ANACONDACON

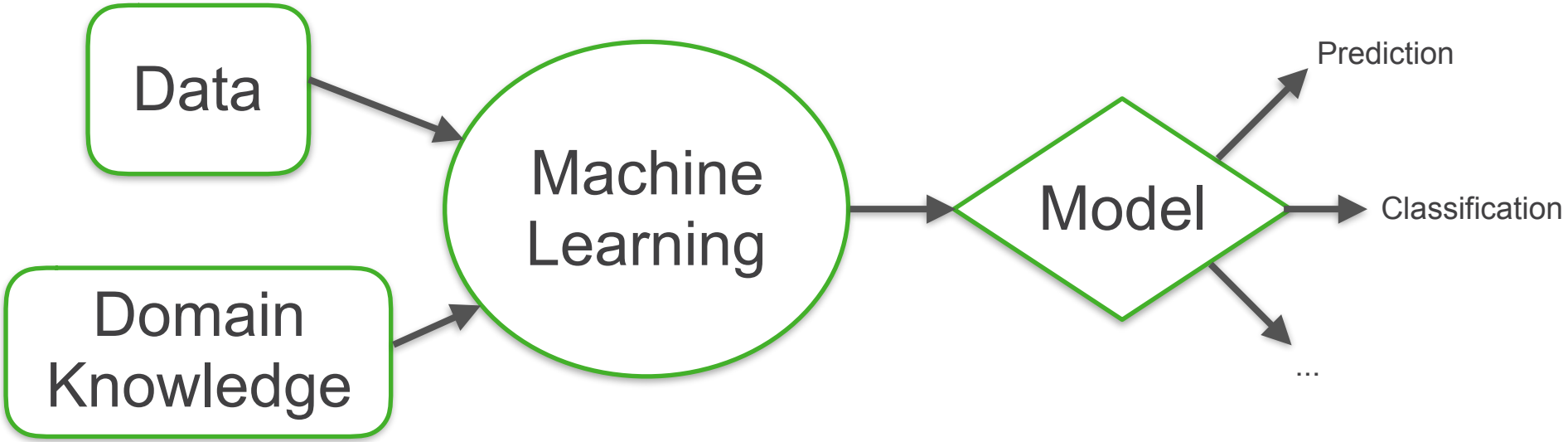
Practical Data Science and Machine Learning with GPUs

Stanley Seibert

Sr. Director of Community Innovation

Part 1: Machine Learning Basics in Anaconda

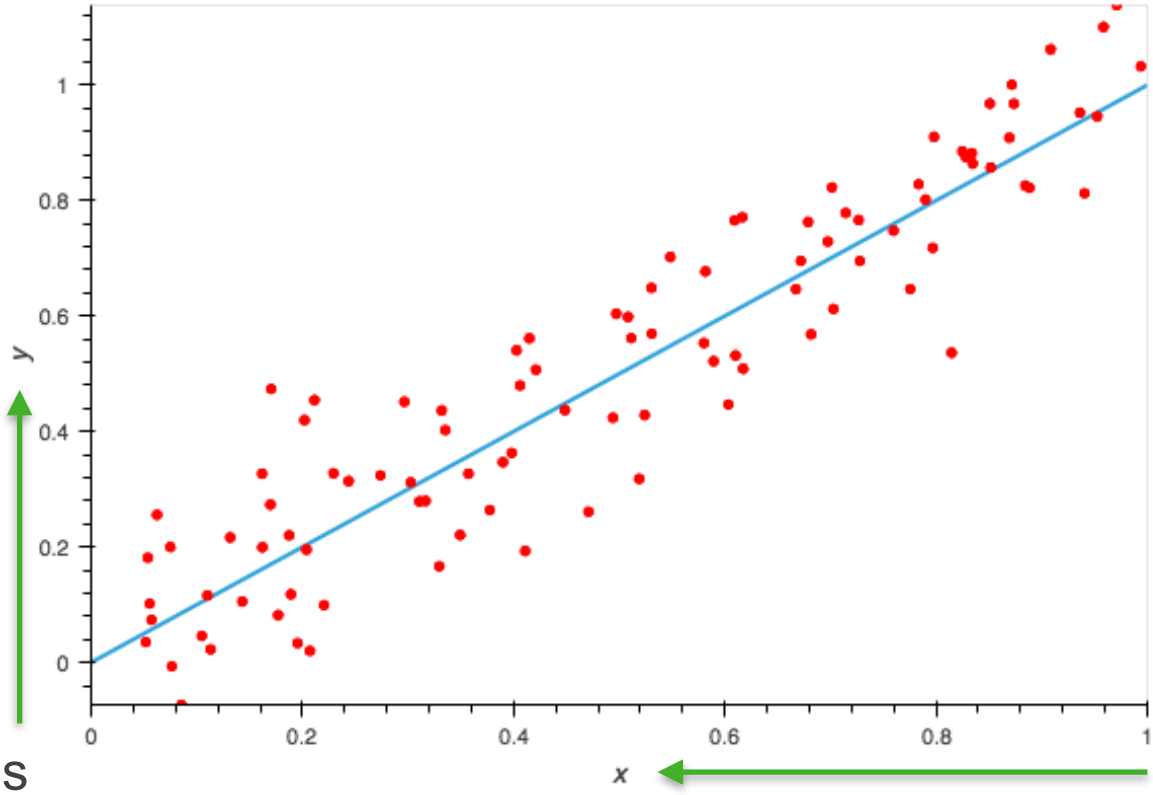
What is Machine Learning?



What can machine learning help us do?

- **Predict the future:**
  - ✓ *"How much corn will I sell next month?"*
- **Reveal hidden information:**
  - ✓ *"How many truck tires likely fail to meet our durability specification based on the temperature data collected during their manufacture?"*
- **Identify structure in large data sets:**
  - ✓ *"Can business loan applications be grouped by common attributes?"*
- **Find unusual trends:**
  - ✓ *"Which items in my grocery store have seen an abnormal sales decrease outside of historical seasonal variation?"*

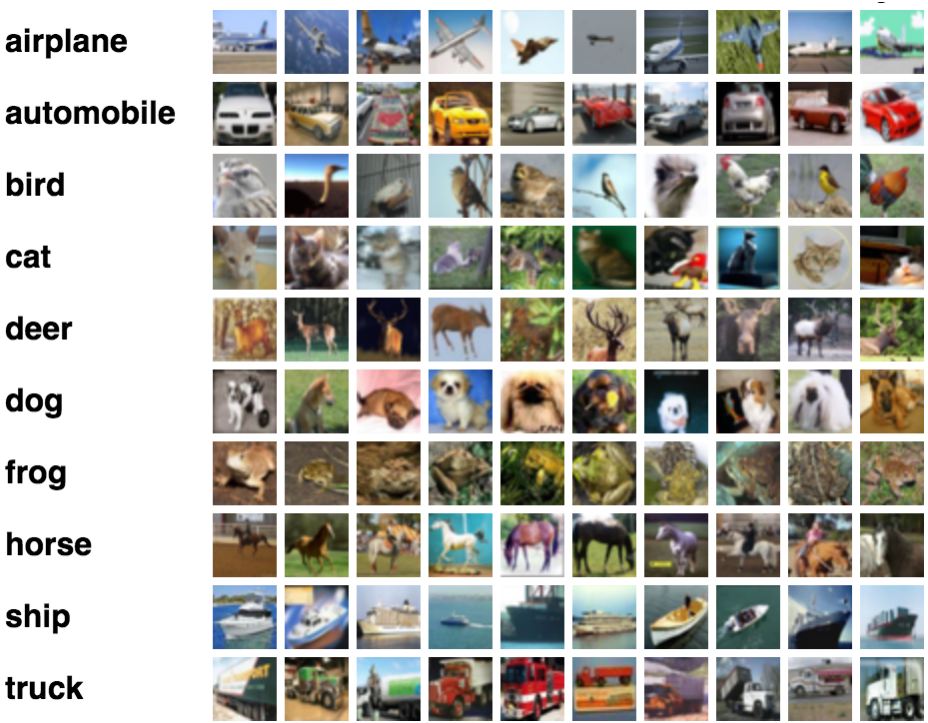
ML Categories: Regression



predict this

observe this

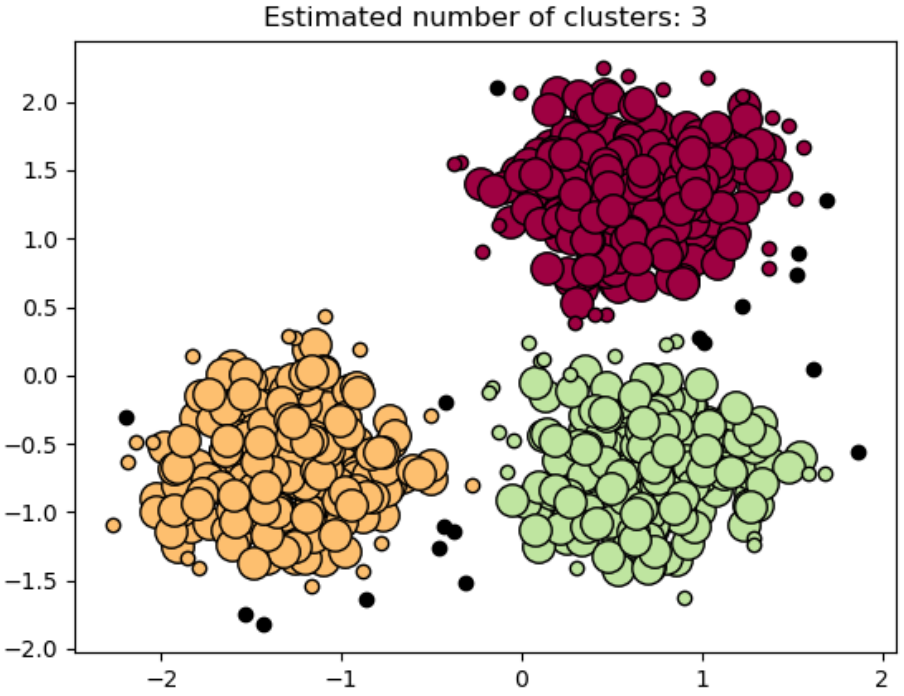
ML Categories: Regression



- Assign each sample to one or more categories.
- Categories known ahead of time
- Can be **exclusive** or **inclusive**

