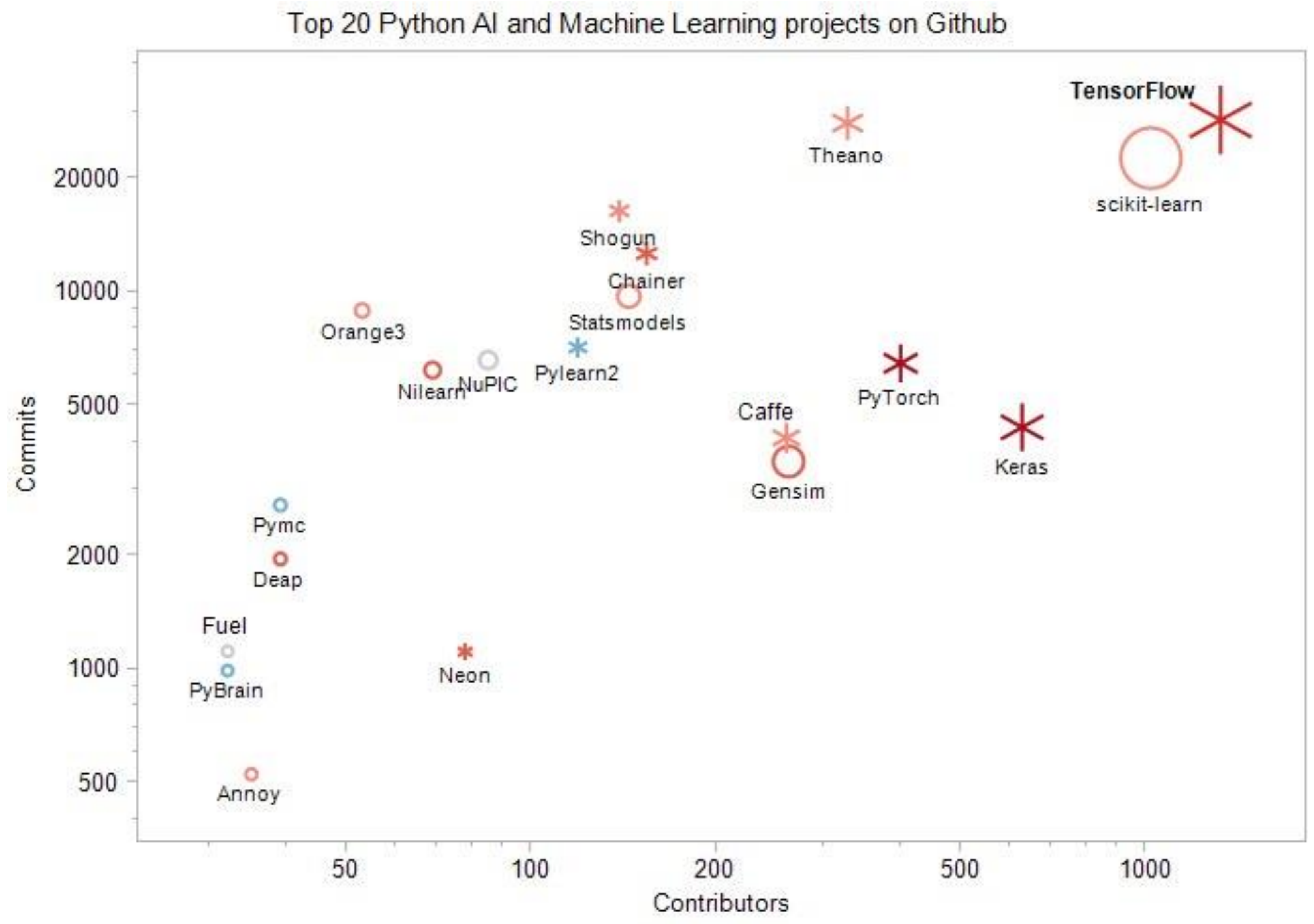
Topic 3
Python Deep Learning (DL)

AI HUMAN INTERFACE

Learning Outcomes

- Use of Deep Learning (DL) Tools
  - Describe some available python deep learning tools
  - Install and configure Keras & Tensorflow
- Deep Learning architectures
  - Using Model Zoo with pre-trained networks
  - Review of deep learning architectures

Python Deep Learning



source https://www.kdnuggets.com/2018/02/top-20-python-ai-machine-learning-open-source-projects.html

Open source ML

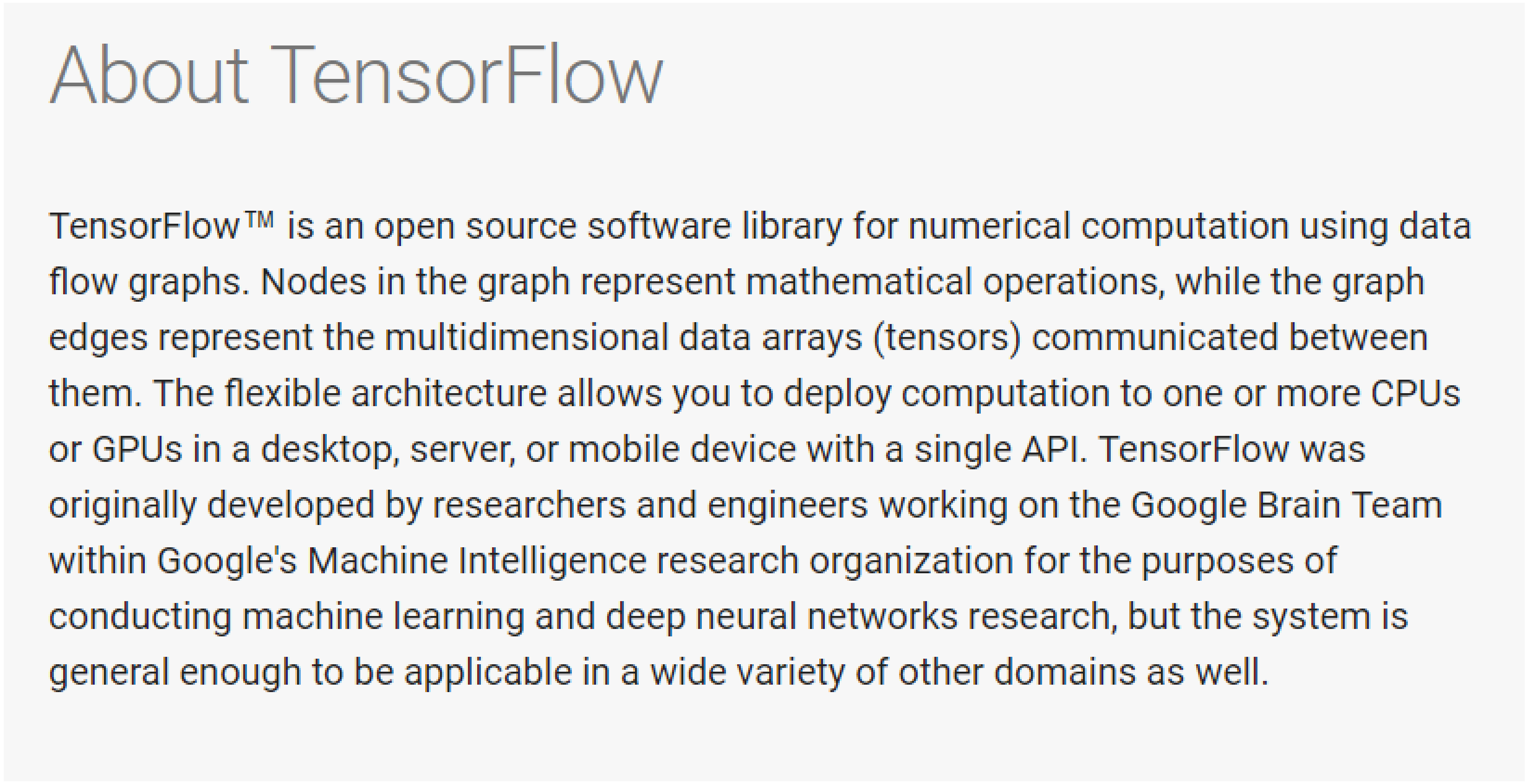
- **TensorFlow** was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization. The system is designed to facilitate research in machine learning, and to make it quick and easy to transition from research prototype to production system.
  - https://github.com/tensorflow/tensorflow
  - https://www.tensorflow.org/

