Image Classification Transfer Learning

Definition

Project Overview

Computer vision is one of the fields where machine learning has made great strides in this decade. It involves gaining high-level understanding from digital images or videos. It can be thought of as an application of artificial intelligence to automate the tasks that the human visual system can do. Computer vision itself has many sub-domains depending on the objective one is trying to achieve. Some of these are scene reconstruction, event detection, object recognition, motion estimation, image restoration[1] etc. Object recognition can also be called object or image classification and the one we shall focus on here is with classify multiple types of images with labels and hence is a case of multi-label classification.

Image classification is important as it helps in other areas of computer vision like object detection and localization used in applications like self driving cars. Large amounts of effort have gone into research of this area and huge datasets have been collected for benchmark to measure the progress. The breakthrough happened with the introduction of Convolution Neural Networks(CNNs) for image classification task and LeNet architecture[2] was applied to classify MNIST dataset. More bigger datasets like ImageNet[3] came after that and the architectures applied on this benchmark dataset changed the course of AI. Some state of the art CNN architectures used on ImageNet are AlexNet[4], ZFNet[5], VGGNet[6], GoogLeNet[7], ResNet[8] etc. The history shows the importance and efforts put in the task of image classification. We shall use CIFAR-100 dataset as our input data and is available for download here [9].

Problem Statement

Given a dataset of images with multiple categories, train a model to classify the images into their respective categories. The problem is best resolved by applying CNN but finding a good architecture is the challenge. In our particular case, our goal is to correctly classify as many as CIFAR-100 images as possible.

Many state of the art CNN architectures have been designed and used to get high classification performance on CIFAR-100. The top performing benchmarks available are listed in [12]. Deep neural networks like DenseNet and Wide ResNet have reported over 80% accuracy on the test set. But these architectures have high number of parameters and require high computing power like GPUs and also long duration to train. So, we would explore other options like transfer learning with bottleneck features extraction and fine tuning to train a model in shorter period of time and also achieve a decent performance.

Metrics

Different metrics exist for comparing performance of deep learning models and their suitability depends on the distribution of data to be examined. The CIFAR-100 dataset has equal number of images for each class and superclass defined. So, the data is uniformly distributed among the categories and we do not have to consider metrics like precision, recall, F-score, etc. that are essential for skewed distribution. We shall use accuracy as our metric for comparing the performance of models. Accuracy is defined as

$$accuracy \ = \ correct\ predictions\ /\ total\ predictions$$

We shall calculate accuracies on both the sets of training examples and test examples. We shall mention training accuracy as accuracy and test accuracy as validation accuracy in our code and notebooks. The performance shall be compared based on the validation accuracy as the models can very easily overfit on the training data. This distinction shall be same for both the coarse label and fine label predictions and the accuracies shall be calculated separately.

