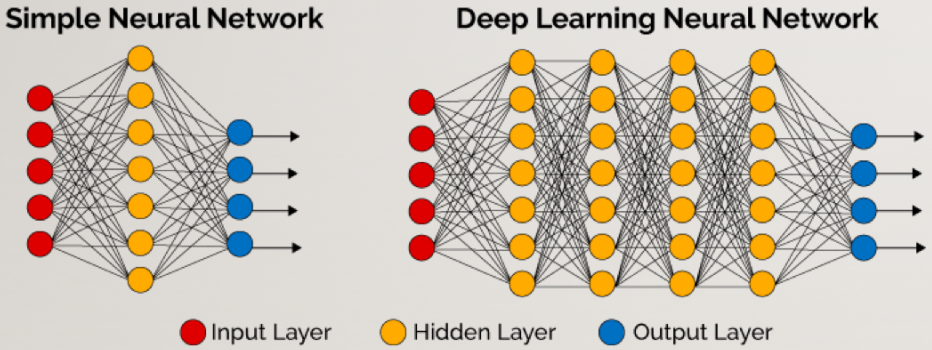


IMAGE CLASSIFICATION IN DEEP LEARNING

CUNY 698 – MS in Data Science

Walt Wells – Spring 2018

INTRODUCTION

Deep learning techniques are one of the most exciting and effective new approaches to building a machine learning system that can operate with accuracy and precision in a complex real-world environment. This is important because for difficult problems with highly dimensional data like image classification, deep learning models like Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs) are among the best performers in benchmarked competitions and are even outpacing their human subject matter expert counterparts at image classification tasks.

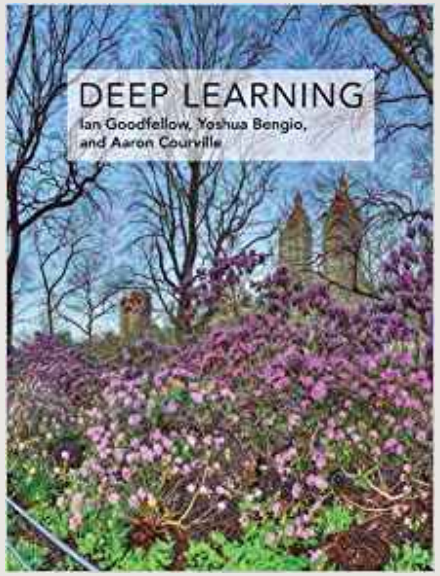
But how do they work?

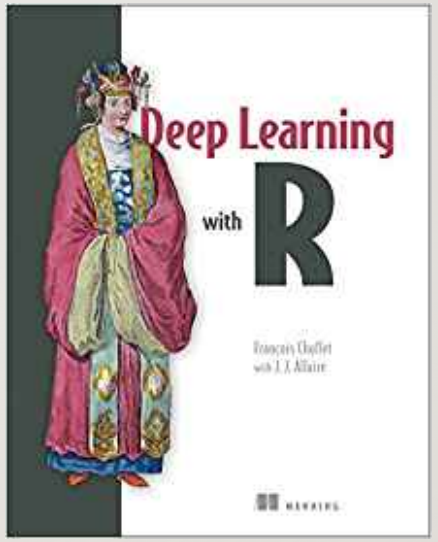
This study will seek to explore a variety of deep learning techniques by building a convolutional neural network and conducting image classification over a popular benchmarking dataset.

