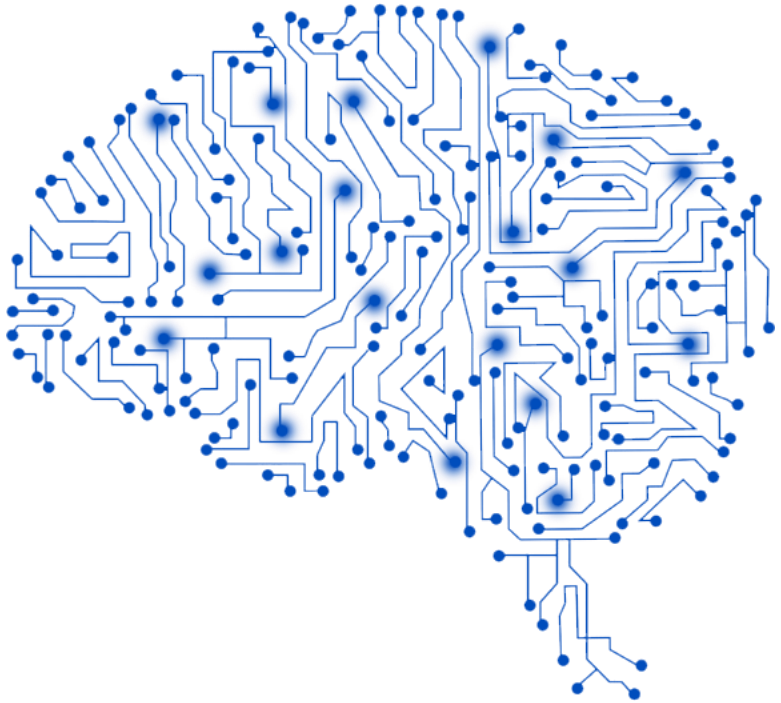
Separating Hype from Reality in Deep Learning

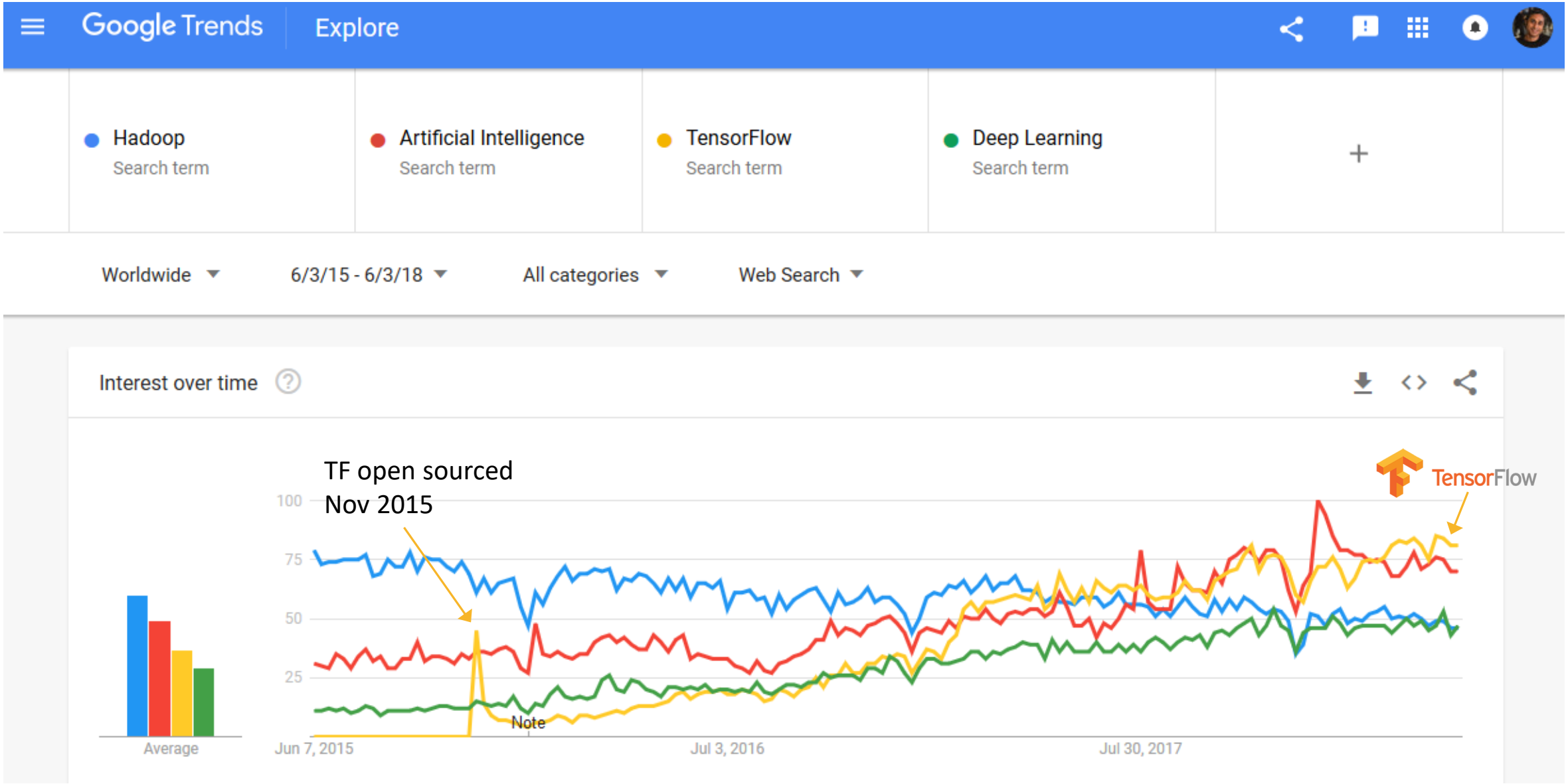


Sameer Farooqui

Slides:
http://bit.ly/hype-deep-learning

Presented June 2018 @ SPARK SUMMIT

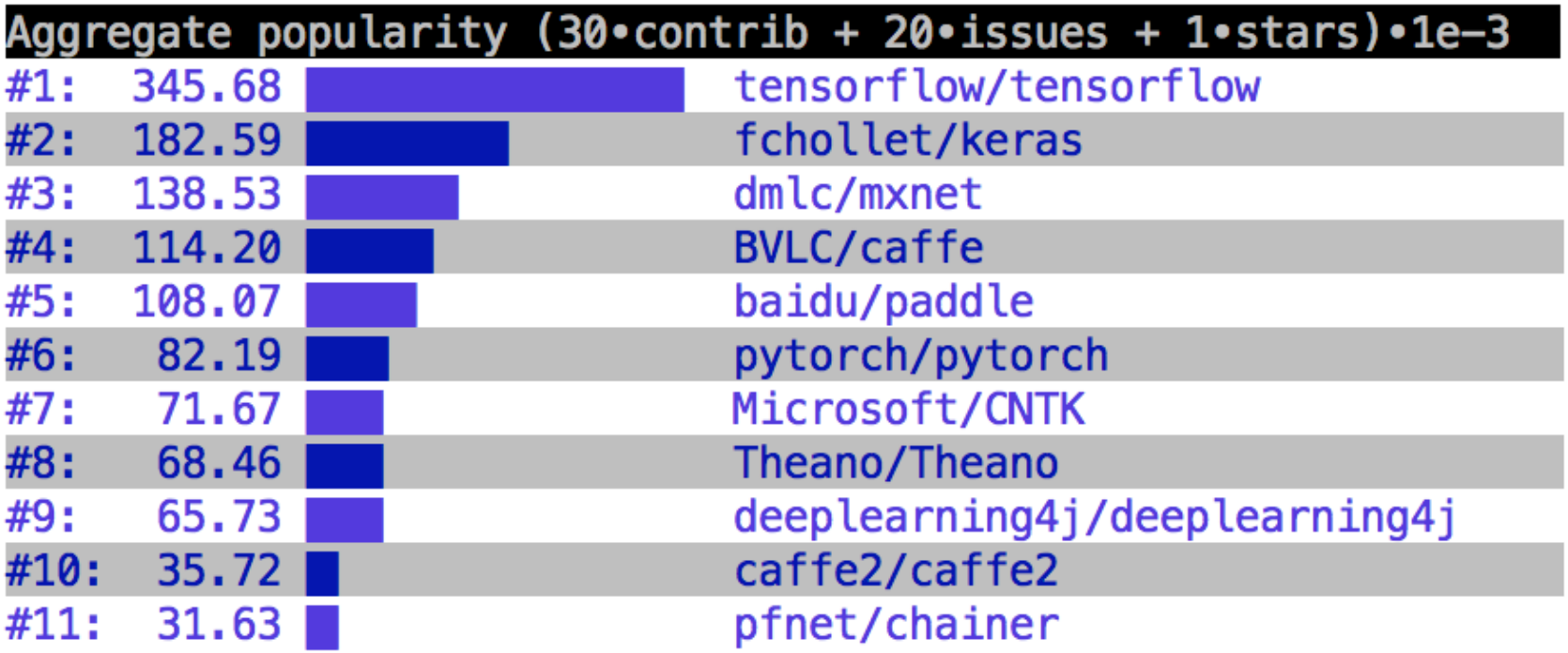
Last 3 Years Trends



François Chollet 
@fchollet

Following

And here are the GitHub popularity metrics at this time.



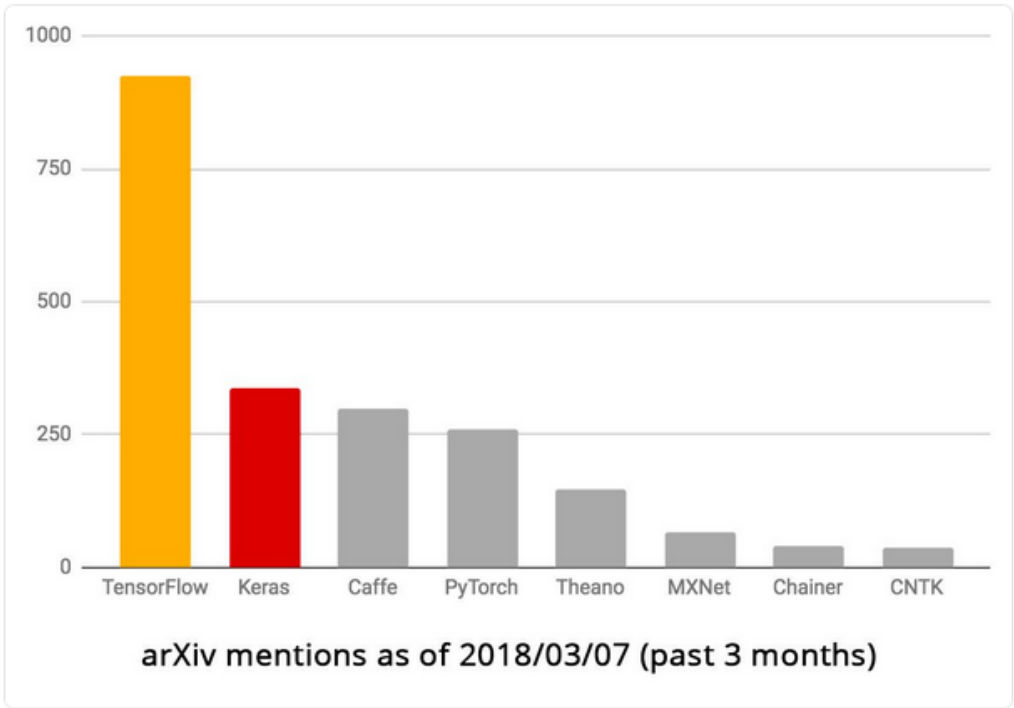
12:02 AM - 9 Mar 2018

43 Retweets **164** Likes 

François Chollet
@fchollet

Following

TensorFlow is the platform of choice for deep learning in the research community. These are deep learning framework mentions on arXiv over the past 3 months



4:40 PM - 8 Mar 2018

335 Retweets **794** Likes

TensorFlow

- Open-source software library for machine learning
- System for building, training and serving neural network models
- Can run on CPUs, GPUs and TPUs
- Developed by the Google Brain team for internal use:
  - **Google Search** (RankBrain)
  - **Image Classification** (Inception architecture for ImageNet 2014)
  - **SmartReply** (Deep LSTM model to auto-generate email responses)
  - **Google Translate** (On-Device Computer Vision for Optical Character Recognition for real time translations)
  - Drug Discovery (in collaboration with Stanford University; deep neural network model for identifying promising drug candidates)

Initial Release: November 2015

License: Apache 2.0

Written in: C++, Python, CUDA
~ 1 million LOC

Platforms: Linux, macOS, Win, Android

Current version: 1.7.0 *(March 27, 2018)*

APIs: Python, C++, Java, Go

https://github.com/tensorflow/tensorflow

https://twitter.com/tensorflow

Common Neural Networks

- **Deep Feed Forward Neural Network:** Regression or Classification
  (predict a #) (predict a category)
- **Convolutional Neural Network:** Computer Vision
- **Recurrent Neural Network (or LSTM):** Time Series or Language
- **Generative Adversarial Network:** Creating fake images, Creativity

Who am I?

Sameer Farooqui

Google Cloud

- 300+ PB in HDFS
- 20k+ Hadoop nodes

Resigned from Databricks in Dec 2016 to study Deep Learning

Currently: Strategic Cloud Engineer for Big Data @ Google assigned to Twitter

Taught 150+ classes on Hadoop, NoSQL, Cassandra, HBase, Apache Spark

Nerd famous. YouTube videos with 328k, 120k and 60k views.

Traveled to 40+ countries (❤️: Palau, Machu Picchu, Andaman Islands)

When not in front of laptop: Snowboarding, Scuba diving, Wakeboarding

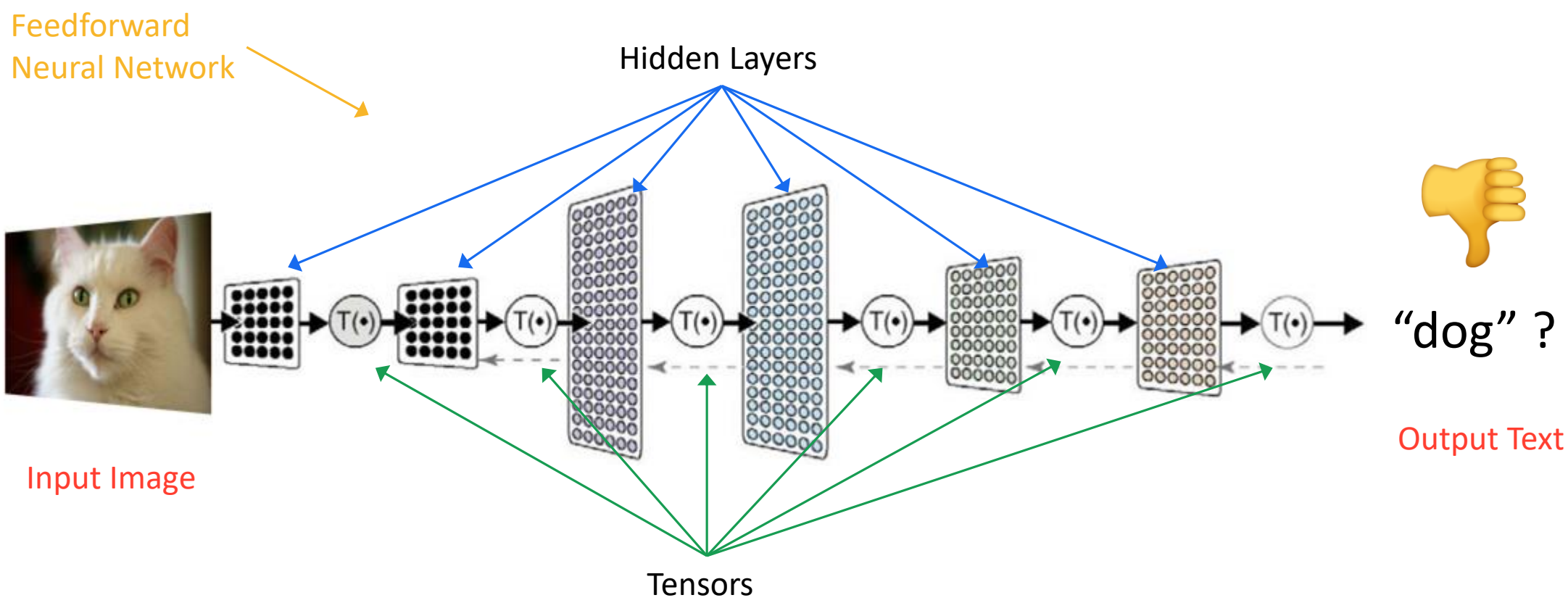
This Talk

- **Introduction:** 15 min – technical introduction to Neural Networks
- **AutoML Vision:** 10 min – demo, classify images of clouds
- **AutoML Vision:** 10 min – How it works… transfer learning & RL
- **CNN:** 15 min – Convolutional Neural Networks
- **Transfer Learning:** 5 mins – Steal pre-trained layers
- **GANs & MultiModel:** Maybe… if there's time.

Deep Neural Networks

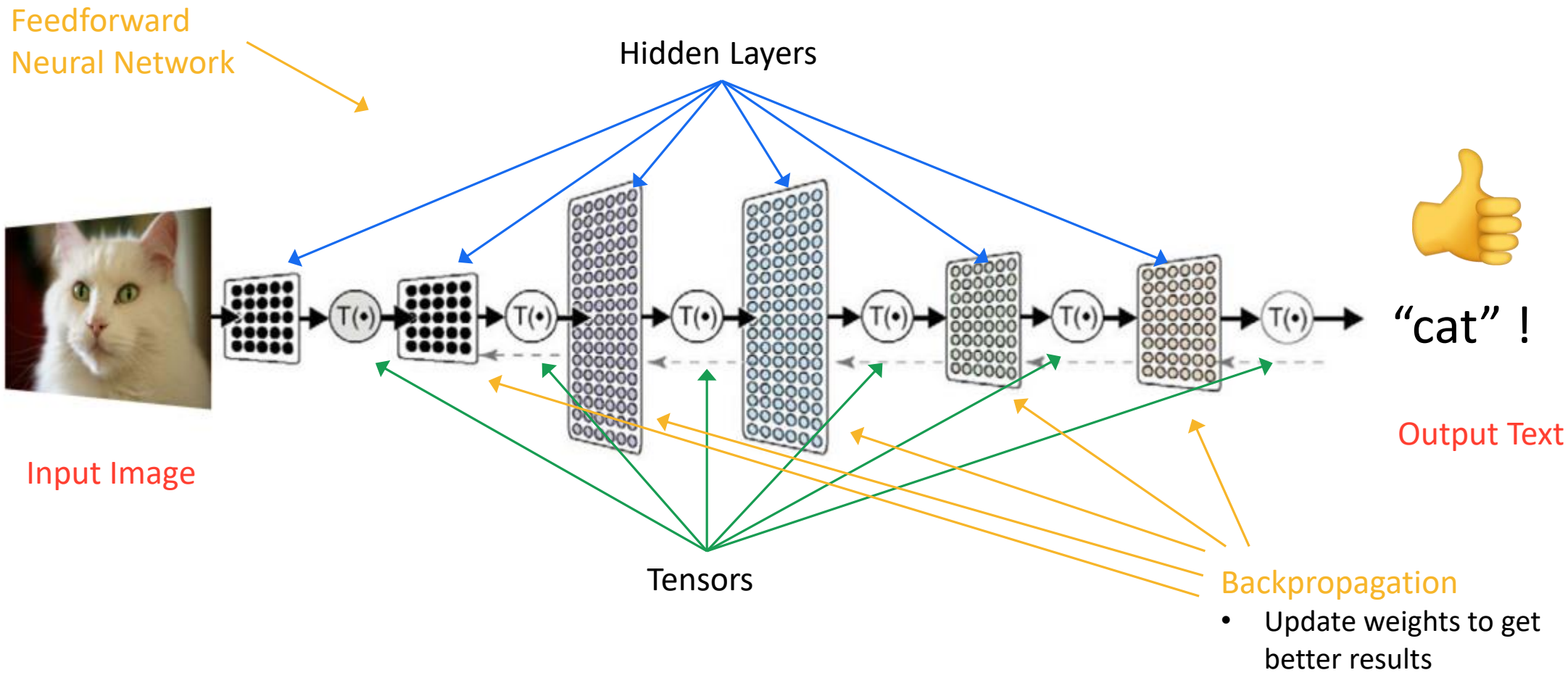
the fastest 15 min crash course in Deep Learning…

An object detection Neural Network



Source: Jeff Dean

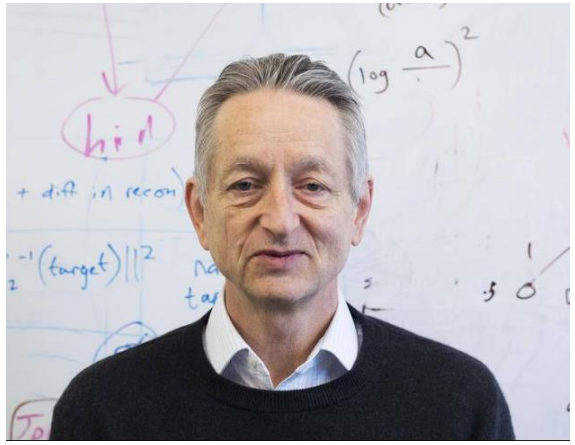
An object detection Neural Network



Source: Jeff Dean

Pioneers in AI: Geoffrey Hinton

(Perhaps the most important person currently in the AI field!)



- Born: 1947 in London *(fun fact: great-great-grandson of George Bool)*
- One of the first researchers to demonstrate use of backpropagation algorithm for training neural networks ("His (1986) paper was basically the foundation of the second wave of neural nets" – Yann LeCun)
- Invented dropout & Deep Belief Nets
- Co-invented Bolzmann machines (a stochastic RNN)
- Two of his students work the ImageNet computer vision contest in 2012, by inventing a neural network that identified objects with almost 2x the accuracy of the nearest competitor
- Currently divides his time working for Google Brain and University of Toronto
- His research group in Toronto made major breakthroughs in Deep Learning that revolutionized speech recognition and object classification

One of Hinton's PhD students!

Reddit AMA

Homepage at U of Toronto

Research @ Google

What is a Tensor?

- A container for numerical data
- A generalization of matrices to an arbitrary number of dimensions
  (A matrix is just a 2D tensor)
- Creating tensors from your data is the first step to using a neural network
- Tensors are fed into the neural network to run deep learning algorithms

2D Tensors

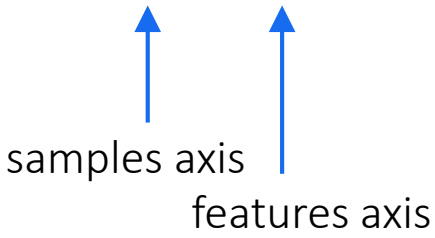
- Most common case
- Each data collection (sample) is a vector of features
- A bunch of vectors make a 2D matrix

Education Feature encoding:

0 – Did not complete High School
1 – High School
2 – Associate's Degree
3 – Bachelor's Degree
4 – Master's Degree
5 – PhD / MD

Example #1:

- A dataset of 100,000 people with each person's age, education and income.
- Each record/sample has 3 values
- 2D Tensor shape: (100000, 3)




```
27, 3, 80000
35, 5, 160000
18, 1, 35000
    .
    .
```

100k samples

3D Tensors

- Used for timeseries or sequence data
- 3rd axis is for time

Example #1:

- A dataset of stock prices for 250 days
- Every minute record the current, highest and lowest price of the stock in that minute
- 3D Tensor shape: (250, 390, 3)

Minutes in a trading day

A 3D timeseries data tensor:

Image source:

3D Tensors

- Used for timeseries or sequence data
- 3rd axis is for time

Example #2:

- A dataset of 1 million tweets
- Each tweet is a sequence of 140 characters out of an alphabet of 128 unique characters
- 3D Tensor shape: (1000000, 140, 128)

Each char encoded as binary vector of 128

A 3D timeseries data tensor:

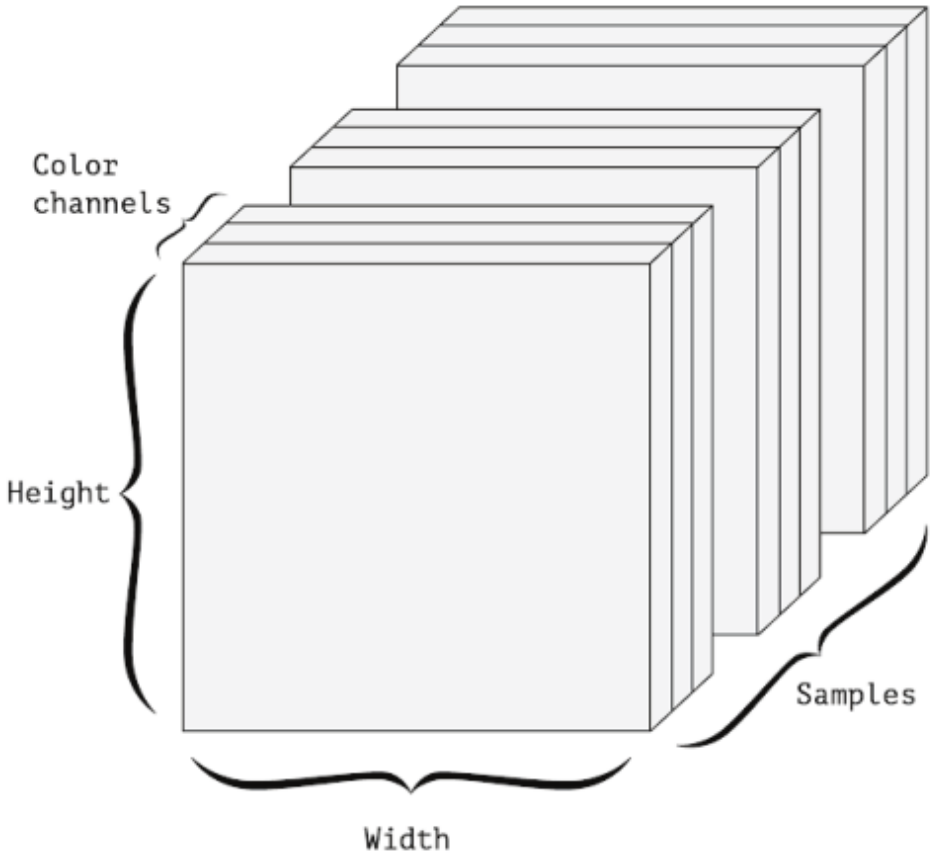
128 size Binary Vector
(all zeros, except for one index corresponding to the char)

Image source:

4D Tensors

- Each image is 3 dimensions (width, height & color depth)
- A batch of images becomes a 4D Tensor

A 4D images tensor:



Example:

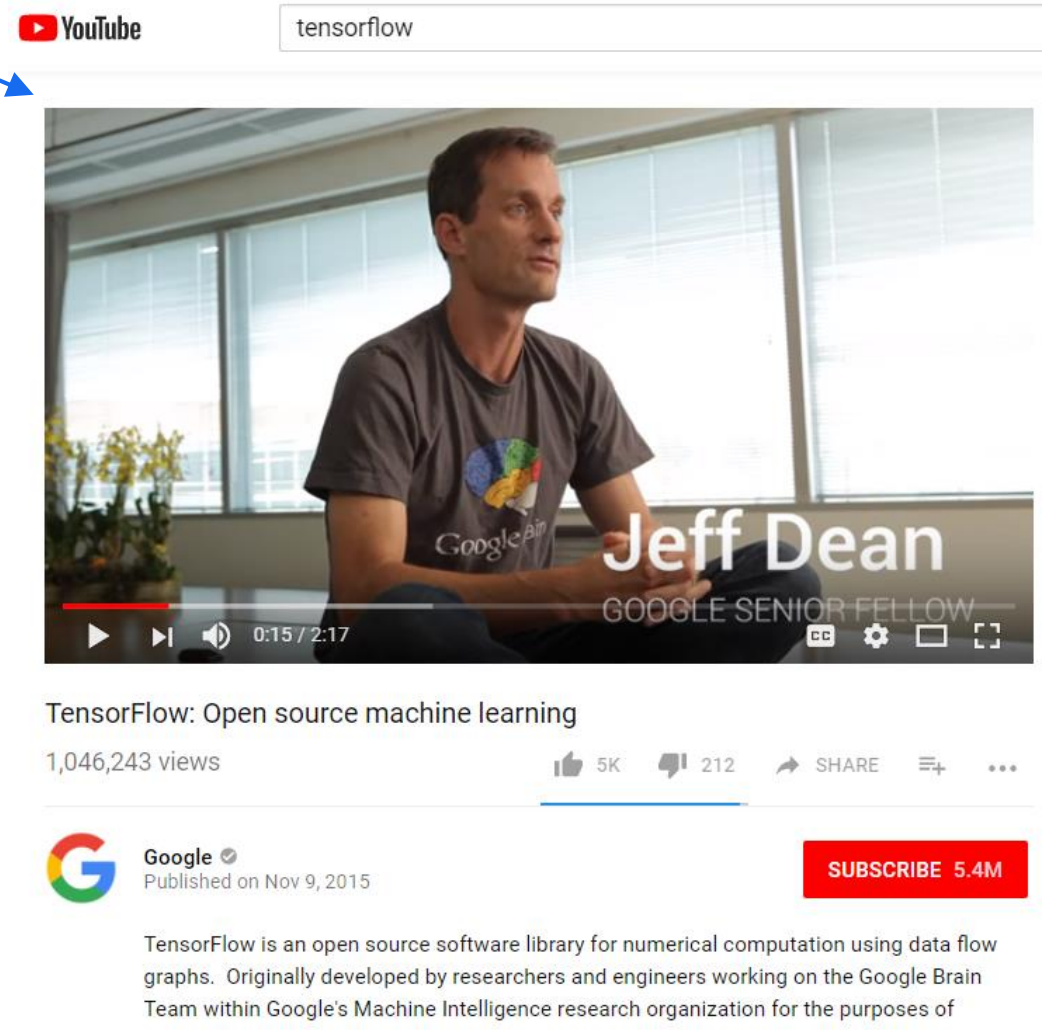
- Width and Height are just pixels
- 4D Tensor shape: (samples, width, height, color_depth)

RBG: red, blue, green channels

Image source:

5D Tensors

Most popular YouTube video for “tensorflow”:



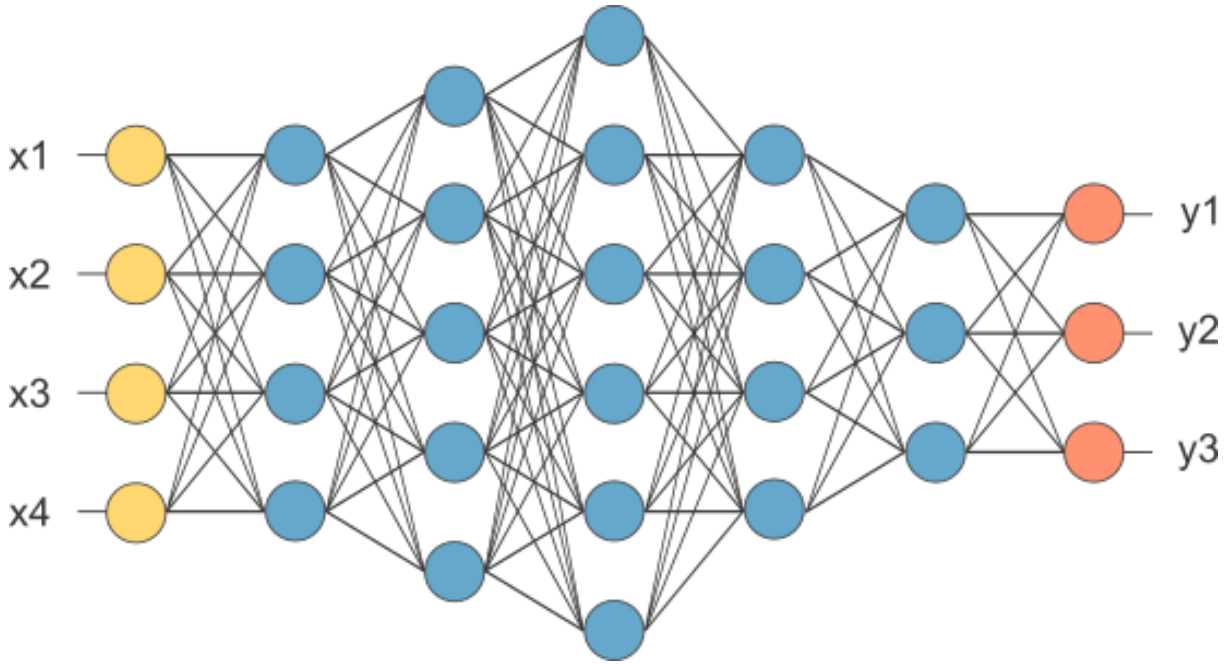
- One of the few times you need 5D tensors
- Video is a sequence of frames (images)

Example:

- 5D Tensor shape:
  (samples, frames, width, height, color_depth)

- A 60-second, 256x144 YouTube video clip sampled at 4 frames per second would have 240 frames
- A batch of 10 such video clips would be stored in a tensor of shape: (10, 240, 256, 144, 3)
- That’s 265,420,800 values!

You still need to do feature extraction and standardization if using Deep Learning!

Tensors → 

→ Results

Neural Network

Build a Pass/Fail predictor: Binary Classification

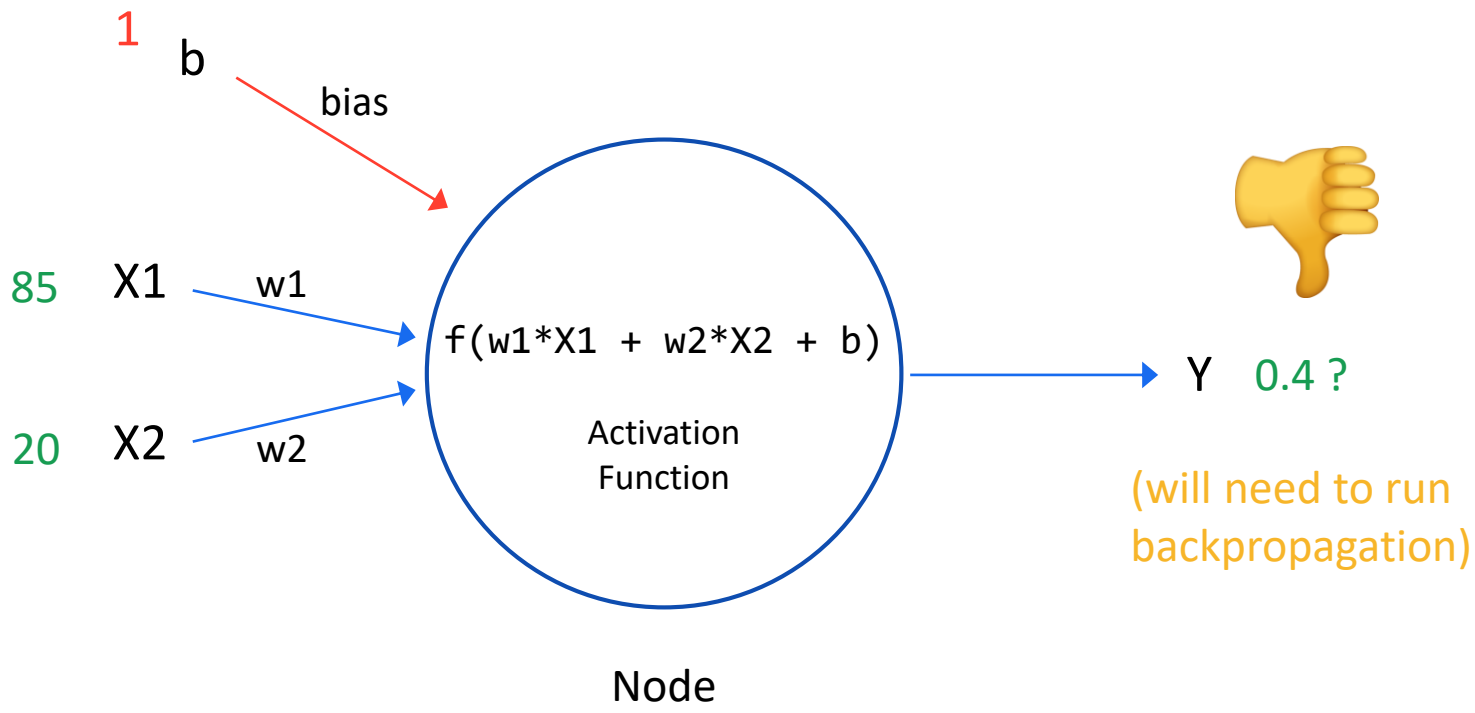
Training Data

Mid-term Grade	Hours Studied	Final Class Result
85	20	1 (pass)
67	15	0 (fail)
94	25	1 (pass)
..	..	..

Test Data

Mid-term Grade	Hours Studied	Final Class Result
75	30	?

Training A Single Neuron



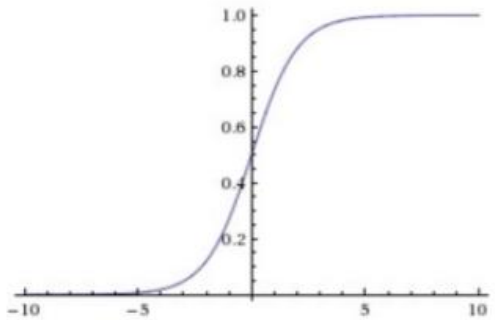
Weights may be randomly assigned initially, so initial results are bad…

Activation Functions

Sigmoid

- takes a real-valued input and squashes it to range between 0 and 1

$$\sigma(x) = 1 / (1 + \exp(-x))$$

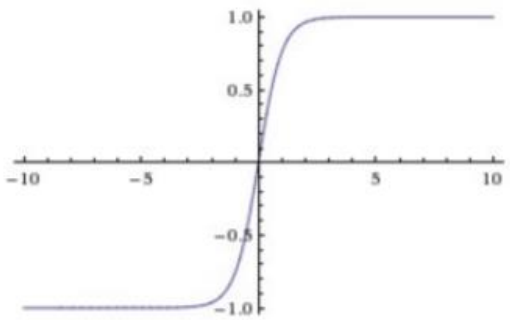


Sigmoid

tanh

- takes a real-valued input and squashes it to the range [-1, 1]

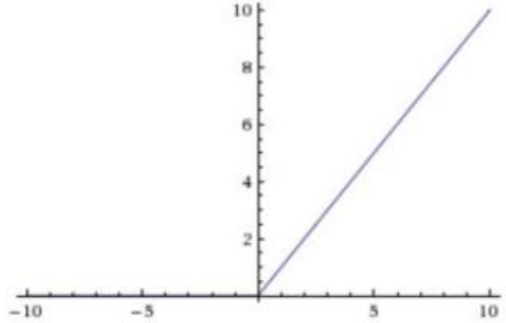
$$\tanh(x) = 2\sigma(2x) - 1$$



ReLU

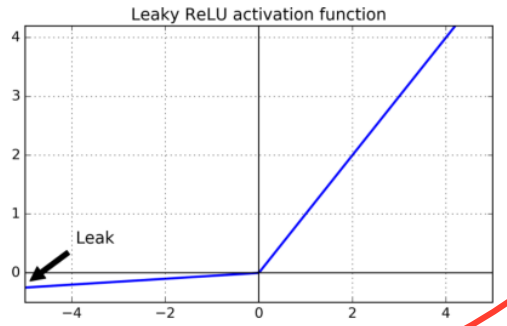
- ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero *(replaces negative values with zero)*

$$f(x) = \max(0, x)$$

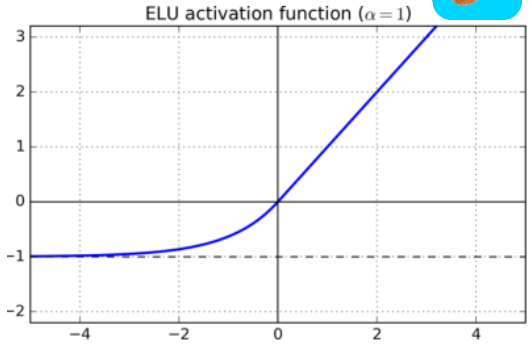


Others

LeakyReLU:



ELU:



[2016 ELU Paper]

Source:

Bias

Maybe you only want the neuron to activate if the weighted sum (of incoming activations * weights) is > 100.

