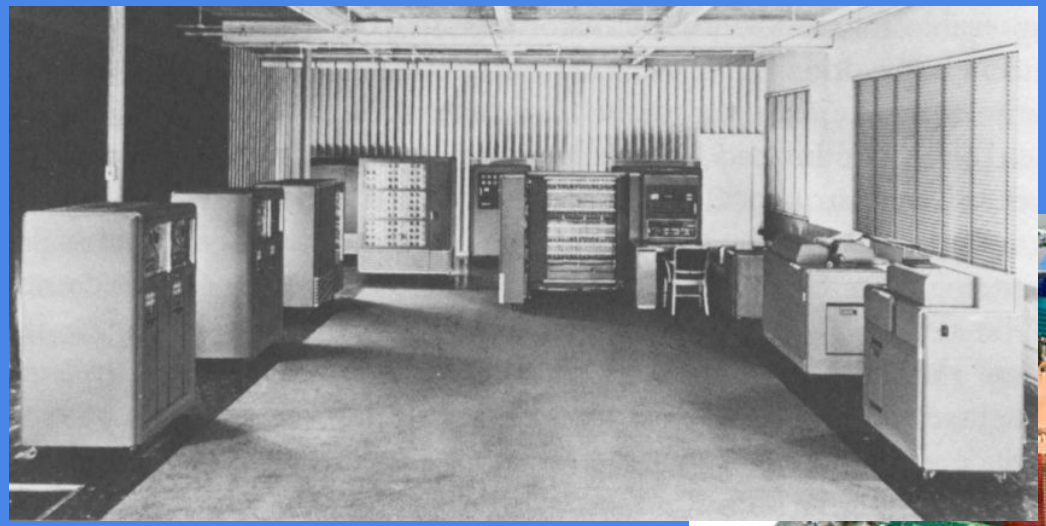
The development of AI thanks to advancements in hardware

Iker García Ferrero
Eritz Yerga Gutierrez

Motivation



IBM 701

IBM Minsky

Initial optimism

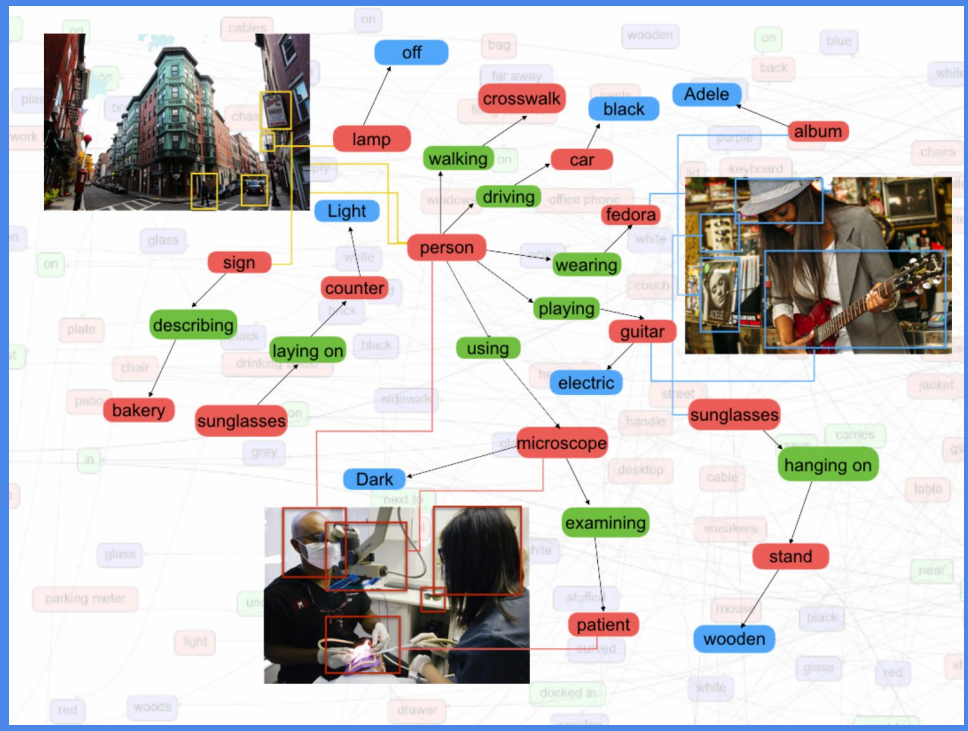
“ I’m not trying to impress you or leave you amazed, but the easiest way I have to sum it up is to say that at the moment machines capable of thinking, learning and creating exist in the world. Furthermore, their ability to do that will quickly increase until (in a near future) the magnitude of problems they will be able to solve will be on par with human mind itself on the same tasks. ”

- Herbert Simon. 1957.