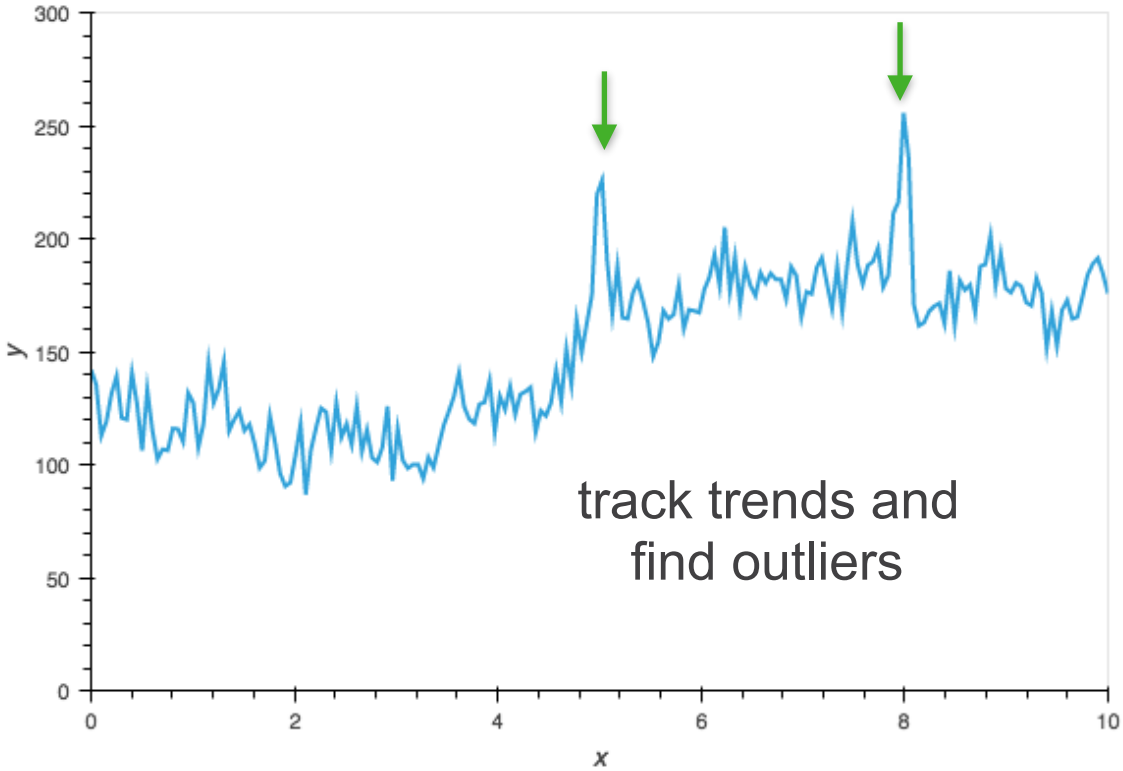
ML Categories: Clustering

- Divide a set of items into groups
- Categories are **not** known ahead of time



http://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

ML Categories: Anomaly Detection

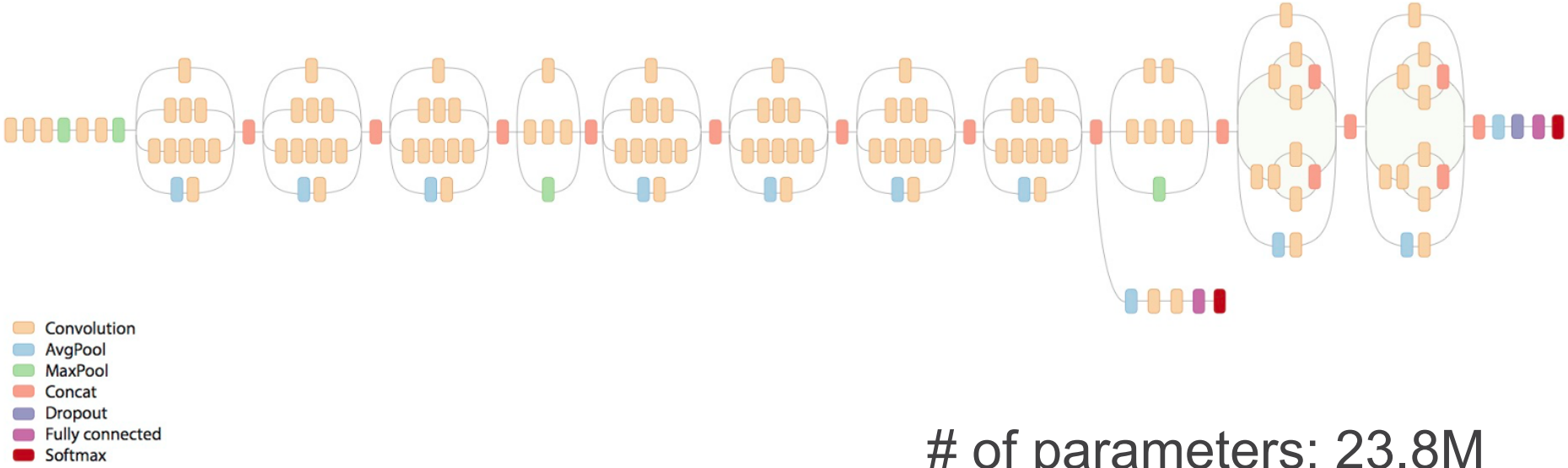


ML Categories: Generation

http://genekogan.com/works/style-transfer/

Deep Learning

Inception v3 Network



# of parameters: 23.8M
# of layers: ~140

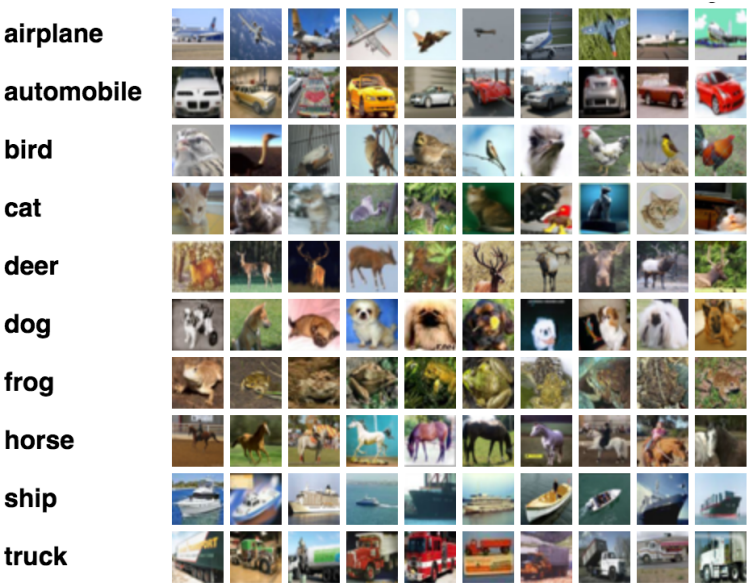
Critical ML Questions, Part 1

- **What is the purpose of the model I am trying to create?**
- What specific, quantitative model output will help me achieve that purpose?
- Are there publications, blog posts, conference talks describing a similar use case?
- What relevant data is available for this model? Can more be gathered?
- Is there additional domain knowledge that would be helpful for selecting training data?
- Are there experts in the data I can talk to?

Critical ML Questions, Part 2

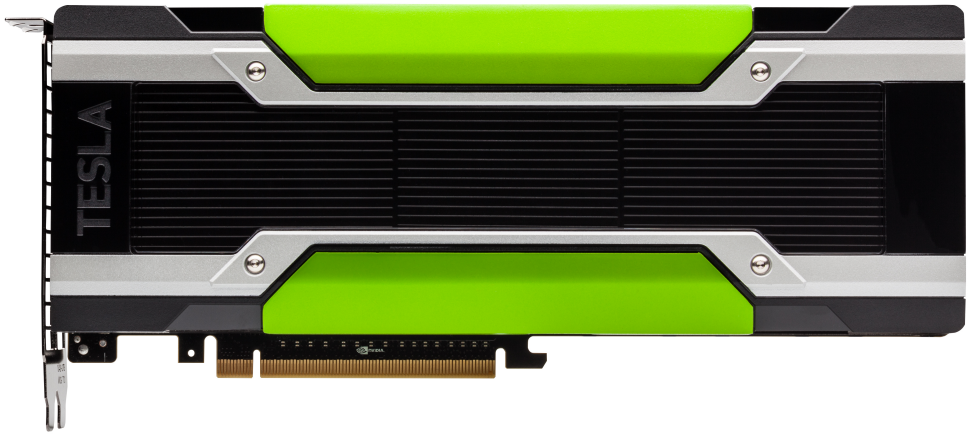
- Are there security or ethical concerns associated with the training data and the future uses of this model?
- What accuracy has already been achieved in our organization, and how much improvement is required for success?
- Can the business benefits of improved accuracy be quantified?
- How will I know the model is working once deployed?

Today's (Toy) Problem



- **Can we classify small color images into one of 10 categories?**
- Goal is better than 75% accuracy
- We will build a deep learning model and train it with GPUs.

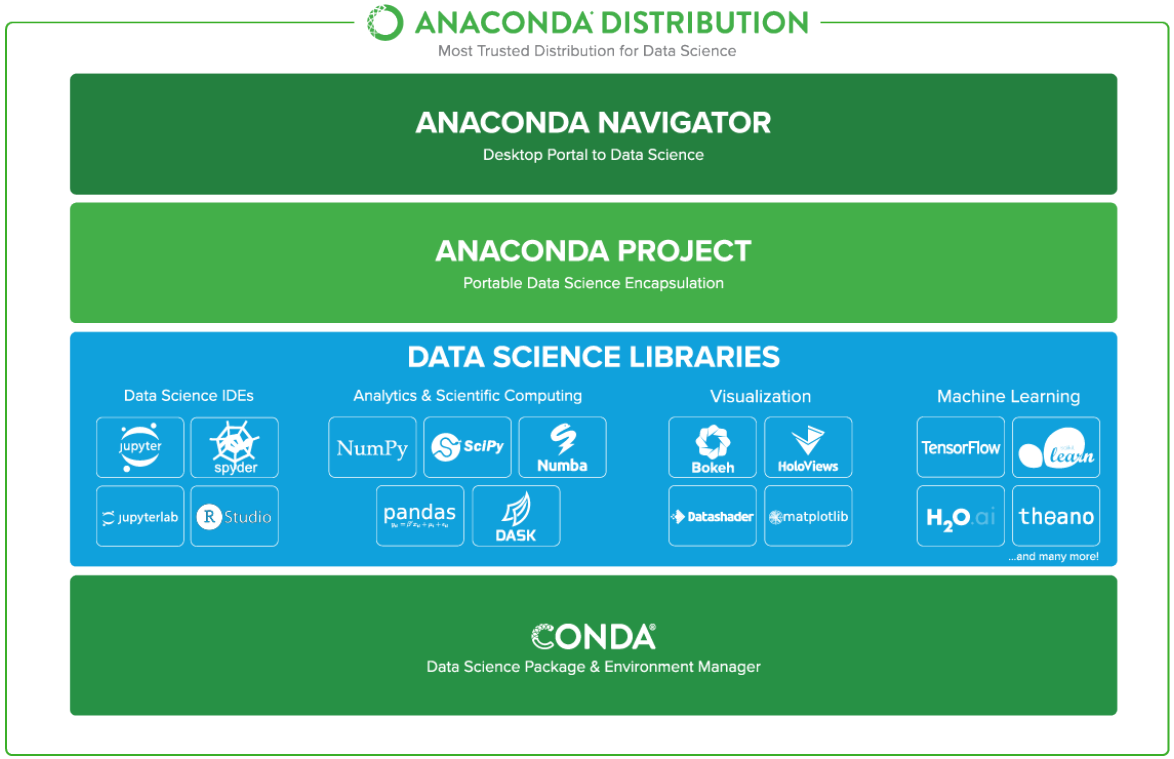
Hardware: NVIDIA GPUs



Tesla P100
3584 CUDA cores
9.3 TFLOPS (single precision)
16 GB Memory
732 GB/sec

× 56

Software: Anaconda Distribution



Already installed in your tutorial environment

GPU-Accelerated Packages in Anaconda

GPU-Accelerated Packages in Anaconda

PYTORCH

Caffe

Chainer

CuPy

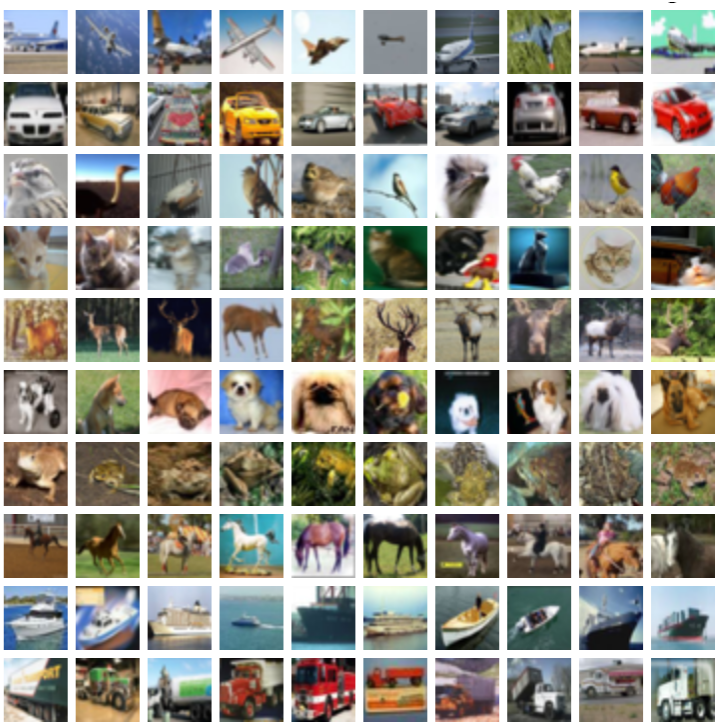
Numba

mxnet

dmlc
XGBoost

Data: CIFAR10 Image Set

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



https://www.cs.toronto.edu/~kriz/cifar.html

Classic data set for benchmarking image classification

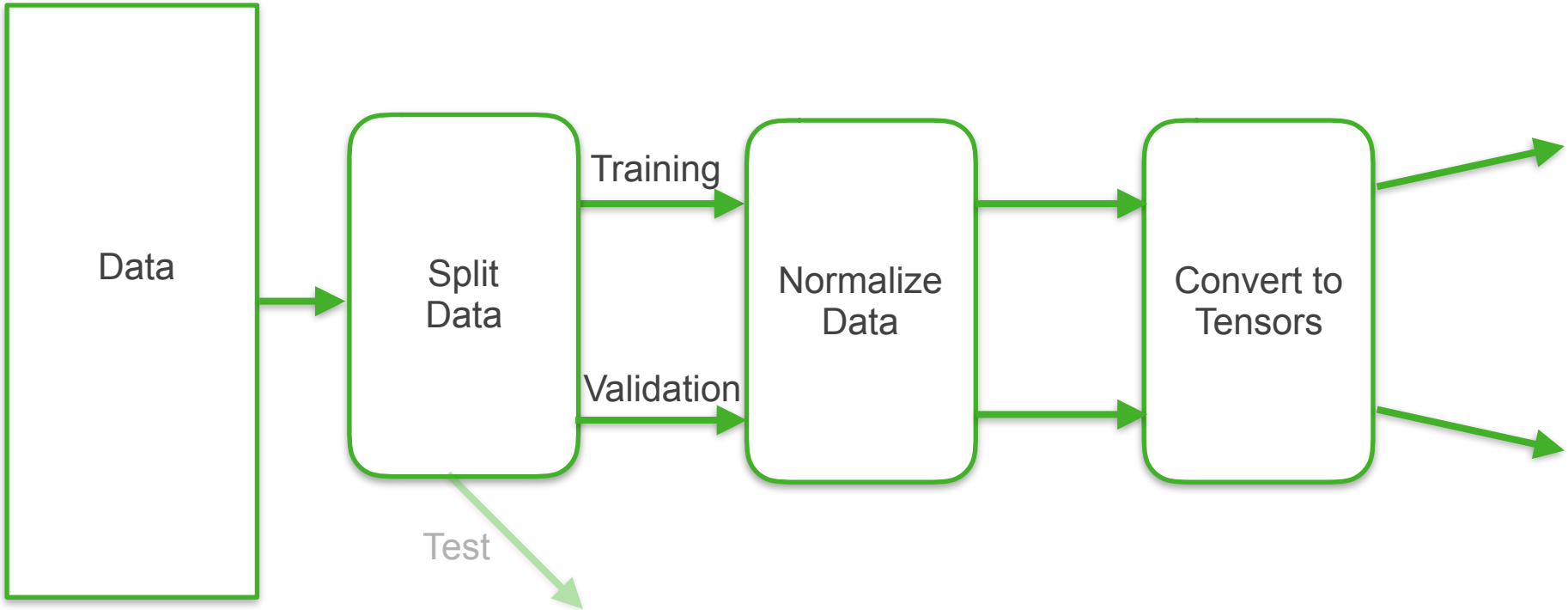
- 60000 images
- 32x32 color pixels
- 10 classes
- 6000 images from each class
- Classes are mutually exclusive

