Indoor Scene Classification: A Horserace

Cory Cutsail, Chirantan Ghosh

University of Delaware

May 15, 2020

Overview

What do we mean by indoor scene classification?

Given an image taken indoors, how do we identify the type of location?

This is a hard problem – even for humans!

A More Formal Definition

"An indoor scene is the abstract of various semantic cues which include multiple objects of which classes are open-set and contextual relationships between them." - Li et al, 2019

Given a set of images X, learn mapping $f : X \mapsto Y$, $Y = \{0, 1\}^n$

Example: You see an image that contains a bookshelf. Is it in a library, a home office, a living room, a restaurant? How do we know by looking at a picture?

Why should we care about indoor scene classification?

- Autonomous navigation
- Localization of criminal activity
- Understanding user behavior in different physical settings
- Advertising and marketing

What has been done?

- Szummer & Picard, 1998, kNN + MSAR
- Payne & Singh, 2005, Benchmarking, Data Requirements
- Quattoni & Torralba, 2009 (CVPR09), Dataset, ElasticNet, Multiclass

And then come CNNs!

Data: CVPR09

- The CVPR09 dataset contains 67 Indoor categories with a total of 15620 images.

- The number of images varies across categories, with at least 100 images per category in .jpeg format.

Methodology: An Architectural Horserace

We test 4 network architectures on 4 CPU/GPU configurations in Keras/TensorFlow

Methodology: Network Architectures

- ResNet50
- ResNet101
- DenseNet121
- InceptionV3

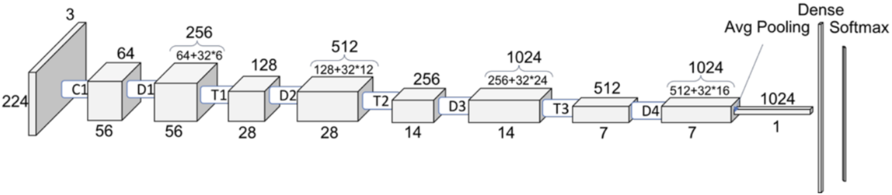


Figure: DenseNet-121 Architecture Visualization from Towards Data Science. D_i is a dense block, T_i is a transition block

Methodology: Configurations

Configurations:

- CPU1: Intel i9, TensorFlow back-end
- CPU2: Intel i7, TensorFlow back-end
- GPU1: GeForce RTX 2080 max Q, TensorFlow-GPU back-end
- GPU2: AMD Radeon Pro 575, PlaidML back-end

Early problems

- The CPUs are both too slow to run to completion
  - Epoch runtimes between 15 and 20 minutes
- nan loss and slow convergence on PlaidML back-end
  - Potentially due to random initializations
  - Required several kernel restarts
- Bad accuracy on PlaidML back-end
  - 2017 PR suggests poor accuracy is a known issue

Solution: model accuracy results taken from GeForce RTX 2080 max Q w. TF GPU BE. Accuracy and compute time results for first 10 epochs shown for all available configurations and architectures

Results: Fully Trained Architectures

Model	Epochs	Training Acc.	Val. Acc.	MS Per Epoch
DenseNet121	134	0.9457	0.4641	226s
ResNet101	109	0.9100	0.0326	245s
ResNet50	155	0.8828	0.0392	248s
InceptionV3	234	0.9062	0.5022	233s

Immediate Takeaways

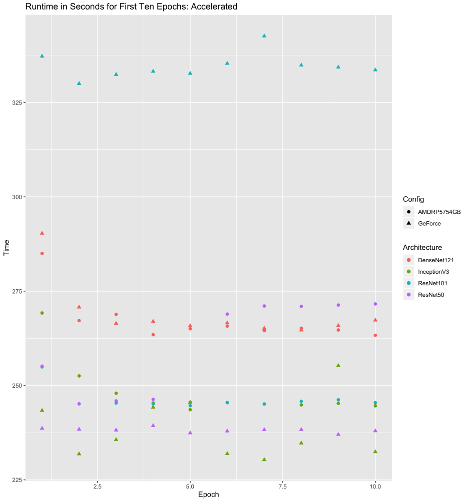
- Variation in number of epochs to convergence
- Roughly consistent in training accuracy at convergence (good)!
- Wild variation in validation accuracy at convergence on ResNet architectures

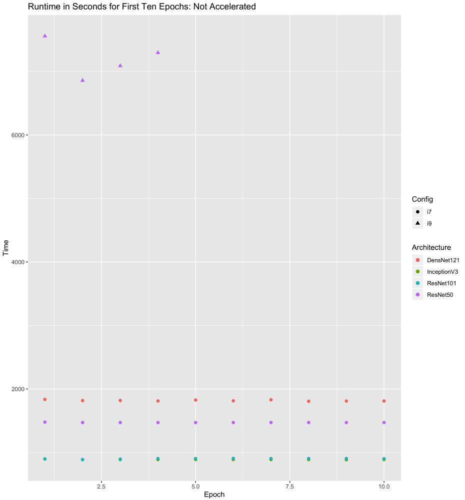
What is going on with our validation accuracy?

A few different things:

- Keras' default methodology for splitting into training and validation sets was used
  - Uses user-specified proportion to split dataset based on ordering
  - i.e., user specifies 90% of data as training, 10% as validation, $\Rightarrow$ the validation set contains the 'last' 10% of the dataset
  - Depending on how the data is loaded (sequentially by category here), the validation set may contain only a small fraction of the total number of classes.
- The ResNet models are drastically overfitting
  - This has to do with layer 'freezing'
    - i.e., specifying certain layers that should not be updated when training
  - ResNet models are known to overfit in certain problem spaces, and the recommended solution is to freeze the batch normalization layer
  - This was not done in our 'horserace', as we are comparing common out-of-the-box models
  - Experimenting with freezing different layers of each out-of-the-box model is a natural next step for this project

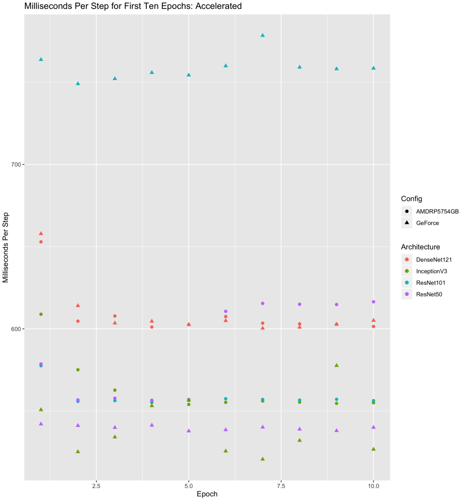
Results: Model Runtime Across Architectures, Config

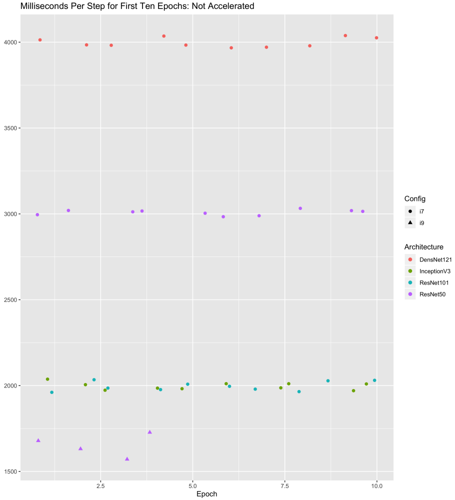




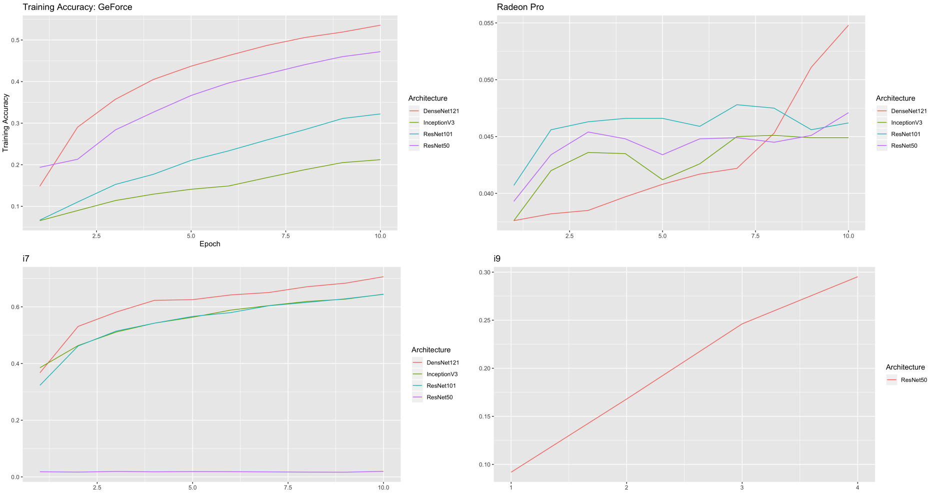
Note that these figures have y-axes that are an order of magnitude apart

Results: MS per Step Across Architectures, Config

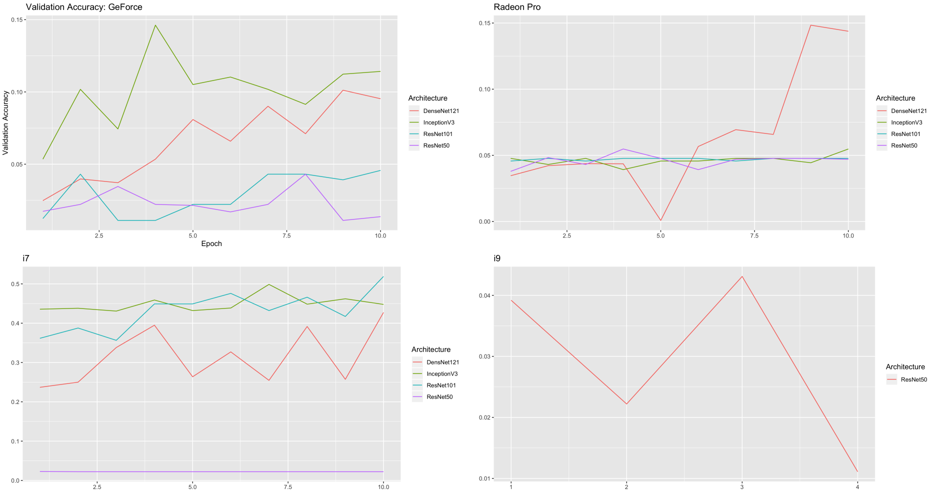




Results: Training Accuracy Across Architectures, Config



Results: Validation Accuracy Across Architectures, Config



Conclusions

- PlaidML: Easy to set up, results may vary
- Standard TF-GPU back-end with InceptionV3 architecture wins across the board in early stages
  - Among the lowest in runtime, highest in validation accuracy
  - Final validation accuracy is highest, number of epochs to converge is very high, runtime per epoch is average
- DensNet121 took the longest to converge in terms of number of epochs, but comes in second from an accuracy perspective
  - Time per epoch is much faster than the others