CS5242 Project 1

Group 33: Zheng Zhijian (E0146072) and Teekayu Klongtruajrok (E0210381)

November 14, 2017

1 Task Description

There exists a dataset of over 90,000+ images of 132 Chinese and Singaporean dishes, with Chinese food images collected by Chen [1] and Singaporean food images collected by Database System Research Group of SoC, NUS. These image are to be classified according to each of the 132 types.

2 Approach

The family of algorithms for state of the art image classification is convolutional neural networks (CNN). Due to the lack of theoretical justification for constructing an architecture of a CNN with guaranteed capabilities, the best approach is transfer-learning from existing, reportedly good, architectures. Tensorflow and Kears (with Tensorflow backend) is selected as a deep learning framework for this task due to its large and consistent community support. According to Tensorflow researchers, the top three best architectures have Top-1 accuracy above 80%, when trained against ImageNet's ILSVRC-2012-CLS dataset [2] as seen on Table 1.

Tensorflow provides pre-trained model checkpoints for each of this architecture. Transfer-learning involves having a code representation of the architecture, loading the pre-trained checkpoints (with all its pre-trained weights) onto the local architecture, customize each layer as needed, then finally customize the final layer so that the output matches the number of classes for a specific application.

This is preferable to retraining from scratch because pre-trained weights contains an implicit value of generalizability of its dataset, which means a certain amount of pattern had already been discerned and its knowledge represented by the weights. This, coupled with the representative nature of the ImageNet's dataset, guarantees that the pre-trained weights in this instance have the generality that is valuable to any image classification in general. The only discrepancy is the classification output layer, which doesn't match this particular application (the ImageNet dataset have 1,000 classes while this application have 132), therefore the final layer needs to be revised.

To help with the training, NVIDIA's GTX 1080 TI GPU is used to help speed up the computation speed. It have 11 GB memory, 3,584 cores, 1,582 MHz clock, 484 GB/s memory bandwidth, and consumes 600 W of power.

The approach to this task is to perform transfer-learning on each of the architectures listed in Table 1. Changes that are experimented with are:

1. Dropout rates - 0.2, 0.5, and 0.8
2. Learning rate optimizer - Adam and SGD with Nesterov momentum
3. Epochs - 30-200 epochs
4. Batch size - 8, 32, and 48

Model	Top1 Accuracy	Top5 Accuracy
NASNet-A_Large_331 [3]	82.7%	96.2%
Inception-ResNet-v2 [4]	80.4%	95.3%
Inception V4 [4]	80.2%	95.2%

Table 1: Pre-trained CNN models available on Tensorflow accuracy report.

Model	Training Accuracy	Validation Accuracy	Testing Accuracy
Inception-ResNet-v2 [4]	99.73%	88.39%	89.41%
Inception V4 [4]	99.75%	87.90%	82.25%
NASNet-A_Large_331 [3]	84.73%	84.12%	82.60%

Table 2: CNN models accuracy report on the food classification problem.

3 Findings

All three architectures in Table 1 are used to train the dataset, and the best performing model is Inception_ResNet_v2, scoring the best testing accuracy with the same number of epochs, as shown in Table 2. Each of the models reported are trained for 70 epochs. The maximum number of training is set to 200 epochs with early stopping implemented. Most of the time, the validation accuracies saturate at about 50-80 epochs.

Experiments on multiple dropout rates had shown than the accuracy performance stays roughly the same when the dropout rate are 0.5 and 0.8. The testing accuracy for Inception-ResNet-v2 is exactly the same for both dropout rates, staying at 89.41%. When the dropout rate is reduced to 0.2, the performance drops by 10%. The validation accuracy for Inception V4 dropped to 71% when using 0.2 dropout rate.

Due to memory limitations, only a few experiments can be done with batch sizes. To help speed up the training, a GPU is used with 11 GB memory. As a result, the batch sizes allocated can only be as much as to match the available memory, as shown in Table 3.

Due to the time limit, a thorough comparison of learning rate optimizers are not performed. Adam optimizer is used for Inception-ResNet-v2 and NASNet-A_Large_331, while SGD with Nesterov momentum and Adam are used for Inception V4. What is expected is that models using Adam will converges faster but have less generalization capability, while models using SGD with Nesterov momentum will converge slower but generalize better and should lead to better accuracy in general [5]. When using Adam, Inception V4 converges faster than SGD, as to be expected. Inception-ResNet-v2 also converges quickly (in under 100 epochs as explained earlier), which can also be attributed to Adam. NASNet-A_Large_331 converges the slowest, despite also using Adam, which is also to be expected due to its sheer size. Training Inception V4 with Adam was only done to gauge the convergence speed, so the accuracy measure was never tested. The accuracy report on Table 2 for Inception V4 was for training with SGD with Nesterov momentum. Despite not having a conclusive answer as to whether to use Adam and SGD with Nesterov momentum, using Wilson [5] as ground truth, it can be extrapolated that SGD with Nesterov momentum should be better, but does not result in Inception V4's accuracy surpassing Inception-ResNet-v2 because Inception-ResNet-v2 is simply a better performing model, with more generalization capability implicit in its architecture.

In addition to the learning rate optimizer, exponential decay learning rate is also used. Two decay rates are experimented with: initial learning rate being 0.0002 that decays by 70% every

Model	Batch Size
NASNet-A_Large_331 [3]	8
Inception-ResNet-v2 [4]	32
Inception V4 [4]	48

Table 3: Batch sizes for each models for GPUs with 11 GB memory.

Model	min./epoch
Inception-ResNet-v2 [4]	12
Inception V4 [4]	15
NASNet-A_Large_331 [3]	60

Table 4: Training speed per 1 epoch for each models.

2 epochs, and initial learning rate being 0.001 that decays by 97% every 2 epochs. The former learning rate decay rule results in a faster increase of training accuracy than the latter for all models - for Inception-ResNet-v2, after 4 epochs, the former rule yield a 65% increase in training accuracy while the latter rule yield a 43% increase.

4 Performance

The fastest model to train is Inception-ResNet-v2, followed by Inception V4, then finally NASNet-A_Large_331. The speed per epoch is shown in Table 4. The training speed is subject to the hardware, and this performance report came from training while using the GPU described in section 2. Without the GPU, the performance severely fell. Inception-ResNet-v2 takes over 4 hours per epoch while training on CPU alone. Therefore, any deep learning experiments should be done with at least one GPU.

5 Problems and Future Plans

5.1 Speed

The first few initial experiments were performed without GPU, which results in the training speed to be above 1 hour per epoch for Inception-ResNet-v2. This is much too slow to be practical, therefore a GPU is used to alleviate the training process. Using a GPU drives down the speed for Inception-ResNet-v2 to be 12 minutes.

5.2 Model Choice

The initial model of choice is NASNet-A_Large_331 due to its good performance on the ImageNet dataset. This model converges much more slowly and achieved less accuracy than other models, as seen on Table 2 and 4. Therefore, NASNet-A_Large_331 was abandoned in favor for a faster and more accurate Inception-ResNet-v2.

5.3 Dropout Rates

Initial experiments for Inception V4 uses the dropout rate of 0.2 while Inception-ResNet-v2 and NASNet-A_Large_331 uses 0.8. The accuracy performance difference is clear, as Inception V4 suffers from overfitting (the validation accuracy never goes beyond 65% while the training accuracy converges at 96%) while the other two model's validation accuracy converges above 80%. This makes it clear that: a higher dropout rate is required for these models, and these models are very sensitive to overfitting. The dropout rate of 0.5 was then experimented on Inception V4 and the validation accuracy increased to be above 80%.

In later experiments, numerous model checkpoints with good results were acquired. Attempts were made to make use of these checkpoints as starting point for further experimentations. One of those experiments were to observe the effects of different dropout rates. Most checkpoint were of models that had learnt with dropout rate of 0.8, and these were used to continue to train with, now, a dropout rate of 0.5. After 50 epochs, the model converges to the accuracy 0-3% worse than what it started out with. This made it clear that after a certain point, the model "memorized" the dataset, and any regularization method that affects the network's structure (like the dropout rate) would then affect the accuracy. Therefore, dropout rates should be kept consistent throughout each session of training.

Flexibility	Training Accuracy	Validation Accuracy
Full	99.56%	88.24%
L6	99.83%	84.86%
L8	99.51%	88.27%

Table 5: Accuracy report before and after freezing bottom layers of Inception-ResNet-v2, training with 0.5 dropout rate over 200 epochs. *Note: "Full" means no layers are frozen and the model have full flexibility, and "L<n>" means the model is frozen for all except the top "n" layers.*

5.4 Underfitting

Underfitting is a problem observed with NASNet-A_Large_331. The NASNet class of architectures is learnt from the dataset directly, thereby automating the architecture engineering process. The architecture for the model used is learnt from the CIFAR-10 dataset, and it yield state-of-the-art results or better on both the CIFAR-10 and ImageNet dataset [3]. Therefore, the fact that this model underperforms in this application is unexpected. However, looking more closely at NASNet's success with the ImageNet dataset, despite using the architecture learnt from CIFAR-10 when training ImageNet, the parameters are trained from scratch. The model used in this application uses the ImageNet's parameter checkpoints. What should have been done instead is to abandon the checkpoint and train this current dataset from scratch. This is the most likely reason that NASNet-A_Large_331 does not performing as well as it should, and more experiments need to be done in this direction.

5.5 Overfitting

Overfitting is a problem observed with both Inception-ResNet-v2 and Inception V4. Attempts to address this issue had been: changing the dropout rate, using SGD with Nesterov momentum in place learning rate optimizers, increasing the training data, and freezing the lower layers of the model. Experiments with the dropout rate and learning rate optimizers had already been discussed above. Increasing the training dataset is done by not holding out any data for validation and just train on the entire dataset. This results in an improvement of 1-2% in testing accuracy for all models. Freezing the lower layers came out of the understanding that a model overfits when it does not have enough data or it is too expressive (hence its ability to memorize the dataset and thus overfit). All regularization can be formulated according to this narrative - e.g. dropout acts to limit the expressiveness of the network to allow more quality learning on each neuron. Freezing the lower layers of the model is simply one way to limit the model's expressiveness by preserving all the good work that the pre-trained model had already done in uncovering latent patterns in images. The idiosyncrasy of the images for the current application are thus dealt with in the upper layer instead. As shown in Table 5, training only on the top 6 layers achieves worse accuracy than the fully trainable network, and training only on the top 8 layers achieve a better validation accuracy than a fully trainable network. This makes it clear that while too little flexibility will underfit the dataset, limited flexibility that is large enough have potential to achieve a higher accuracy. More experiments need to be performed in this direction.

5.6 Framework's Unexpected Behaviors

The first issue is Keras sometimes produce a higher validation accuracy than the training accuracy. The reason for this is because the way Keras calculates the accuracy measures. The training accuracy is calculated while all the regularizers are turned on, including the dropout regularizer, while the validation accuracy is calculated with the regularizers turned off [6]. Therefore, naturally, the validation accuracy, with the network being able to express itself fully, will yield better result than the training accuracy. The second issue is that both Tensorflow and Keras produces a 5-10% jump in training accuracy every time training is resumed from a checkpoint. This issue is reproduceable in the authors' work environment. So far, there had been no explanation for this behavior.

References

[1] J. Chen and C.-w. Ngo, "Deep-based Ingredient Recognition for Cooking Recipe Retrieval," *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*, pp. 32–41, 2016.

[2] N. Silberman and S. Guadarrama, "TensorFlow-Slim image classification model library." https://github.com/tensorflow/models/tree/master/research/slim, 2017.

[3] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," 2017.

[4] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," 2016.

[5] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The Marginal Value of Adaptive Gradient Methods in Machine Learning," pp. 1–14, 2017.

[6] 5Ke and danidc, "Validation accuracy is always greater than training accuracy in Keras." https://stackoverflow.com/questions/45135551/validation-accuracy-is-always-greater-than-training-accuracy-in-keras, 2017.