Fooling Yourself With Data

Non-representative Training Data

Non-stationary distribution

Data Preparation Process



One-Hot Encoding of Category Variables

Category #
1
0
2
3
3
1
2

0	*1*	*2*	*3*
0	**1**	0	0
1	0	0	0
0	0	**1**	0
0	0	0	**1**
0	0	0	**1**
0	**1**	0	0
0	0	**1**	0

Exercise 1: Data Exploration and Prep

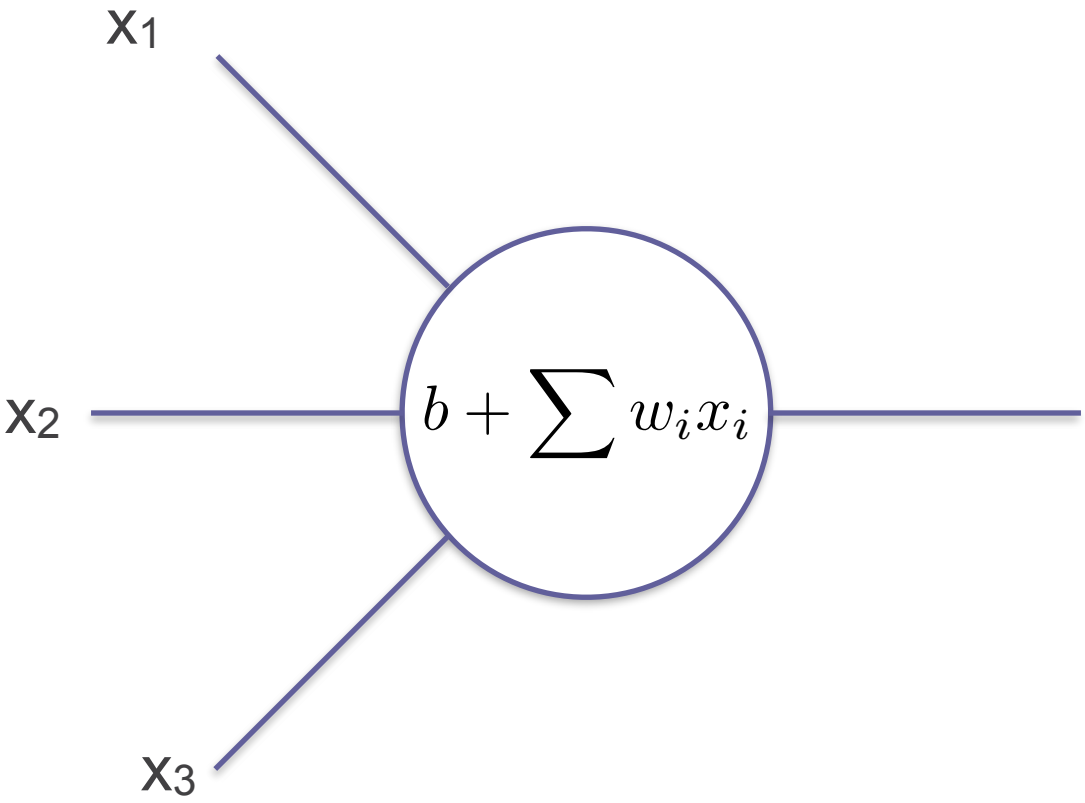
Part 2: Introduction to Deep Learning with Keras

What is a neural network?

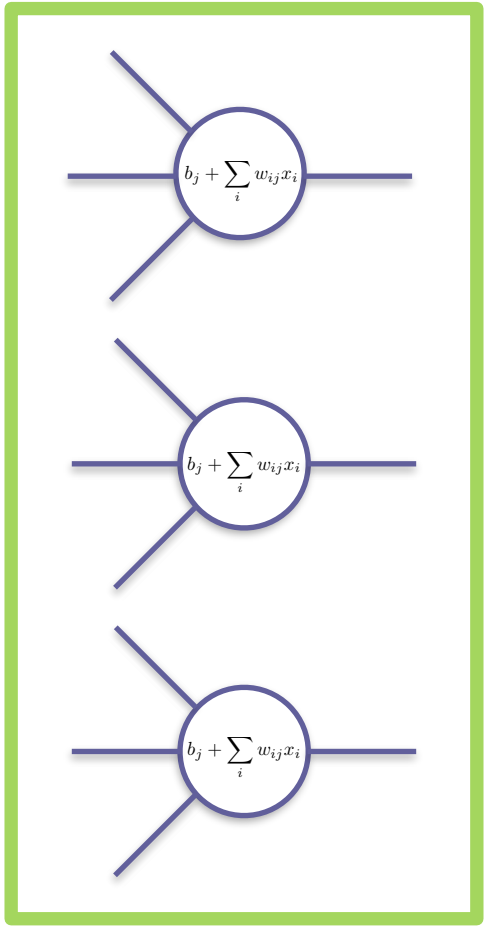
- *A "biologically-inspired" method for making models out of simple computational units connected to form a large mesh*
- Very flexible.
- Require a lot of data to train.
- Although described in physical terms (nodes, layers, etc), they are always implemented in software using array math.
- **No relation to neuroscience, artificial general intelligence, or other sci-fi stuff.**