- **Keras**, a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
  - https://github.com/keras-team/keras
  - https://keras.io/

Open source ML

- **PyTorch** Tensors and Dynamic neural networks in Python with strong GPU acceleration (No official support for Windows)
  - https://github.com/pytorch/pytorch
  - http://pytorch.org/
- **CNTK** The Microsoft Cognitive Toolkit (CNTK) is an open-source toolkit for commercial-grade distributed deep learning. It describes neural networks as a series of computational steps via a directed graph.
  - https://github.com/Microsoft/CNTK
  - https://www.microsoft.com/en-us/cognitive-toolkit/

TensorFlow



About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Note: It is not specifically built for Deep Learning

What can TensorFlow do?

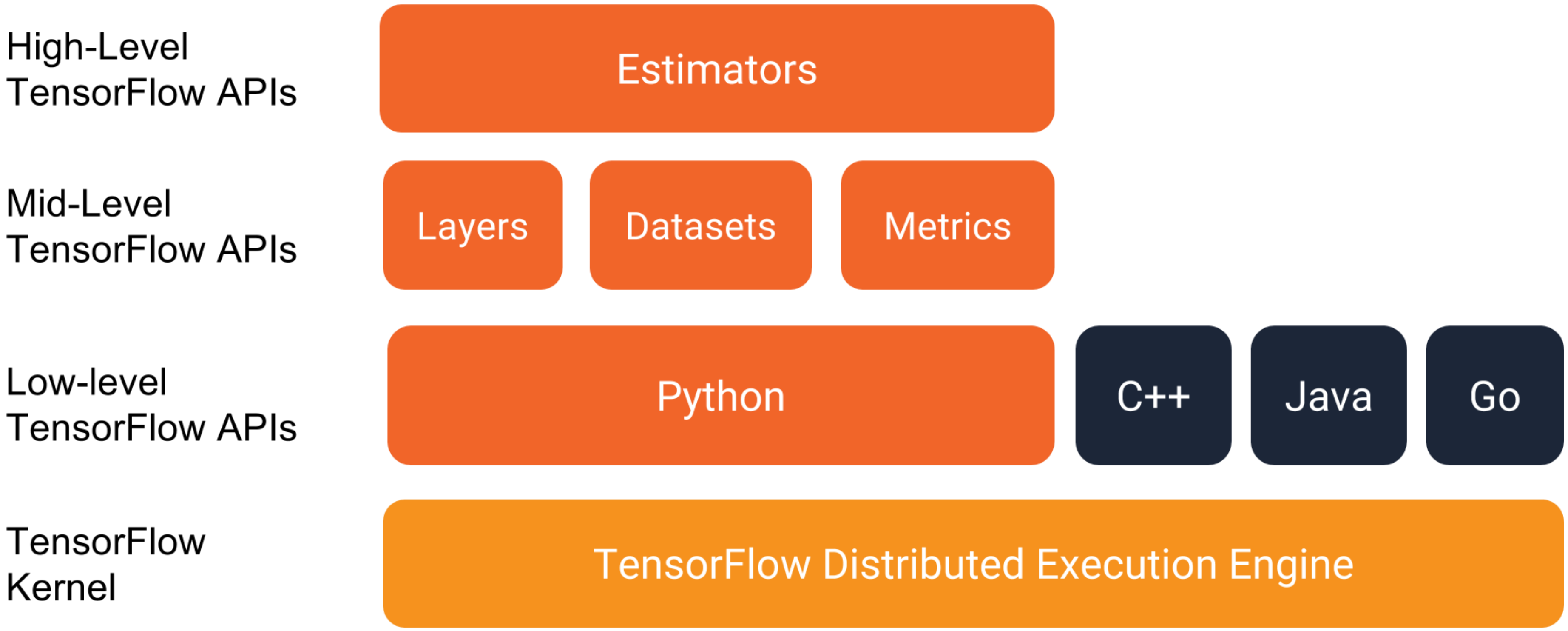
Linear Models: Linear Regression and Logistics Regression

Neural Network: Shallow NN and Deep Learning

Kernel Methods: Support Vector Machines (SVM)