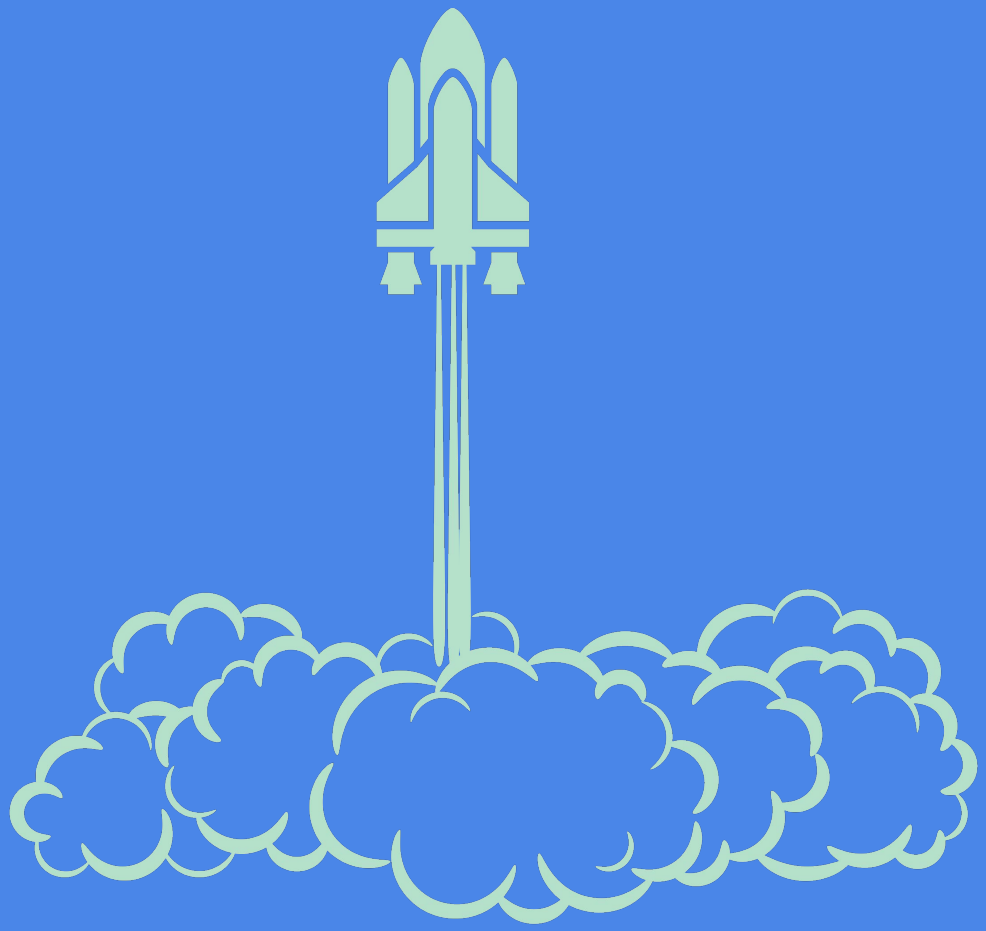
February 24, 1956. Arthur Samuel demonstration of his program in an IBM 701 computer

A crash with reality

- Problems when facing complex tasks.
- US translation project: **“the spirit is strong but the flesh is weak” -> “the vodka is good but the flesh is rotten**”.
- Impossibility to handle combinatory explosion.



AI is skyrocketing. Why now?



Advancements in CPUs

169,200 TIMES FASTER

APPLE II
1977
500,000 FLOPS
US$1298

Ryzen 7 1800X
2017
84,600,000,000 flops
Full System: US$800

1982		2017	WHAT'S DIFFERENT?
 APPLE][plus		 APPLE iMAC 21.5"	
48 KILOBYTES	MEMORY	8 GIGABYTES	166,666x MORE
1 MHZ	CPU SPEED	1.6 GHZ	1600x FASTER
140 KILOBYTES	STORAGE	1 TERABYTE	7,508,684x MORE
595 (WITH DRIVE) $4353 (2017 DOLLARS)	PRICE	$1099	3.9x LESS
 FORD MUSTANG 4CYL		 FORD MUSTANG ECOBOOST 4CYL	
22 MPG	MPG	20 MPG	0.9x LESS
16.2 SEC	0-60 TIME	5.5 SEC	2.9x FASTER
88 HP	HORSEPOWER	310 HP	3.5x MORE
$6789 $17,437 (2017 DOLLARS)	PRICE	$26,195	1.5x MORE

SO, IF CARS DEVELOPED AT THE PACE OF COMPUTERS, A 2017 FORD MUSTANG ECOBOOST WOULD HAVE:

660,764,192 HP
GET FROM 0-60 IN 0.0034 SEC
GET ABOUT 3,666,652 MPG AND COST $4,471

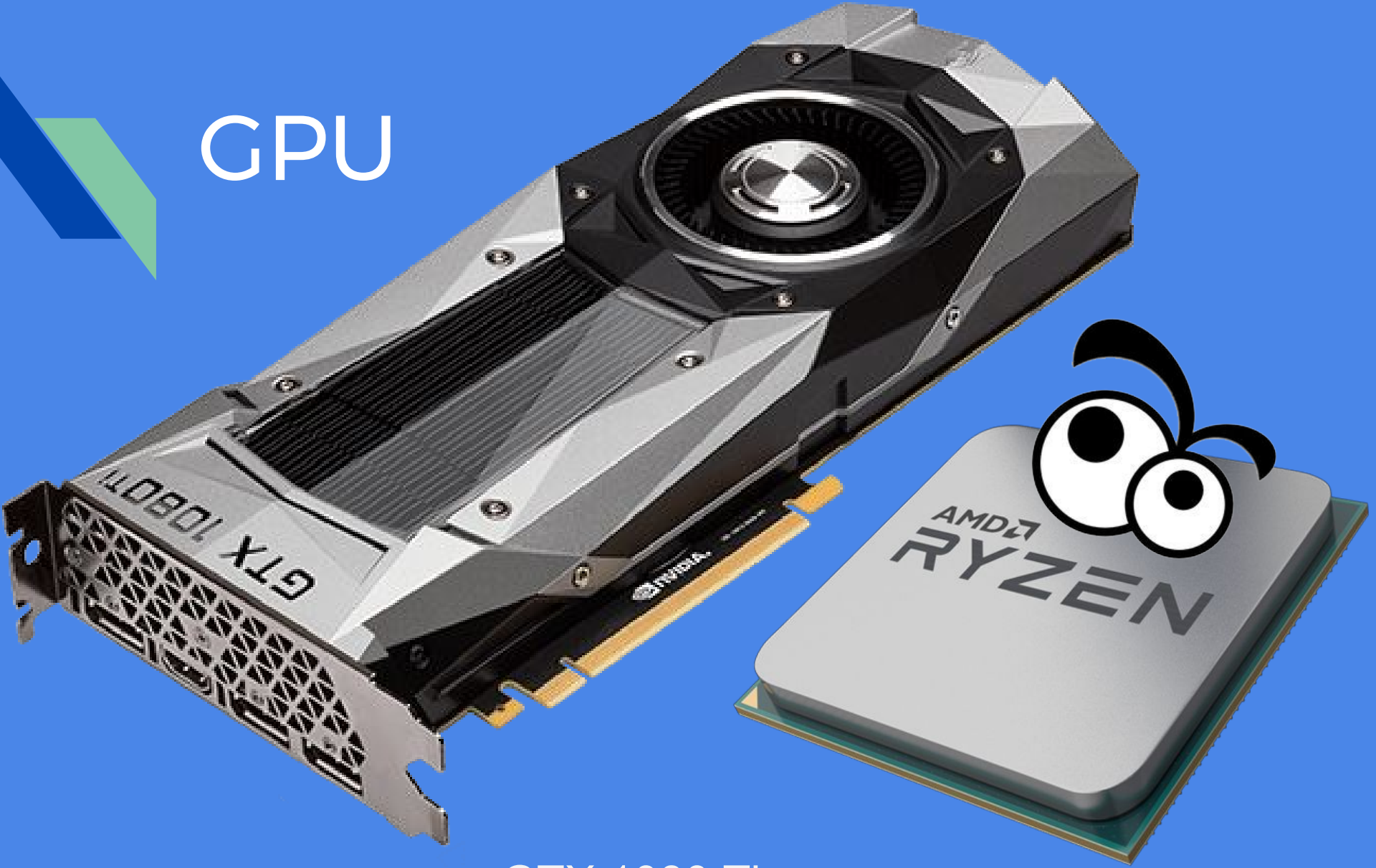
BUT, IF COMPUTERS DEVELOPED AT THE PACE OF CARS, A 2017 APPLE iMAC 21.5" WOULD HAVE:

43.2K OF RAM MEMORY,
WOULD RUN AT 2.9MHz
WOULD BE ABLE TO STORE 490K OF DATA
AND WOULD COST $6,529

IT'S SORT OF A STUPID COMPARISON. BUT IT'S INTERESTING.

JALOPNIK

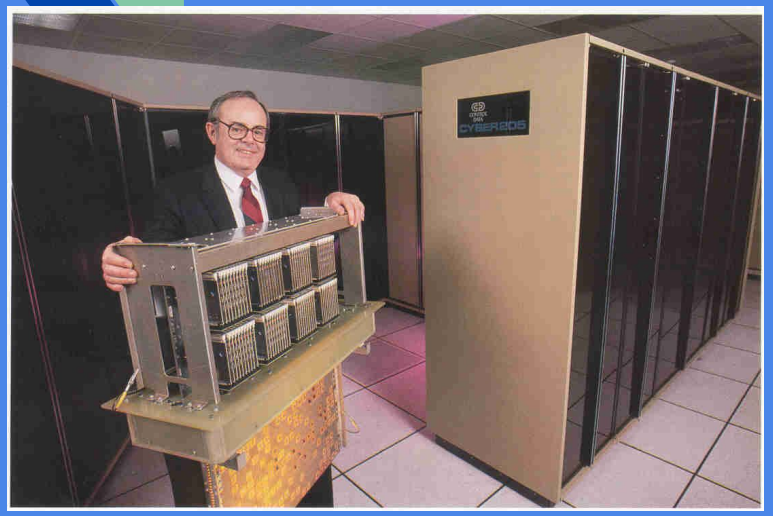
GPU

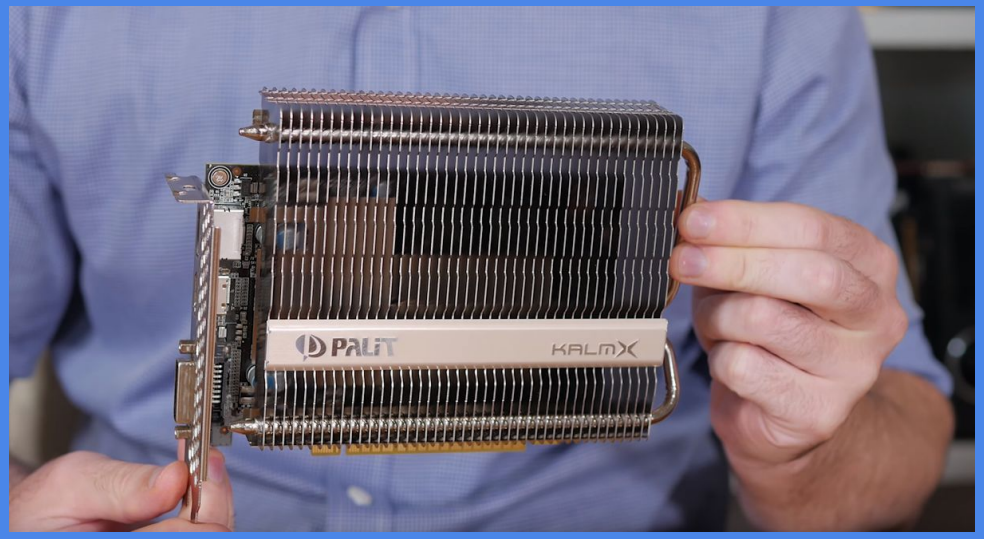


GTX 1080 Ti
11000000000000 flops
130 times faster than the Ryzen 7 1800X

High computational power available to everyone thanks to GPUs

Millions of researchers, projects, applications... around the world





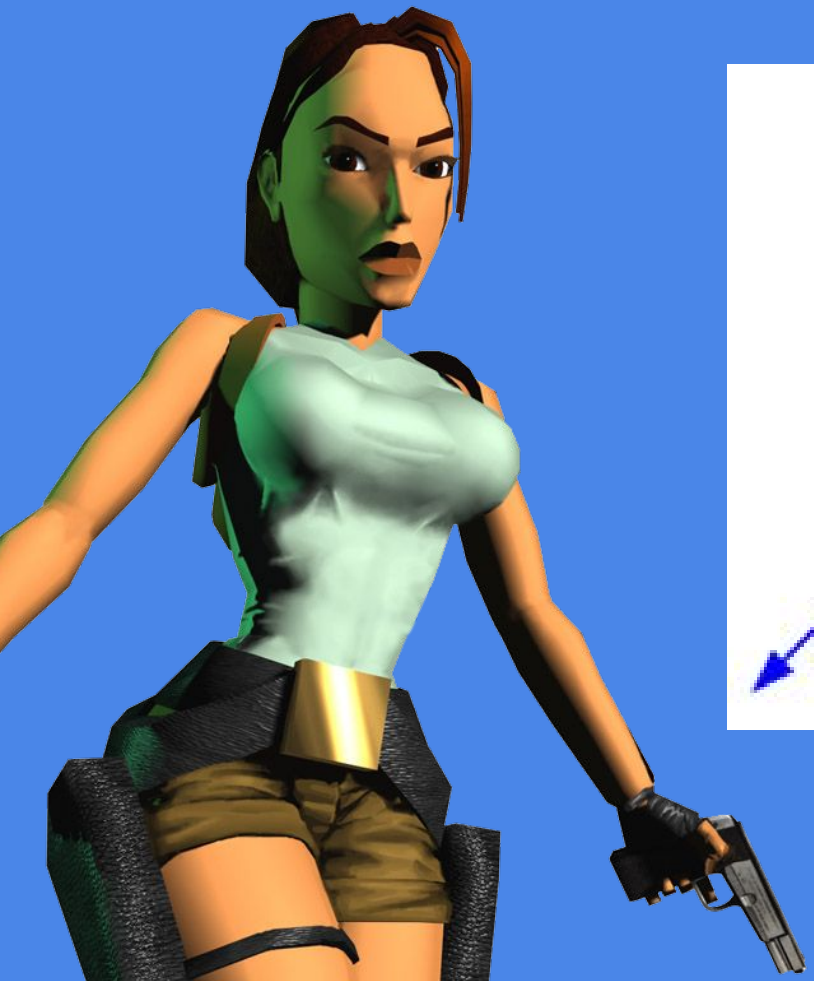
ETA10

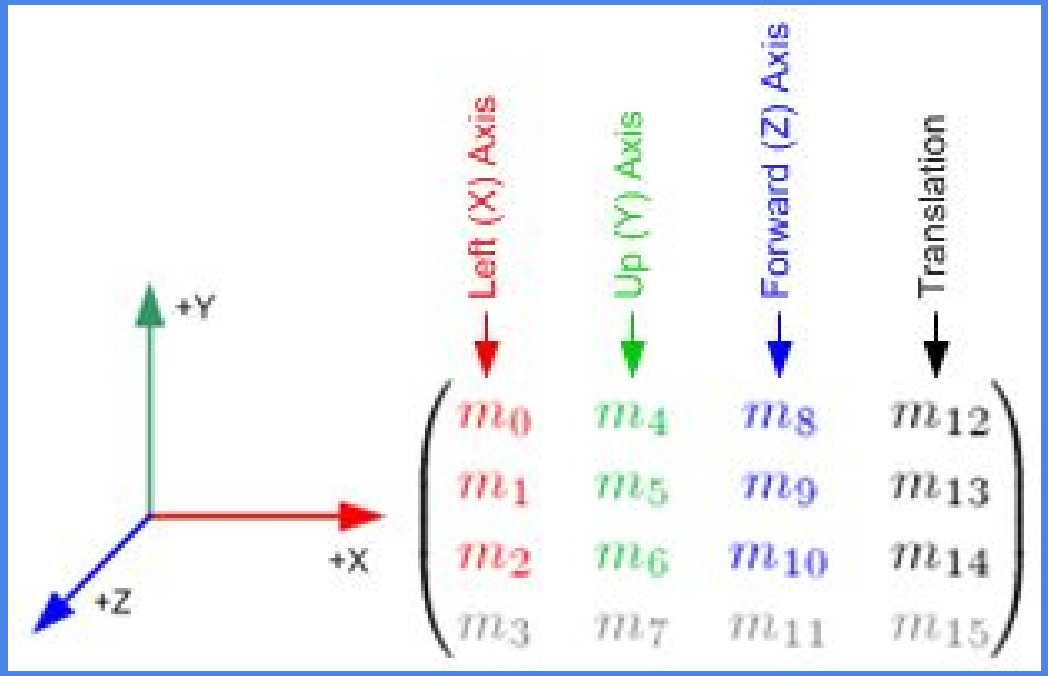
- 1987 Supercomputer
- 10 GLOPS
- High power consumption
- Liquid nitrogen cooling
- Very expensive