https://www.flickr.com/photos/usdagov/16802162424

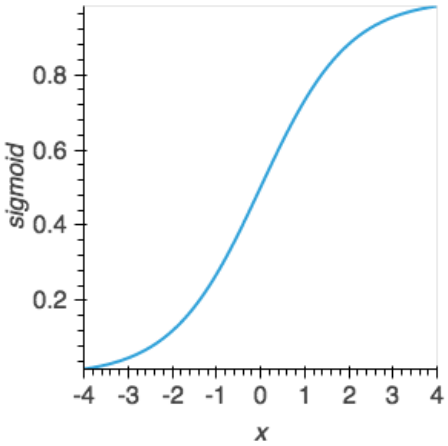
Node

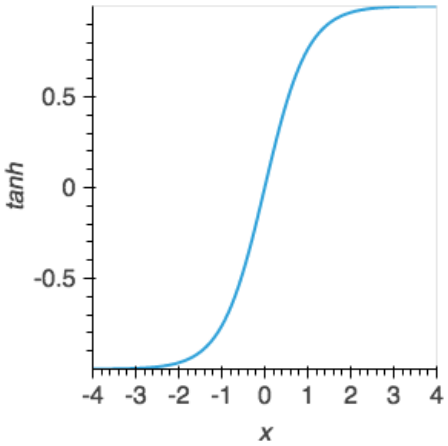


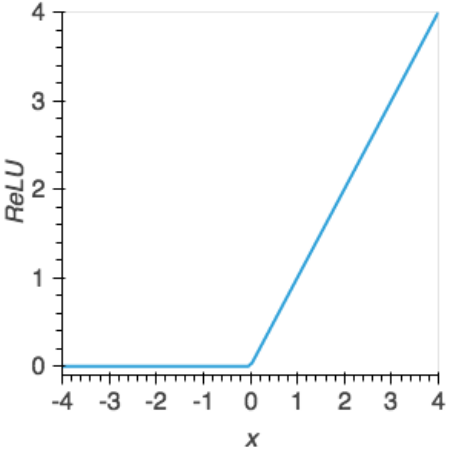
Layer

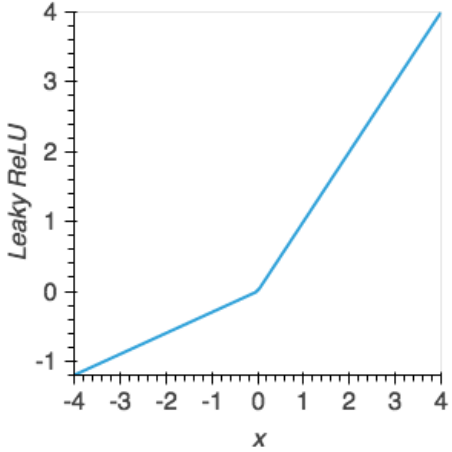


Activation Functions

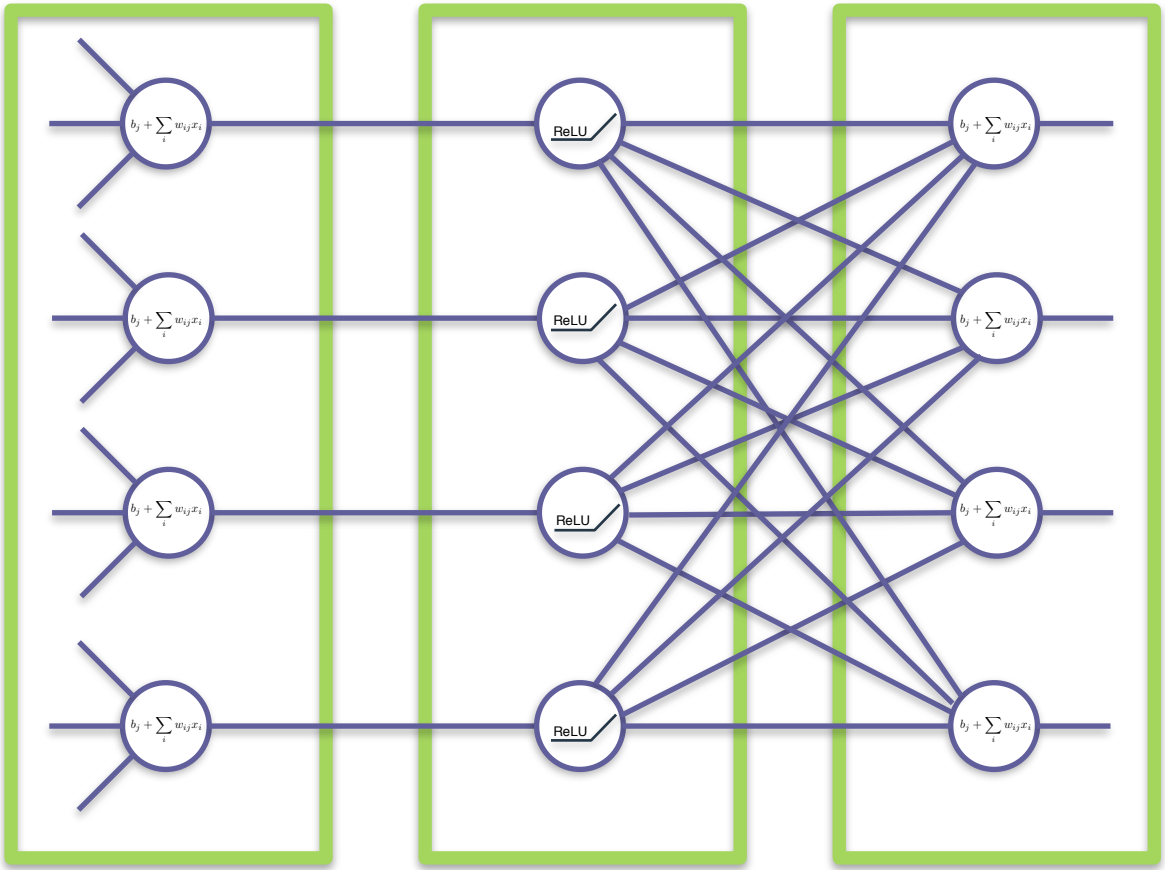








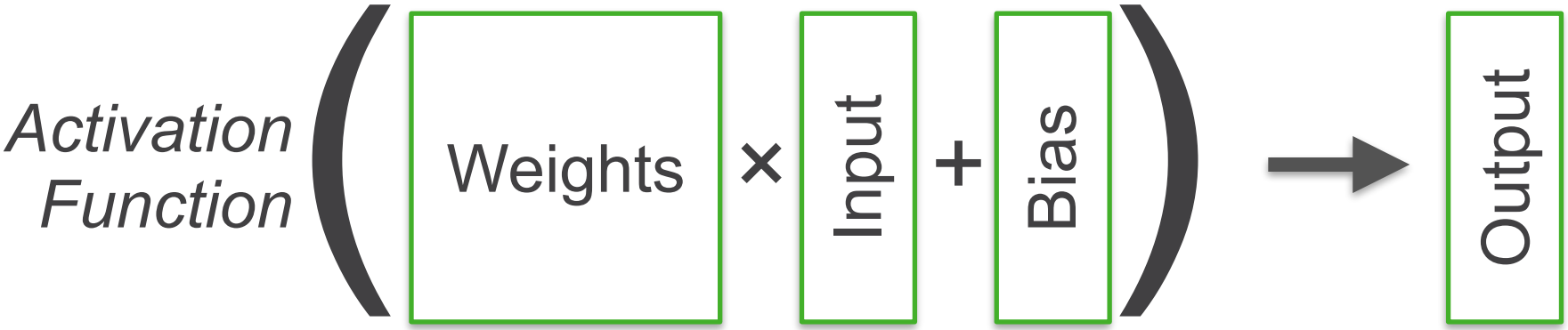
Network



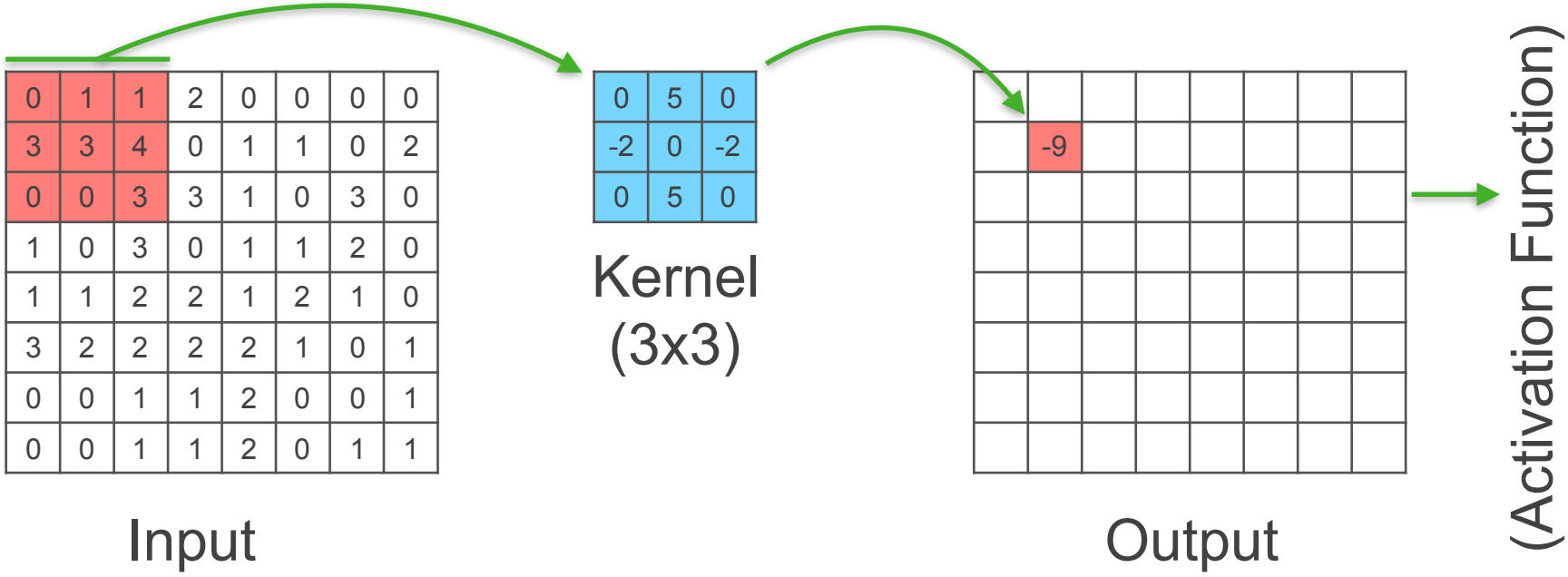
Layer Types

- "Dense" or "Fully connected"
- Convolutional
- Pooling
- LSTM
- (output) Softmax
- (output) Sigmoid

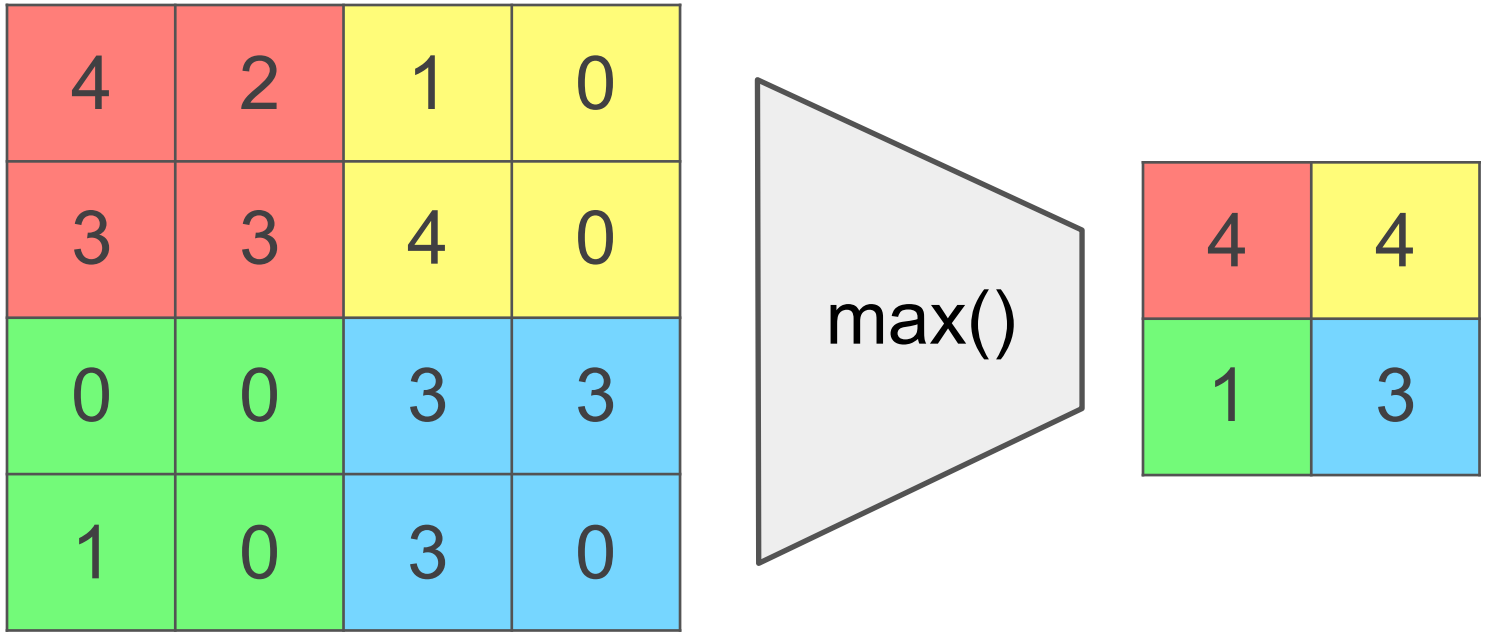
Dense Layers



Convolutional Layers



Pooling Layers



Category Output Layers: Sigmoid vs. Softmax

If you set a threshold, can accommodate non-exclusive classes and "none of the above" results

airplane	0.010
automobile	0.110
bird	0.005
cat	0.160
deer	0.003
dog	0.007
frog	0.150
horse	0.010
ship	0.100
truck	0.005

Sigmoid

Membership in a class is determined by having the largest value

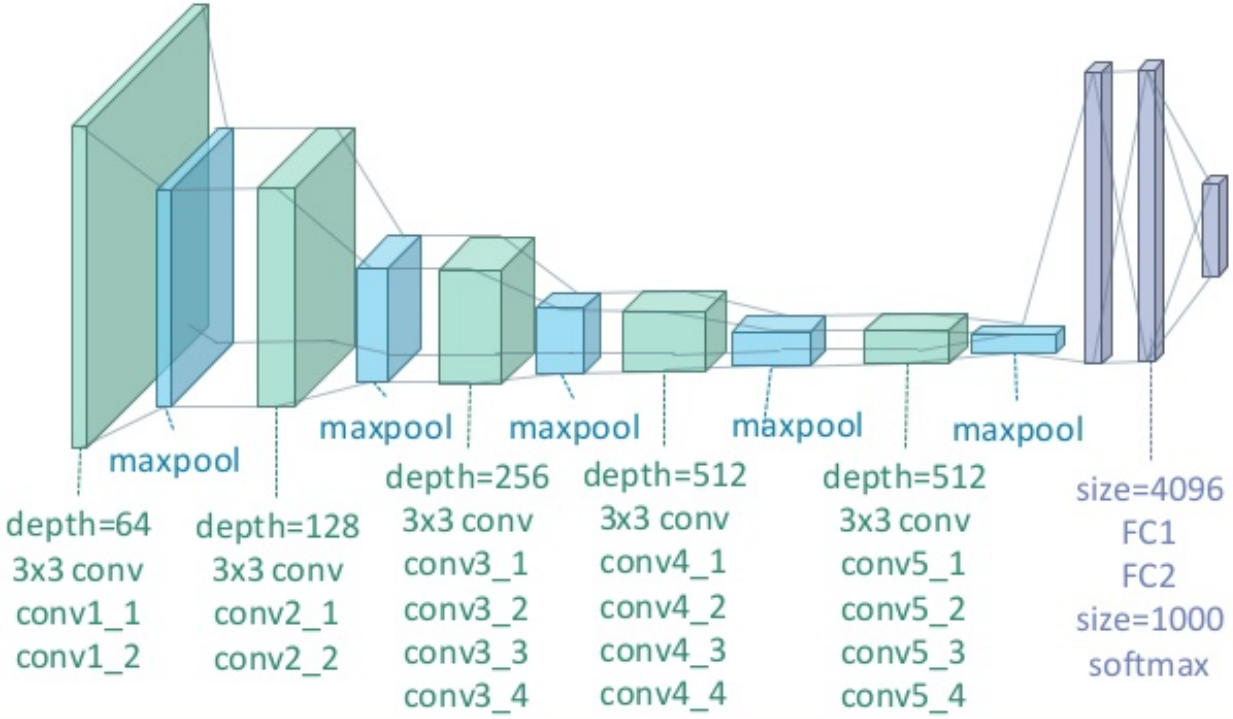
(Output always sums to 1)

airplane	0.017
automobile	0.196
bird	0.009
cat	0.286
deer	0.005
dog	0.013
frog	0.268
horse	0.018
ship	0.179
truck	0.009

Softmax

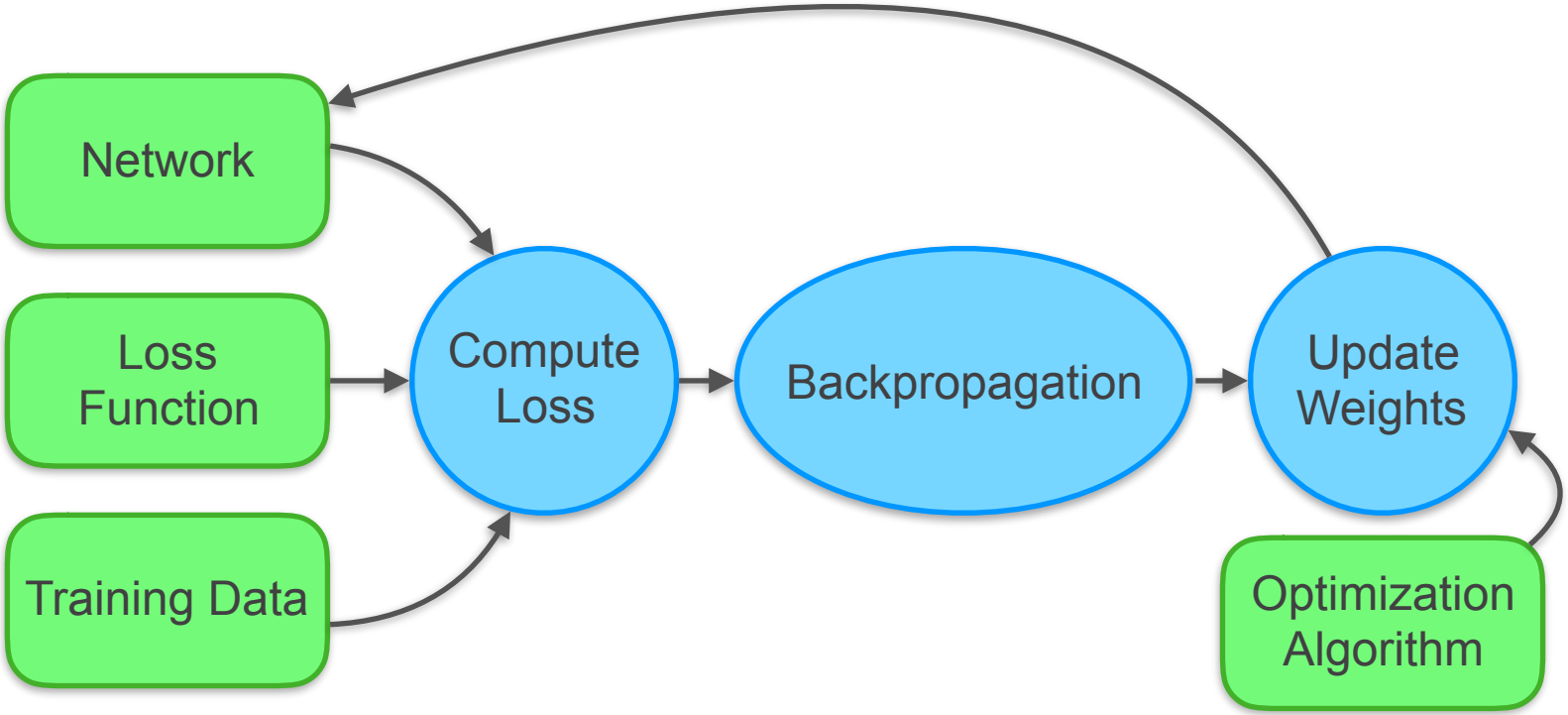
An Image Classification Network

VGG19



https://www.slideshare.net/ckmarkohchang/applied-deep-learning-1103-convolutional-neural-networks