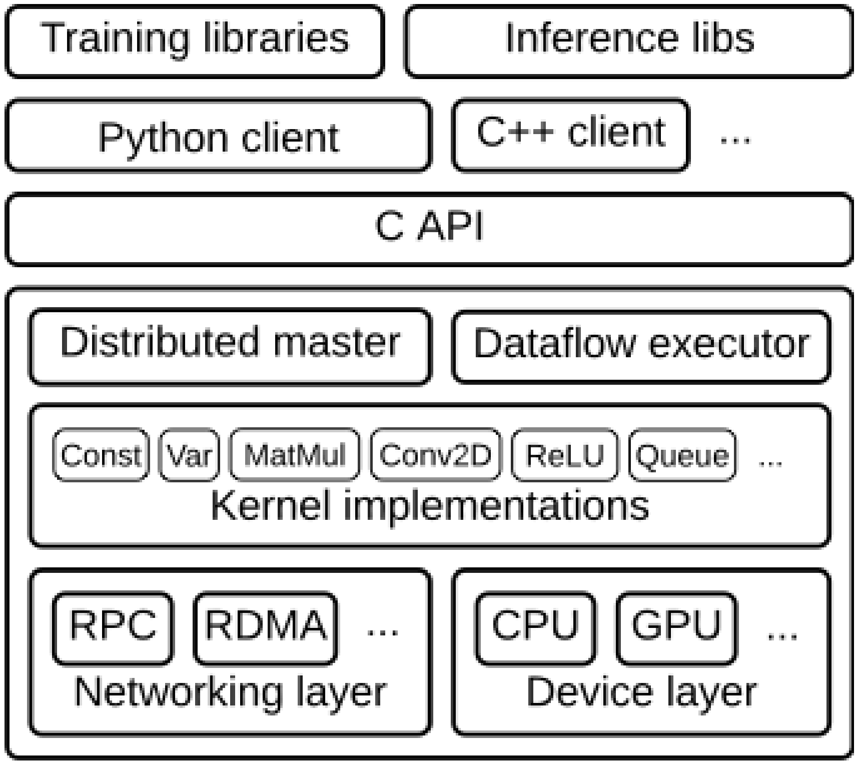
Other Mathematical Computations: Partial Differential Equations (PDEs)

TensorFlow Programming


High-Level TensorFlow APIs
Estimators
Mid-Level TensorFlow APIs
Layers
Datasets
Metrics
Low-level TensorFlow APIs
Python
C++
Java
Go
TensorFlow Kernel
TensorFlow Distributed Execution Engine

source https://www.tensorflow.org/get_started/get_started_for_beginners

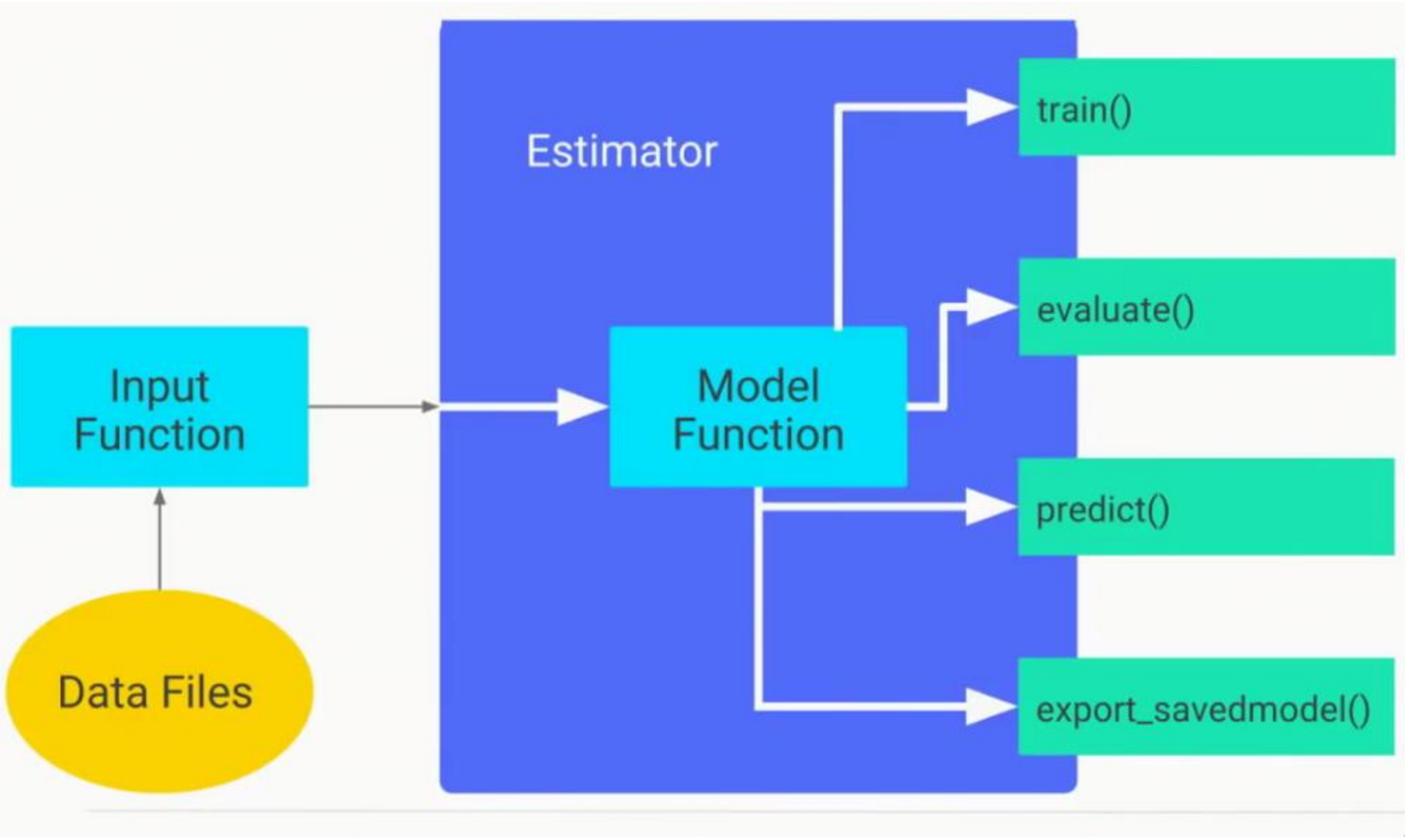
TensorFlow (Runtime stack)



source https://www.tensorflow.org/extend/architecture

TensorFlow API

- TensorFlow supports several programming languages: Python, C++. Java, Go.
- TensorFlow has APIs available in several languages both for constructing and executing a TensorFlow graph. The Python API is at present the most complete and the easiest to use, but other language APIs may be easier to integrate into projects and may offer some performance advantages in graph execution.
- Python API: https://www.tensorflow.org/api_guides/python/


Estimator
Input Function
Model Function
Data Files
train()
evaluate()
predict()
export_savedmodel()