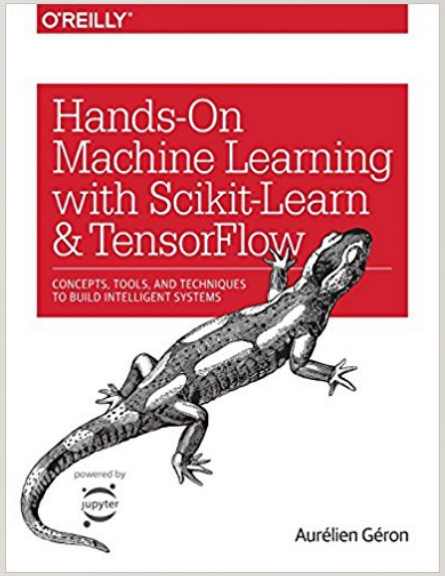
LITERATURE REVIEW

DEEP LEARNING SURVEY

- Overview of Deep Learning Techniques
- How to Build, Train, and Test Models in Python and R
- Computational Architecture and Data Engineering







Geoffrey Hinton and Yann LeCunn

IMAGE CLASSIFICATION USING DEEP LEARNING

- State of the Field
- Innovative Academic Papers
- Major Touchstones

MATERIALS AND METHODS

THE DATASET

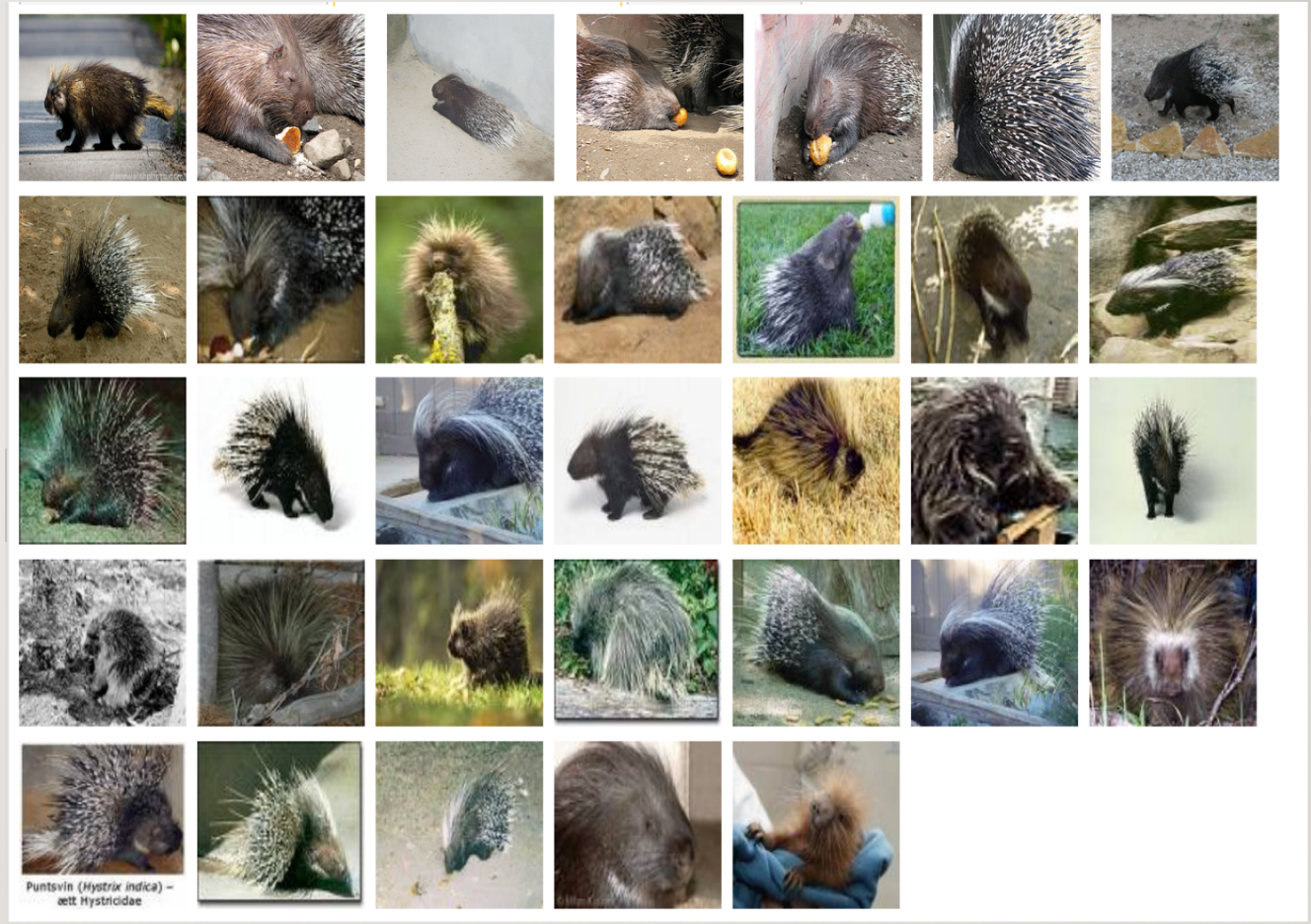
ImageNet and TinyImageNet

- Stanford Vision Lab
- Major Touchstone in Field of Computer Vision

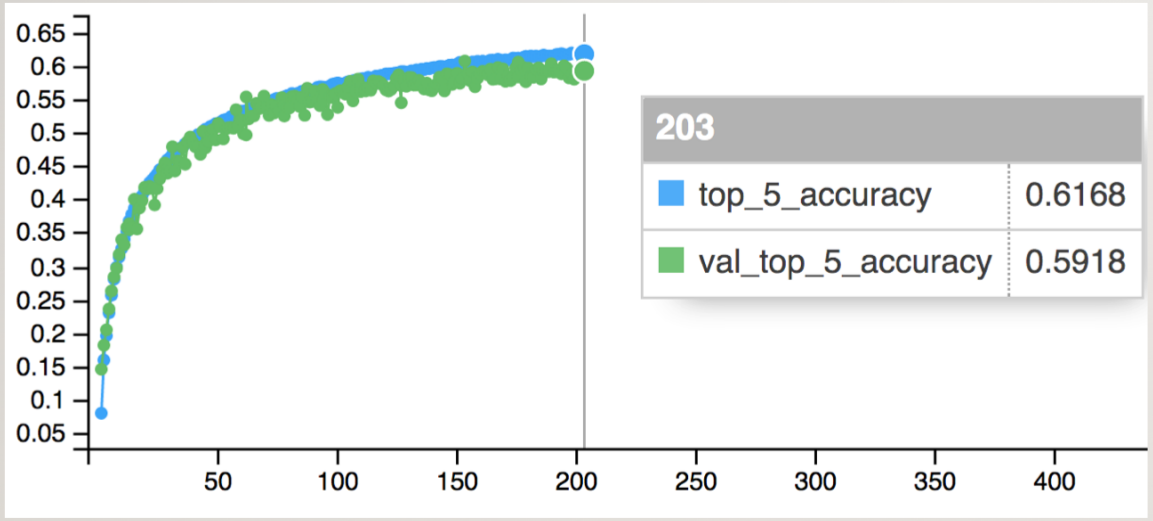
1000 / 200 Labeled Classes

Split into Training and Validation Sets

~150 GiB / ~1 GiB



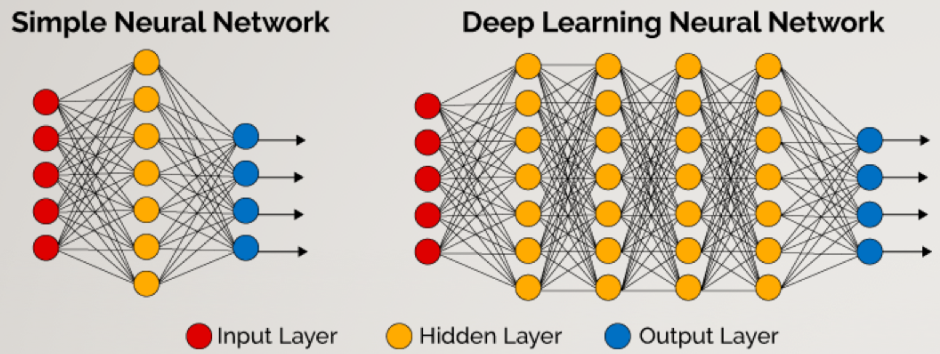
THE TASK

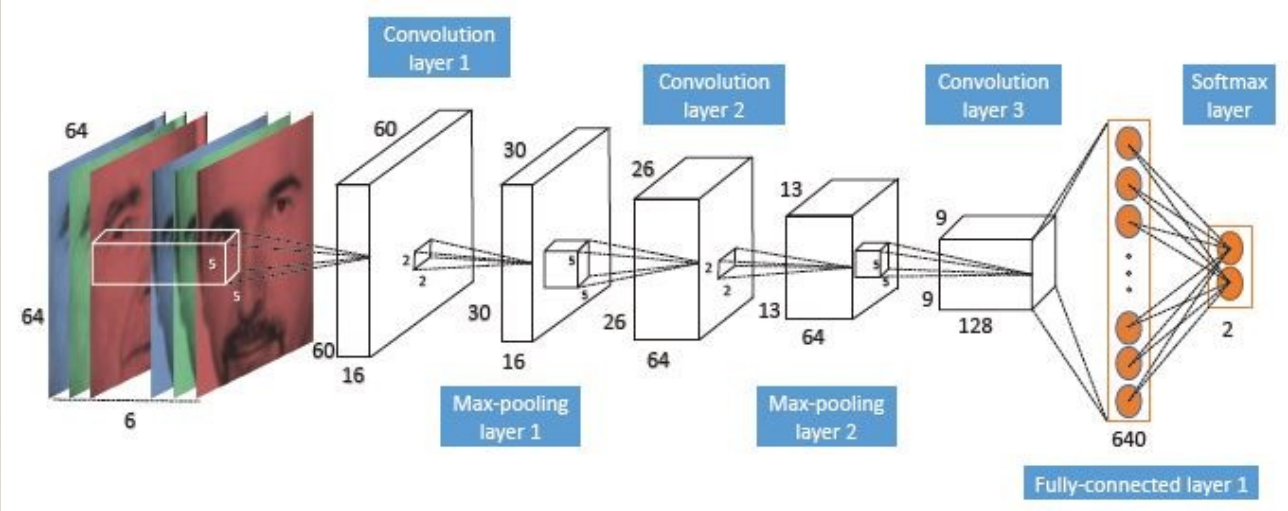


- Build a model that, when given an image as input, will produce a list of 5 classes that it believes represent some aspect of the input image.
- We will judge success and improve our model based on this ”top 5 accuracy” metric.
- Since an image may contain multiple classes, (has a fish and a cat and a fork and a…) this method allows a model to identify multiple objects in an image and not be penalized if one of the objects identified was in fact present, but not included in the ground truth.

THE TOOLS

- We used relatively new tools and libraries in R (keras and cloudml) released or updated in 2018.
- Keras with Tensorflow as the backend has excellent and intuitive functions for designing model architecture, training and optimizing models, data generation, data augmentation, and hyper parameter tuning.
- The cloudml package provides a simple way to submit R Keras jobs to the Google ML Engine for training, testing, and prediction. Using this package abstracts away the need for a data scientist to manage and tune the hardware required for deep learning, making it as simple as sending a Keras script to the Google API.





THE CONVOLUTIONAL NEURAL NETWORK

- Convolutional neural networks are a feed-forward deep learning technique that excels at image classification tasks.
- A feed-forward neural network is a series of interconnected layers that pass information forward from the input using a series of activation functions (relu, sigmoid, etc.).
- The connections between layers are weights that determine the strength of the values passed forward from the previous layer.
- Convolutions are a method extracting knowledge about a feature found in one part of an image (an eye, an ear), and being able to apply it to another image where the feature is in a different location.