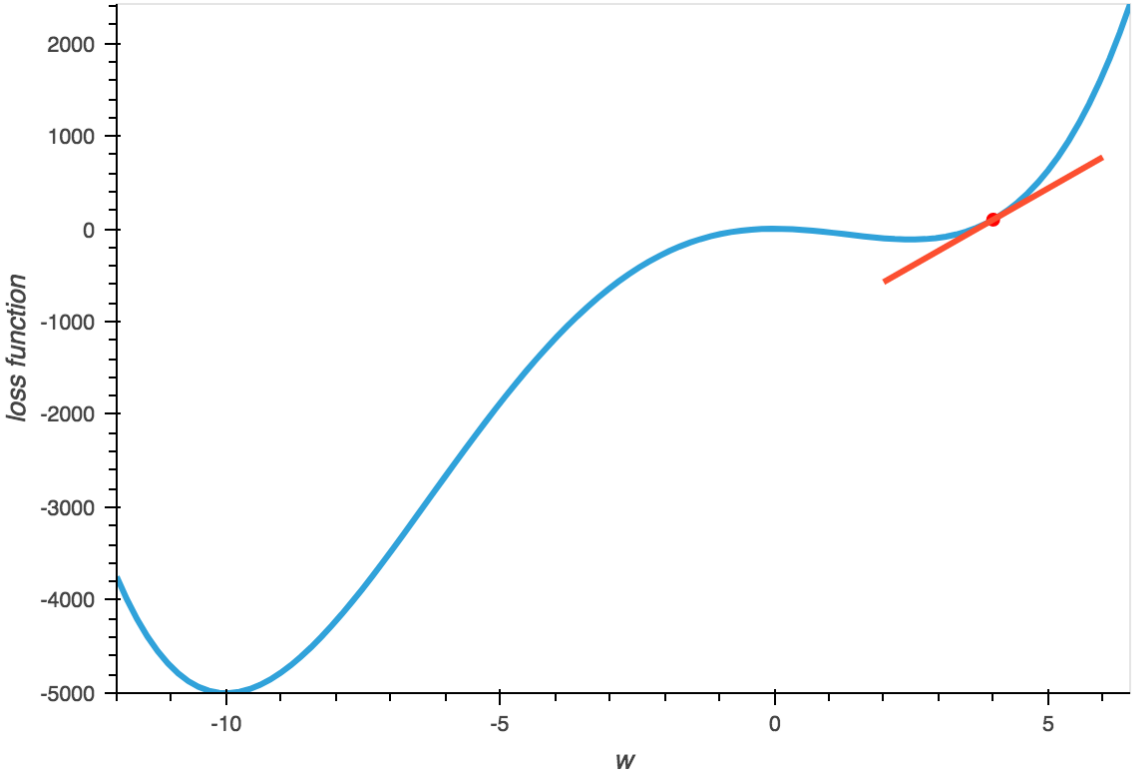
Training a Network



Loss Functions

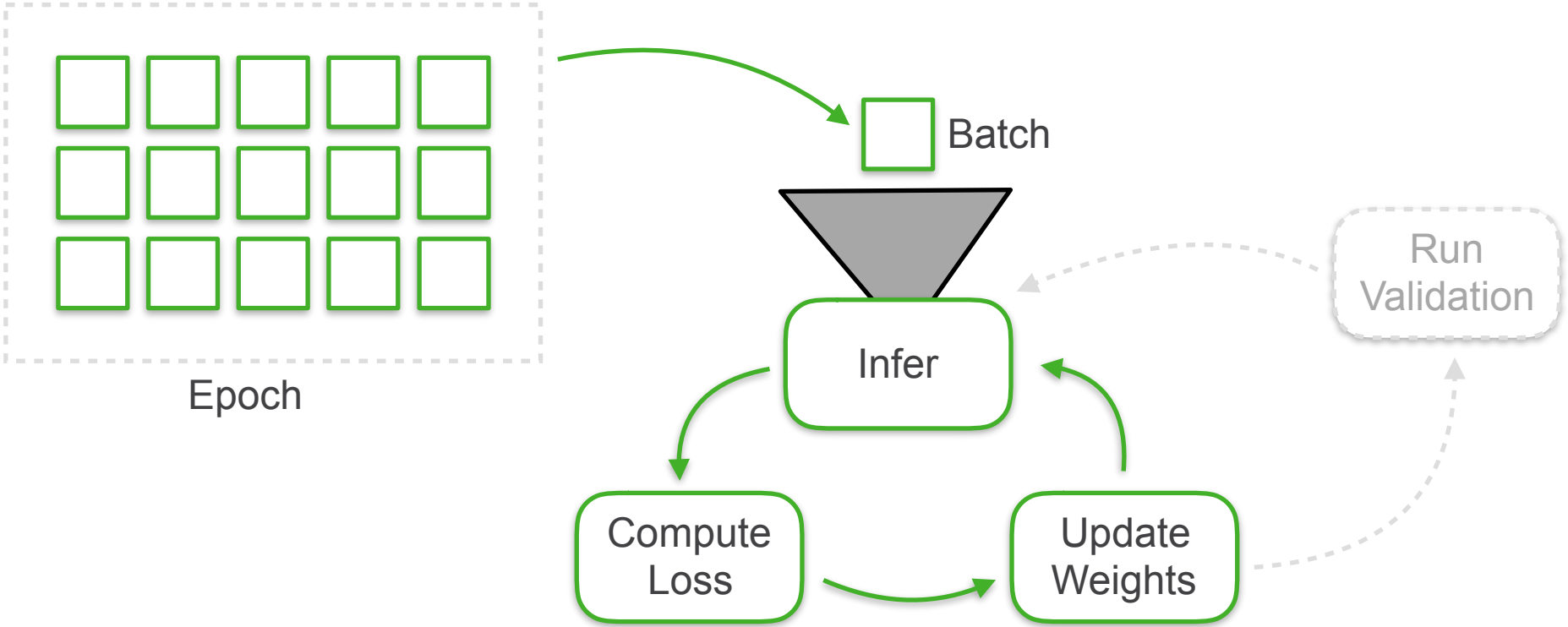
- The optimizer will try to minimize the loss function on the training data
- *Picking a loss function depends on the kind of model you are building*
- Some good default choices:
  - Regression: **mean_squared_error**
  - Binary classification: **binary_crossentropy**
  - Multi-category classification: **categorical_crossentropy**

Backpropagation & Optimization



- Work backwards through network to compute each weight affects loss function
- Optimization algorithm adjusts each weight according to some strategy.

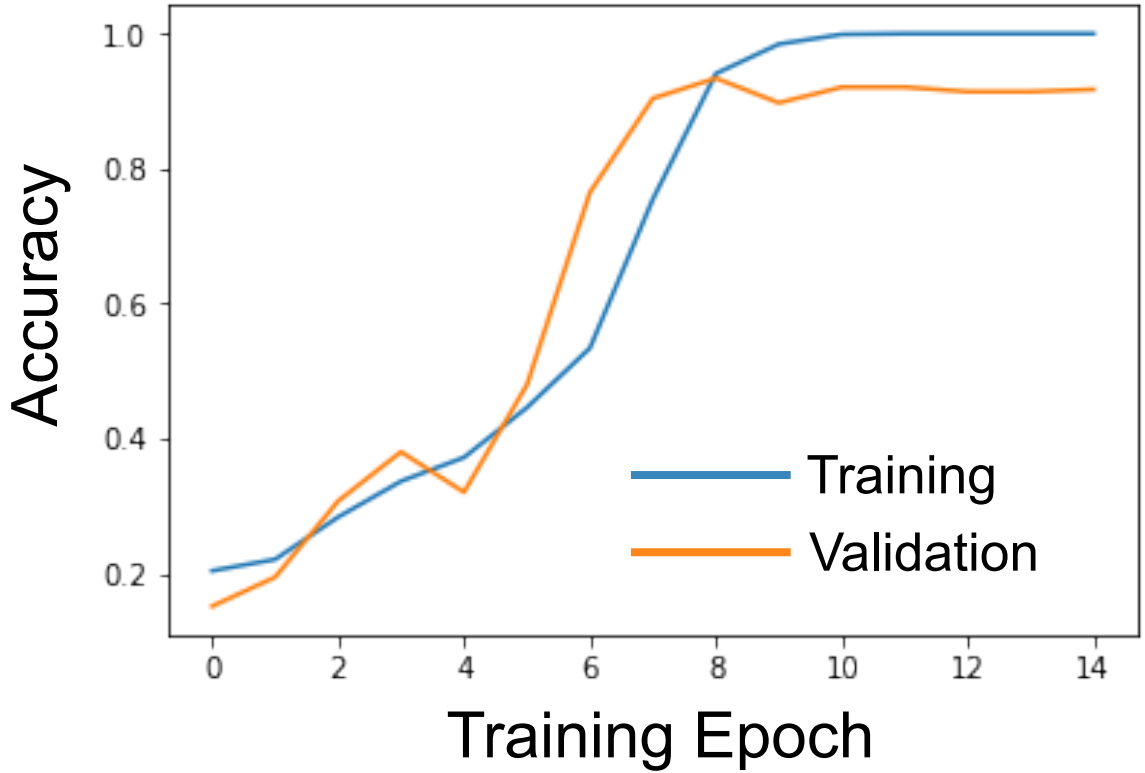
Batches and Epochs



Exercise 2: Deep Learning with Keras

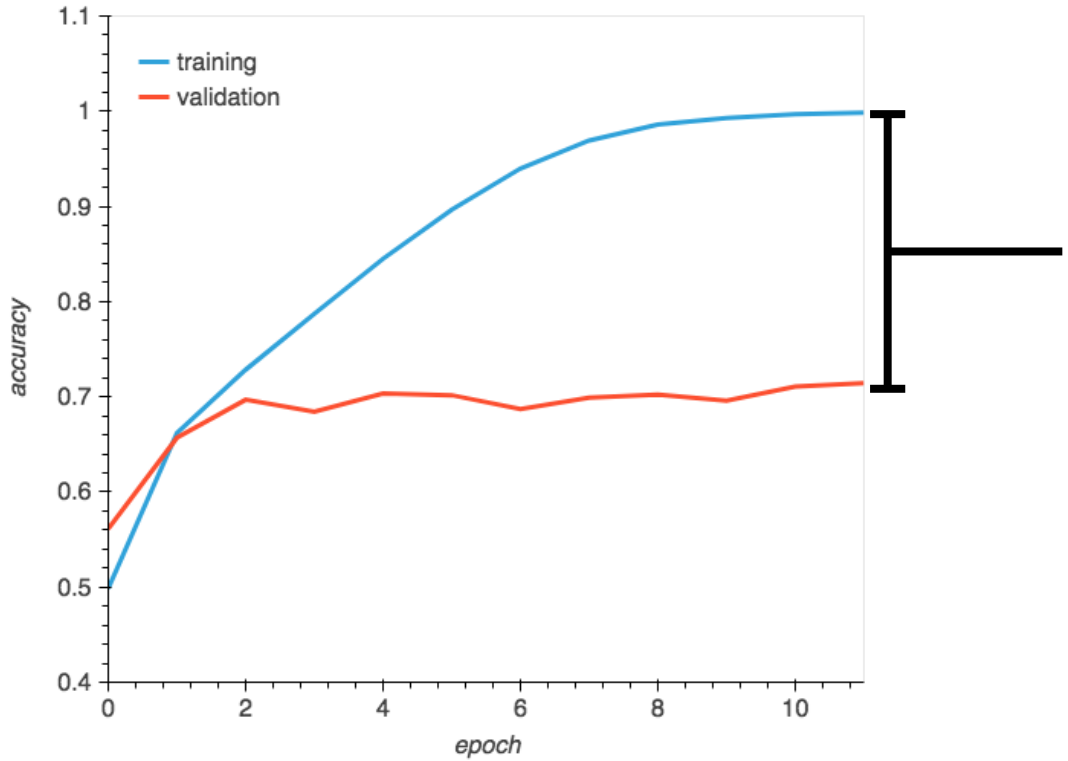
Part 3: Evaluating Models and Troubleshooting

Most important question in ML



How do I know when I am done?

Overfitting



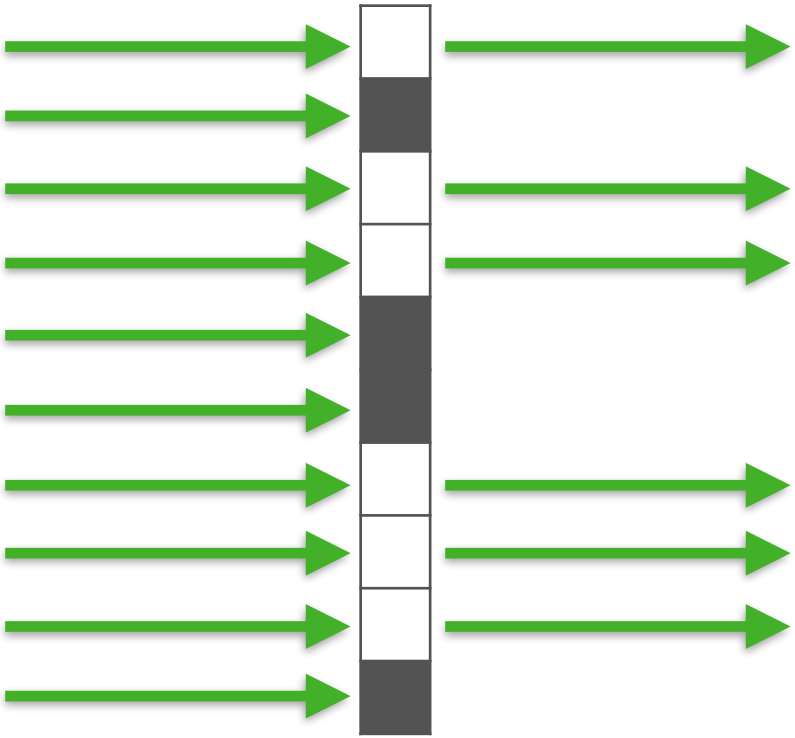
Gap caused by model overfitting

What to do about overfitting?

- **Overfitting does not mean your model is bad.**
  - Most models will overfit after enough training epochs
  - *The validation data accuracy is what you care about*

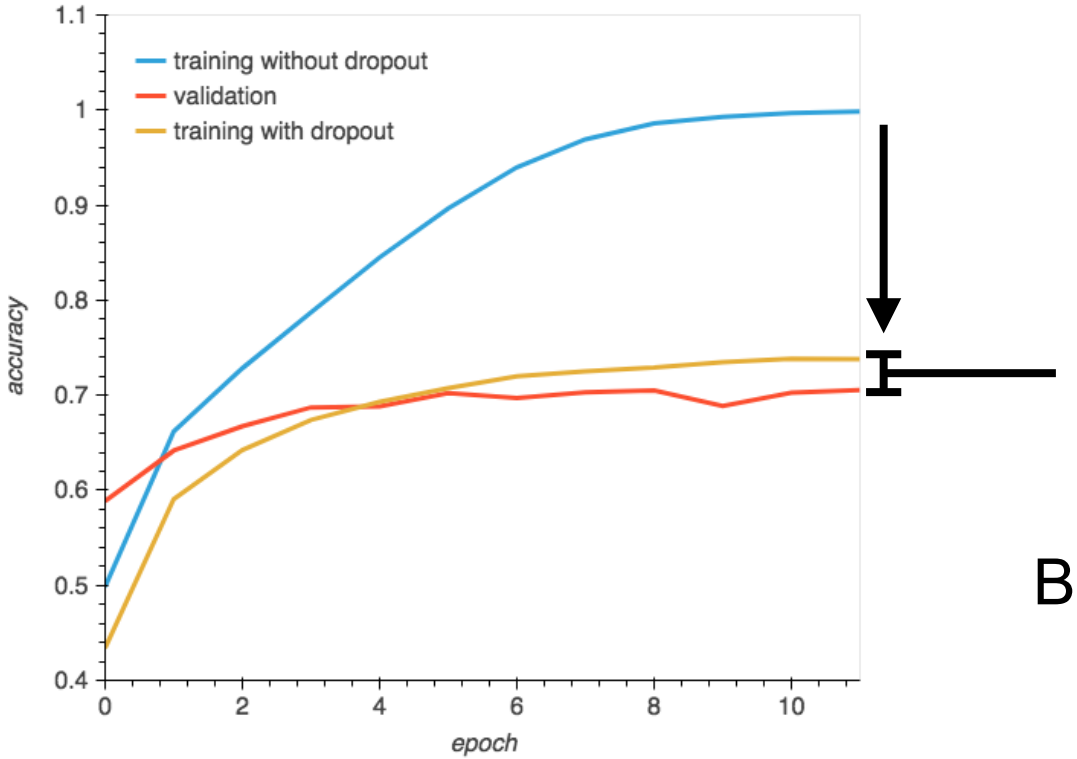
- Techniques exist to control overfitting:
  - Regularization
  - Dropout layers
  - Reduce size of network
  - Get more data

Dropout Layers



- Randomly zero subset of output values
- Only active during training
- Fraction is an adjustable parameter
  - *Usually 0.25 to 0.5*

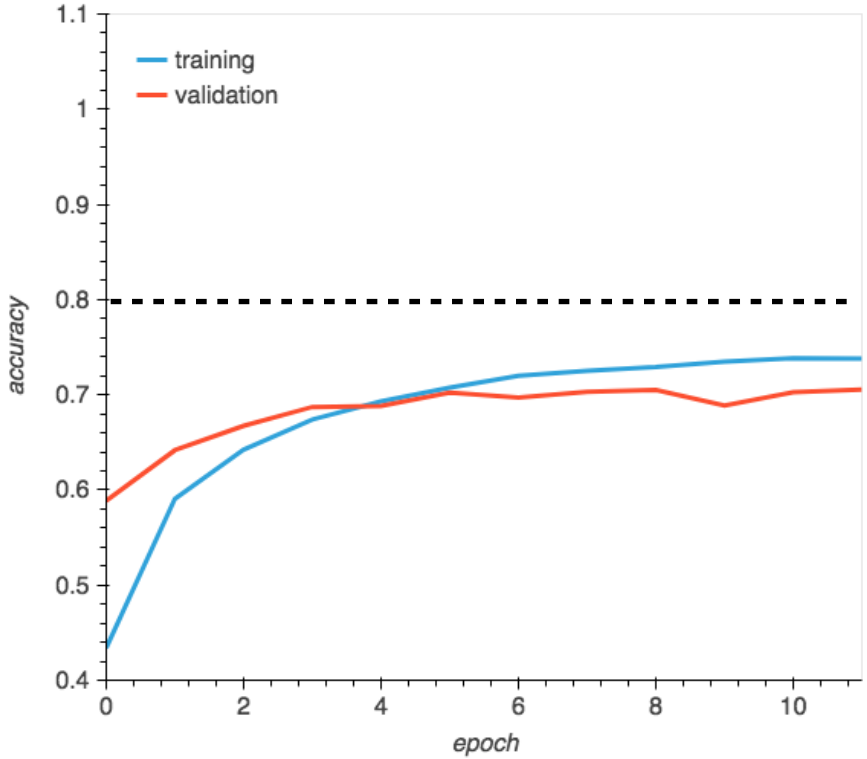
After Adding Dropout



Much less overfitting

But now we have a new problem...

Underfitting

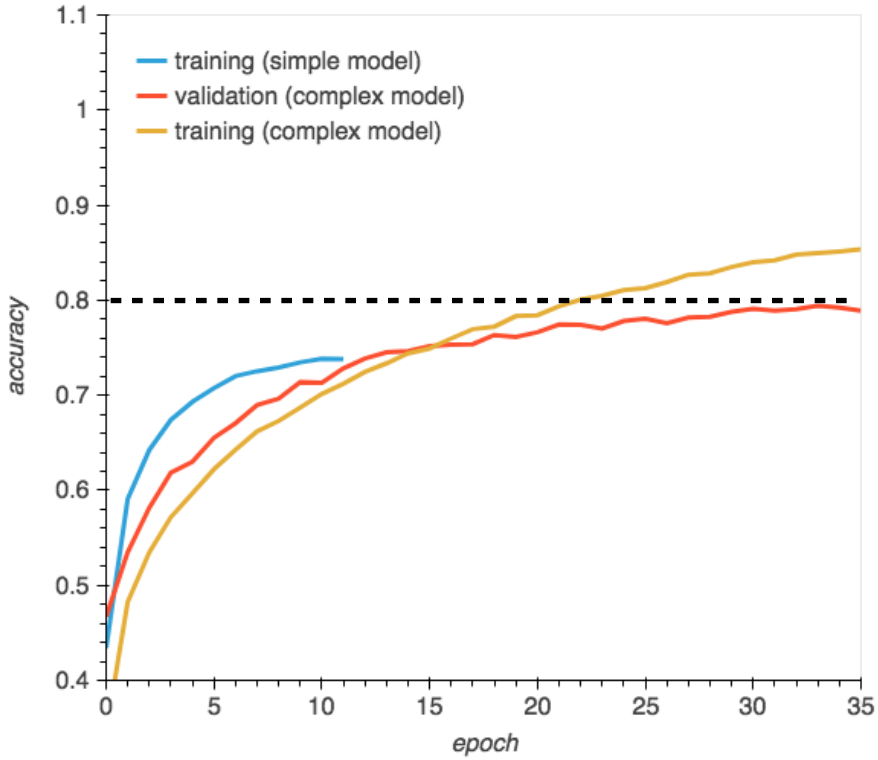


Model accuracy never reaches the goal.

Possible causes:

- Model too simple?
- Not enough training data?
- Mislabeled data?
- Optimizer learning rate?
- ...

Fixing Underfitting



Changed 2 things

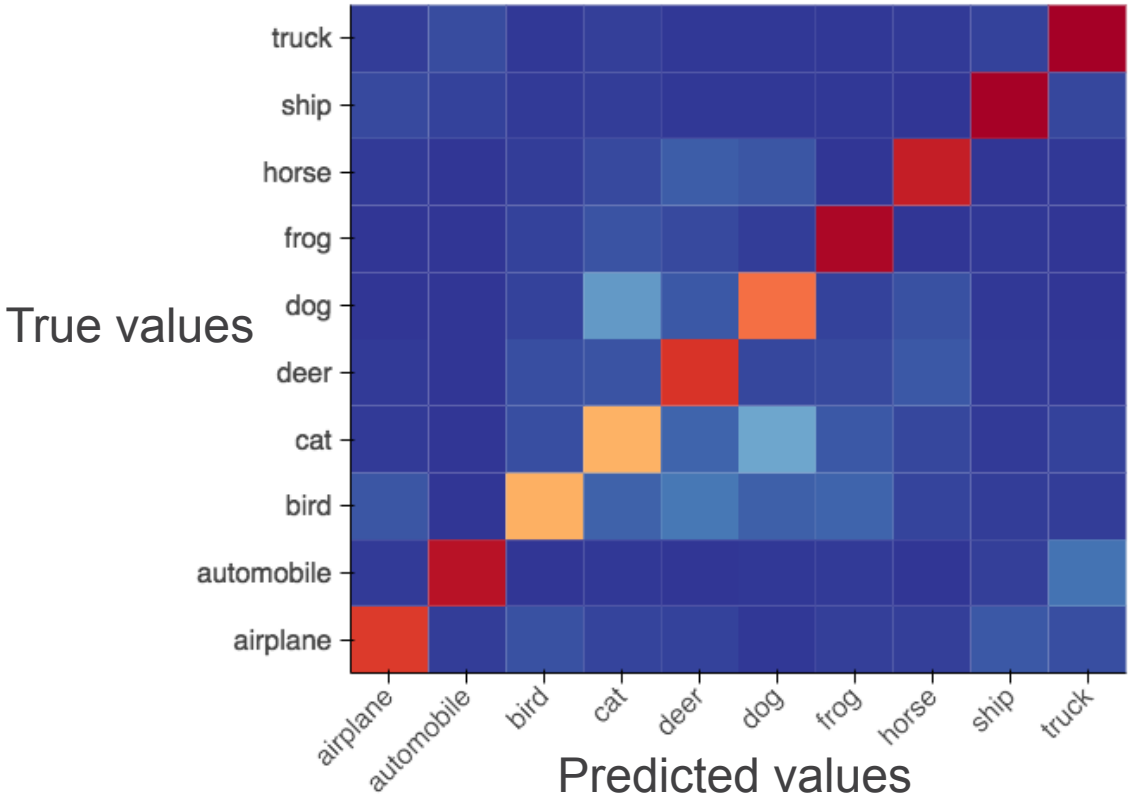
- Increased model complexity (added extra convolutional layers)
- Changed optimizer algorithm

Trial and error is still the norm...

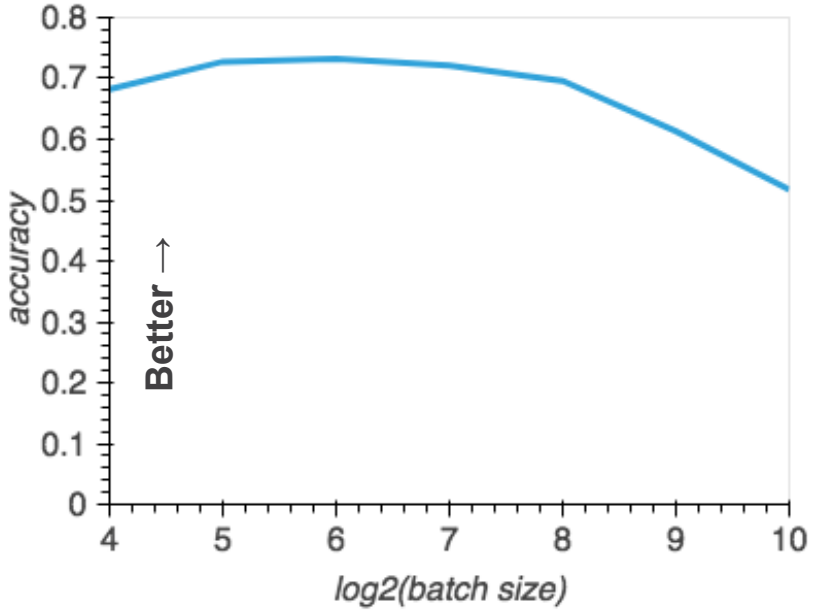
Limitations of Accuracy

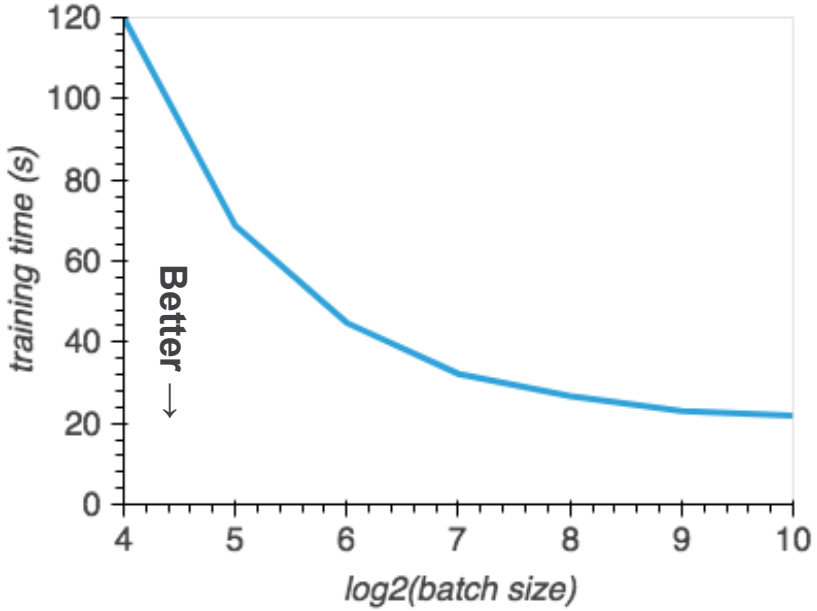
- Accuracy is like the "check engine light"
- Can hide some problems
- ***Interpretation is hard if your training data is unbalanced***
- Important to look at other measures:
  - False positive rate
  - False negative rates
  - Confusion matrix
  - Inspect fail cases!