Keras.io

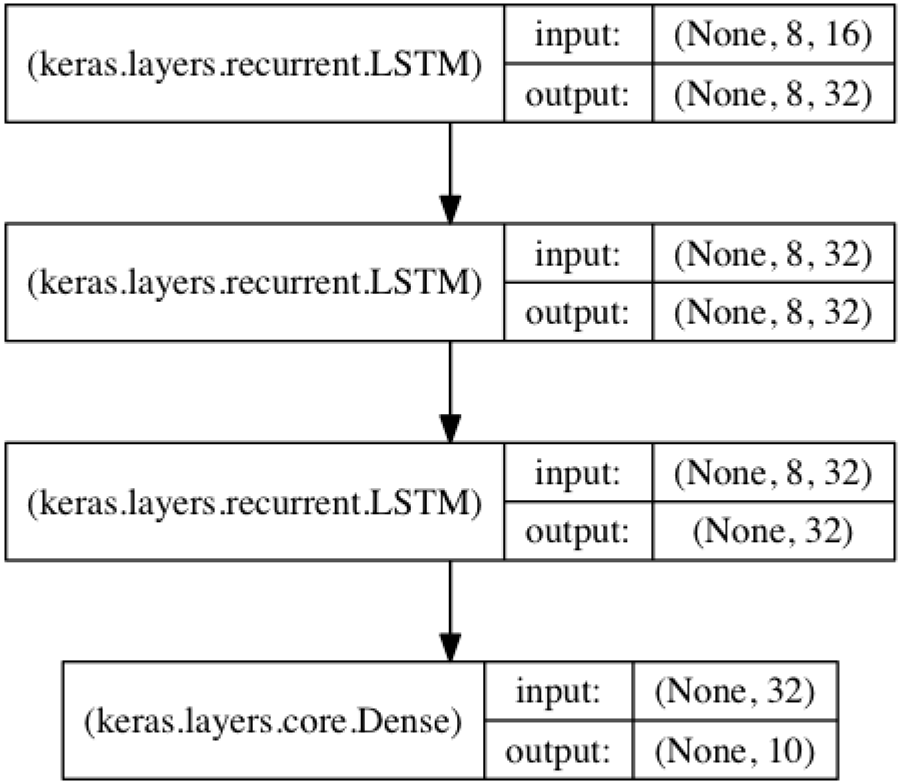
- **Keras** is a high-level neural networks API, written in Python and capable of running on top of **TensorFlow**, **CNTK**, or **Theano**. It was developed with a focus on enabling fast experimentation.
- Use Keras if you need a deep learning library that:
  - Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
  - Supports both convolutional networks and recurrent networks, as well as combinations of the two.
  - Runs seamlessly on CPU and GPU.

Keras

- The core data structure of Keras is a **model**, a way to organize layers.
- The simplest type of model is the **Sequential model**, a linear stack of layers.
- For more complex architectures, you should use the Keras **functional API**, which allows to build arbitrary graphs of layers.

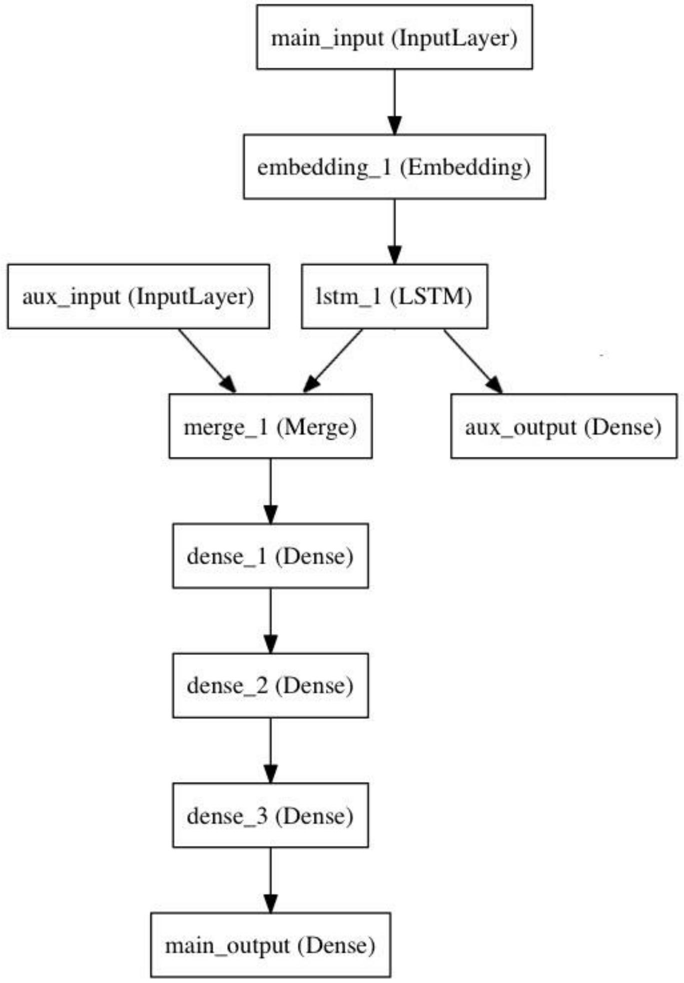
Stacked LSTM for sequence classification

The Sequential model is a linear stack of layers



source https://keras.io/getting-started/sequential-model-guide/

The Keras functional API is the way to go for defining complex models, such as multi-output models, directed acyclic graphs, or models with shared layers.



Distributed Training

Distributed Training with Tensorflow

Source: https://www.tensorflow.org/guide/distributed_training

- tf.distribute.Strategy is a TensorFlow API to distribute training across multiple GPUs, multiple machines or TPUs. Using this API, you can distribute your existing models and training code with minimal code changes.
- tf.distribute.Strategy has been designed with these key goals in mind:
  - Easy to use and support multiple user segments, including researchers, ML engineers, etc.
  - Provide good performance out of the box.
  - Easy switching between strategies.
- tf.distribute.Strategy can be used with a high-level API like Keras, and can also be used to distribute custom training loops (and, in general, any computation using TensorFlow).