Analysis

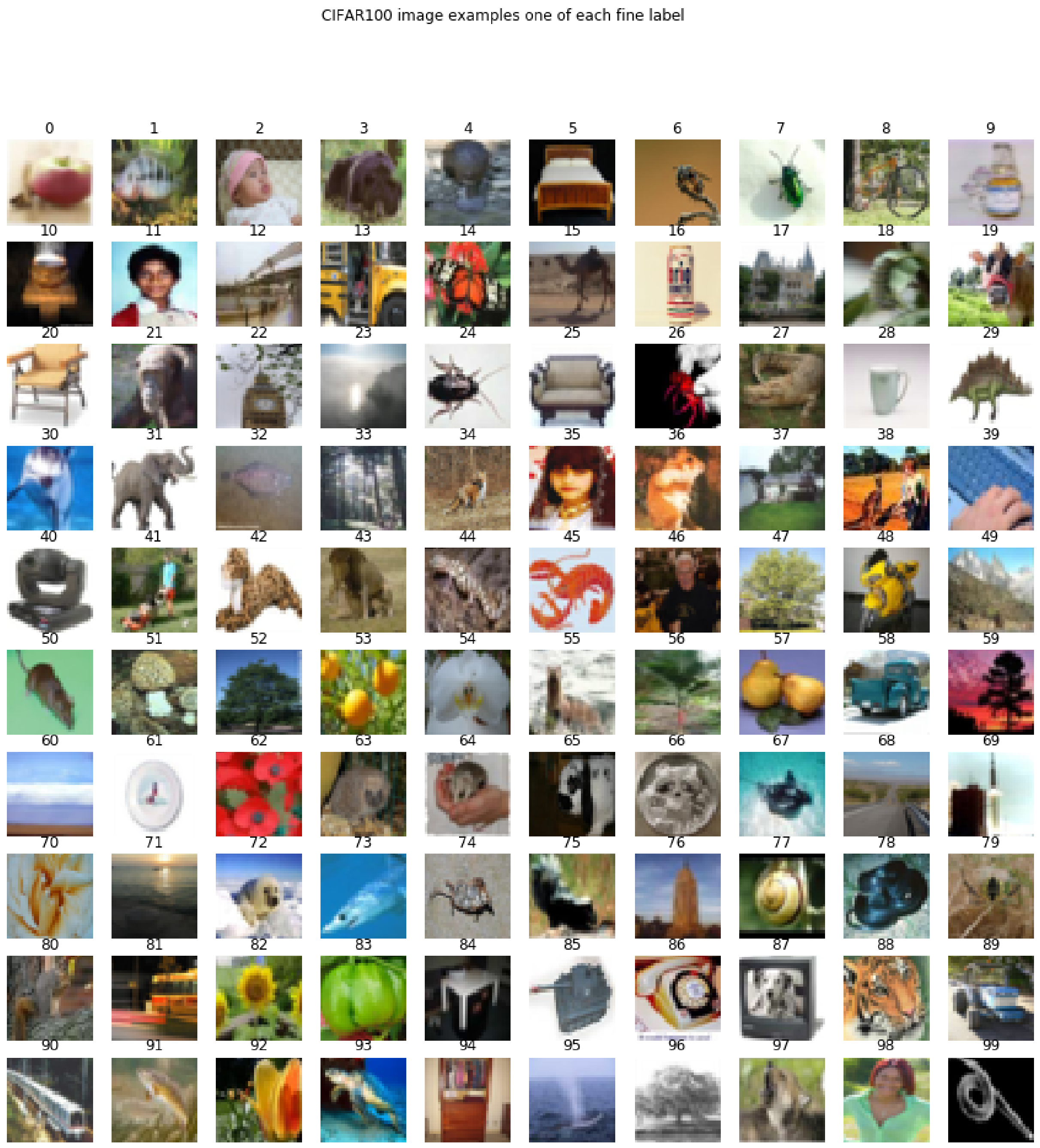
Data Exploration

CIFAR-100 is a labeled subset of the 80 million tiny images dataset[10] and was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton[11]. The dataset consists of 60000 32x32 colour images of which 50000 are training images and 10000 are test images. CIFAR-100 has 100 classes with 600 images per class and 20 superclasses.

The classes are stored as fine labels and superclasses as coarse labels. So, each CIFAR-100 image has two labels associated with it. We have done classification for both the labels in our code. The list of fine and coarse label categories can be viewed in the data_exploration.ipynb notebook at our code repository[13] in github. We also have listed samples belonging to each one of the fine labels there.

Exploratory Visualisation

There is not much statistical data of significance to visualize in the CIFAR-100 dataset, as there are 100 classes with 600 images belonging to each class. However, each superclass has different number of classes and the superclass to class mapping is shown in [9]. The following image consists of sample images belonging to each one of the 100 classes with the class index to provide us a glimpse of the data. The index to class name mappings are available at data_exploration.ipynb in [13].



Algorithms and Techniques

We have trained a simple CNN and also state of the art Wide ResNet from scratch for comparison purposes. But our prime focus here is to explore the techniques of transfer learning using bottleneck feature extraction and fine tuning.