So you want a bias for it to be inactive.

So just add -100 to the weighted sum before plugging it into the current activation function.

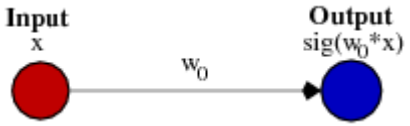
Bias = how high the weighted sum needs to be before the neuron gets meaningfully active.

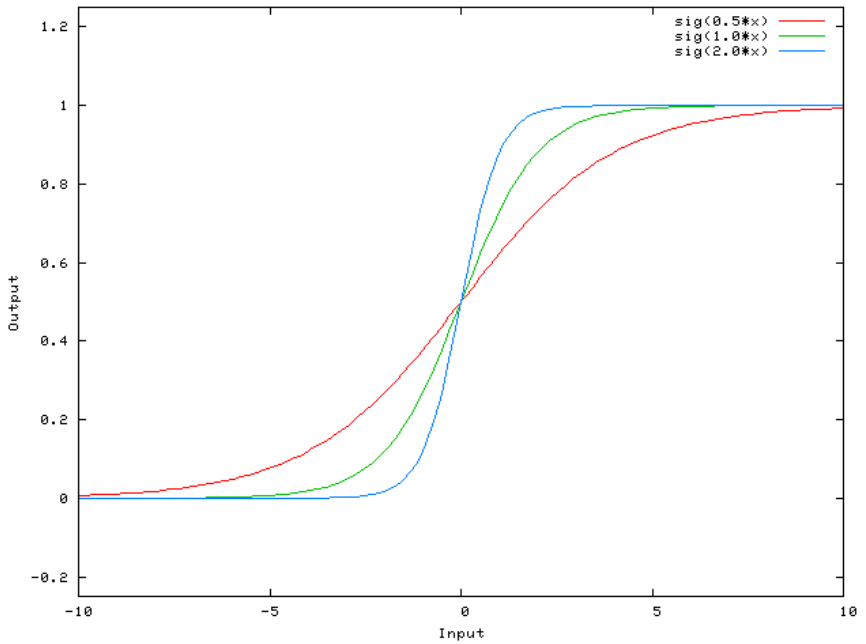
Bias

(which typically just outputs 1)

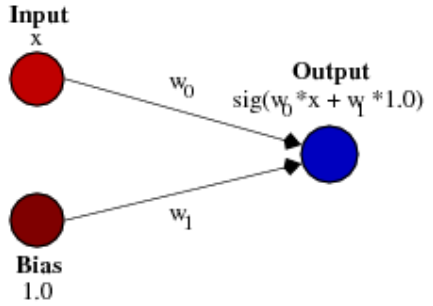
A bias value allows you to shift the activation function to the left or right:

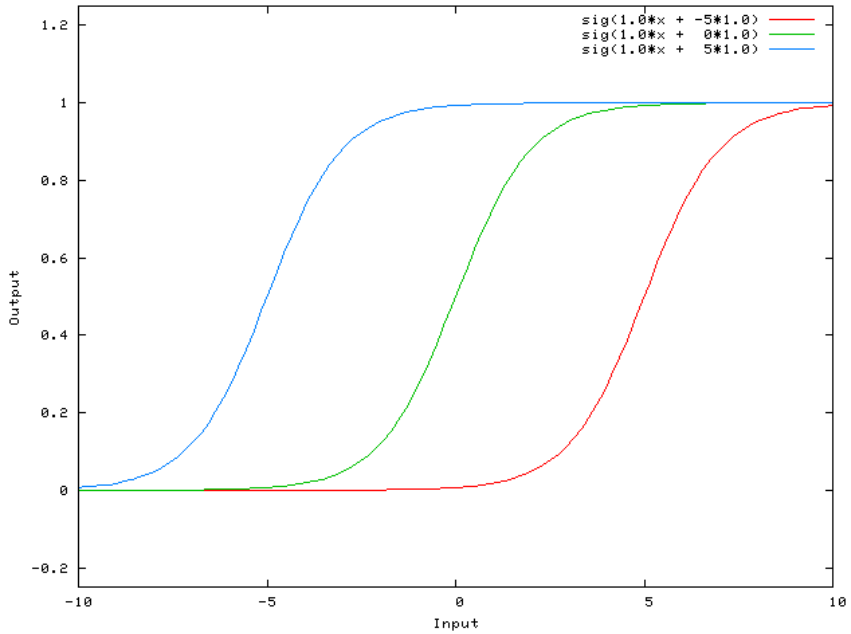
1-input, 1-output network that has no bias:





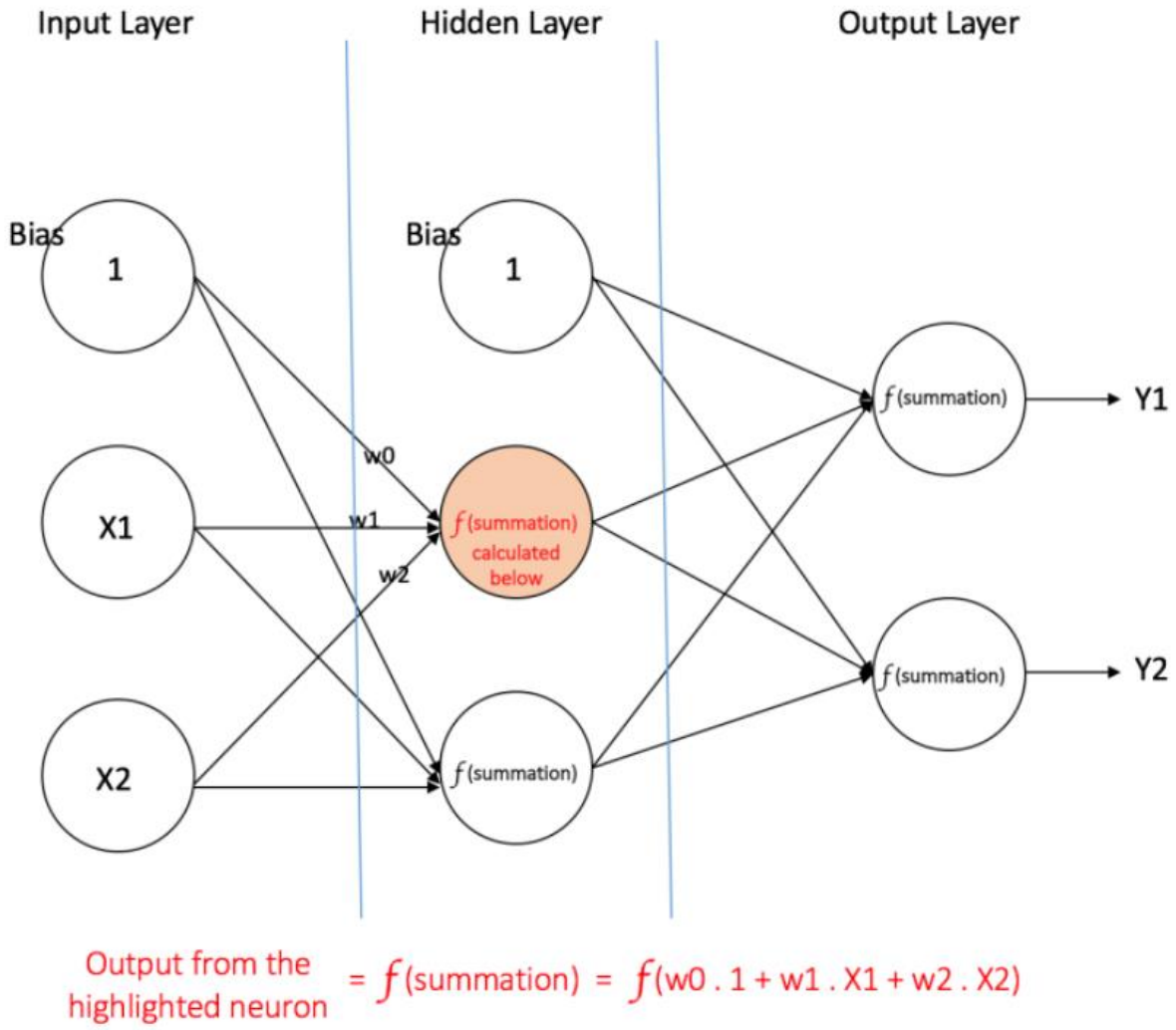
With bias added:





Source: https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

Feedforward Neural Network



- **Input Layer:** Bias node has value of 1. X1 and X2 are external numeric values. No computation is performed here, so outputs from these nodes are fed directly into the first hidden layer.
- **Hidden Nodes:** Bias is again 1. Output of the other 2 nodes depends on the inputs 1, x1, x2 and their associated weights. The two bottom nodes get activated based on the activation function
- **Output Layer:** The two nodes take inputs from the hidden layer and perform similar activation functions. Y1 and Y2 are the outputs of the neural net.

Backpropagation

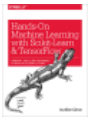
- The process by which a Multi Layer Perceptron learns the correct weights
- It is a supervised training scheme, so it learns from labeled data
- Initially all the edge weights are randomly assigned

But in modern practice, Xavier or He initialization is used

- For every input in the training dataset, the ANN is activated and its output is observed
- This output is compared with the desired output that we already know, and the error is “propagated” back to the previous layer
- This error is noted and the weights are “adjusted” accordingly. This process is repeated until the output error is below a predetermined threshold
- After all the input records are fed in and after multiple backprops, the Artificial Neural Network can be tested on new, unseen data

Backpropagation Details

- Backpropagation is a technique used to train artificial neural networks
- It first computes the gradients of the cost/loss function with regards to every model parameter (all the weights and biases), and then it performs a Gradient Descent step using these gradients.
- This backpropagation step is typically performed thousands or millions of times, using many training batches, until the model parameters converge to values that (hopefully) minimize the cost function.
- To compute the gradients, backpropagation uses reverse-mode autodiff.
- Reverse-mode autodiff performs a forward pass through a computation graph, computing every node's value for the current training batch, and then it performs a reverse pass, computing all the gradients at once

Source: 

Stochastic Gradient Descent

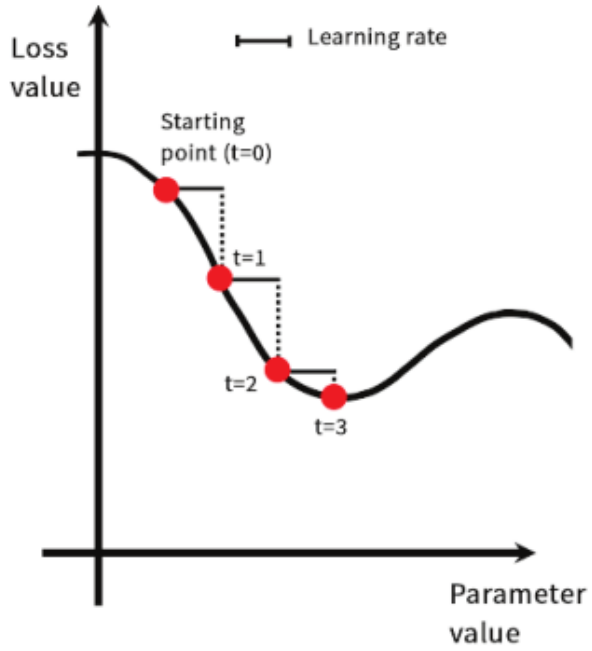


Figure 2.10 SGD down a 1D loss curve (1 learnable parameter).

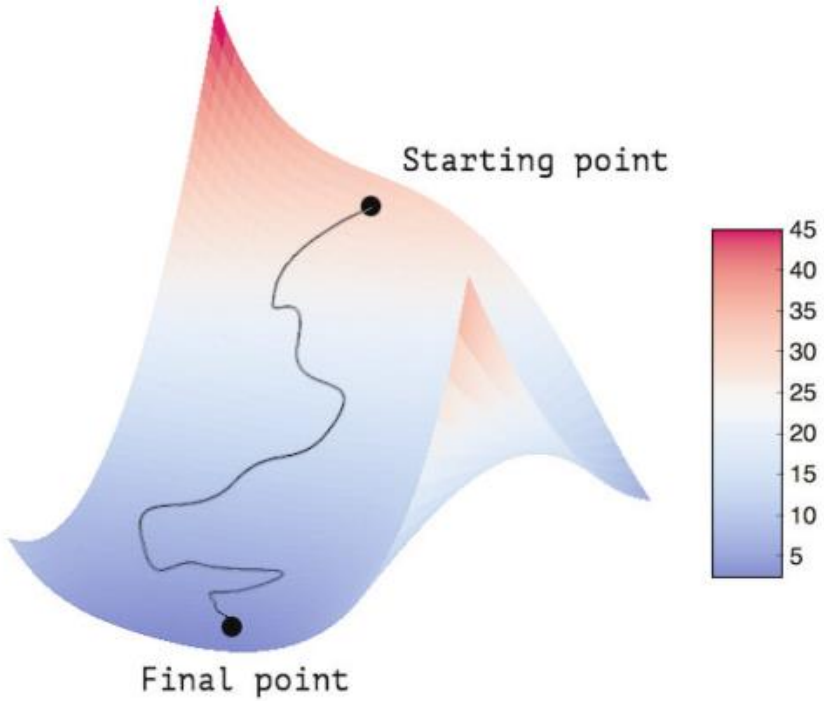


Figure 2.11 Gradient descent down a 2D loss surface (2 learnable parameters).

Image source:

Stochastic Gradient Descent

- The term "stochastic" refers to the fact that each record of data is drawn at random
- Minibatch SGD means the NN pulls a batch of data at a time, before running backprop
- Additionally, there exist multiple variants of SGD that differ by taking into account previous weight updates when computing the next weight update, rather than just looking at the current value of the gradients.
- There is, for instance, "SGD with momentum", but also “Adam”, “Adagrad", RMSprop", and several others. Such variants are known as "optimization methods" or "optimizers".

Image source: 

Loss vs Optimizer

- You have to define a “loss” function and an “optimizer” before running training
- The "loss" is the quantity that you will attempt to minimize during training, so it should represent a measure of success on the task you are trying to solve.
  (Mean Squared Error is commonly used for regression, or binary crossentropy for binary classification, or categorical crossentropy for multi-class classification)
- The "optimizer" specifies the exact way in which the gradient of the loss will be used to update *parameters (Most Google ML SCEs are using Adam)*

Image source: 

Pop Quiz!

Q) What are the different hyper-parameters you can tune in a basic Multi-Layer Perceptron (an Artificial Neural Network)?

A)

- # of hidden layers
- # of neurons in each hidden layer
- The activation function used in each hidden layer (typically ReLU or Leaky ReLU) and the output layer
- For the output layer, use logistic activation function for binary classification, softmax activation function for multiclass classification, or no activation function for regression

Neural Nets Training Pattern

1. Read a batch of input data and the current parameter values
2. Compute the loss function via a forward pass through the network
3. Compute gradients for each of the parameters (a backward pass)
4. Update gradients for next forward pass (repeat)
5. Test on new data

Demo!

Cloud AutoML Vision ALPHA

image analysis simplified

Google Cloud

Computer Vision in Google Cloud

Cloud Vision API

GA: May 2017

- Detect labels in images like:
  - Animals
  - Modes of transportation
  - Adult/violent content
  - Logos
  - Landmarks
  - Optical character recognition
  - Face detection
  - Dominant colors
- Can be used to build metadata on your image

Cloud ML Engine

GA: March 2017

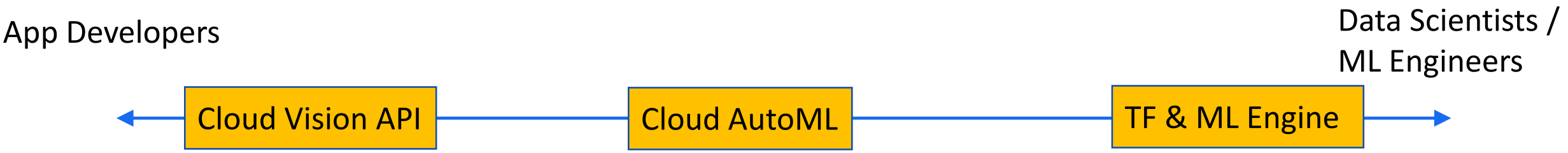
- Fully managed platform for TensorFlow and Keras
- Distributed training with GPUs/TPUs
- Hyperparameter tuning with HyperTune
- Supports online predictions for TF, Keras, scikit-learn and XGBoost

Cloud AutoML Vision

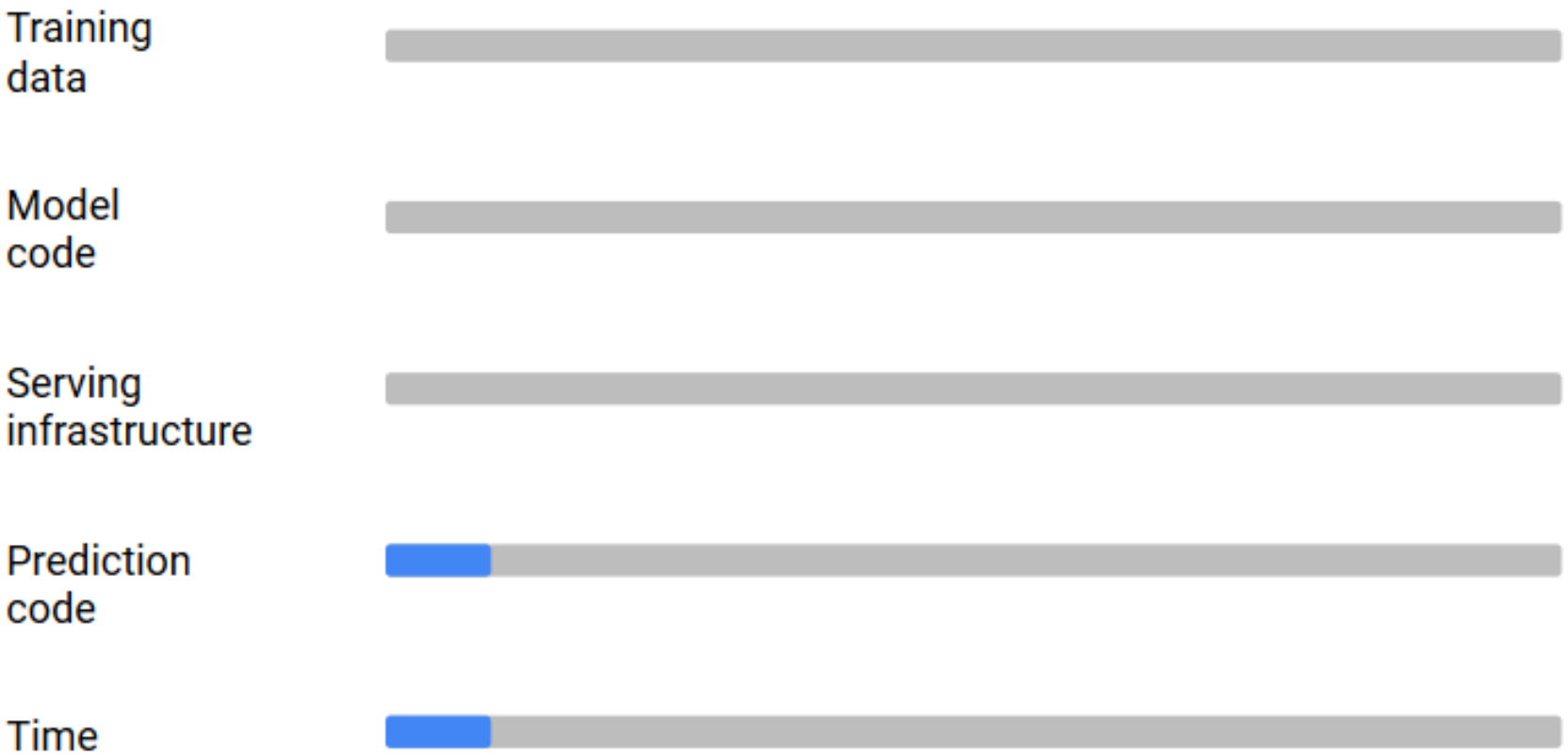
Alpha: Jan 2018

- Bring your training data to create your own custom vision model with minimum ML skills required
- Can start with 10 training samples
- Simple GUI interface, takes minutes to create a custom model
- Leverages Transfer Learning and Reinforcement Learning (for automated model construction)

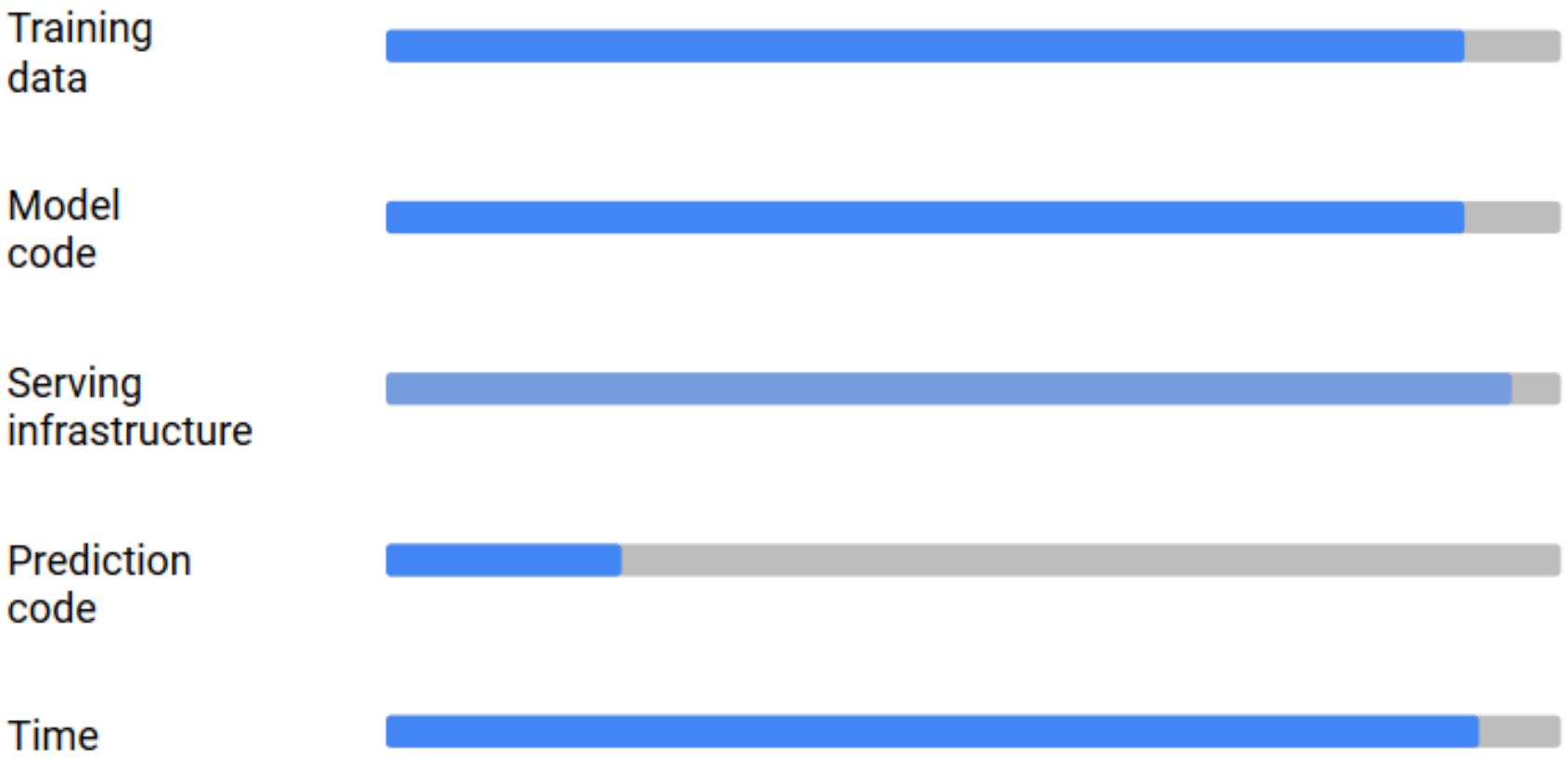
Computer Vision in Google Cloud



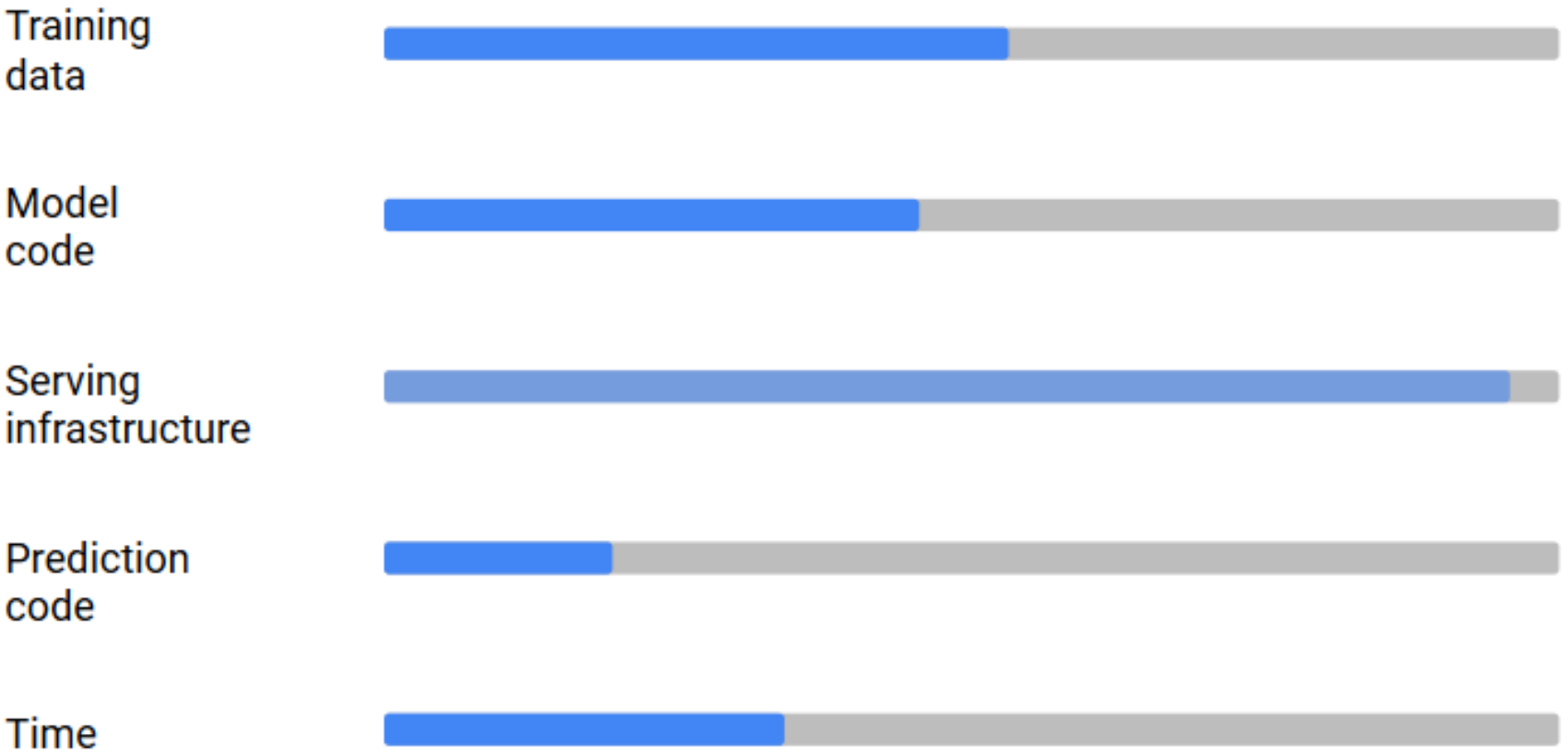
Machine Learning as an API



TF and ML Engine (build custom models from scratch)



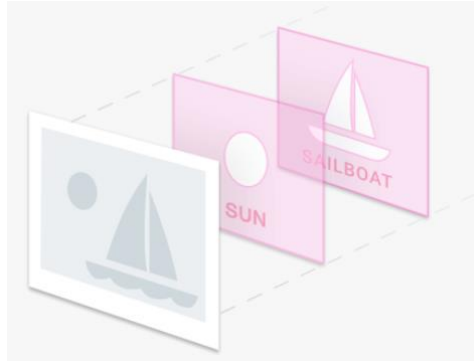
AutoML (transfer learning)



Google Cloud

CLOUD VISION API

Derive insight from images with our powerful Cloud Vision API



Try the API

Upload your image

Powerful Image Analysis

Google Cloud Vision API enables developers to **understand the content of an image** by encapsulating **powerful machine learning models** in an easy to use REST API. It quickly **classifies images** into thousands of categories (e.g., "sailboat", "lion", "Eiffel Tower"), **detects individual objects and faces within images**, and finds and reads printed words contained within images.

Google AI Blog

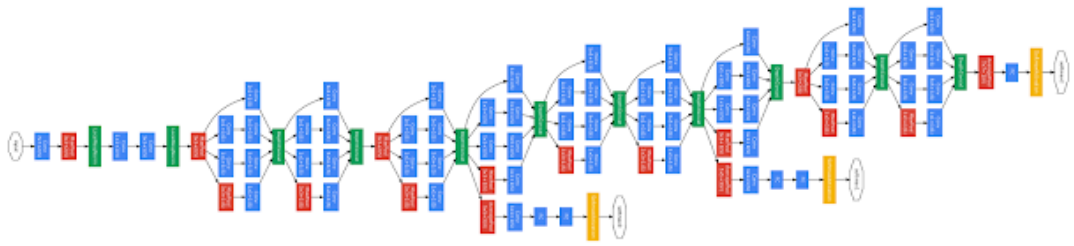
The latest news from Google AI

Using Machine Learning to Explore Neural Network Architecture

Wednesday, May 17, 2017

Posted by Quoc Le & Barret Zoph, Research Scientists, Google Brain team

At Google, we have successfully applied deep learning models to many applications, from image recognition to speech recognition to machine translation. Typically, our machine learning models are painstakingly designed by a team of engineers and scientists. This process of manually designing machine learning models is difficult because the search space of all possible models can be combinatorially large – a typical 10-layer network can have ~10^{10} candidate networks! For this reason, the process of designing networks often takes a significant amount of time and experimentation by those with significant machine learning expertise.



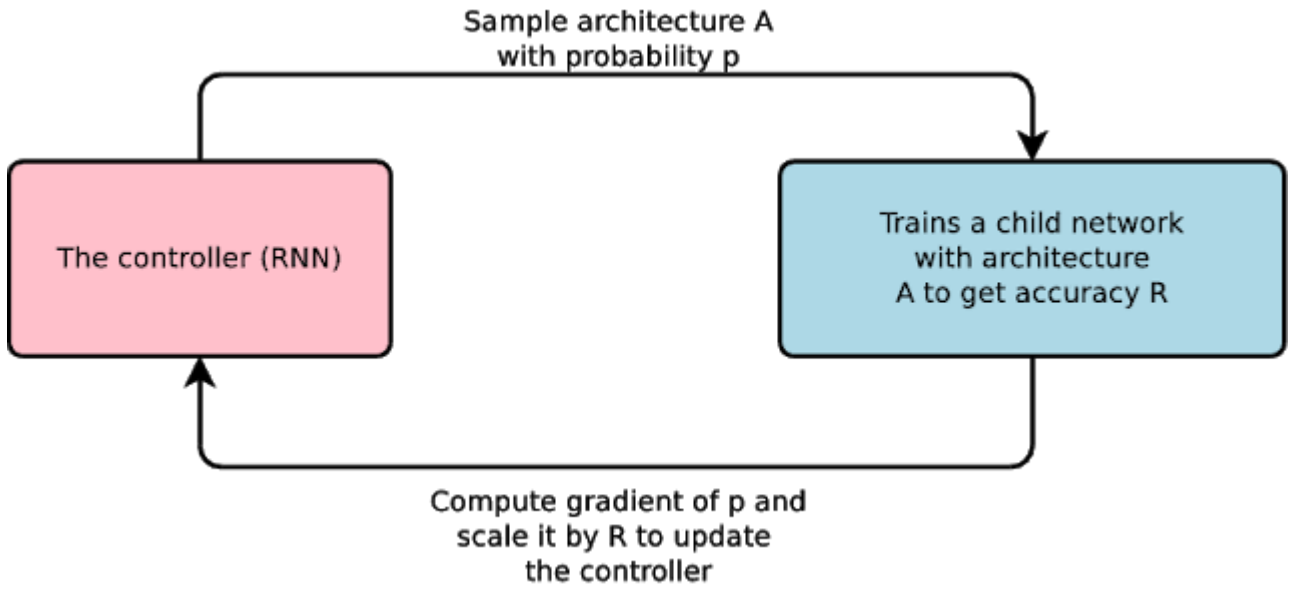
Our GoogleNet architecture. Design of this network required many years of careful experimentation and refinement from initial versions of convolutional architectures.

"Typically, our machine learning models are painstakingly designed by a team of engineers and scientists.

This process of manually designing machine learning models is difficult because the search space of all possible models can be combinatorially large — a typical 10-layer network can have ~1010 candidate networks!"

GoogleNet computer vision architecture took many years of careful experimentation and refinement

AutoML internals

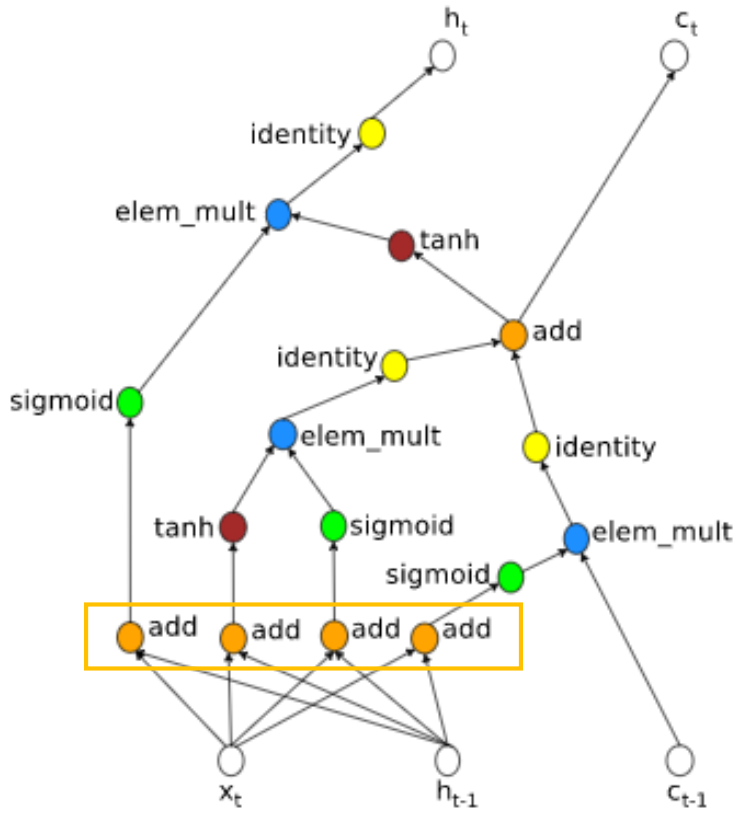


We have applied this approach on the CIFAR-10 image recognition dataset and the Penn Treebank language modeling datasets.

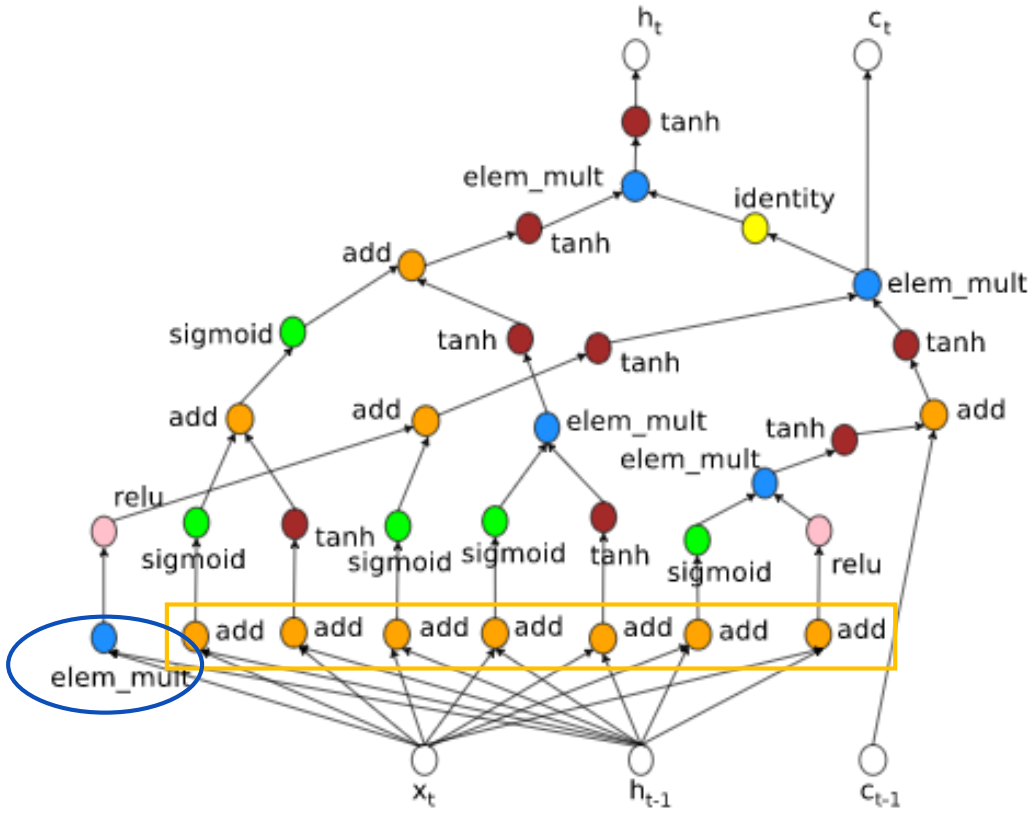
On both, our approach can design models with accuracies on par with state of the art models.

AutoML: Penn Treebank

Designed by human experts



Created by AutoML



AutoML Vision: Demo

Problem: Predict weather trends and flight plans from images of clouds

Dataset:

- 1820 labeled images of clouds:
  - Cumulus: 537
  - Cirrus: 496
  - Cumulonimbus: 453
  - Altocumulus: 199
  - Altostratus: 135

Training:

- 2 Model types:
  - Base model (Free to train; 1 hour)
  - Adv mode ($550, up to 24 hours)

Evaluate/Predict:

- 192 images in test set
- We will look at:
  - Area Under Curve (Precision/Recall curves)
  - Confidence Threshold Curves
  - Confusion Matrix
- Make predictions from the trained models

Google AI Blog

The latest news from Google AI

AutoML for large scale image classification and object detection

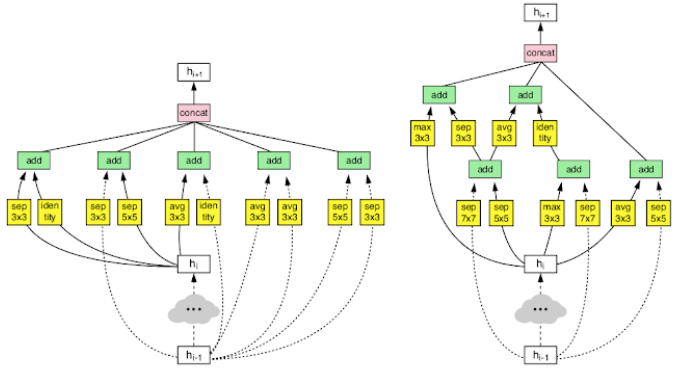
Thursday, November 2, 2017

Posted by Barret Zoph, Vijay Vasudevan, Jonathon Shlens and Quoc Le, Research Scientists, Google Brain Team

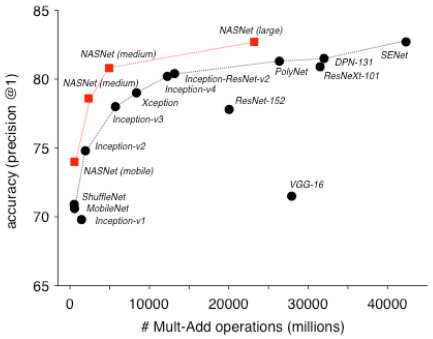
A few months ago, we introduced our AutoML project, an approach that automates the design of machine learning models. While we found that AutoML can design small neural networks that perform on par with neural networks designed by human experts, these results were constrained to small academic datasets like CIFAR-10, and Penn Treebank. We became curious how this method would perform on larger more challenging datasets, such as ImageNet image classification and COCO object detection. Many state-of-the-art machine learning architectures have been invented by humans to tackle these datasets in academic competitions.

CIFAR-10 and Penn Treebank are small academic datasets.

Google's research scientists became curious about how AutoML would perform on larger, more challenging datasets like ImageNet image classification and COCO object detection.



Our NASNet architecture is composed of two types of layers: Normal Layer (left), and Reduction Layer (right). These two layers are designed by AutoML.



Accuracies of NASNet and state-of-the-art, human-invented models at various model sizes on ImageNet image classification.

What is ImageNet?

- Since 2010, the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been held where research teams evaluate their algorithms on a given dataset to get the highest accuracy in computer vision
- Dataset: ~3-10 million images with ~1k – 20k potential labels (descriptions) that are crowdsourced via Amazon Mechanical Turk (there are also 250 categories of dog breeds to showcase fine-grained classification)
- In 2011, a good classification error rate was 25%
- By 2012, a CNN achieved 16%
- By 2014, the error rates fell to a few percent
- In 2015, researchers reported that software exceeded human ability at the narrow computer vision tasks!
- But, in the contest the algorithms only have to identify images as belonging to one of a thousand categories; humans can recognize a much larger number of categories

ImageNet: A Large-Scale Hierarchical Image Database

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei
Dept. of Computer Science, Princeton University, USA
{jiadeng, wdong, rsocher, jial, li, feifeili}@cs.princeton.edu

Abstract

The explosion of image data on the Internet has the potential to foster more sophisticated and robust models and algorithms to index, retrieve, organize and interact with images and multimedia data. But exactly how such data can be harnessed and organized remains a critical problem. We introduce here a new database called "ImageNet", a large-scale ontology of images built upon the backbone of the WordNet structure. ImageNet aims to populate the majority of the 80,000 synsets of WordNet with an average of 500-1000 clean and full resolution images. This will result in tens of millions of annotated images organized by the semantic hierarchy of WordNet. This paper offers a detailed analysis of ImageNet in its current state: 12 subtrees with 5247 synsets and 3.2 million images in total. We show that ImageNet is much larger in scale and diversity and much more accurate than the current image datasets. Constructing such a large-scale database is a challenging task. We describe the data collection scheme with Amazon Mechanical Turk. Lastly, we illustrate the usefulness of ImageNet through three simple applications in object recognition, image classification and automatic object clustering. We hope that the scale, accuracy, diversity and hierarchical structure of ImageNet can offer unparalleled opportunities to researchers in the computer vision community and beyond.

1. Introduction

The digital era has brought with it an enormous explosion of data. The latest estimations put a number of more than 3 billion photos on Flickr, a similar number of video clips on YouTube and an even larger number for images in

content-based image search and image understanding algorithms, as well as for providing critical training and benchmarking data for such algorithms.

ImageNet uses the hierarchical structure of WordNet [9]. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are around 80,000 noun synsets in WordNet. In ImageNet, we aim to provide on average 500-1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated as described in Sec. 3.2. ImageNet, therefore, will offer tens of millions of cleanly sorted images. In this paper, we report the current version of ImageNet, consisting of 12 "subtrees": *mammal, bird, fish, reptile, amphibian, vehicle, furniture, musical instrument, geological formation, tool, flower, fruit.* These subtrees contain 5247 synsets and 3.2 million images. Fig. 1 shows a snapshot of two branches of the mammal and vehicle subtrees. The database is publicly available at `http://www.image-net.org`.

The rest of the paper is organized as follows: *We first show that ImageNet is a large-scale, accurate and diverse image database* (Section 2). In Section 4, we present a few simple application examples by exploiting the current ImageNet, mostly the mammal and vehicle subtrees. Our goal is to show that *ImageNet can serve as a useful resource for visual recognition applications such as object recognition, image classification and object localization.* In addition, the construction of such a large-scale and high-quality database can no longer rely on traditional data collection methods. Sec. 3 describes how ImageNet is constructed by leveraging Amazon Mechanical Turk.

2. Properties of ImageNet

ImageNet is built upon the hierarchical structure pro-

What is ImageNet?

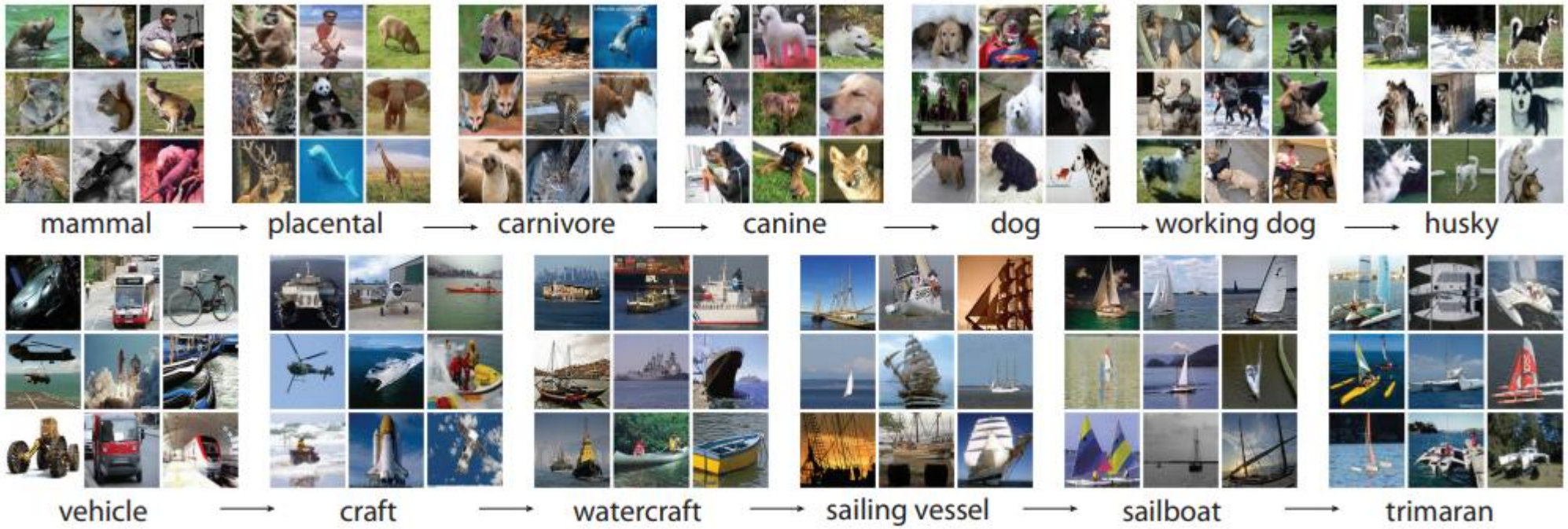


Figure 1: A snapshot of two root-to-leaf branches of ImageNet: the **top** row is from the mammal subtree; the **bottom** row is from the vehicle subtree. For each synset, 9 randomly sampled images are presented.

Source: http://www.image-net.org/papers/imagenet_cvpr09.pdf

AutoML: (for large scale image classification and object detection)

In Nov 2017, to apply AutoML to ImageNet & COCO:

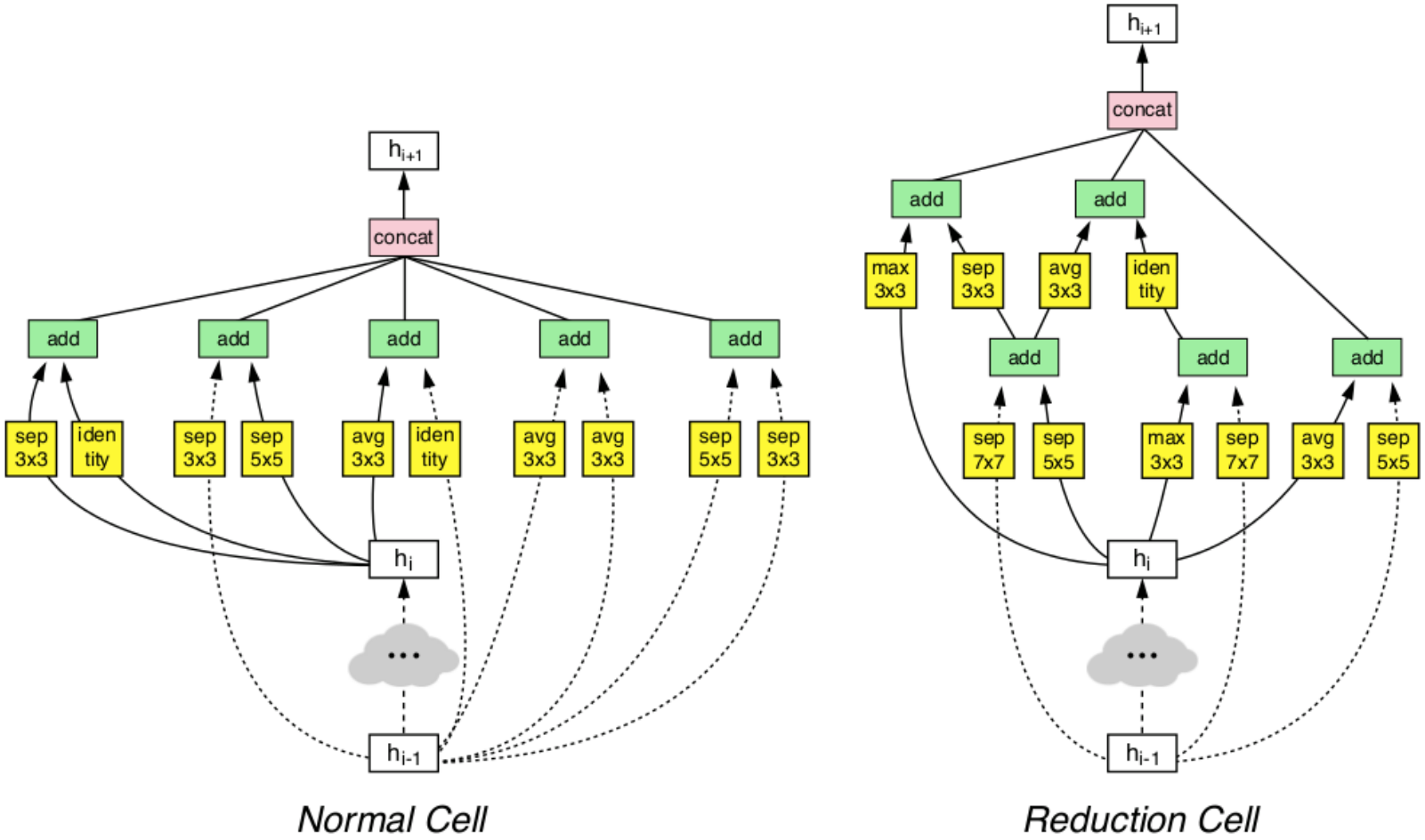
- We redesigned the search space so that AutoML could find the best layer which can then be stacked many times in a flexible manner to create a final network.

- We performed architecture search on CIFAR-10 and transferred the best learned architecture to ImageNet image classification and COCO object detection.

With this method, AutoML was able to find the best layers that work well on CIFAR-10 but also work well on ImageNet classification and COCO object detection.

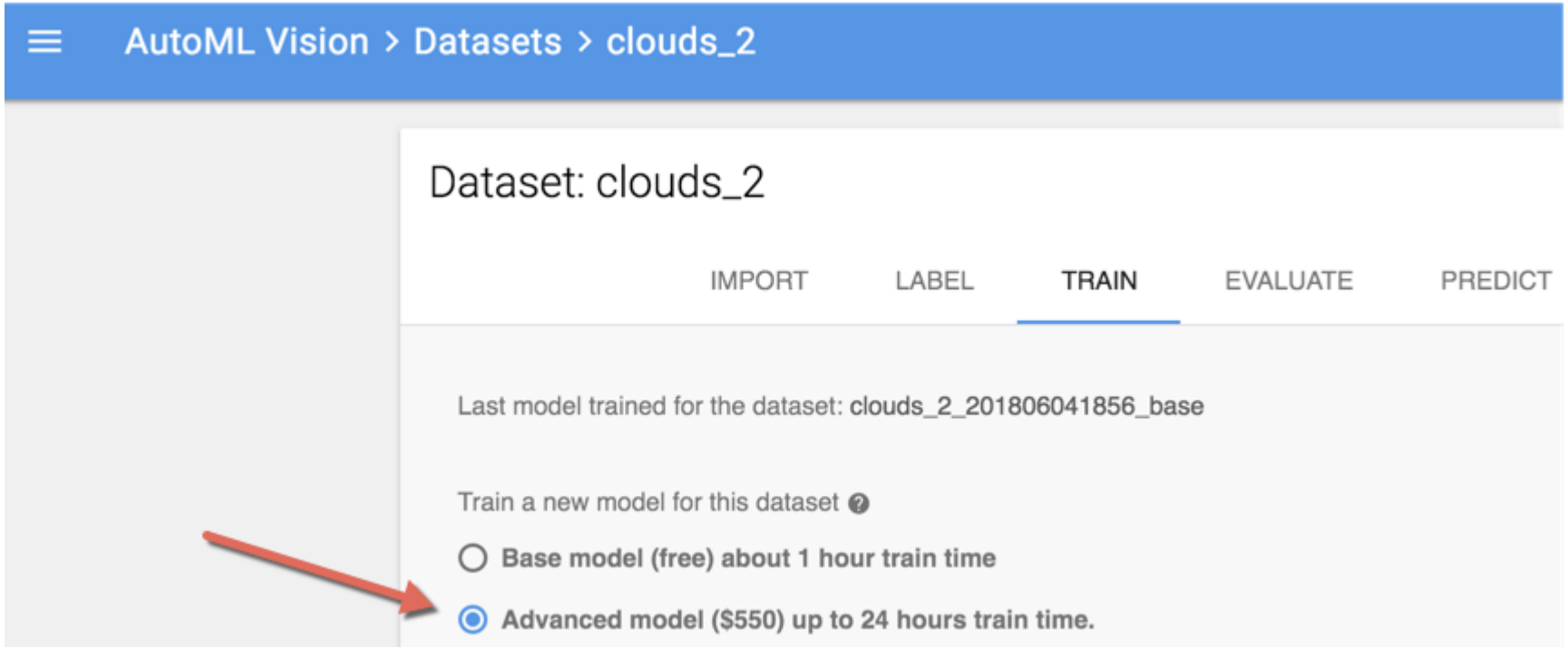
These two layers are combined to form a novel architecture, which we call "NASNet".

NASNet Architecture: normal and reduction layers



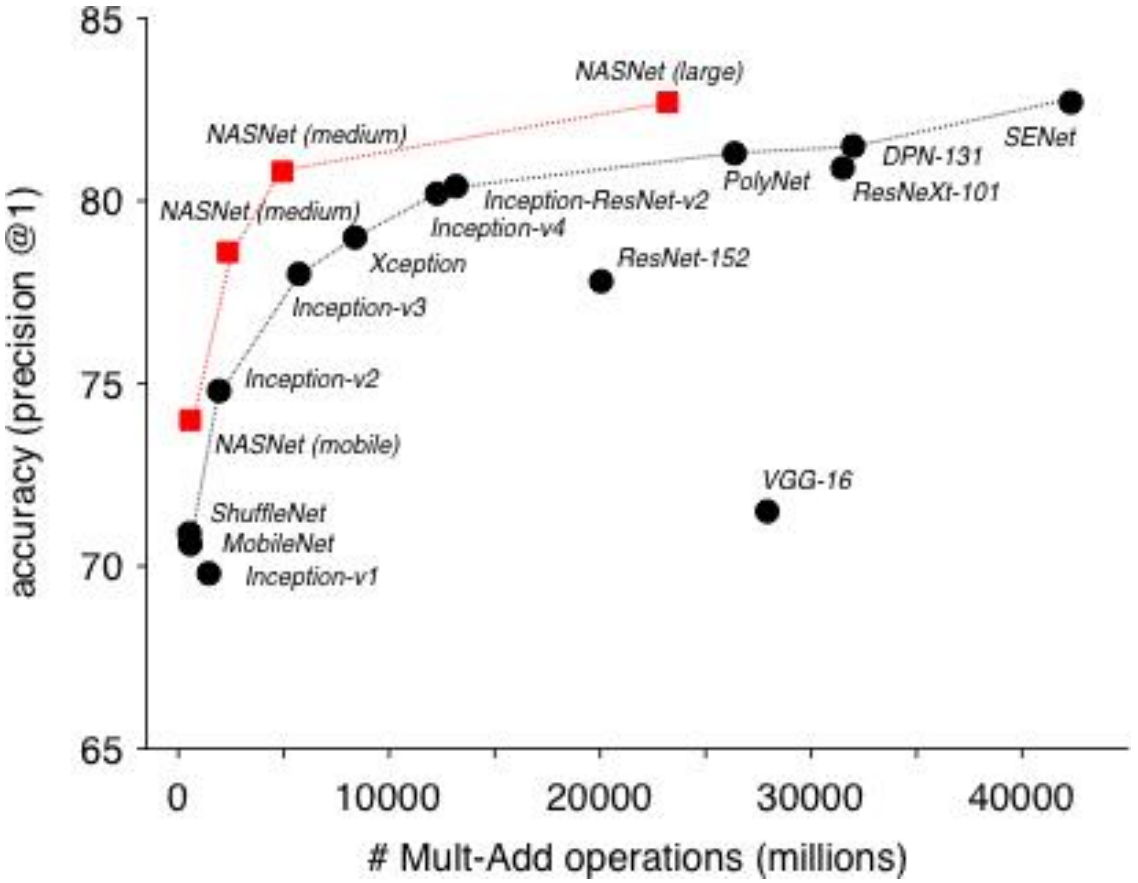
Our NASNet architecture is composed of two types of layers: Normal Layer (left), and Reduction Layer (right).
These two layers are designed by AutoML.

AutoML: Advanced Mode



- Advanced mode uses Google's learning2learn technology, specifically NASNet (along with transfer learning)
- NASNet uses ML for optimizing ML, the meta-level ML model tries to find the best deep learning model design for a specific training dataset.

NASNet Performance (Nov 2017)



Accuracies of NASNet and state-of-the-art, human-invented models at various model sizes on ImageNet image classification.

"On ImageNet image classification, NASNet achieves a prediction accuracy of 82.7% on the validation set, surpassing all previous Inception models that we built at Google.

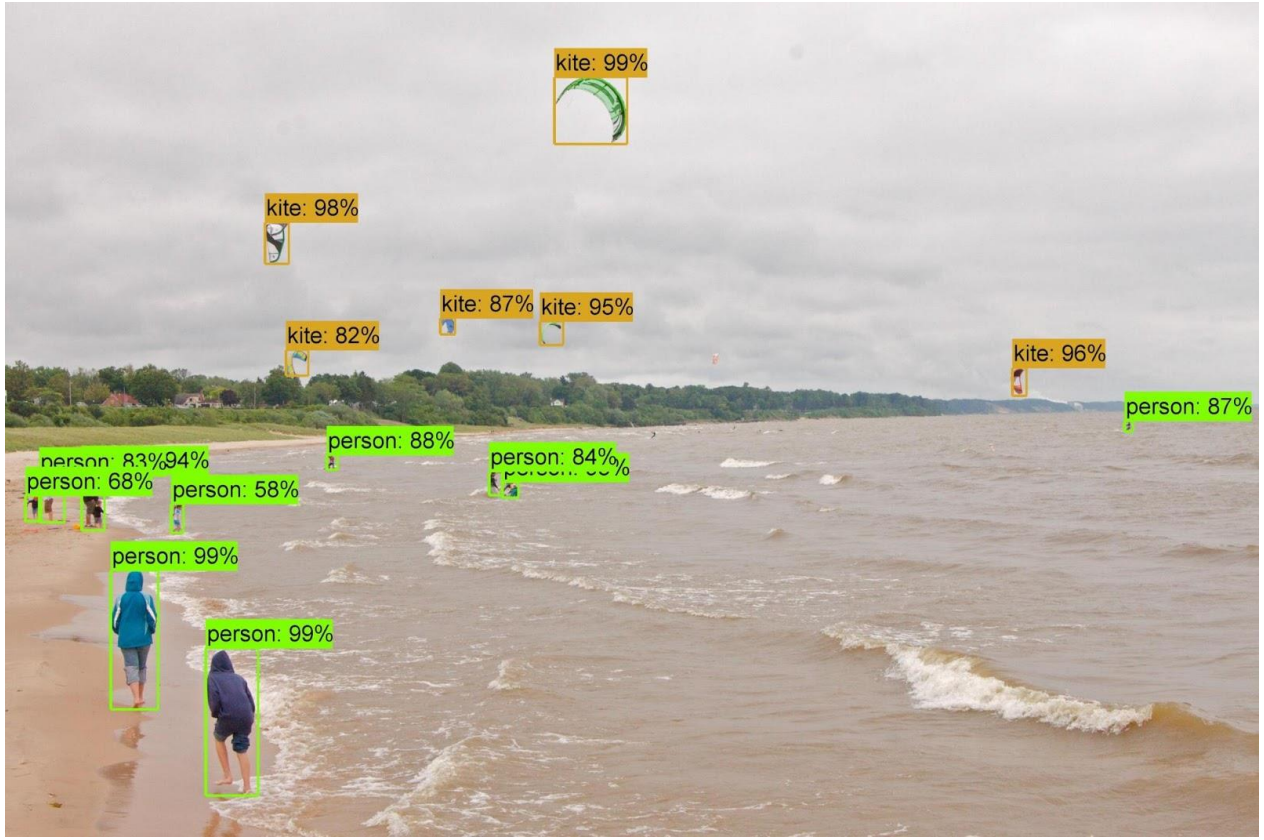
Additionally, NASNet performs 1.2% **better than all previous published results**."

"Furthermore, NASNet may be resized to produce a family of models that achieve good accuracies while having very low computational costs.

For example, a **small** version of NASNet achieves **74% accuracy**, which is 3.1% better than equivalently-sized, state-of-the-art models for mobile platforms.

The **large** NASNet achieves **state-of-the-art accuracy** while **halving** the computational cost of the best reported result on arxiv.org."

NASNet (Object Detection)



Example object detection using Faster-RCNN with NASNet.

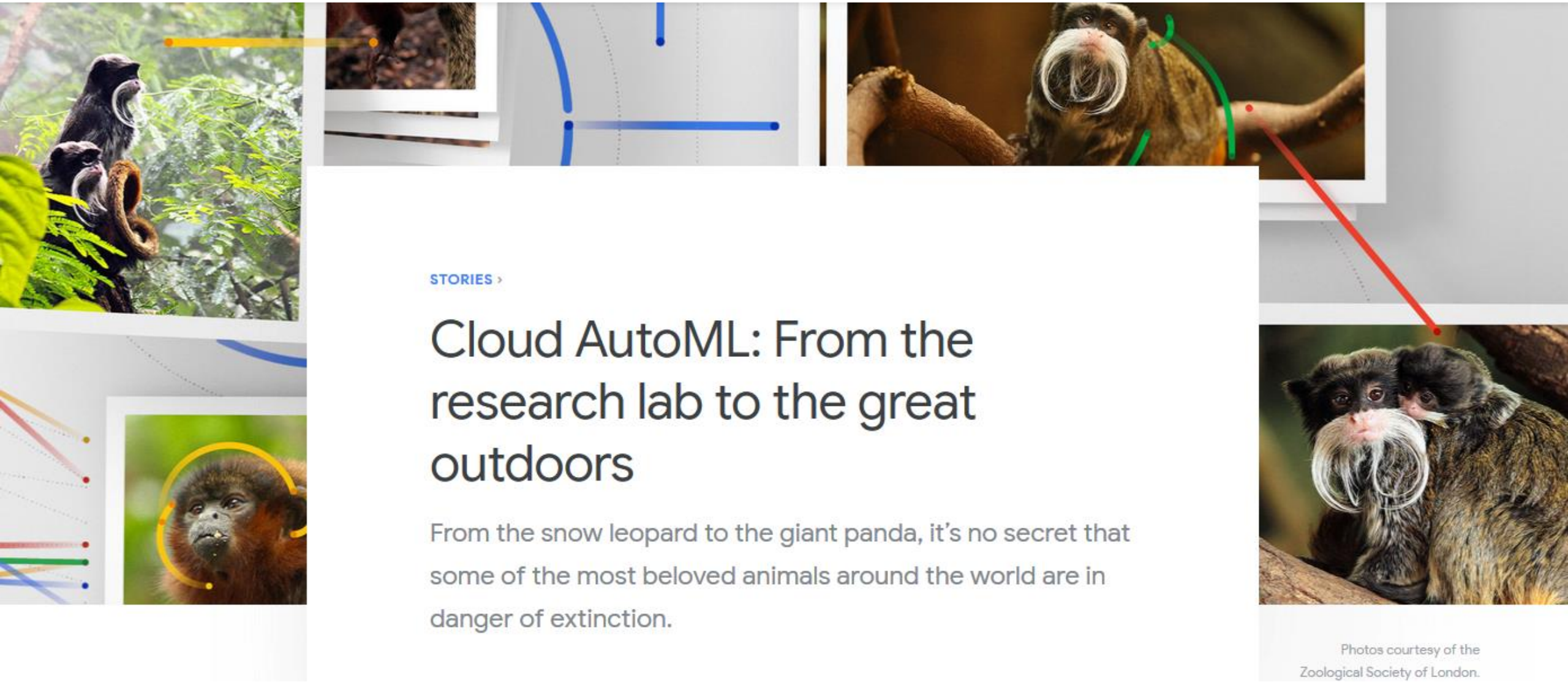
"We also transferred the learned features from ImageNet to object detection.

In our experiments, combining the features learned from ImageNet classification with the Faster-RCNN framework **surpassed previous published, state-of-the-art predictive performance** on the COCO object detection task in both the largest as well as mobile-optimized models.

Our largest model achieves 43.1% mAP which is **4% better** than the previous, published state-of-the-art."

"We suspect that the image features learned by NASNet on ImageNet and COCO may be reused for many computer vision applications.

Thus, we have open-sourced NASNet for inference on image classification and for object detection in the Slim and Object Detection TensorFlow repositories."



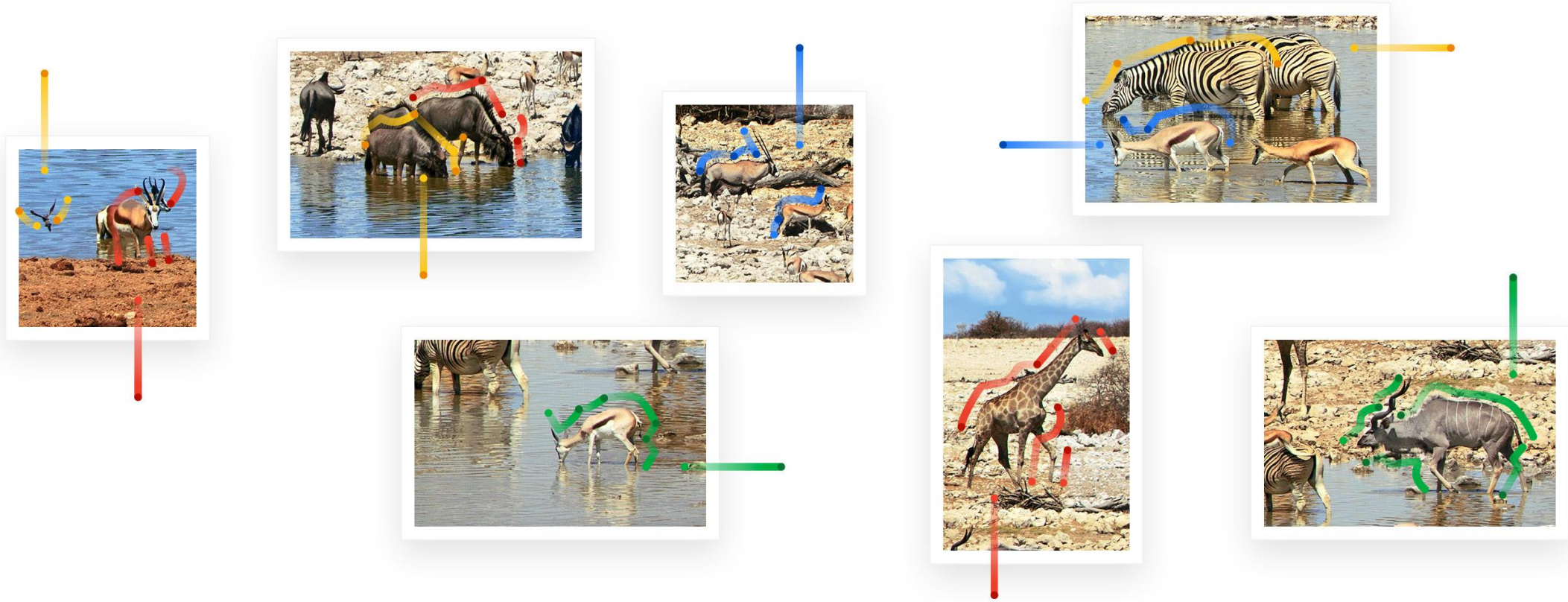
STORIES ›

Cloud AutoML: From the research lab to the great outdoors

From the snow leopard to the giant panda, it's no secret that some of the most beloved animals around the world are in danger of extinction.

Photos courtesy of the Zoological Society of London.

Turning Moonshot Research into a Product



Zoological Society of London uses camera traps w/ motion & heat sensors to take a picture every time a human or animal passes.

Using pre-trained models were only good for general classification (elephant vs tiger).

ZSL needed specificity down to the sub-species level.

ZSL worked with Google Brain and the Cloud AI team to use AutoML vision to start to identify subspecies, such as a Red titi monkey versus an Emperor tamarin monkey.

Convolutional Neural Networks

computer vision

Pioneers in AI: Yann LeCun

- Born: 1960 in France
- Fun fact: In 2017 he declined an invitation to lecture at King Abdullah University of Science and Technology in Saudi Arabia b/c he was considered a terrorist in the country in view of his atheism
- Considered the founding father of Convolutional Neural Networks (CNN)
- Famous for his work on optical character recognition and computer vision
- Currently Director of AI Research @ Facebook and Director of NYU Center for Data Science

Personal Homepage

Research @ Facebook

Facebook Profile

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

1988 paper on using CNNs for optical character recognition

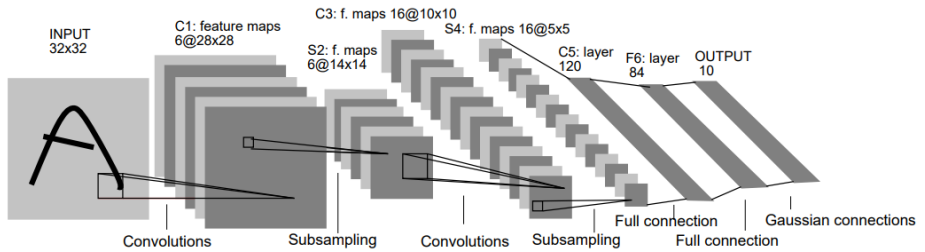
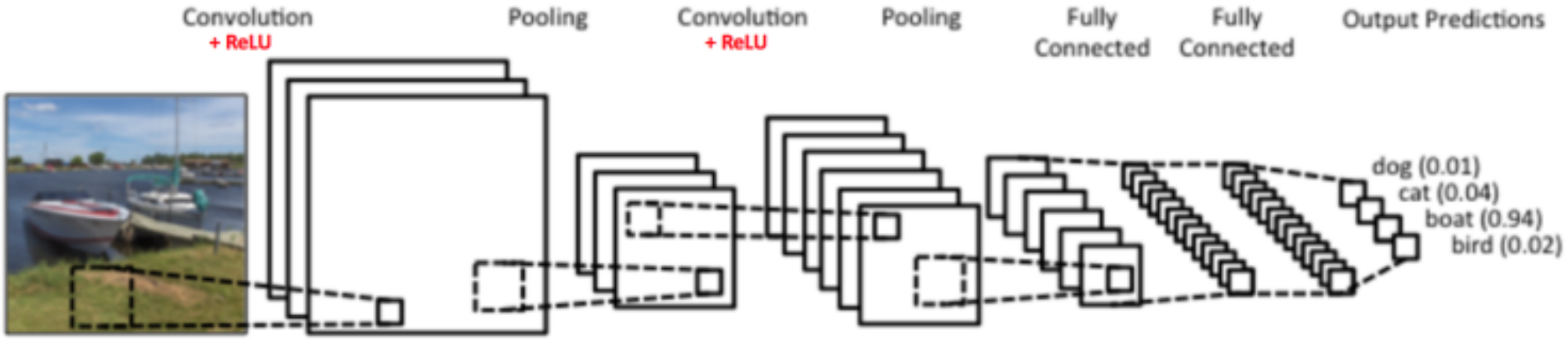


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CNN



- This CNN classifies the input image into 4 categories: dog, cat, boat, bird
- Here, the highest probability for this image is boat (95%)
- The sum of all probabilities in the output layer should be one

The 4 main operations in a ConvNet:

1. Convolution
2. Non Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

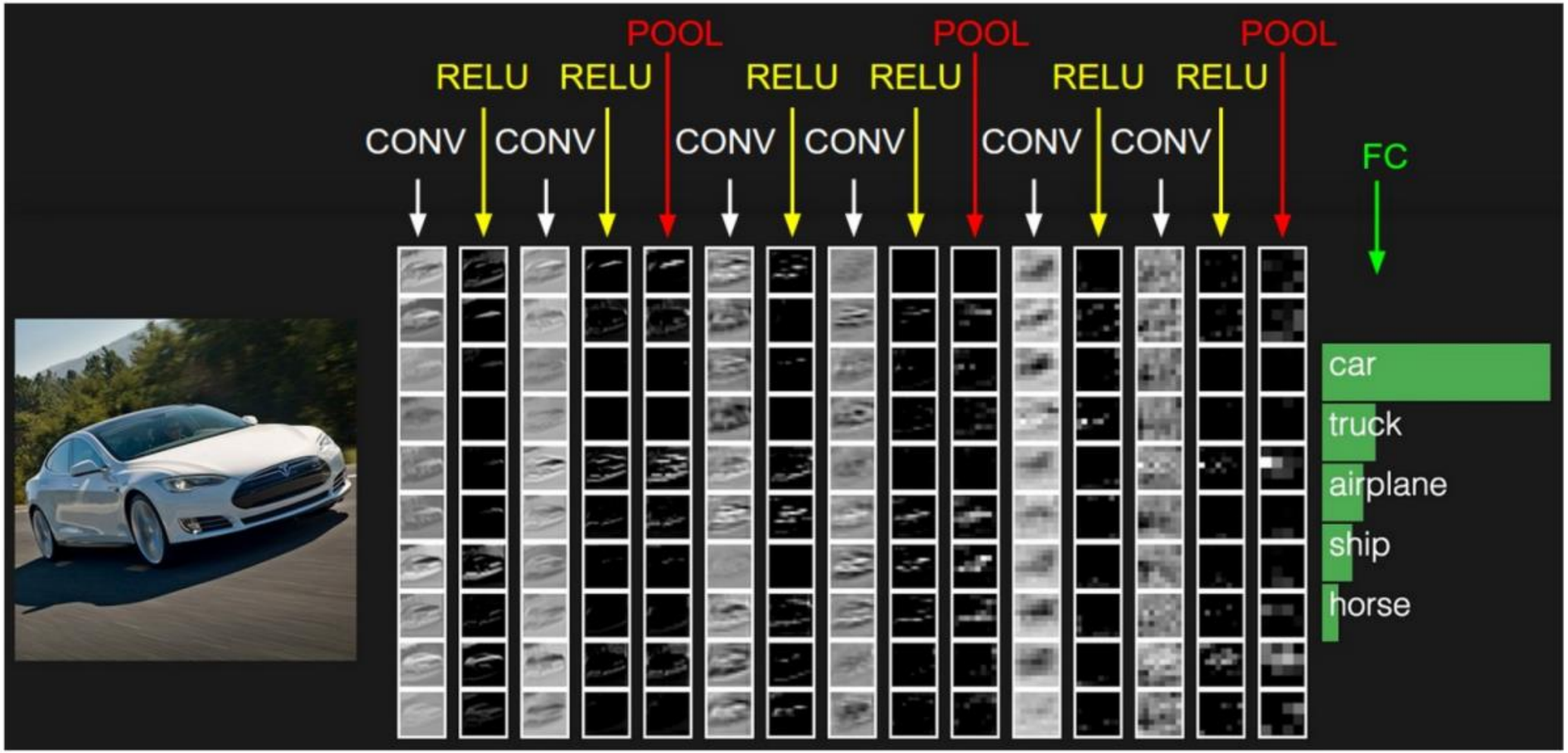
LeNet5

Table 13-1. LeNet-5 architecture

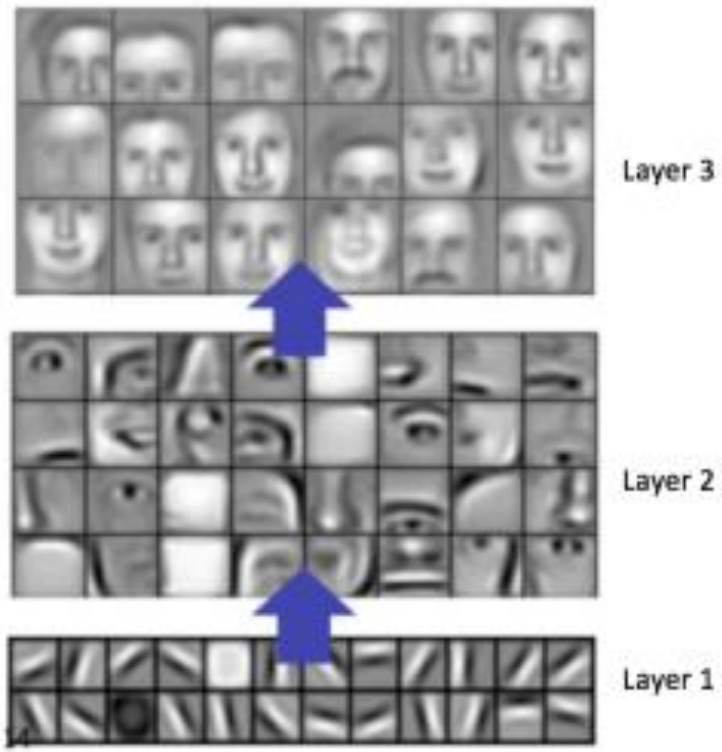
Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

- LeNet5 (1998) is 7 layers, not counting the input layer
- Input image is 32x32 pixels

Source:



- Some of the best performing ConvNets today have tens of Convolution and Pooling layers
- It is not necessary to have a Pooling layer after every Convolutional Layer
- We can have multiple Convolution + ReLU operations in succession before having a Pooling operation



- In facial recognition, a ConvNet may learn to detect edges from raw pixels in the first layer
- Then use the edges to detect simple shapes in the second layer
- Then use these shapes to deter higher-level features, such as facial shapes in higher layers
- These features (left) were learnt using a Convolutional Deep Belief Network (by Andrew Ng)

The Convolution Step

- Primary purpose of convolutions is to extract features from the input image
- Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data

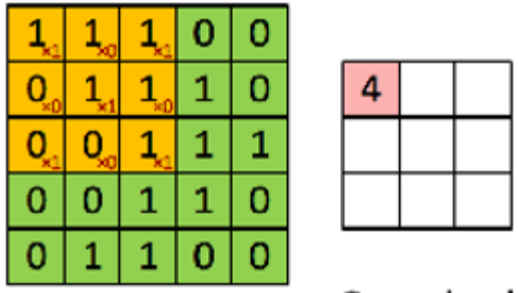
Consider a 5 x 5 image whose pixel values are only 0 and 1:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Also, consider another 3 x 3 matrix:

1	0	1
0	1	0
1	0	1

The Convolution of the 5 x 5 image and the 3 x 3 matrix can be computed as shown:



The Convolution Step

Consider a 5 x 5 image whose pixel values are only 0 and 1:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

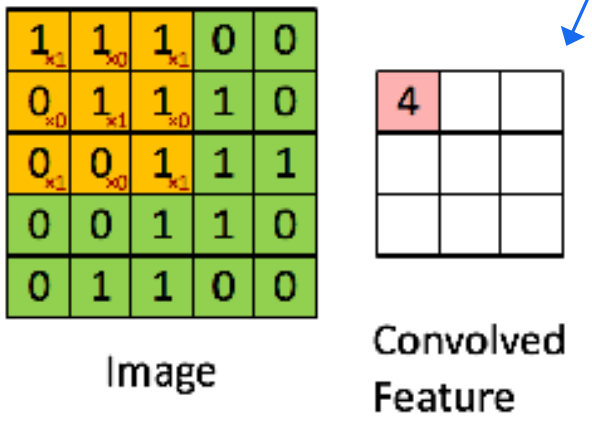
Also, consider another 3 x 3 matrix:

1	0	1
0	1	0
1	0	1

Called a ‘filter’ or ‘kernel’

(filter acts as feature detectors from the original input image)

The Convolution of the 5 x 5 image and the 3 x 3 matrix can be computed as shown:



Called a ‘feature map’

- Slide the orange matrix over the original image by 1 pixel (called a ‘stride’) and
- For every position compute element wise multiplication between the two matrices and add the multiplication outputs to get the final integer to form the final pink output matrix.

The Convolution Step

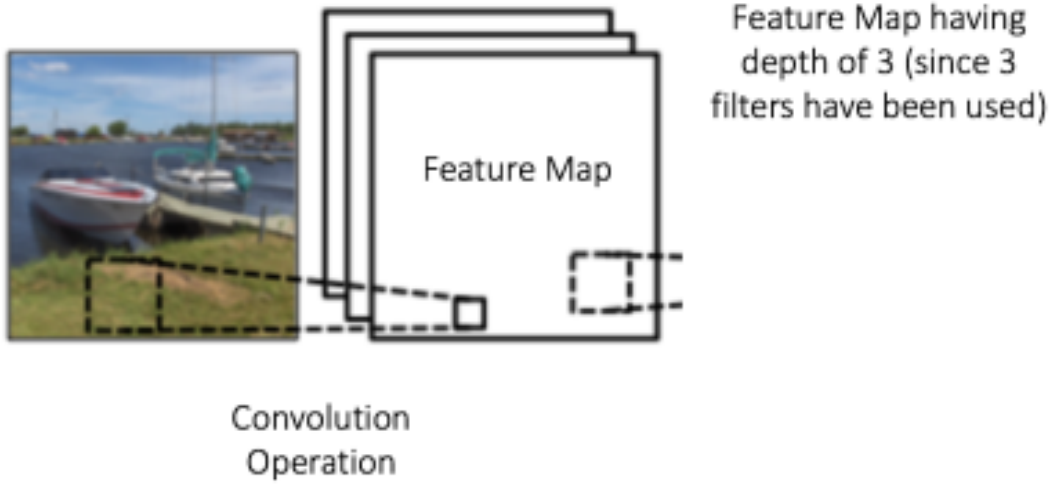
- Different values of the filter matrix will produce different Feature Maps for the same input image
- Consider this image:

- On the right, look at the effects of convolution with different filters
- We can perform operations like edge detection, sharpen and blur just by changing the numeric values of the filter matrix

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Filters and Feature Map

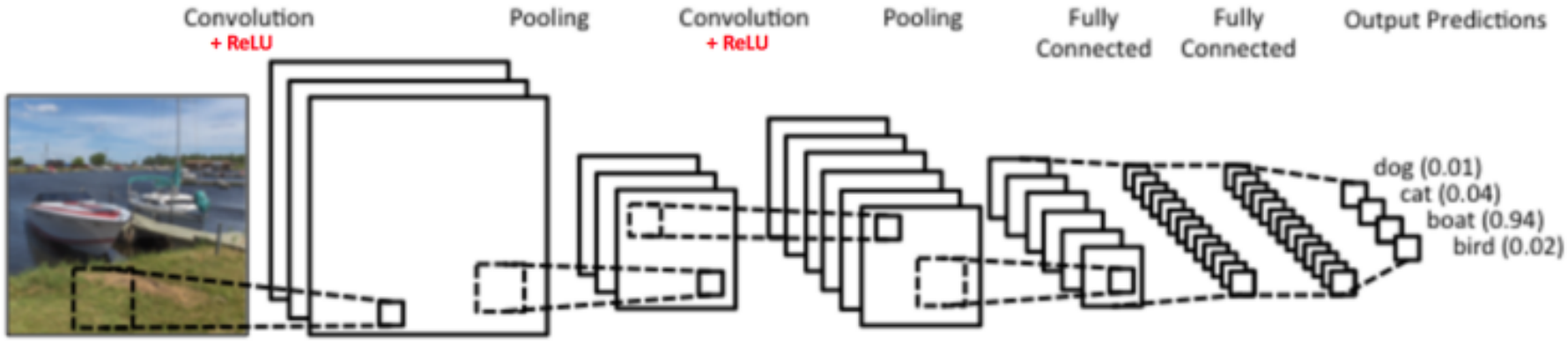
- A CNN learns the values of the filter on its own during the training step
- But we still need to specify filters, filter size, architecture of the network, etc
- The more number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images



The size of the feature map is controlled by 3 parameters:

1. Depth (# of filters)
2. Stride (# of pixels filter slides by)
3. Zero-padding (input matrix bordered by 0s)

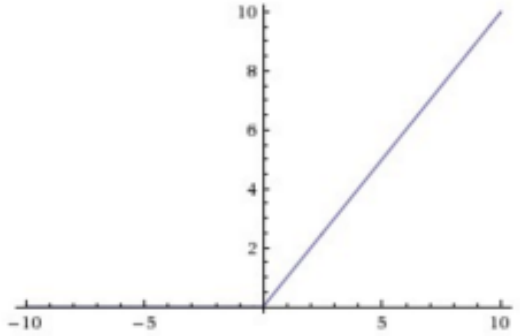
Introducing non-linearity (ReLU)



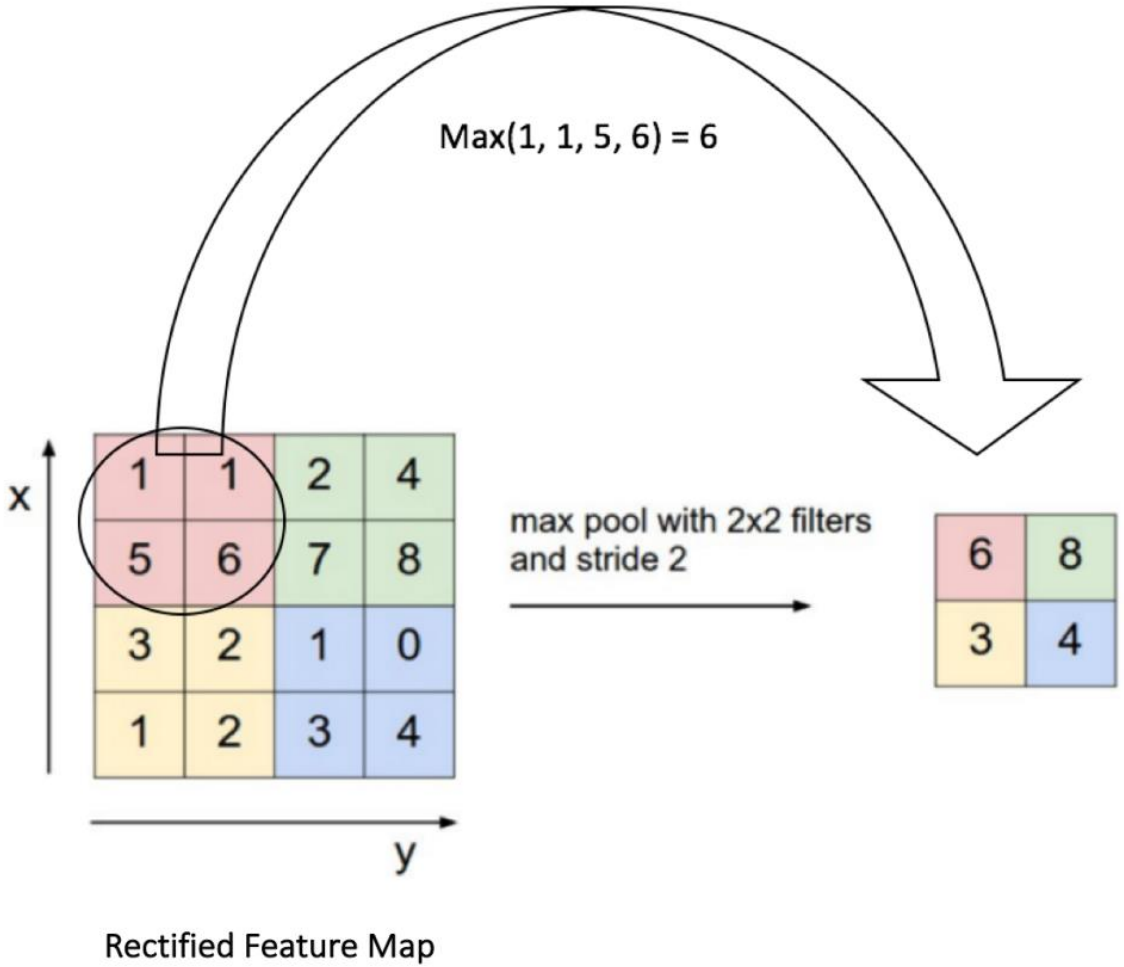
- Rectified Linear Unit (ReLU) is used after every convolution operation

- ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero

Output = Max(zero, Input)

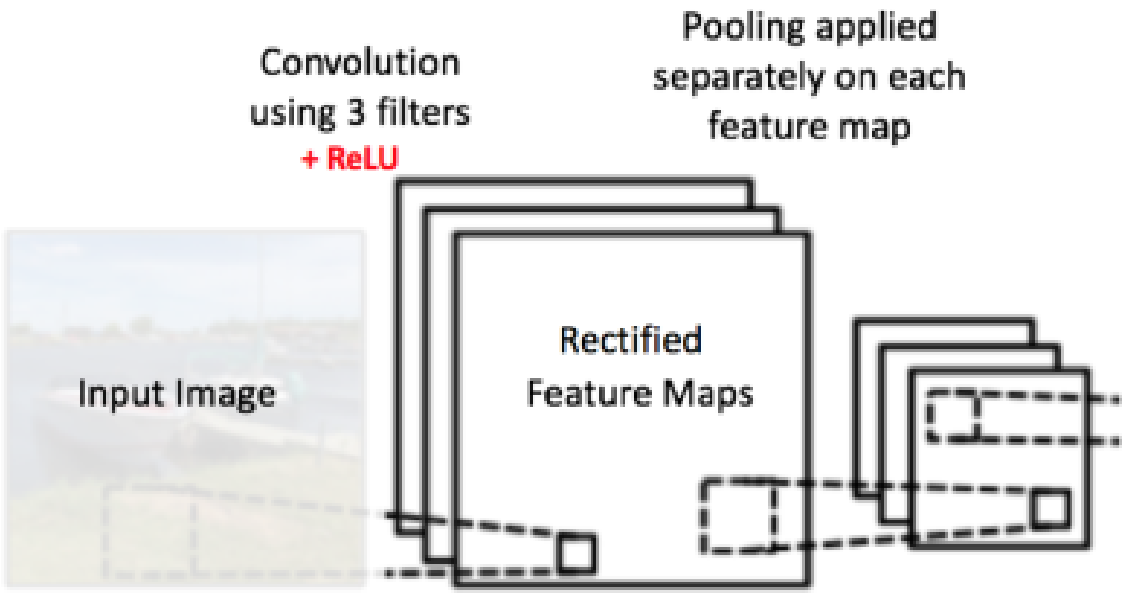


The Pooling Step



- Spatial Pooling reduces the dimensionality of each feature map but retains the most important information
- Spatial Pooling can be of different types: Max, Average, Sum, etc
- In Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window
- This diagram shows an example of Max Pooling operation on a Rectified Feature map (obtained after convolution + ReLU operation) by using a 2×2 window
- We slide our 2 x 2 window by 2 cells (also called ‘stride’) and take the maximum value in each region

The Pooling Step



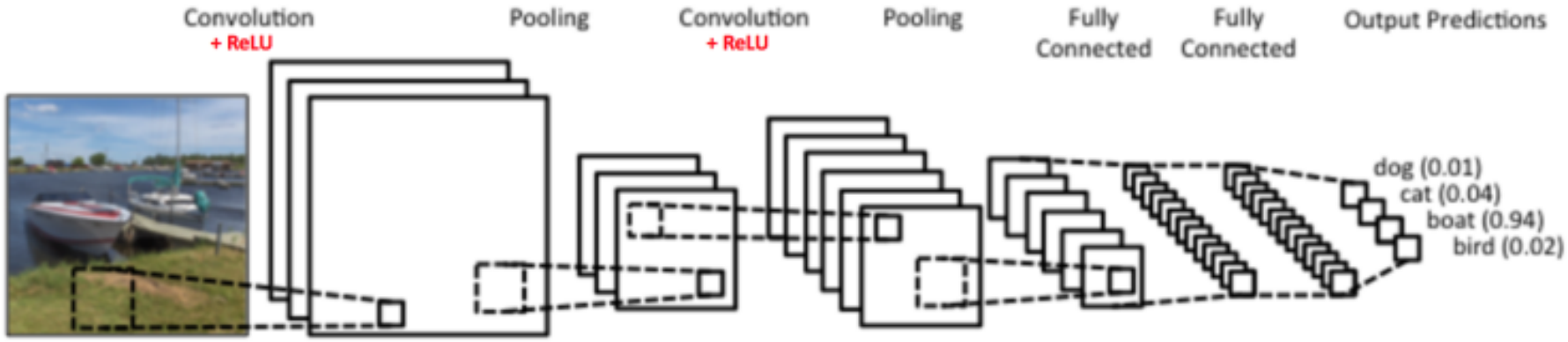
- The pooling operation is applied separately to each feature map
- Notice that, due to this, we get three output maps from three input maps

The Pooling Step

The function of Pooling is to progressively reduce the spatial size of the input representation. It does the following:

- makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network, therefore, controlling overfitting
- makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling)
- helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located

Story so far

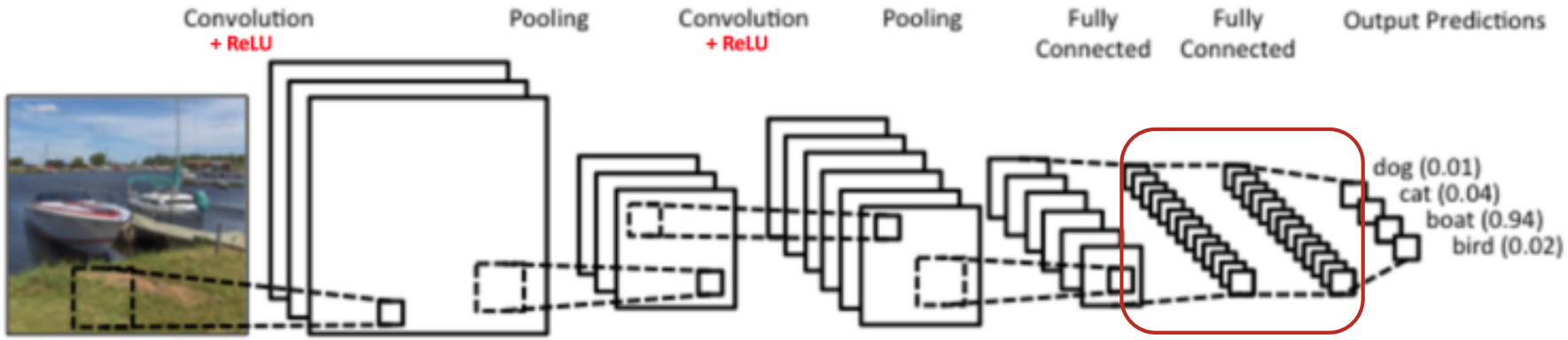


- We have two sets of Convolution, ReLU & Pooling layers – the 2nd Convolution layer performs convolution on the output of the first Pooling Layer using six filters to produce a total of six feature maps

- ReLU is then applied individually on all of these six feature maps. We then perform Max Pooling operation separately on each of the six rectified feature maps.