Distributed Training with Keras

Source:
https://www.tensorflow.org/guide/distributed_training#using_tfdistributestrategy_with_tfkerasmodelfit

tf.distribute.Strategy is integrated into tf.keras which is TensorFlow's implementation of the Keras API specification. tf.keras is a high-level API to build and train models. By integrating into tf.keras backend, we've made it seamless for you to distribute your training written in the Keras training framework using model.fit.

```
mirrored_strategy = tf.distribute.MirroredStrategy()

with mirrored_strategy.scope():
  model = tf.keras.Sequential([tf.keras.layers.Dense(1, input_shape=(1,))])

model.compile(loss='mse', optimizer='sgd')
```

Distributed Training with Keras

Source: https://www.tensorflow.org/guide/distributed_training#using_tfdistributestrategy_with_tfkerasmodelfit

This example usees MirroredStrategy so you can run this on a machine with multiple GPUs. strategy.scope() indicates to Keras which strategy to use to distribute the training. Creating models/optimizers/metrics inside this scope allows us to create distributed variables instead of regular variables. Once this is set up, you can fit your model like you would normally. MirroredStrategy takes care of replicating the model's training on the available GPUs, aggregating gradients, and more.

Source: https://www.tensorflow.org/tutorials/distribute/keras
Source: https://www.tensorflow.org/tutorials/distribute/multi_worker_with_keras

Horovod

Source: https://horovod.ai/

Horovod is a distributed deep learning training framework for TensorFlow, Keras, PyTorch, and Apache MXNet

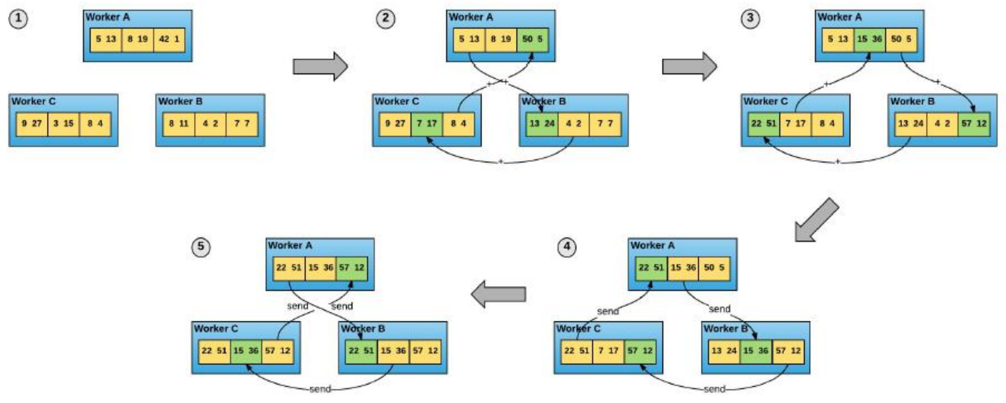
Source: https://github.com/horovod/horovod

Source: https://horovod.readthedocs.io/en/stable/

Distributed DL with Horovod

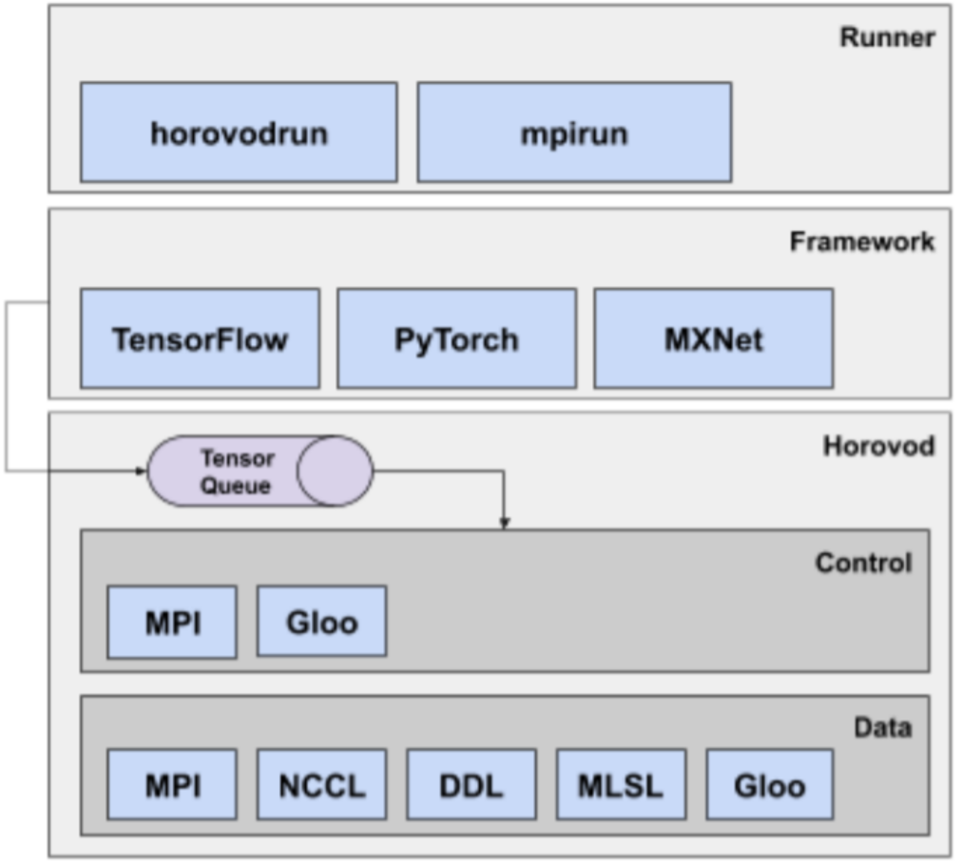
Source: https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9321-distributed-deep-learning-with-horovod.pdf

Horovod Technique: Ring-Allreduce



Patarasuk, P., & Yuan, X. (2009). Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2), 117-124. doi:10.1016/j.jpdc.2008.09.002

Source: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.5642&rep=rep1&type=pdf
Source: https://www.adaltas.com/en/2019/11/15/avoid-deep-learning-bottlenecks/



Horovod Stack

Model Zoo

A REPOSITORY OF PRE-TRAINED MODELS FOR A PARTICULAR DEEP LEARNING FRAMEWORK

Tensorflow Model Zoo

- https://github.com/tensorflow/models
  - This repository contains a number of different models implemented in TensorFlow:
    - The official models are a collection of example models that use TensorFlow's high-level APIs. They are intended to be well-maintained, tested, and kept up to date with the latest stable TensorFlow API. They should also be reasonably optimized for fast performance while still being easy to read. We especially recommend newer TensorFlow users to start here.
    - The research models are a large collection of models implemented in TensorFlow by researchers. They are not officially supported or available in release branches; it is up to the individual researchers to maintain the models and/or provide support on issues and pull requests.
    - The samples folder contains code snippets and smaller models that demonstrate features of TensorFlow, including code presented in various blog posts.