Confusion Matrix



Effect of Batch Size





8 training epochs

Hyperparameters

- **Choices that affect the structure or training of the network, but are not optimized during training.**

- Examples include:
  - Number of nodes in each layer
  - Batch size
  - Dropout rates

- Encapsulate your training code into a Python function that you can call many times for different hyperparameter choices.

- Be wary of overfitting! Hold aside a test dataset to check at the very end.

Exercise 3: Evaluating Models

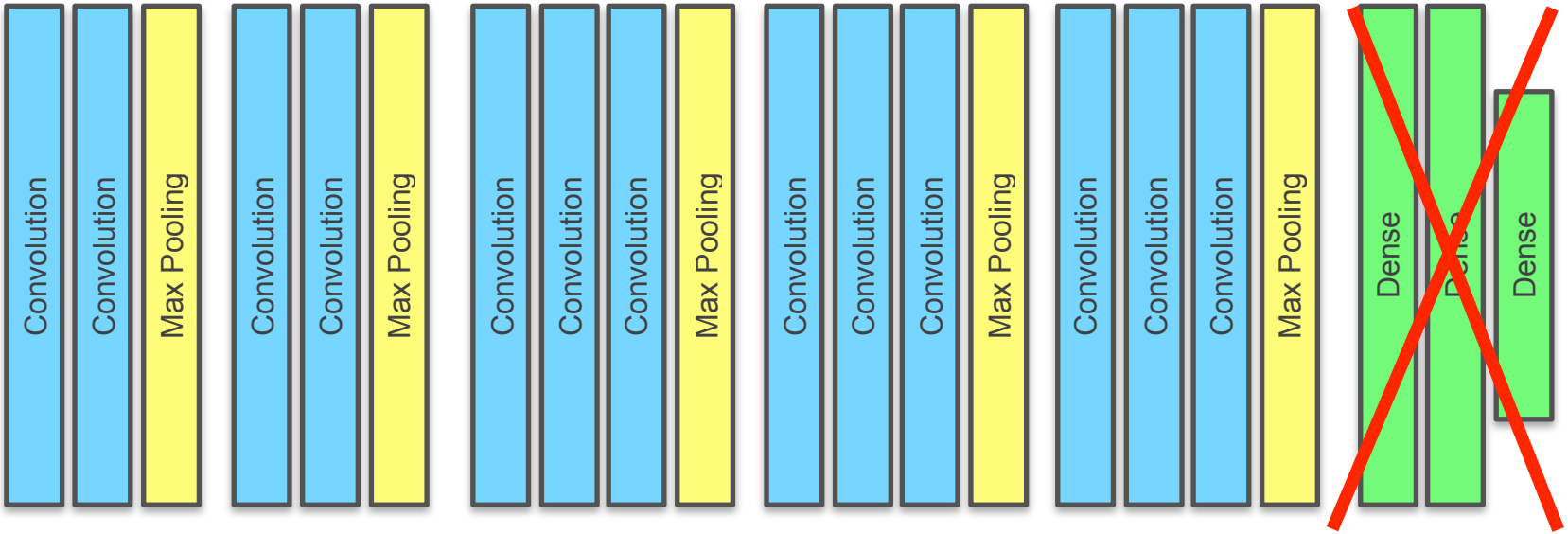
Part 4: Advanced Topics and Deployment

Building on Existing Models

- In practice, you will generally use models defined by researchers, rather than construct new ones from scratch. Keras comes with quite a few:
  - VGG16
  - ResNet50
  - InceptionV3
  - MobileNet
  - NASNet
  - https://keras.io/applications/#documentation-for-individual-models

- Three approaches to retraining (slower to faster):
  1. Retrain existing architecture from scratch
  2. Retrain existing architecture starting from existing trained weights
  3. **Retrain only the final dense layers**

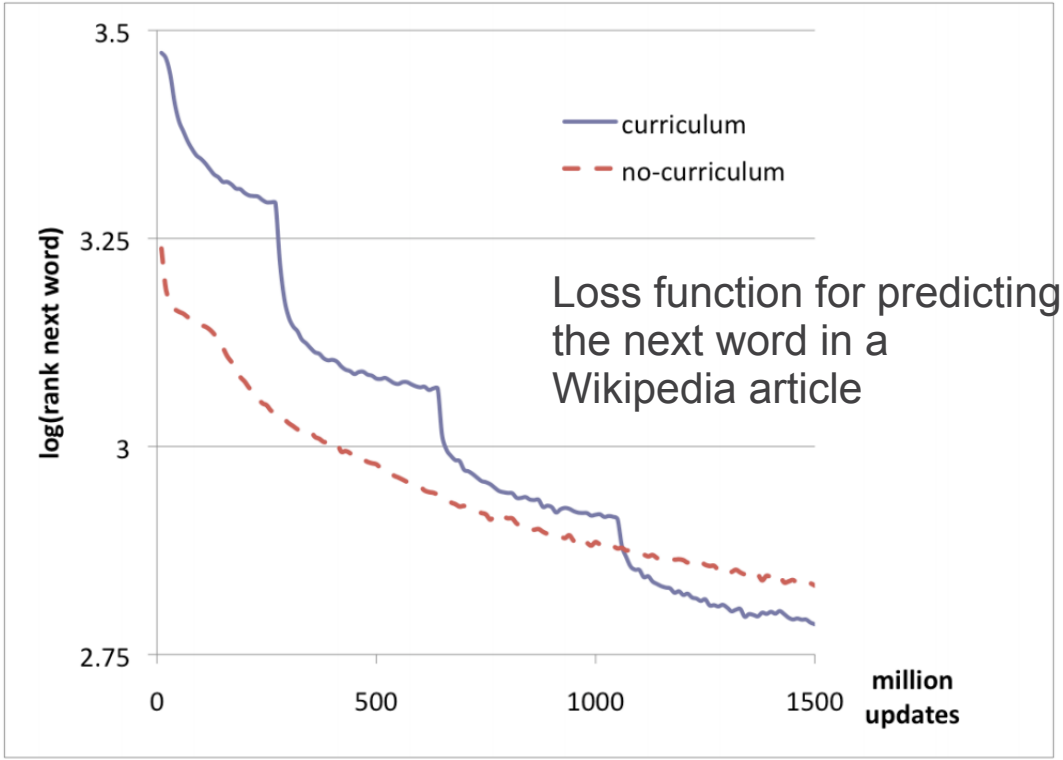
Transfer Learning



VGG16 Pretrained on ImageNet

Chop top layers off and retrain your own layers

Curriculum Learning

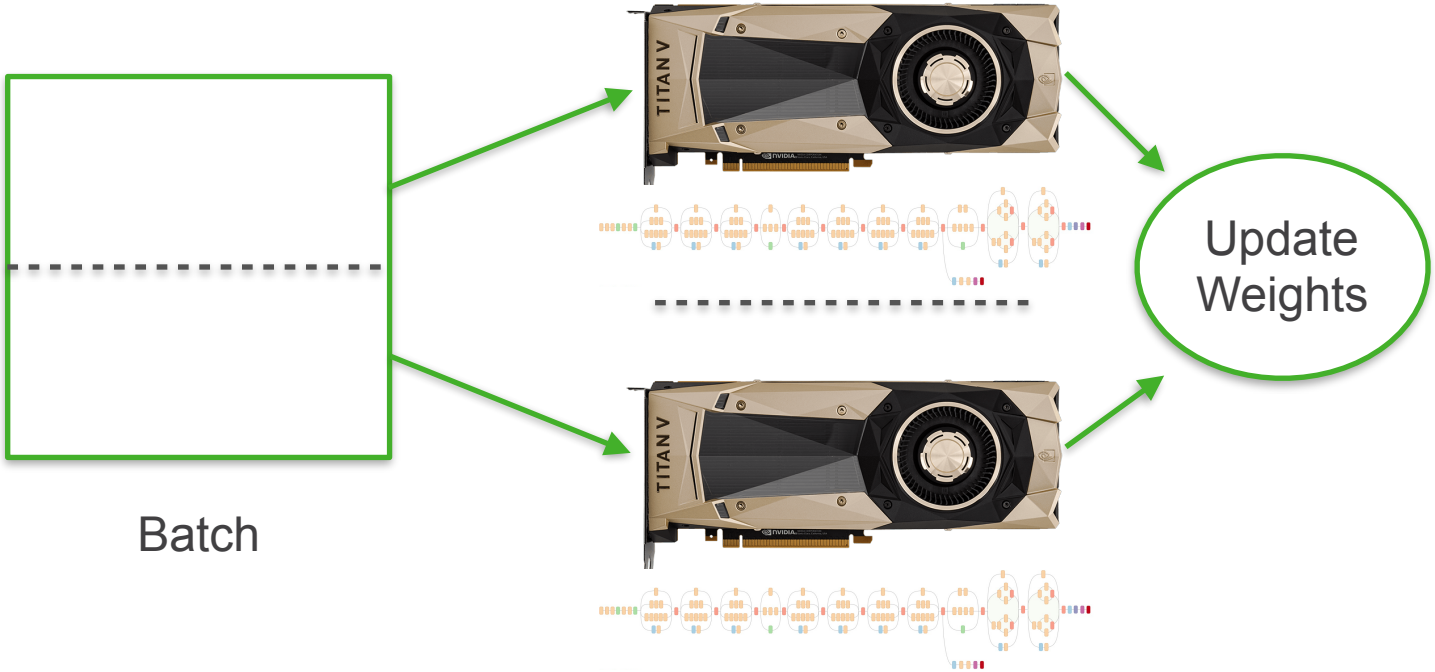


- Train the model in multiple passes
- Use increasingly hard examples on each pass
- Easy if your data has a "difficulty" knob

Bengio, et al, ICML 2009

Using Multiple GPUs

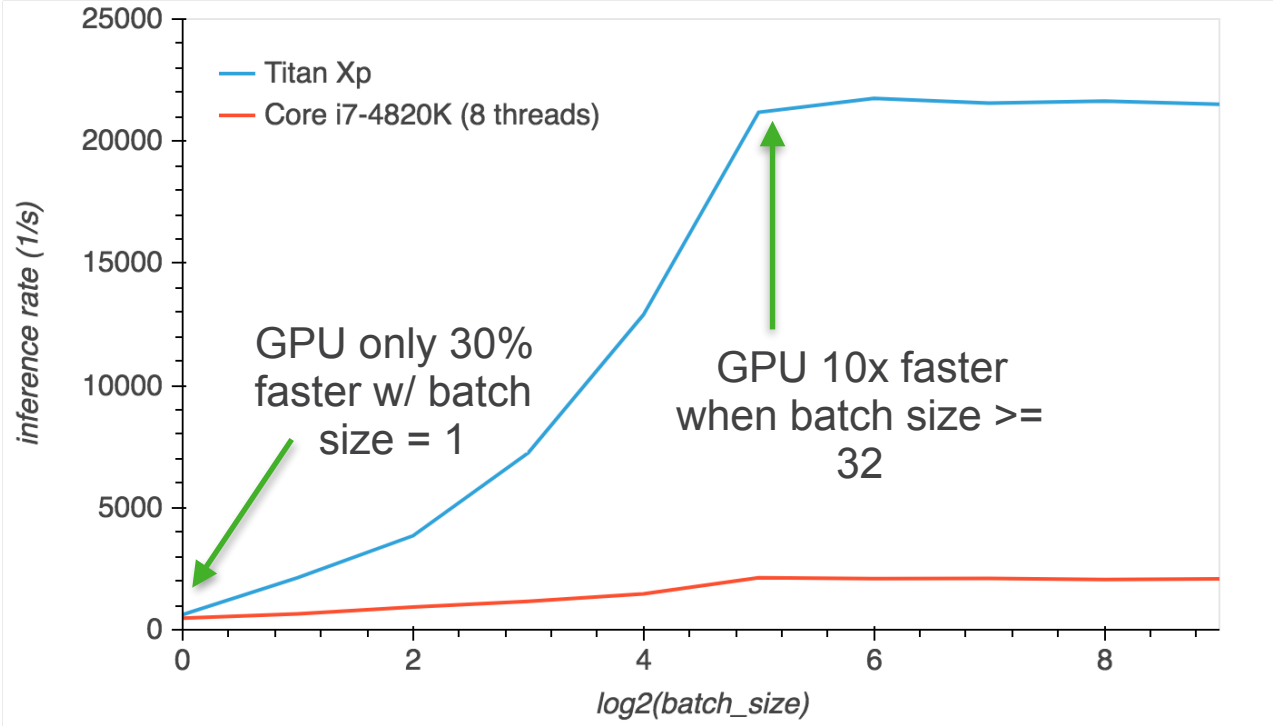
`keras.utils.training_utils.multi_gpu_model()`



Getting to Deployment

- **This was the point of doing all this, right?**
- Tools and best practices evolving rapidly
- Questions to consider:
  - What hardware is available in production?
  - Bulk processing or online processing? (or both?)
  - What is the performance requirement (throughput, latency)?
  - Does the model need optimization? (memory size or speed)
  - How should I package it, along with dependencies?

Determining Hardware Needs

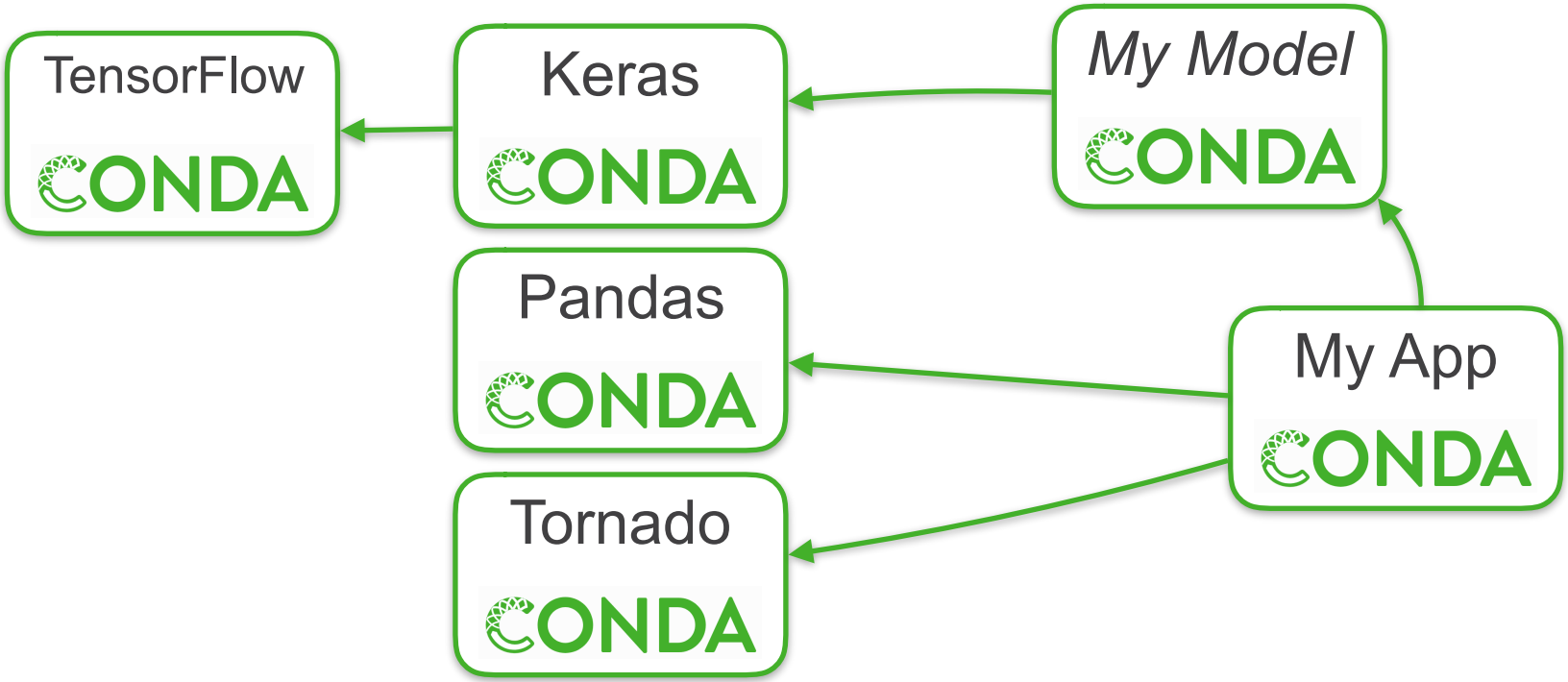


Inference in batches, or one at a time?

Do you need a GPU in production?

Smaller GPUs may also be be sufficient

Packing Models with Conda



A Data Scientist's Job Is Never Done

- Important to monitor deployed models:
  - Does accuracy on new data match training/validation data?
  - Snapshot incorrect predictions for study
  - Keep expanding your training data
  - *Make sure any data collection complies with your security and privacy policies*
- Version your models, just like software!

Exercise 4: Transfer Learning / Loading & Saving Models

Conclusion

The Process

1. Define your problem
2. Identify your dataset
3. Design a network
4. Train the network
5. Check your work and iterate as needed
6. Package for production
7. Monitor deployed models for effectiveness

GPU Requirements: Hardware & OS

- **NVIDIA GPU**
  - Best results: Pascal, Volta, or Turing architectures, >4 GB of GPU memory
  - Suggestions:
    - GeForce: GTX 1070, GTX 1080, RTX 2070, RTX 2080
    - Titan: Xp, V, RTX
    - Quadro: many models
    - Tesla: P100, V100
  - *You can use older GPUs to learn, but don't expect great performance.*
- **CPU:**
  - Quad core
  - 2x more CPU memory than your GPU
- **OS:**
  - Linux 64-bit is preferrable
  - Windows 64-bit also works
  - macOS: no suitable NVIDIA GPUs in Macs made since 2014, external NVIDIA GPUs not officially supported, CUDA drivers incompatible with macOS 10.14

- *Can also use cloud servers!*

Cloud Availability

Tesla K80

- Two GPUs on one card
- 2 x 2496 CUDA cores @ 0.56 GHz
- 2 x 12 GB memory
- *Released: Nov 2014*

Azure Google Cloud Platform aws

Tesla P100

- 3584 CUDA cores @ 1.3 GHz
- 16 GB memory
- 732 GB/sec
- NVLink 1
- *Released: April 2016*

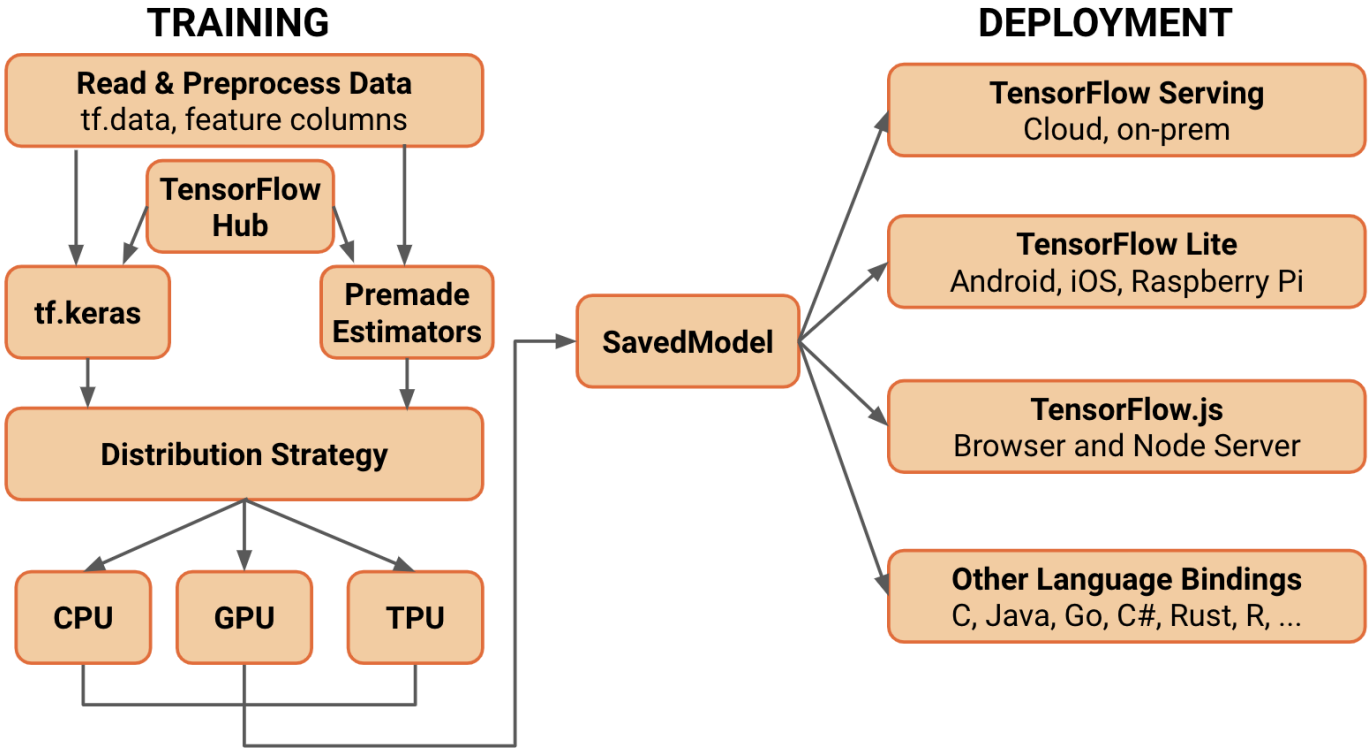
Azure Google Cloud Platform

Tesla V100

- 5120 CUDA cores @ 1.3 GHz
- 16/32 GB memory
- 900 GB/sec
- NVLink 2
- Tensor Cores for DL
- *Released: June 2017*

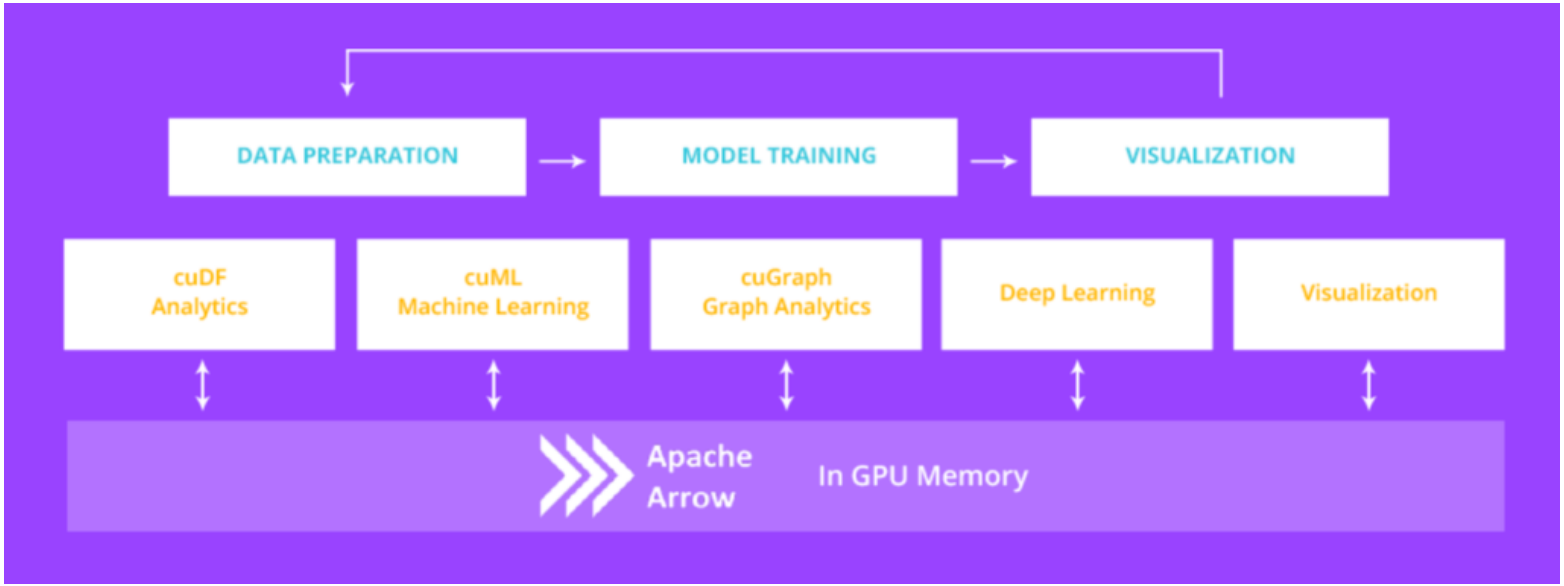
Azure Google Cloud Platform aws

Learn More: TensorFlow 2.0



https://medium.com/tensorflow/whats-coming-in-tensorflow-2-0-d3663832e9b8

Learn More: RAPIDS



GPU-accelerated dataframes and traditional ML algorithms

https://medium.com/rapids-ai/the-road-to-1-0-building-for-the-long-haul-657ae1afdfd6

Further Reading on Deep Learning

- **Tutorial Notebooks:**
  https://github.com/ContinuumIO/ac2019-dl-gpu
- *Deep Learning with Python*, by François Chollet
  https://www.manning.com/books/deep-learning-with-python
- Neural Network Playground (experiment in your browser!)
  http://playground.tensorflow.org/
- Keras Documentation
  https://keras.io/
- **Want code examples? Google search for "Keras [use case]".**
  Examples:
  "keras image segmentation"

Thank you.