Training SoTA CNN models from scratch on very large datasets would surely give better performance than the techniques discussed. But they most of the times require high computational power setups with GPUs and also a long duration to train. They may even give poor performance if the dataset is not large enough to learn the parameters. So, the techniques discussed may help us to leverage pre-trained models parameters to our advantage. We may use the parameters of these pre-trained models to learn features of our dataset in smaller duration of time and less computational power, and even with smaller datasets. Various possible scenarios along with suitable techniques are discussed in [14].

Bottleneck Features Extraction

This approach is suitable for datasets similar in characteristics to the pre-trained model dataset but with much smaller number of data. In this approach, we take the pre-trained model without the final fully-connected classification layer and then use it to predict the features of our input data. These features are referred to as bottleneck features or CNN codes. We then feed these features to a new classifier for training and classifying our images.

Fine Tuning

This is suitable for similar datasets but with large amounts of data. Apart from removing the last classification layer and adding a new dense layer, here we also train all the previous convolutional layers or some of the last layers. Training these layers is what we refer as fine tuning. So, time required may be more than the previous technique and is also prone to overfitting in case of small datasets.

Benchmark

We use a CNN with few convolutional layers as our benchmark model. We have taken the architecture from Keras examples available in [15]. Although the performance would be low compared to SoTA CNN models for the simple benchmark model we have chosen, it would still be better than a random guess accuracy of 1% for fine labels and 5% for coarse labels as we have 100 classes and 20 superclasses.

Methodology

Data Preprocessing

Data Preprocessing is very crucial for converting input into suitable form expected by the framework chosen and also for faster convergence.

A. Input dataset converted to (n_examples, 32, 32, 3) form as tensorflow expects ‘channels last’ dimension ordering for rgb images.[16]
B. Zero mean center and std normalization has been applied to input images as a preprocessing step in all the models.
C. Data augmentation using shifting, horizontal flip and rotation is applied for the benchmark CNN and Wide ResNet model.
D. Resizing of input images of size 32 x 32 to 224 x 224 has been done for bottleneck features extraction and fine tuning on the pretrained ResNet50 model.

Implementation

A total of four models have been trained for performance comparison. First three models have each been implemented twice, once for fine labels and once for coarse labels. Last model has only fine label implementation. Keras meta framework with Tensorflow backend is used and all documents and source code are available at [13].

1. **Simple CNN model:** Benchmark model with architecture from Keras examples[15]. Source *cifar100_cnn.py* run for fine and coarse labels and output logs and plot available at [17] and [18].
2. **Bottleneck features with ResNet50:** Pretrained ResNet50 model on ImageNet from Keras applications[19] is used as base model. Input image had to be resized to 224x224 as expected by base model. It caused memory issues in Keras ImageDataGenerator as it converts image dtype to float from uint8 and leads to machine crash or out of memory error. So, did not use data augmentation and created a custom generator to resize and do preprocessing steps like zero centering and scaling on each image. Also used smaller batch sizes in generator to avoid memory issues. *bottleneck_features_100.ipynb* and *bottleneck_features_20.ipynb* are the fine and coarse label notebook implementations.
3. **Fine tuning with ResNet50:** Base model is same pretrained ResNet50 model as in case of bottleneck features above. Memory issues and resolution are same as above. We freeze all the previous layers before the last residual layer and then fine tune the latter. *fine_tuning_100.ipynb* and *fine_tuning_20.ipynb* are the fine and coarse label notebook implementations.

4. **Wide ResNet 28-8:** Wide Residual Networks[20] have attained over 82% accuracy on the CIFAR-100 dataset. WRN model with 28 depth and 8 widen factor using Keras contrib[21] with modified implementation as in [22] is trained from scratch for few epochs only for fine labels for training time and performance comparison. *wide_resnet_100.ipynb* is the notebook implementation.

Results

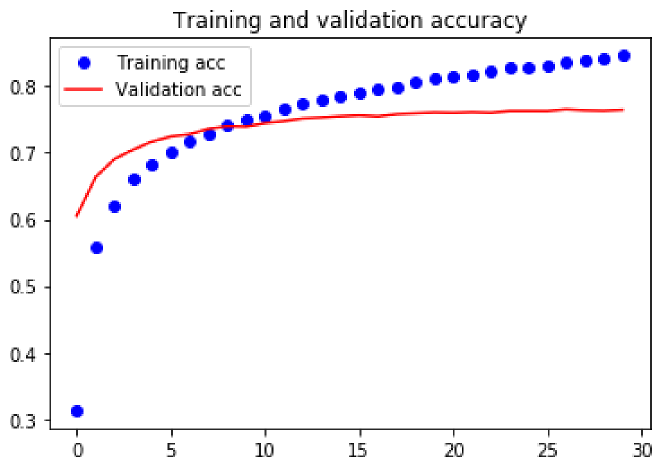
Model Evaluation and Validation

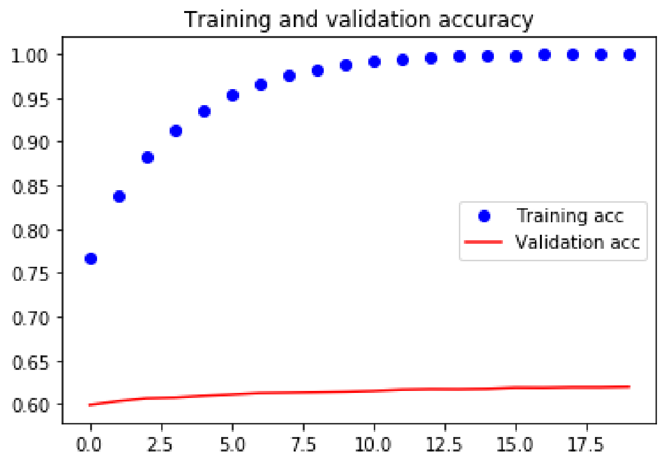
Table below shows the validation accuracy in percentage obtained by the models for coarse and fine labels.

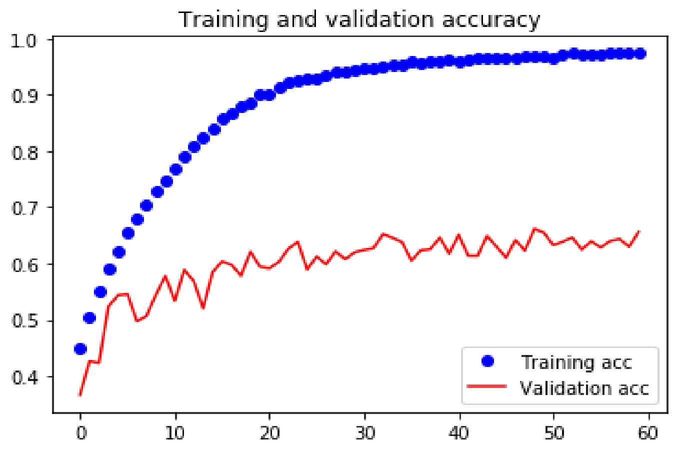
	Benchmark CNN	Bottleneck Features	Fine tuning	WRN 28-8
Fine	44.30	76.41	61.93	65.63
Coarse	56.40	84.87	75.24	-

Epochs	100	30	20	60

Benchmark models mainly report and focus on accuracies of fine labels and we shall do the same here. All future references of performance comparison shall be in the context of fine labels only. Validation accuracy and training accuracy plots for bottleneck, fine tuning and WRN are shown in respective order below.







The models have been trained on FloydHub Tesla K80 free GPU version and Paperspace P4000 paid GPU machine. The approximate training time in seconds of the models are listed in table below.

	Benchmark CNN	Bottleneck	Fine Tuning	WRN 28-8
Tesla K80	23	-	-	-
P4000	-	15	397	315

All the three models result in higher validation accuracies in less number of epochs than the naive benchmark CNN model. Training time is high for fine tuning and wrn model but bottleneck has comparable training time per epoch to the benchmark. Fine tuning starts to overfit with very less number of epochs.

Justification

The validation accuracies along with the plots show that bottleneck feature transfer learning achieves the highest accuracy without overfitting in the least amount of epochs. WRN has less performance because we only trained for 60 epochs. [23] has reported over 82% validation accuracy for fine labels using learning rate decay and trained for about 100-200 epochs. We trained it to compare its training time with other techniques.

The results show that transfer learning with bottleneck features is the best method to use in our case of CIFAR-100 dataset as it achieves performance comparable to many good benchmark models[12] with few epochs and less time. The computation could even be performed with less computational resources like a CPU as is evident from the training times. Fine tuning and WRN would require computational machines like GPUs and take over 5 mins per epoch even in GPUs. The results match the concepts of bottleneck features and fine tuning as explained in references [14]. Keras implementation details referred from [24] - [27].

Conclusion

Reflection

[14] presents the theory and concepts of training models on new datasets and how to approach them. It enumerates the possible scenarios and possible techniques to apply depending on the size and characteristics of dataset. It advises how to take advantage of models and weights trained by other people and leverage that to our use cases. We have implemented the first two techniques of bottleneck features or CNN codes and fine tuning to evaluate how they fare in practice. The CIFAR-100 dataset has much smaller number of images than the ImageNet but similar goal and characteristics. So, it falls under the category of CNN codes. Our results show the applicability of the techniques and matches the theory. It also validates the second point of fine tuning overfitting very quickly on the small CIFAR-100 dataset.

Improvement

For further improvement in validation accuracy of the best technique, transfer learning with bottleneck features, we could try other classifiers like SVM. Also we could train for more epochs and use other regularisation techniques or different values of dropout to prevent overfitting. We could experiment with other optimizers and different values of hyperparameters like learning rate. There are many hyperparameters which could be tuned to come up with higher performance.

References:

1. https://en.wikipedia.org/wiki/Computer_vision
2. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998d). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.
3. Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.
4. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
5. Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European Conference on Computer Vision. Springer International Publishing, 2014.
6. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014).
7. Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
8. He, Kaiming, et al. "Deep residual learning for image recognition." arXiv preprint arXiv:1512.03385 (2015).
9. https://www.cs.toronto.edu/~kriz/cifar.html
10. http://groups.csail.mit.edu/vision/TinyImages
11. Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.
12. CIFAR-100 benchmarks https://benchmarks.ai/cifar-100
13. https://github.com/lapjarn/capstone_udactiy
14. http://cs231n.github.io/transfer-learning/
15. Keras examples https://github.com/keras-team/keras/tree/master/examples
16. https://github.com/guillaume-chevalier/caffe-cifar-10-and-cifar-100-datasets-preprocessed-to-HDF5
17. https://www.floydhub.com/lapjarn/projects/cifar100_capstone/5
18. https://www.floydhub.com/lapjarn/projects/cifar100_capstone/6
19. https://keras.io/applications/
20. Wide Residual Networks (BMVC 2016) http://arxiv.org/abs/1605.07146 by Sergey Zagoruyko and Nikos Komodakis.
21. https://github.com/keras-team/keras-contrib
22. https://github.com/titu1994/Wide-Residual-Networks
23. Torch implementation of WRN https://github.com/szagoruyko/wide-residual-networks
24. Keras blog at https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
25. CIFAR-10 experiments https://github.com/rnoxy/cifar10-cnn
26. https://www.learnopencv.com/keras-tutorial-transfer-learning-using-pre-trained-models/
27. https://www.learnopencv.com/keras-tutorial-fine-tuning-using-pre-trained-models/