Keras Model Zoo

- https://keras.io/applications/#available-models
- **Keras Applications** are **deep learning models** that are made available alongside **pre-trained weights**. These models can be used for prediction, feature extraction, and fine-tuning..
- Weights are downloaded automatically when instantiating a model. They are stored at ~/.keras/models/.

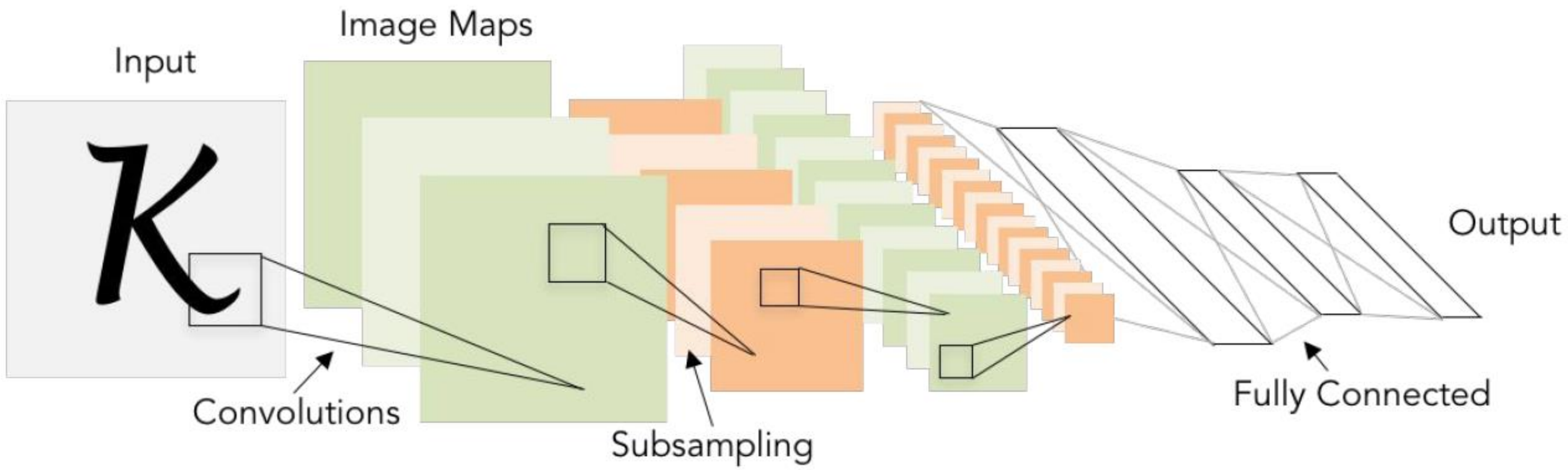
- Models for image classification with weights trained on ImageNet:
  - Xception
  - VGG16
  - VGG19
  - ResNet50
  - InceptionV3
  - InceptionResNetV2
  - MobileNet
  - DenseNet
  - NASNet

Source https://github.com/fchollet/deep-learning-models

Review of Deep Learning Architectures

Review: LeNet-5

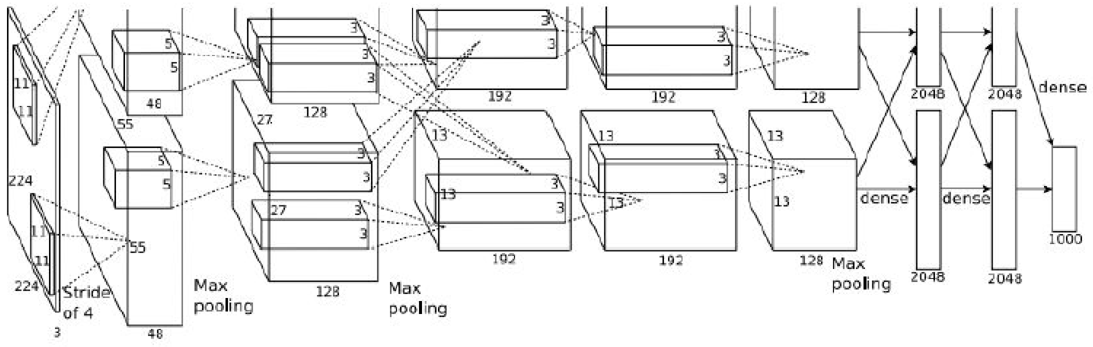
[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

Case Study: AlexNet

[Krizhevsky et al. 2012]



Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

Case Study: VGGNet

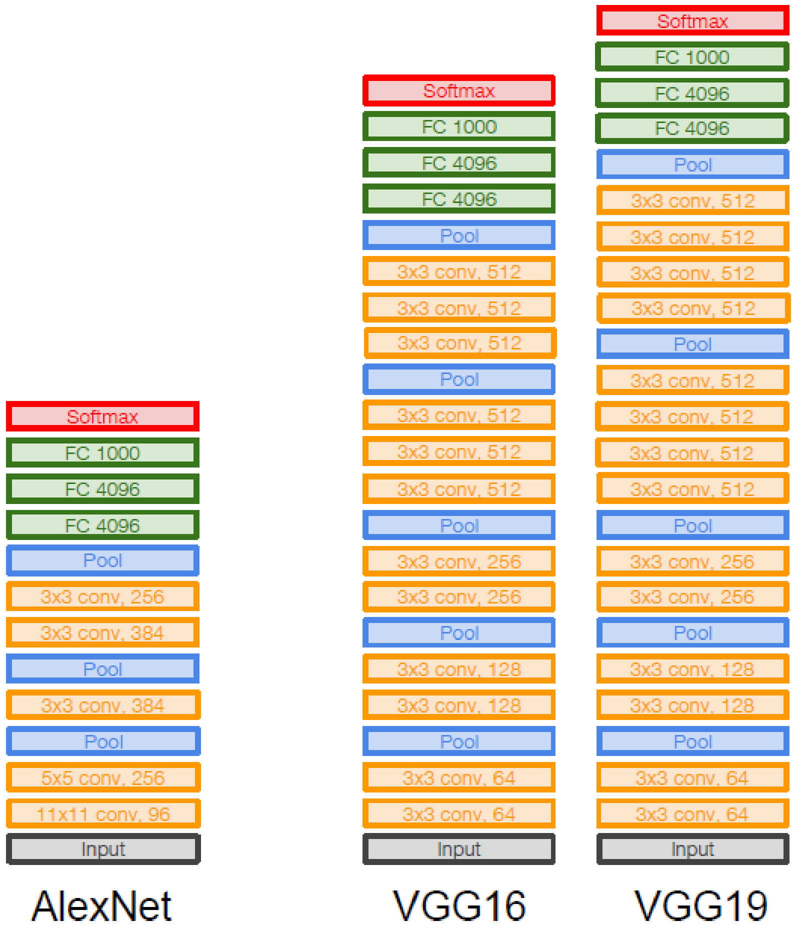
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14