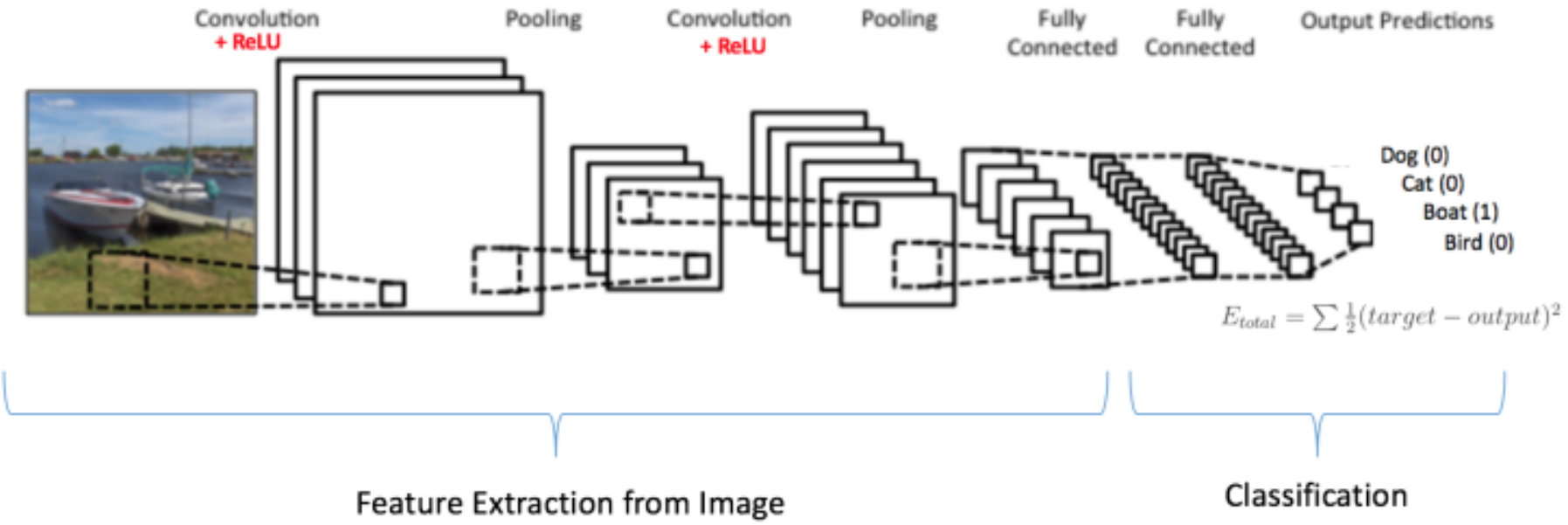
- Together these layers extract the useful features from the images, introduce non-linearity in our network and reduce feature dimension while aiming to make the features somewhat equivariant to scale and translation

Fully Connected Layers



- Each Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer
- The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer
- The output from the convolutional and pooling layers represent high-level features of the input image
- The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

Training using Backpropagation

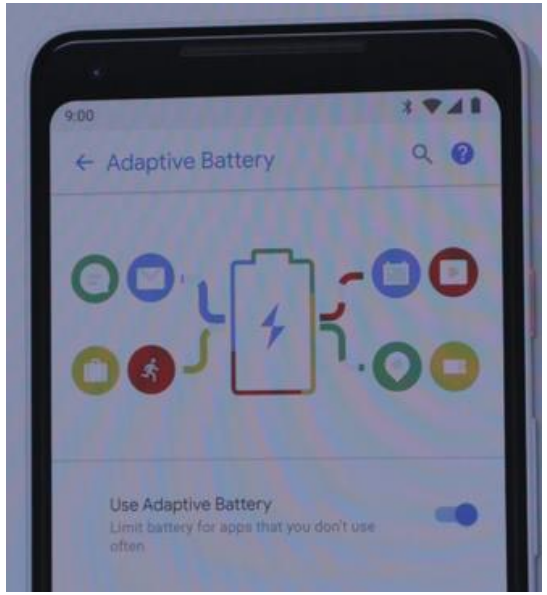


- Since the input image is a boat, the target probability is 1 for Boat class and 0 for other three classes, i.e.

- Input Image = Boat
- Target Vector = [0, 0, 1, 0]

- Use Backpropagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.

CNN Use Case: Adaptive Battery + Brightness



- **Adaptive Battery:** A smart battery management system that uses machine learning to anticipate which apps you'll need next, providing a more reliable battery experience. Feature is capable of "anticipating actions," resulting in 30-percent fewer CPU wakeups.

- **Adaptive Brightness:** A personalized experience for screen brightness, built on algorithms that learn your brightness preferences in different surroundings.

Source: DM Blog

Transfer Learning

steal pre-trained layers from an existing model…

Reusing Pretrained Layers: Transfer Learning

- It is generally not a good idea to train a very large DNN from scratch
- Instead, try to find an existing neural network that accomplishes a similar task, then just reuse the lower layers of the network

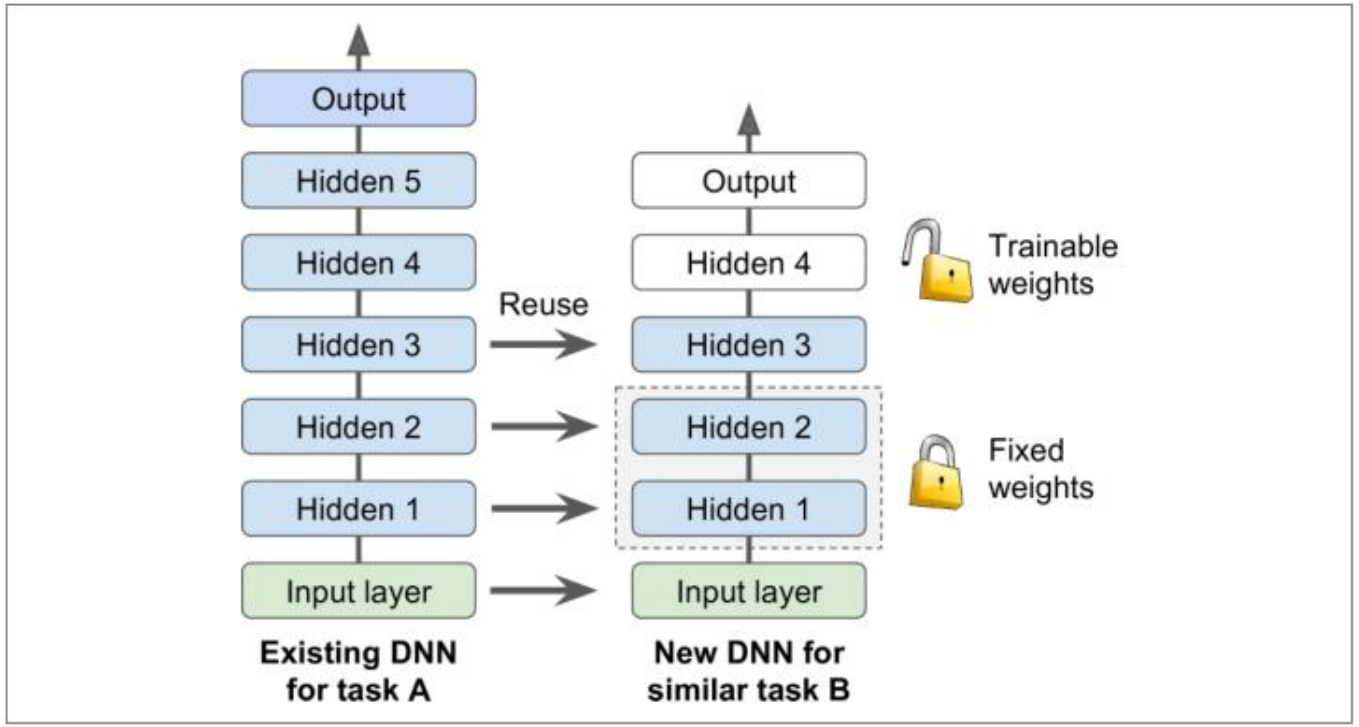


Figure 11-4. Reusing pretrained layers

Source:

Reusing Pretrained Layers: Transfer Learning

Freezing the Lower Layers

In [67]:
```
reset_graph()

n_inputs = 28 * 28  # MNIST
n_hidden1 = 300 # reused
n_hidden2 = 50  # reused
n_hidden3 = 50  # reused
n_hidden4 = 20  # new!
n_outputs = 10  # new!

X = tf.placeholder(tf.float32, shape=(None, n_inputs), name="X")
y = tf.placeholder(tf.int64, shape=(None), name="y")

with tf.name_scope("dnn"):
    hidden1 = tf.layers.dense(X, n_hidden1, activation=tf.nn.relu, name="hidden1")       # reused
    hidden2 = tf.layers.dense(hidden1, n_hidden2, activation=tf.nn.relu, name="hidden2") # reused
    hidden3 = tf.layers.dense(hidden2, n_hidden3, activation=tf.nn.relu, name="hidden3") # reused
    hidden4 = tf.layers.dense(hidden3, n_hidden4, activation=tf.nn.relu, name="hidden4") # new!
    logits = tf.layers.dense(hidden4, n_outputs, name="outputs")                         # new!

with tf.name_scope("loss"):
    xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y, logits=logits)
    loss = tf.reduce_mean(xentropy, name="loss")
```

Source: 

Not Enough Data?

4 Tricks:

- Use a model zoo
- Transfer Learning
- Unsupervised Pretraining
- One Shot Learning

Generative Adversarial Networks (GANs)

perhaps the most important idea in AI in the past decade…

GANs

Quora

What are some recent and potentially upcoming breakthroughs in deep learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU

Answered Jul 28, 2016 · Upvoted by Joaquin Quiñonero Candela, studied Machine Learning and Gokul Krishnan, M.Sc Computer Science & Machine Learning, ETH Zurich (2018)

There are many interesting recent development in deep learning, probably too many for me to describe them all here. But there are a few ideas that caught my attention enough for me to get personally involved in research projects.

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.

The idea is to simultaneously train two neural nets. The first one, called the Discriminator — let's denote it D(Y) — takes an input (e.g. an image) and outputs a scalar that indicates whether the image Y looks "natural" or not. In one instance of adversarial training, D(Y) can be seem as some sort of energy function that takes a low value (e.g. close to 0) when Y is a real sample (e.g. an image from a database) and a positive value when it is not (e.g. if it's a noisy or strange looking image). The second network is called the generator, denoted G(Z), where Z is generally a vector randomly sampled in a simple distribution (e.g. Gaussian). The role of the generator is to produce images so as to train the D(Y) function to take the right shape (low values for real images, higher values for everything else). During training D is shown a real image, and adjusts its parameter to make its output lower. Then D is shown an image produced from G and adjusts its parameters to make its output D(G(Z)) larger

Ian Goodfellow
@goodfellow_ian

Following

Two years of GAN progress on class-conditional ImageNet-128

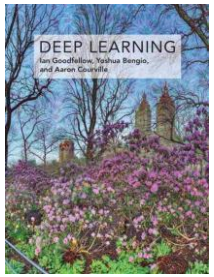
7:14 PM - 30 May 2018

375 Retweets **1,185** Likes

19 375 1.2K

Pioneers in AI: Ian GoodFellow

LinkedIn Profile

(co-author) 

- Yann LeCun, Facebook's chief AI scientist, has called GANs "the coolest idea in deep learning in the last 20 years."
- 2014, Montreal bar: How can a computer create photos by itself?
- Ian's breakthrough: "What if you pitted two neural networks against each other?"
- GANs give machines something like an imagination…
- GANs == big leap forward in unsupervised learning in AI
- Currently research scientist at Google Brain
- Previously was a researcher at OpenAI
- At Google, he developed a system enabling Google Maps to automatically transcribe addresses from photos taken by Street View cars

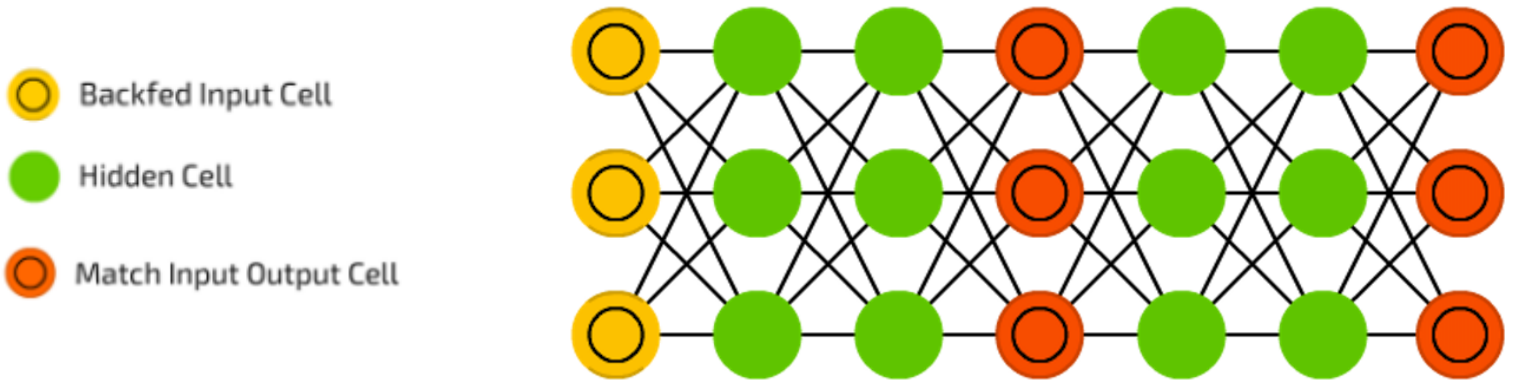
Generative Adversarial Networks (GANs)

Generator Network

- Produces synthetic output (photos or handwriting)
- Generator adjusts its parameters for creating new images till discriminator can no longer tell real from bogus

Discriminator Network

- Judges the generator's output against genuine images and tries to determine which are real/fake
- Reports results back to generator why it thinks an image may be fake



Typically a Deconvolutional Neural Network

Typically a Convolutional Neural Network

Future Applications of GANs

Ian: “There are a lot of areas of science and engineering where we need to optimize something,” such as **medicines** that need to be more effective or **batteries** that must get more efficient. “That’s going to be the next big wave.”

- Generating fake medical records almost as good as the real thing
- Generate images from text descriptions

Source: MIT Technology Review

MultiModel Neural Networks

generalizing neural networks…

Future of Deep Learning

You Retweeted

Andrej Karpathy @karpathy · Jun 18

"One Model To Learn Them All" another step in Google's attempt to turn all of itself into one big neural network arxiv.org/abs/1706.05137

23 660 1.3K

Single Neural Network model is trained on:

- WSJ speech corpus
- ImageNet
- Image Captioning via COCO dataset
- Multiple translation tasks (English, German, French)
- WSJ English parsing task

Neural Network contains:

- Convolutional layers
- An Attention mechanism
- Sparsely-gate layers

One Model To Learn Them All

Łukasz Kaiser
Google Brain
lukaszkaiser@google.com

Aidan N. Gomez*
University of Toronto
aidan@cs.toronto.edu

Noam Shazeer
Google Brain
noam@google.com

Ashish Vaswani
Google Brain
avaswani@google.com

Niki Parmar
Google Research
nikip@google.com

Llion Jones
Google Research
llion@google.com

Jakob Uszkoreit
Google Research
usz@google.com

Abstract

Deep learning yields great results across many fields, from speech recognition, image classification, to translation. But for each problem, getting a deep model to work well involves research into the architecture and a long period of tuning. We present a single model that yields good results on a number of problems spanning multiple domains. In particular, this single model is trained concurrently on ImageNet, multiple translation tasks, image captioning (COCO dataset), a speech recognition corpus, and an English parsing task. Our model architecture incorporates building blocks from multiple domains. It contains convolutional layers, an attention mechanism, and sparsely-gated layers. Each of these computational blocks is crucial for a subset of the tasks we train on. Interestingly, even if a block is not crucial for a task, we observe that adding it never hurts performance and in most cases improves it on all tasks. We also show that tasks with less data benefit largely from joint training with other tasks, while performance on large tasks degrades only slightly if at all.

1 Introduction

Recent successes of deep neural networks have spanned many domains, from computer vision [13] to speech recognition [8] and many other tasks. Convolutional networks excel at tasks related to vision, while recurrent neural networks have proven successful at natural language processing tasks, e.g., at machine translation [27, 3, 4]. But in each case, the network was designed and tuned specifically for the problem at hand. This limits the impact of deep learning, as this effort needs to be repeated for each new task. It is also very different from the general nature of the human brain, which is able to learn many different tasks and benefit from transfer learning. The natural question arises:

Can we create a unified deep learning model to solve tasks across multiple domains?

Google's MultiModel Architecture

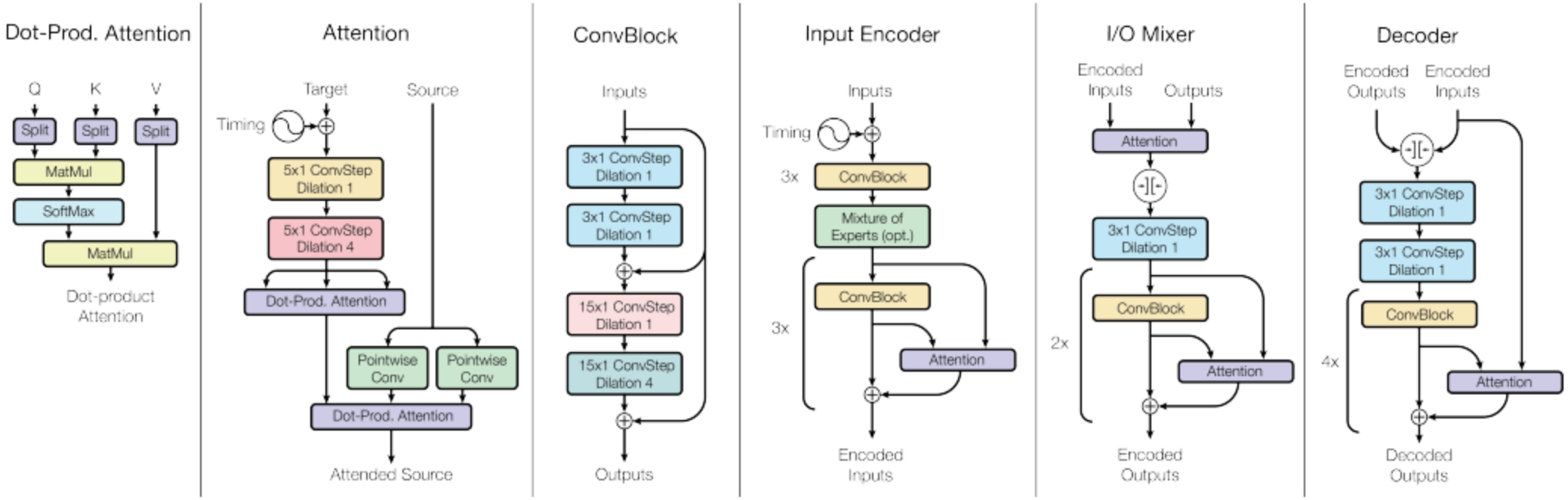


Figure 3: Architecture of the MultiModel; see text for details.

Google's MultiModel Performance

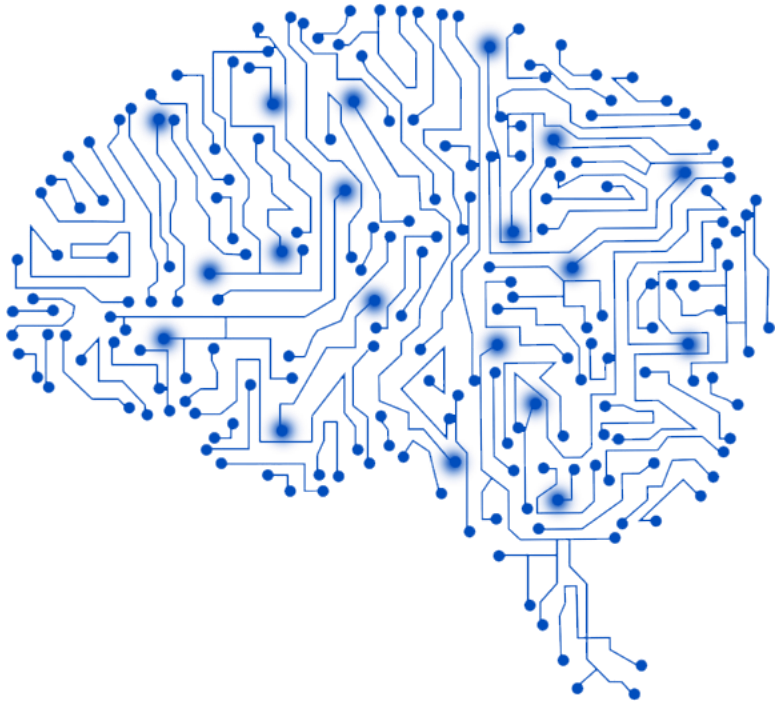
Problem	MultiModel (joint 8-problem)	State of the art
ImageNet (top-5 accuracy)	86%	95%
WMT EN → DE (BLEU)	21.2	26.0
WMT EN → FR (BLEU)	30.5	40.5

Table 1: Comparing MultiModel to state-of-the-art from [28] and [21].

Problem	Joint 8-problem		Single problem	
	log(perpexity)	accuracy	log(perplexity)	accuracy
ImageNet	1.7	66%	1.6	67%
WMT EN→DE	1.4	72%	1.4	71%
WSJ speech	4.4	41%	5.7	23%
Parsing	0.15	98%	0.2	97%

Table 2: Comparison of the MultiModel trained jointly on 8 tasks and separately on each task.

Separating Hype from Reality in Deep Learning



Sameer Farooqui

Slides:
http://bit.ly/hype-deep-learning

Presented June 2018 @ SPARK SUMMIT