RESULTS

Model 1	Model 2
CNN Layers	**CNN Layers**
$InputImage = 64 * 64 * 3$	$InputImage = 64 * 64 * 3$
$ConvLayer, Filters = 32, Kernel = 3$	$ConvLayer, Filters = 32, Kernel = 3$
$MaxPoolingLayer, Size = 2$	$ConvLayer, Filters = 32, Kernel = 3$
$ConvLayer, Filters = 64, Kernel = 3$	$MaxPoolingLayer, Size = 2$
$MaxPoolingLayer, Size = 2$	$BatchNormalizationLayer$
$ConvLayer, Filters = 128, Kernel = 3$	$DropoutLayer$
$MaxPoolingLayer, Size = 2$	$ConvLayer, Filters = 32, Kernel = 3$
$ConvLayer, Filters = 128, Kernel = 3$	$ConvLayer, Filters = 64, Kernel = 3$
$MaxPoolingLayer, Size = 2$	$MaxPoolingLayer, Size = 2$
	$BatchNormalizationLayer$
	$DropoutLayer$
	$ConvLayer, Filters = 128, Kernel = 3$
	$MaxPoolingLayer, Size = 2$
	$ConvLayer, Filters = 128, Kernel = 3$
	$MaxPoolingLayer, Size = 2$
	$BatchNormalizationLayer$
Classifier Layers	**Classifier Layers**
$FlattenLayer$	$FlattenLayer$
$DropoutLayer$	$DropoutLayer$
$DenseLayer, Units = 512$	$DenseLayer, Units = 512$
	$DenseLayer, Units = 1024$
$DenseLayer, Softmax$	$DenseLayer, Softmax$

THE ARCHITECTURE

- Two Different CNN Models were trained over the TinyImageNet Dataset; Roughly based on "AlexNet" architecture that did well in the 2012 ImageNet Challenge.
- Model 1 is fairly simple, with a few different convolutional and pooling layers, increasing complexity slightly as it gets deeper.
- Model 2 is a bit more complex, interspersing convolutional and pooling layers with batch normalizing and dropout layers.
- At the end of each architecture we flatten our features and attach a classifer that predicts the classes.

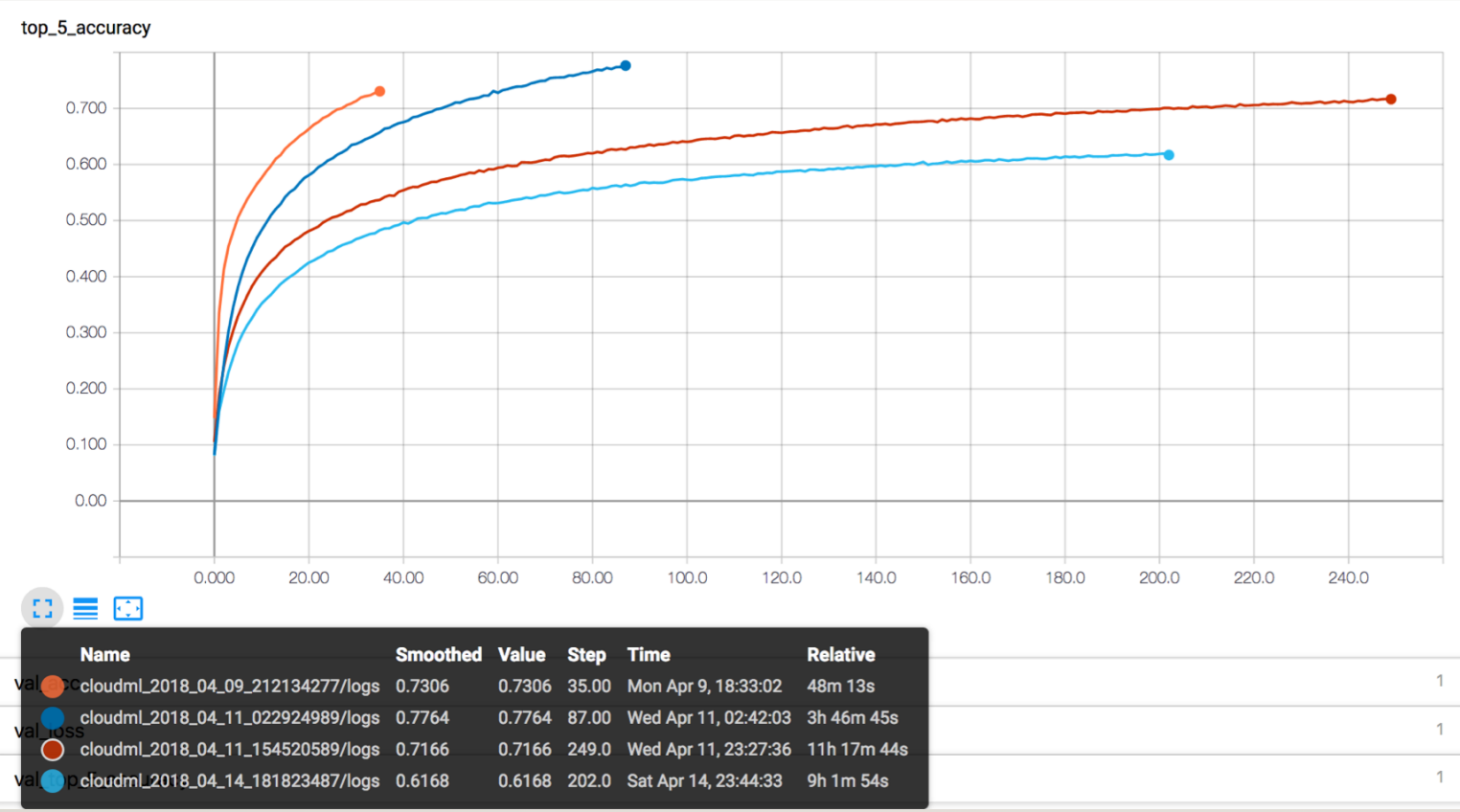
THE TUNING TRICKS

- Deep Learning models perform incredibly well on the training data, so one of the big challenges is to make sure they are not overfitting and generalize well on the holdout sets.
- Dropout: A bizarre method both developed by and denounced by the Hinton group that works well, essentially telling the model to randomly forget weights.
- Batch Normalization: Between layers, subtracting the Batch mean and divide by Batch SD.
- Data Augmentation: Morph the input image so that it essentially expands the dataset, giving the model more data to train on.

RESULTS

- In all, about 25 training and hyperparameter tuning jobs were submitted to CloudML.
- The table shows four of the best overall performers trained over the TinyImageNet dataset. There were a few models not listed here that achieved > 90% top 5 accuracy on the training set, but were overfitting badly and did not generalize well, performing poorly on the validation dataset.
- As a baseline for comparison, randomly guessing an image class from the TinyImagenet Dataset amounts to .5 top 1% accuracy, and 2.5% top 5 accuracy.
- Overall, the more complex model (model 2) w/o data augmentation has the best performance metrics, but it is worth noting that there is a 16% difference in top 5 accuracy between the training and validation datasets indicative of overfitting. What may be preferred about this model is that the top 1 accuracy is also very high, indicating the model is generally identifying classes correctly.

	Model 1	**Model 1**	**Model 2**	**Model 2**
RunID	212134277	181823487	022924989	154520589
TrainAcc	.4384	.3450	.4872	.4300
TrainTop5Acc	.7306	.6168	.7764	.7166
ValAcc	.1767	.3200	.3475	.2981
ValTop5Acc	.3974	.5918	.6086	.5674
Dropout	.5	.5	.5, .5, .75	.5, .5, .5
DataAugmentation	*FALSE*	*TRUE*	*FALSE*	*TRUE*
EarlyStoppingEpoch	38	203	88	250



THE CONCLUSION

Working with new open source tools to implement a Deep Learning CNN model for image classification, we were able to achieve a reasonable predictive classification accuracy.

Questions?