[1]

CS 597 Assignment 1 Report

Team NaN

Chris Dagher {114071706}* and Oliver MacDonald {114174690}†

Boise State University Department of Computer Science and

Boise State University Engineering Plus

(Dated: September 13, 2024)

[1] If you're looking for a good laugh, check the commit messages in our Git repo.

*Electronic address: christopherdaghe@u.boisestate.edu

†Electronic address: olivermacdonald@u.boisestate.edu

1. ARCHITECTURE

Our chosen training dataset was the Food101 dataset [4], which contains 101,000 images of 101 different classes of food.

To train MobileNetV2, we loaded the model from TensorFlow [2], which has a pre-made model that follows the network structure outlined in the MobileNetV2 paper [6]. The network was initialized with random weights and biases, an input shape of (224, 224, 3) and an output shape to match the number of classes in our dataset. Images were resized to match the input shape of the network, and pre-processed using Tensorflow's built-in MobileNetV2 pre-processor.

1.1. Training, Fine-Tuning, and Hyperparameters

We trained the model with the hyperparameters detailed in Table I. In an attempt to find optimal hyperparameters, we experimented with the learning rate, trying the values {1e-2, 1e-3, 1e-4, 3e-4} in an attempt to find a value that would balance training speed and accuracy, as well as the batch size. We trained with smaller batch sizes, but found no significant benefit. As a larger batch size provides a more accurate measure of NN parameters' gradients, we opted chose the largest batch size that could fit in the memory of the GPU we trained on and found that 32 was the largest value we could reliably use. The number of epochs trained for was also varied. We attempted to find a number that would provide a good balance of our desired training metrics without overfitting. This was a process of trial and error, but we found that after 30-40 epochs the network appeared to be overtrained. To prevent overfitting, we trained for 25 epochs so that the fine-tuning would not have to un-learn the overfitted parameters. We did not experiment with varying alpha due to time constraints.

Hyperparameter	**Value**
Number of Epochs	25
Batch Size	32
Optimizer	Adam
Learning Rate	1e-3
alpha	1.0

TABLE I: Hyperparameters used for training MobileNetV2 on the Food101 Dataset.

When transferring the learned model to Cifar100 [5], the trained weights were frozen and a fully-connected layer was appended to the model with an output shape of 100 and a softmax activation function to act as a classifier for our new set of classes. We first trained only the

output layer for 30 epochs with a learning rate of 1e-3, and then unfroze the MobileNetV2 model to train both that and the output for another 30 epochs with a learning rate of 1e-4. We found that this provided a better accuracy by fine-tuning the MobileNetV2 model as opposed to only training the classifier.

2. DISCUSSION & EVALUATION METRICS

The balance of our metrics is good for Food101 and poor for Cifar100 which is heavily biased towards precision. The bias in Cifar100 is likely due to a lack of adequate data augmentation during training, or the model being poorly suited for working with images in our datasets.

The model's comparative performance can be seen in Table II, which shows the test accuracy, precision, recall, and F1 score for each model. It shows that the model learned to heavily prioritize being precise rather than accurate for Cifar100 but was able to remain balanced for Food101 having fairly balanced values for all four metrics. Comparing the two results, Food101 yielded better evaluation metrics than Cifar100. This is due to the better balance between all four metrics where as Cifar100 is unbalanced with a very low recall. We suspect that the low accuracy and recall scores of Cifar100 can be attributed to the following reasons:

Dataset	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Food101	0.5470	0.6406	0.5010	0.5623
Cifar100	0.3474	0.7927	0.0654	0.1208

TABLE II: Test metrics for MobileNetV2 trained on the Food101 dataset and fine-tuned on the Cifar100 dataset.

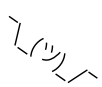
1. **Model architecture** - MobileNetV2 down-samples the input images multiple times. In the case of our implementation, the input of 224x224 is sampled down to 7x7 after the 5 bottlenecks. While this leads to a relatively efficient model, it likely prevents the model from being able to learn fine-grained details. A potential improvement could be adding a residual connection to a small fully-connected network alongside the CNN.

2. **Dataset complexity** - Cifar100 is a complex dataset with many classes. Training a model for this dataset may require more focus on model power than efficiency. For instance, other models can achieve accuracy in excess of 90% such as EfficientNet [3].

3. **Input resolution** - The Cifar100 dataset consists of images with a resolution of 32x32. MobileNetV2 defaults to an input resolution of 224x224, and so we rescaled the Cifar100 images to match this resolution. This could have caused issues in the training because the original images in Cifar100 lack fine detail and details would likely be stretched or blurred during rescaling.

3. CONVERSION TO LITERT (FORMERLY TFLITE)

We were able to convert our trained model to LiteRT [1], which as of September 4th is the new version of TFLite. However, we were not able to evaluate the relative performance compared to the full model. This limitation is due to the LiteRT package having no available distributions through PyPI and a lack of desire to build the library from source ¯_(ツ)_/¯[1]. We can, however compare the model sizes. The disk sizes of the full model and lite model, measured with the `du -h` command, were 28 MB and 9 MB respectively, or a 67.8% reduction in disk usage. This would allow for the model to be stored on storage-constrained embedded systems much more easily.

[1] Yes, this is a custom TikZ drawing. 

[1] LiteRT: a lightweight runtime for resource-constrained environments. URL https://github.com/google-ai-edge/litert.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[3] B. Antonio, D. Moroni, and M. Martinelli. Efficient adaptive ensembling for image classification. *Expert Systems*, Aug. 2023. ISSN 1468-0394. doi: 10.1111/exsy.13424. URL http://dx.doi.org/10.1111/exsy.13424.

[4] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – mining discriminative components with random forests. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 446–461, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10599-4.

[5] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. URL https://arxiv.org/abs/1801.04381.