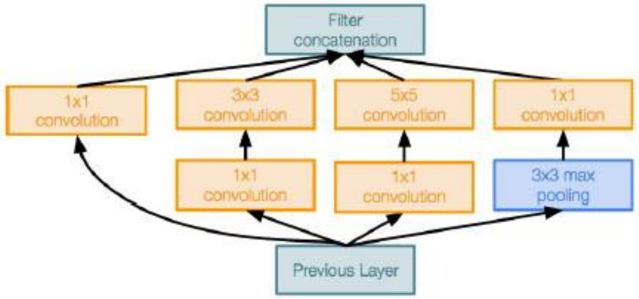


Case Study: GoogLeNet

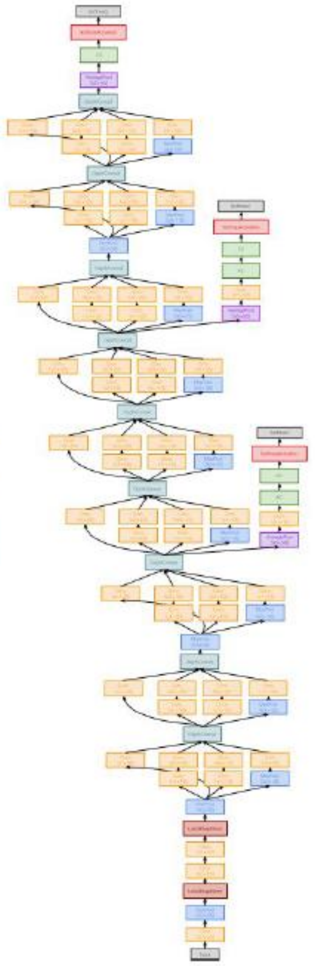
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC’14 classification winner
  (6.7% top 5 error)



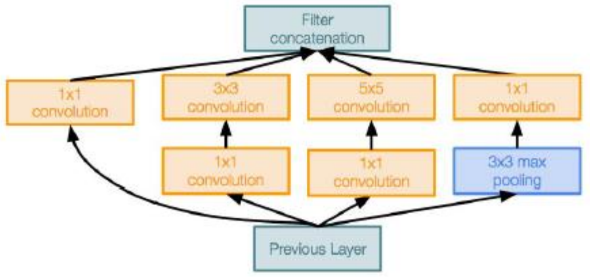
Inception module



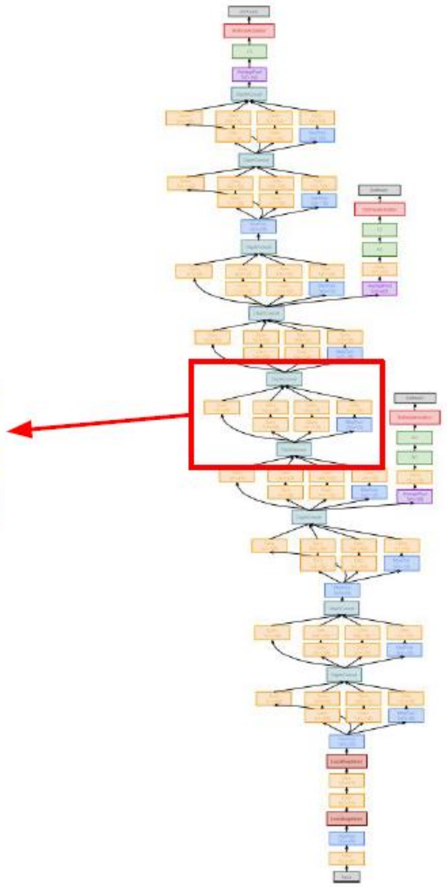
Case Study: GoogLeNet

[Szegedy et al., 2014]

"Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other



Inception module

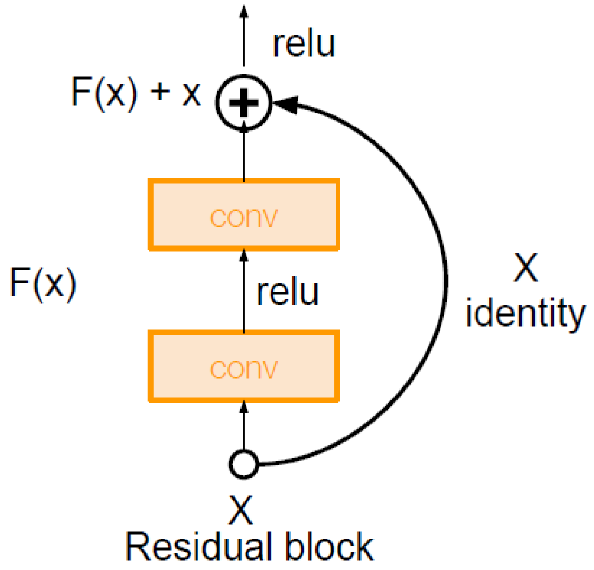


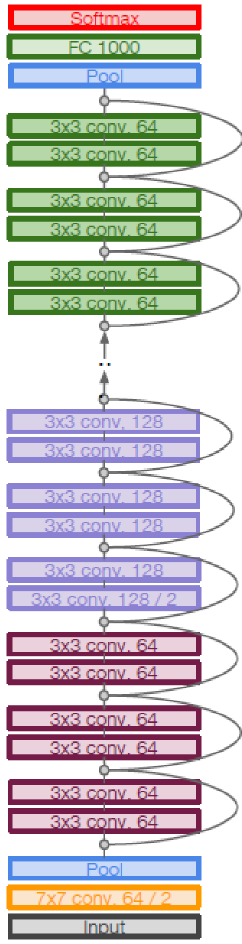
Case Study: ResNet

[He et al., 2015]

