Evaluation of CNN Fine-Tuning for Image Classification

Zachary Maybury

https://github.com/zmaybury/91-523-f15-project

Problem

Convolutional Neural Networks (CNNs) achieve state of the art accuracy when presented with a lot of data

Many datasets related to classification tasks are too small to adequately train a CNN

CNN fine-tuning methods are largely “trial & error”

Goal: Evaluate tuning methods to discover general guidelines for successful fine-tuning

input: image

output: image classification

Background

Recognizing Image Style, Karayev et al.

How transferable are features in deep neural networks?, Yosinski et al.

CNN Features off-the-shelf: an Astounding baseline for Recognition, Razavian et al.

Approach

Create fine-tuned models adapted from “Imagenet” CNN, varying number of samples, learning rates, iterations, and layers to be fine-tuned

Evaluate models to determine performance

Identify important parameters and develop “best-practices” for CNN fine-tuning tasks

Approach - Creating Fine-tuned Models:

Creating fine tuned models requires the following:

1. Creation of training/validation image databases
2. Creation of solver prototxt file, specifying network training parameters
3. Creation of training prototxt file, specifying network topology for training and validation network
4. Creation of deployment prototxt file, specifying network topology for production network
5. Use the above network with Caffe and “Imagenet” CNN model to create fine-tuned network

Approach - Evaluating Fine-Tuned Models

Creation of C++ Program to:

1. Load CNN
2. Process set of images through CNN
3. Determination of TP, FP, and Overall Network Accuracy
4. Creation of Confusion Matrix to illustrate per-class accuracy

Approach - Identify Important CNN Parameters

Fine-tune CNNs while varying the following parameters:

1. Number of Samples
2. Learning rate
3. Training Iterations

Each of the above experiments will be run with two different network fine-tuning configurations:

1. All layers fine-tuned
2. Last two layers fine-tuned, other layers held constant

Data

PubFig: Public Figures Face Database - ~59k images of 200 people

CUB-200_2011 Dataset - ~12k images of 200 bird species

102 Flower Dataset - ~8k images of 102 flower species

Combined Dataset - above datasets combined into 3-class dataset

Experiment setup

Combined (3 classes) and Bird (200 classes) datasets used for experiments

Each dataset initially split into 50/50 for training and testing

Training data further split into training and validation sets for CNN training.

Evaluation Metric: Network Accuracy

Baseline Accuracy

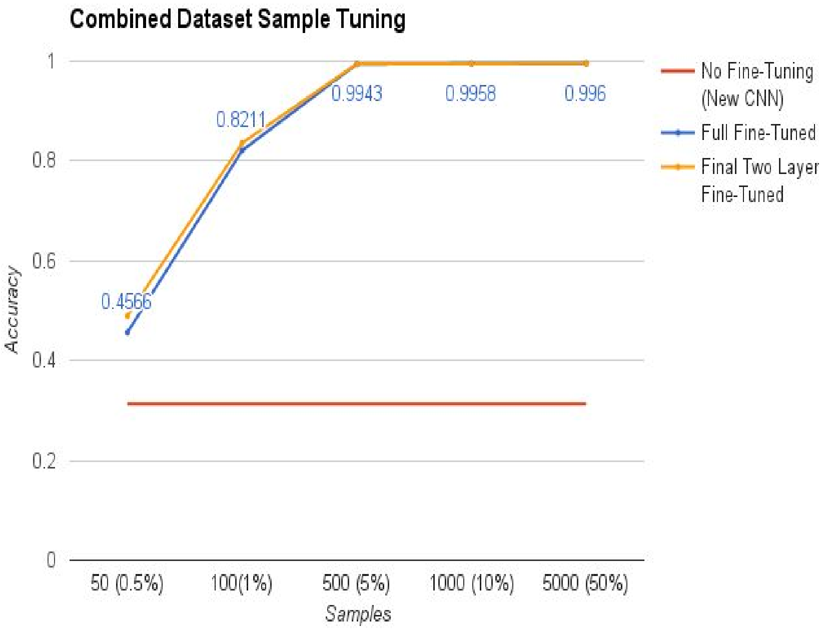
Random Guessing:

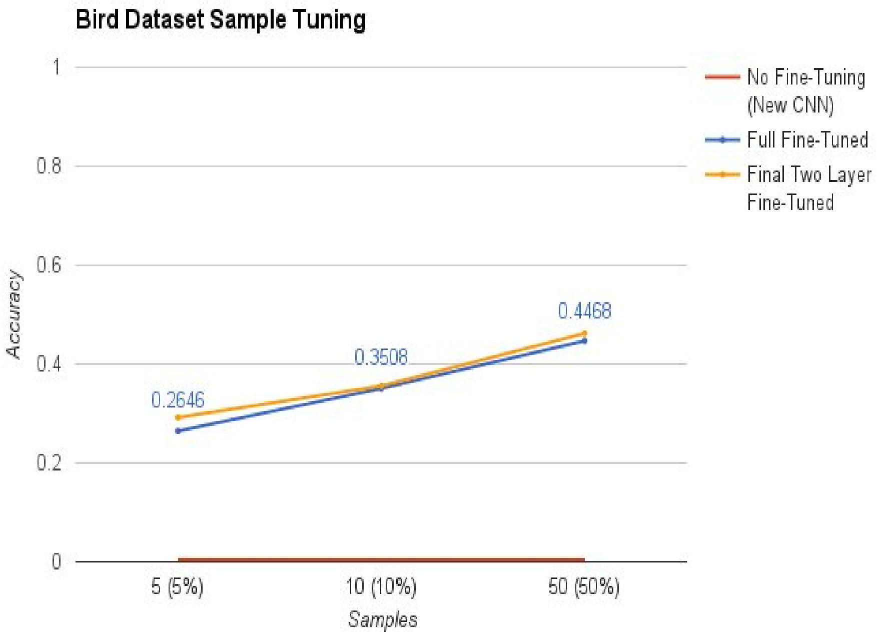
Combined Dataset: 33%   Bird Dataset: 0.5% (1/200)

New CNN trained on train/validation set using no fine-tuning:

Combined Dataset: 31.3%   Bird Dataset: 0.51%

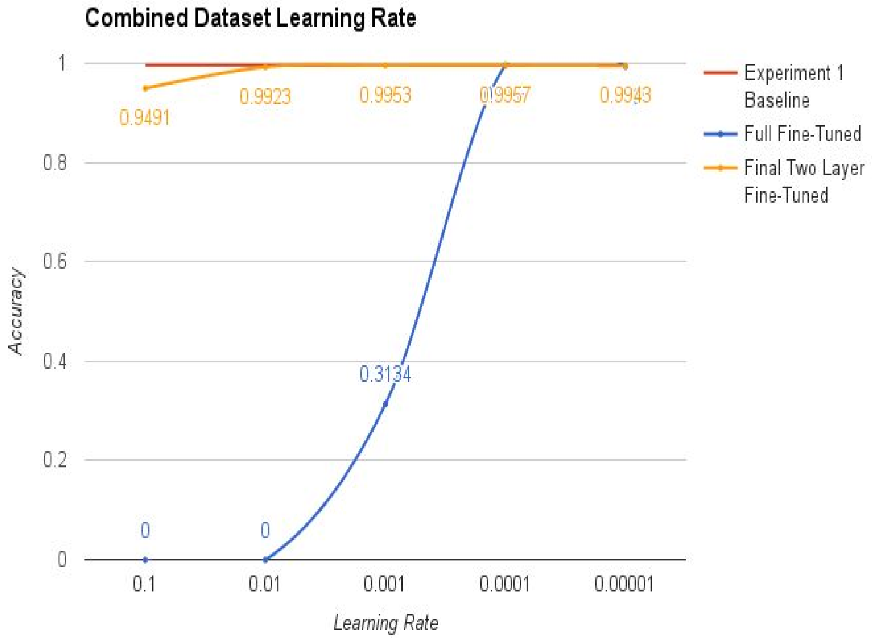
Results - Experiment 1: Number of samples

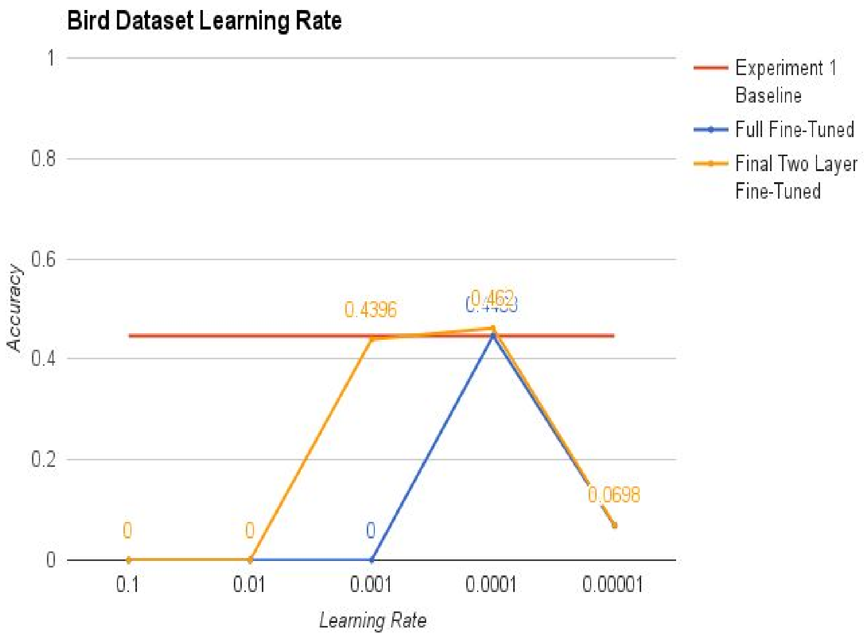




Results - Experiment 2: Learning Rate

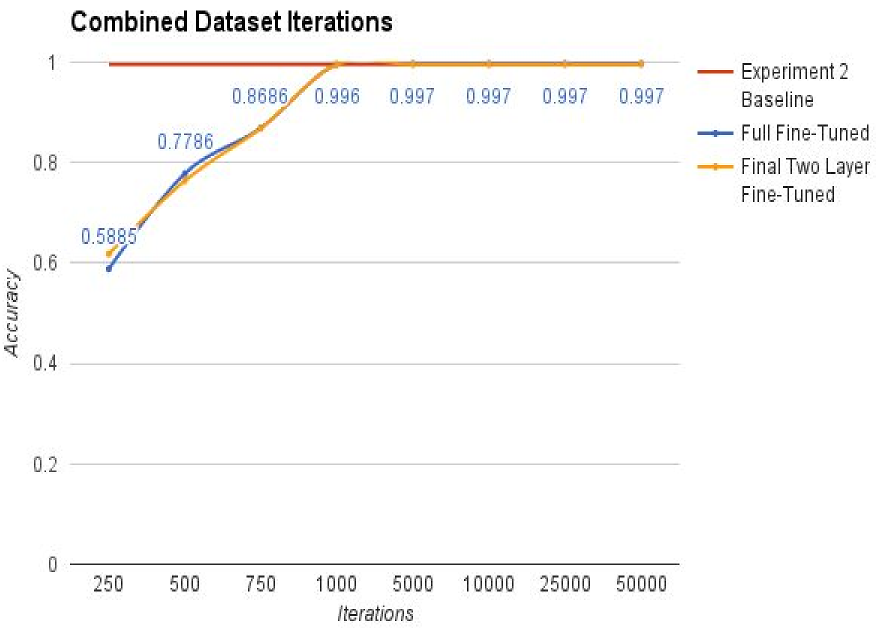
Baseline is Best Network from Experiment 1

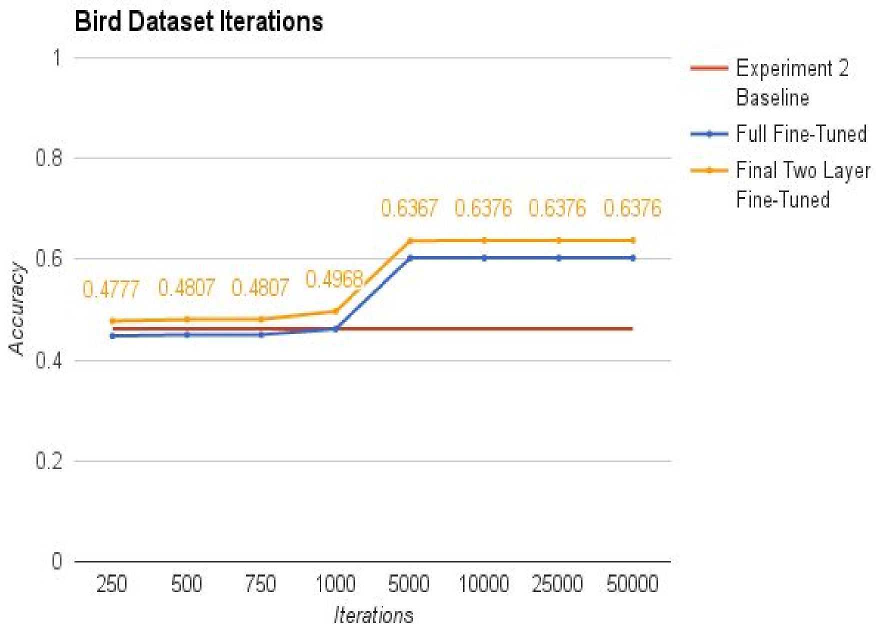




Results - Experiment 3: Iterations

Baseline is Best Network from Experiment 2

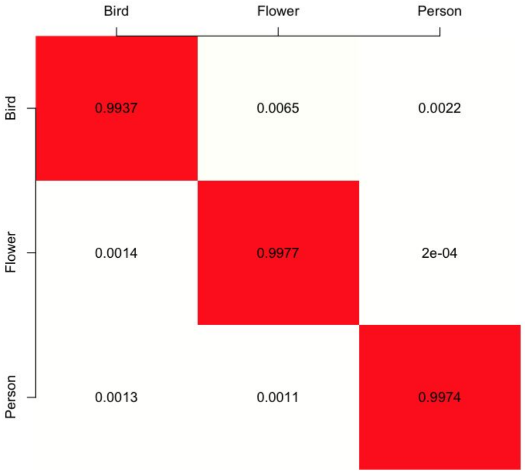




Results - Final (Best) Networks

Combined CNN (3 Classes)
Accuracy: 99.7%

Confusion Matrix

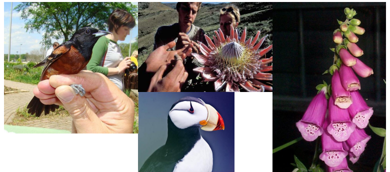


False Positives:

Birds:

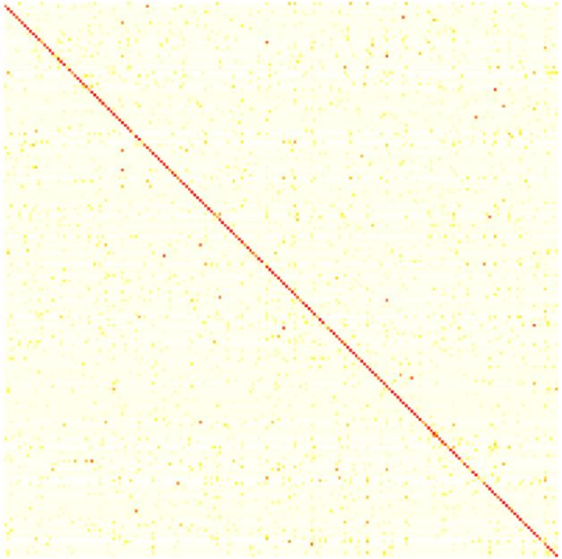
Flowers:

People:



Bird CNN (200 Classes)
Accuracy: 63.76%

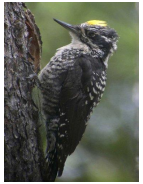
Confusion Matrix



Top Classes:

Green Violetear 90.48%

Geococcyx 88.1%



American Three Toed Woodpecker 86.67%

Conclusions

As expected, CNNs are still data hungry and benefit from utilizing as many samples as available

Fine-tuning learning rate of 0.0001 leads to the best performance, larger or smaller leads to divergence

Fine-tuning becomes stable after a few thousand iterations

Fine-tuning performance better when only tuning last two layers.