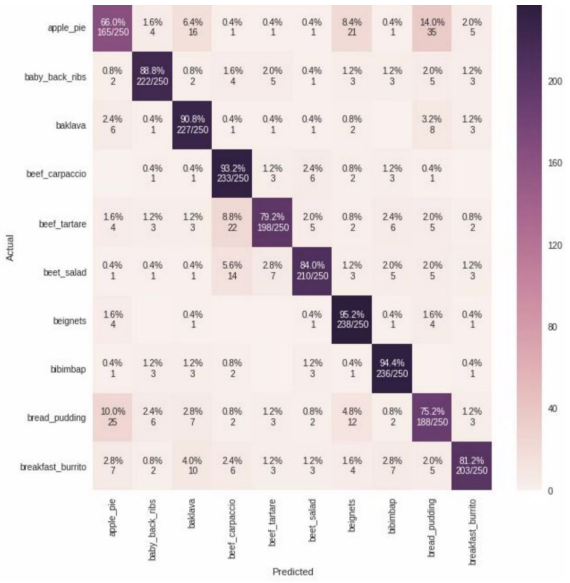


Fig. 6. Confusion Matrix with 10 categories on Resnet-152

were trained for batch size of 64 with Adam as optimizer with learning rate of 0.0001. When trained for a smaller batch size of 8 top-1 error was higher. Due to limited computation resources, ResNet-152 was run for batch size of 8 giving higher error than other ResNet models which might not be the case if batch size was 64. Thus, our results are in accordance with the previously published results [9] on imagenet dataset i.e. top1 error for ResNet-18 > ResNet-34 > ResNet-50 > ResNet-101 ≃ ResNet-151 ≃ Inception-v3. Figure 6 and 7 visualise classification results on 10 classes discussed at length in the progress report.

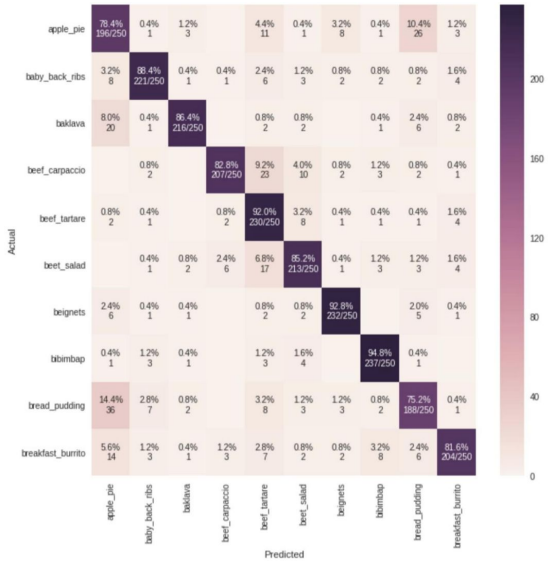


Fig. 7. Confusion Matrix with 10 categories on Inception-v3

Based on the results from 10 food classes, we decided to move forward with ResNet-152 and Inception-v3. The number of classes were then increased to 101. The Inception-v3 and Resnet-152 models were trained with the fine tuned hyper parameters on the Google Cloud Machine Ubuntu 16.04 LTS using 2 Nvidia K80 GPU and 30 GB of memory.

VI. Hyper-Parameter Tuning

We fine tuned four different hyper parameters, the learning rate, batch size, and number of epochs, on the InceptionV3 and ResNet models. We started with a default setting of Adam optimizer, 0.01 learning rate, 64 batch size, and 10 epochs. We later changed the learning rate from 0.01 to 0.0001 (because fine-tuning demands lower learning rate) and increased the epoch to 50 and obtained a better performance. Having tuned the model on 10 categories classification, we use the same setting for classification of 101 categories which saved us time and resources with the only modification of running for 30 epochs instead of 50 owing to resource and time limitations.

VII. Results

A. ETHZ Food-101

Best Top-1 error rate achieved so far on food-101 is 90.27% using wide-slice ResNet [11]. Our results are given below:

TABLE II. Validation Accuracy on Food-101

Top-k Accuracy	Architecture		
	Resnet-152	***Inception-v3***	***Ensemble***
Top-1	75.90%	80.24%	84.85%
Top-5	92.52%	93.80%	96.70%

Fig. 8. Results with 101 categories classification by fine tuning state-of-the-art architectures and their ensemble

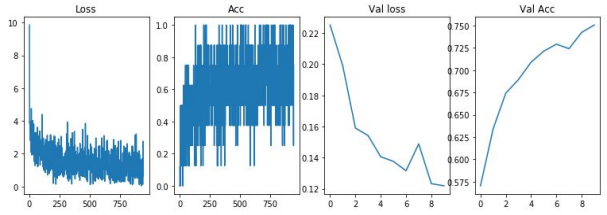


Fig. 9. Plot for training loss, training accuracy, validation loss and validation accuracy for Resnet-152

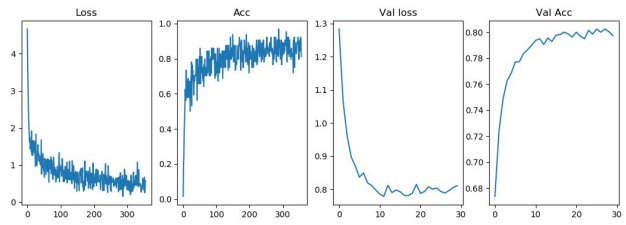


Fig. 10. Plot for training loss, training accuracy, validation loss and validation accuracy for Inception-v3

B. UPMC Food-101

TABLE III. Test Accuracy on UPMC Food-101

Top-k Accuracy	Architecture		
	Resnet-152	***Inception-v3***	***Ensemble***
Top-1	49.08%	51.33%	56.51%
Top-5	70.47%	71.90%	75.92%

Fig. 11. Results with 101 categories classification on test dataset

VIII. Discussion

Initially, the training to evaluate the two models and fine tune the hyperparameters was done on just 10 classes . Then we selected the hyperparameters based on the model performance to train the models for 101 classes since we had limited resources and time to tune hyperparameters for 101 classes.

Food classification is a fine grained recognition task, where the visual differences between food categories are minimal and can be highly affected by factors such as pose, viewpoint, or location of the object in the image. In other words, it is challenging to distinguish images with intra-cuisine features that have large variability but are in the same category compared to images with inter-cuisine features that look similar but are in different categories. Thus, making image classification from purely visual appearance very challenging.

Fig. 12. Inter-cuisine and intra-cuisine feature similarity and variability respectively

The results with 101 categories classification by fine tuning the two models are shown in figure 8, 9 and 10. Our results of the two model performance is not as desired but we were able to obtain reasonable accuracy. We got a top-1 accuracy of 80.24% and top-5 accuracy of 93.80% for the Inception V3 model with Adam optimizer, 0.0001 learning rate, 64 batch size, and 30 epochs. We got a similar top-1 accuracy of 75.90% (for lesser epochs than 30) and top-5 accuracy of 92.52% for the Resnet-152. Due to cuda memory shortage and limitation on time, we used a batch size of 50 not 64 and 10 epochs not 30. We are currently still training the model for 30 epochs. When we ensembeled the two models by averaging the output probabilities of validation, we got a higher accuracy of 84.85% as top 1 and 96.70% as top 5.

The results with 101 categories classification the UPMC Food-101 test dataset are shown on fig.11. The shortage of memory and computational complexity has limited us to train one model rather than multiple models and train for

just 30 epochs rather than thousands of epochs. Thus, giving a relatively low top 1 accuracy of 51.33% and top 5 71.90% for Inception v3, 49.08% as top 1 and 70.47% as top 5 for Resnet-152 which improved by ensembling both models to give 56.51% as top1 and 75.92% as top 5 accuracy. Hence, it is safe to say that even naive ensembling of above two models improve the accuracy by about 5%.

Although no hyperparameter tuning is done for classification of the dataset into 101 categories, the results after 30 epochs of training are still comparable, which is an interesting food for thought. Increasing the number of classification categories increases the size of the dataset to deal with and also increases the training cost. In the specific case of food image classification, the number of classification categories on the standard datasets are much less and are likely to increase in the future. Thus having to fine tune on this ever increasing dataset will increase the time and computational requirements at an even higher rate.

Future work has many possible directions. One is further training of models and introducing more intuitive ensembling techniques such as voting, bagging, boosting and weighted sum with trainable weights. Computation resources and time shall be compared for these techniques as an important deciding factor. Using variants of residual network such as ResNext can be explored to deal computational issues. Inception-Resnet shall be compared to ensemble of inception and resnet to study the changes in performance. Another way of approaching food classification problem is cuisine based categorization. It would be a two step process. First step being, deciding the cruise of the test image and second step would be classification of the food item within a certain cuisine using a model trained for that specific cuisine. Hence, it can be concluded by saying that there is a lot of room for improvement going forward.

IX. Things We Learnt

During the course of this project, we dived deeper into some state of the art architectures and got practical insights into the implementation aspects. Some of the main learnable aspects of the project were as follows:

1. Dealing with huge datasets specifically loading datasets, traversing through them, dividing into training and test examples.
2. Fine-tuning of different existing models (Resnet and inception) to utilize pre-learnt features for giving models a head-start in learning. Pre-trained weights are better initializers of a network thus reaching good accuracy sooner. It saves time, computation resources and effort and gives better training of models to adjust to a somewhat related problem. Over the course of time, we were able to highly appreciate contribution of Imagenet dataset and pre-trained models provided by pytorch towards deep learning community.
3. In the initial part of project, our main concern was to choose various hyperparameters that tested our knowledge greatly. We were able to learn how to choose these hyperparameters systematically.
4. Another learning aspect of the project is the power of ensembling various models to improve accuracy and confidence in our predictions. Although we used averaging to ensemble two networks due to limitation of time and having only two models, but we learnt about other effective techniques such as bagging, boosting and voting etc. Ensembling can improve results but at cost of computational resources.
5. Lastly, working with multiple GPUs, using the google compute instances and dividing workloads for efficiently using the available resources.

X. Advice

A. For Students

Getting familiarity with Google Compute Engine before diving into the project will allow you to estimate the kind of challenge to take up for the project. While working on the project plan, consider training time, training resources, visualising results.

B. For Instructors

Kindly share the advice for students to the next batch before they begin their project proposal. Kindly provide timely reports and share the necessary tutorials beforehand. If possible, give practical advice to students about how much memory and computation time and resources will be needed and is feasible for a project within the limited time frame. Also kindly share the templates for project proposals, progress report and final report in advance.

References

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[3] Chen, H., Xu, J., Xiao, G., Wu, Q., & Zhang, S. (2018). Fast auto-clean CNN model for online prediction of food materials. *Journal of Parallel and Distributed Computing*, *1 17*, 218-227.

[4] Yanai, K., & Kawano, Y. (2015, June). Food image recognition using deep convolutional network with pre-training and fine-tuning. In *M*

[5] Joutou, T., & Yanai, K. (2009, November). A food image recognition system with multiple kernel learning. In *Image Processing (ICIP), 2009 16th IEEE International Conference on* (pp. 285-288). IEEE.

[6] List of Cuisines at url: https://en.wikipedia.org/wiki/List_of_cuisines

[7] Bossard, L., Guillaumin, M., & Van Gool, L. (2014, September). Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*(pp. 446-461). Springer, Cham.

[8] UPMC Food-101 at url: http://visiir.lip6.fr/explore

[9] Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

[10] Image Classification at url: http://www.paddlepaddle.org/documentation/book/en/1.0/03.image_classification/index.html, accessed online on December 7, 2018.

[11] Martinel, N., Foresti, G. L., & Micheloni, C. (2018, March). Wide-slice residual networks for food recognition. In *Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on* (pp. 567-576). IEEE.