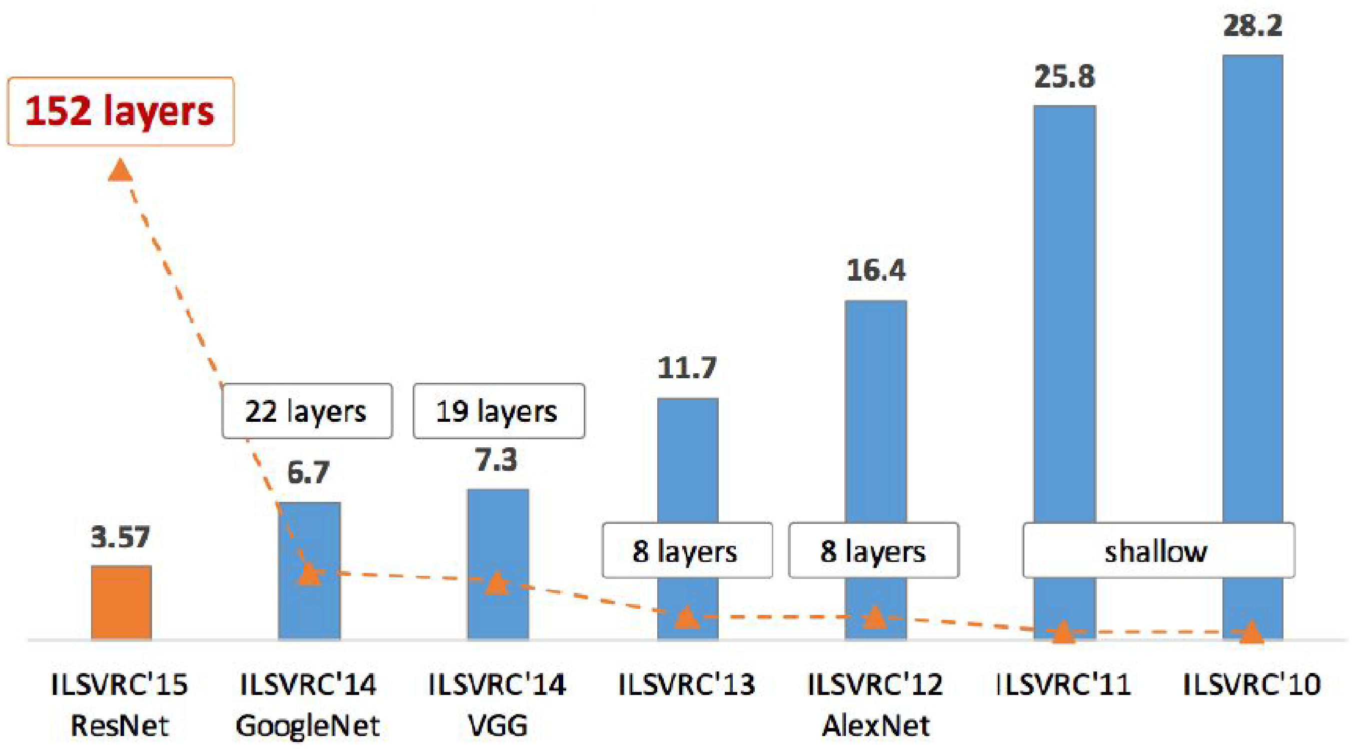
Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

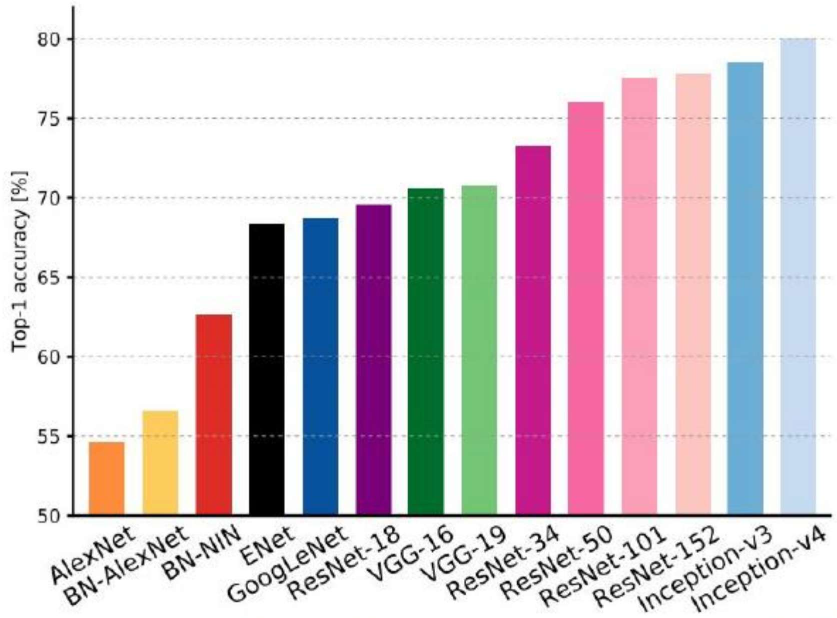




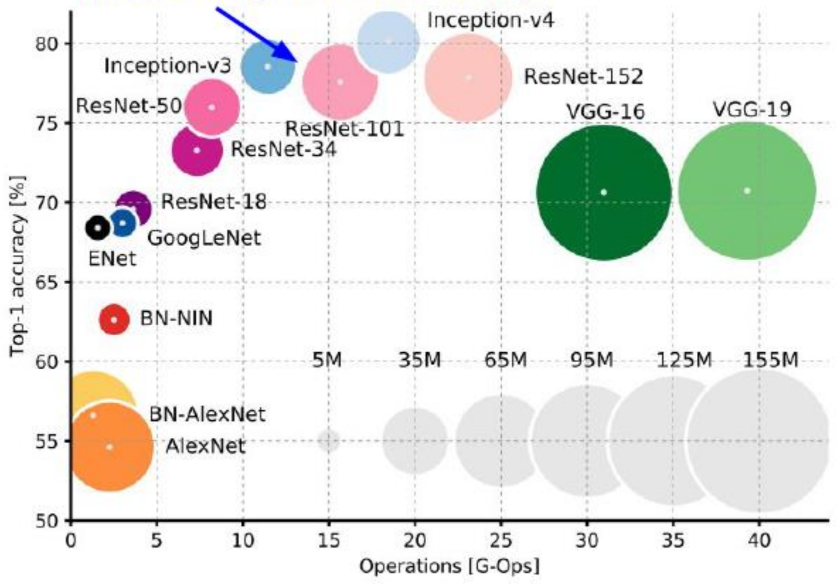
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Comparing complexity...



ResNet:
Moderate efficiency depending on model, highest accuracy



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Summary

What we have learnt:

- Tensorflow and Keras can be used to implement Deep Learning
- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow