CS5242 Project 1

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

November 15, 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 basis for constructing a CNN architecture, the best approach is transfer-learning from existing architectures. Tensorflow and Kears (with Tensorflow backend) iare 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 models in Table 1. 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, customizing 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.

Changes to each models are: reducing the output size from 1,000 (for ImageNet) to 132 for this application; 0.2, 0.5, and 0.8 dropout rates; Adam and SGD with Nesterov momentum to deal with learning rate; training for 30-200 epochs; and trying different 8, 32, and 48 batch sizes.

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.

3 Findings

The best performing model is Inception_ResNet_v2, scoring the best testing accuracy, as shown in Table 3. The maximum number of training is set to 200 epochs with early stopping implemented.

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
Inception-ResNet-v2 [4]	99.73%	88.39%
Inception V4 [4]	99.75%	87.90%
NASNet-A_Large_331 [3]	84.73%	84.12%

Table 2: Models training and validation accuracy on the food classification problem. Training on 70% data and validate on 30% held-out data, 0.8 dropout rate, using Adam optimizer.

Model	Testing Accuracy
Inception-ResNet-v2 [4]	89.41%
Inception V4 [4]	82.25%
NASNet-A_Large_331 [3]	82.60%

Table 3: Models testing accuracy on the food classification problem. All models trained on 100% of the dataset, 0.5 dropout rate. Inception-ResNet-v2 and NASNet-A_Large_331 uses Adam optimizer while Inception V4 uses SGD with Nesterov momentum. Training has upper limit of 200 epochs while employing early stopping strategy.

Usually, 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 4.

Due to the time limit, a comprehensive comparison of learning rate optimizers are not performed. Adam optimizer is used for training all models in validation experiments with the result shown in Table 2 and used on Inception-ResNet-v2 and NASNet-A_Large_331 for testing with results shown in Table 3. SGD with Nesterov momentum is used to train Inception V4 for the testing experiment with results shown in 3. 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 [5]. When using Adam, Inception V4 converges faster than when using SGD. Inception-ResNet-v2 also converges quickly, which can also be attributed to Adam. NASNet-A_Large_331 converges the slowest, despite also using Adam. Despite not having a conclusive answer as to whether to Adam or SGD with Nesterov momentum achieves better result, using Wilson [5] as ground truth, it can be extrapolated that SGD with Nesterov momentum should be better. Given this, Inception V4's accuracy not surpassing Inception-ResNet-v2 must be because Inception-ResNet-v2 is simply a better model to generalize this dataset.

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 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.

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

Table 4: Batch sizes for each models for GPUs with 11 GB memory.

Model	min./epoch
Inception V4 [4]	11.4
Inception-ResNet-v2 [4]	12.6
NASNet-A_Large_331 [3]	50.4

Table 5: Training speed per 1 epoch for each models. Result is achieved with the help of NVIDIA's GTX 1080 TI GPU. Inception V4 and Inception-ResNet-v2 have batch size of 32 for this experiment while NASNet-A_Large_331 have batch size of 8, due to memory restrictions.

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 5. The training speed is subject to the hardware set-up, and this performance report came from training while using the GPU described in section 2. Without the GPU, the performance severely fell, with Inception V4 taking over 4 hours per epoch while training on CPU.

5 Problems and Future Plans

5.1 Model Choice

The initial model of choice is NASNet-A_Large_331 due to its good performance training CIFAR-10 and ImageNet dataset. For this application, this model converges much more slowly and achieved less accuracy than other models, as seen on Table 2, 3, and 5. Therefore, NASNet-A_Large_331 was abandoned in favor for a faster and more accurate Inception-ResNet-v2.

5.2 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.

5.3 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 goal is to learn a general architecture from a dataset that can then be tailored easily to different applications by scaling some well-defined blocks in the architecture. The architecture for the model used in this application is learnt from the CIFAR-10 dataset and tailored for ImageNet by scaling up the number of convolutional blocks. It achieved state-of-the-art accuracy 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, there are two implementation steps that diverged from the current application while performing transfer-learning (which naively transfers the NASNet's ImageNet's architecture

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

Table 6: 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.*

and weights): weights for the ImageNet classification is trained from scratch and the architecture is modified. Underfitting means that the architecture does not model the data as well as it should [6], therefore efforts should be concentrated on how to tailor CIFAR-10's NASNet to fit with the food classification problem. And because the architectures are to be changed, the number of parameters are non-deterministic, therefore for whatever changes that were made to the architecture, it needs to be trained from scratch on the current dataset. Modifying the NASNet's architecture is a welcomed approach because this is the point of NASNet: to easily scale the architecture for specific applications. This presents two ways to further experiment and optimize the results: scale different blocks in NASNet and training it from scratch. The NASNet approach to architecture give rise to the idea that there exists a relationship between model complexity and dataset complexity. It is a worthwhile effort to further pursue defining this relationship, which will then give theoretical basis for scaling NASNet.

5.4 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 6, 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. This presents four ways to further experiment and optimize the results: vary the dropout rates, use SGD with Nesterov momentum [5], find more data to train, and vary the bottom layers to freeze.

5.5 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 [7]. 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] J. Patterson and A. Gibson, *Deep Learning*. O'Reily, 2007.

[7] 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.