- Only available for few universities and research centers, such as Florida State University or Johnson Space Center

NVIDIA GTX 1050

- 2016 entry level gpu
- 1800 GLOPS
- Less than 75 watts
- Passive cooling
- US$ 110
- Available to everyone. You can buy one right now and be using it tomorrow morning.

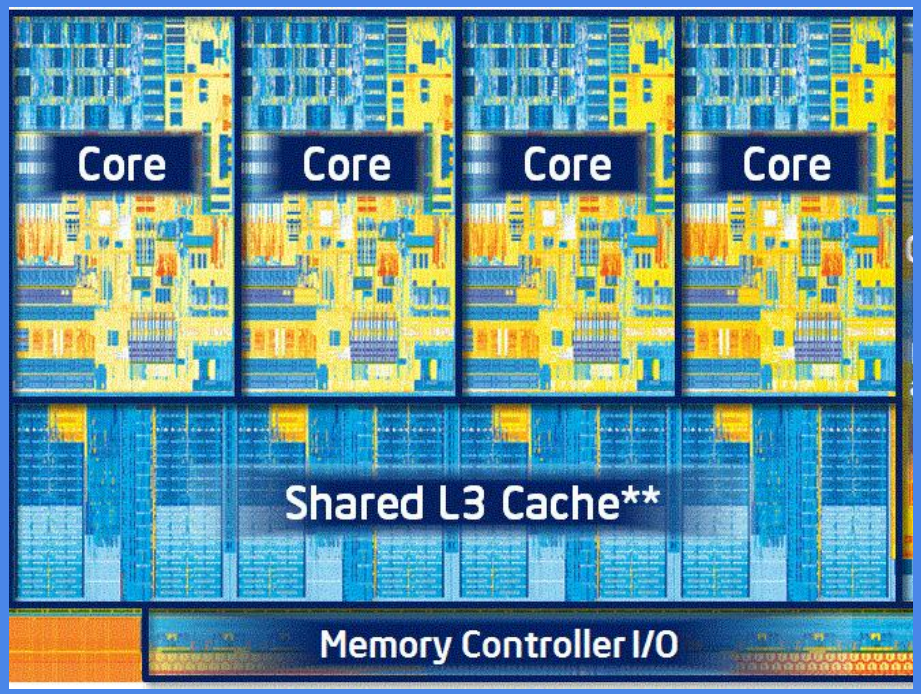
Why GPUs?



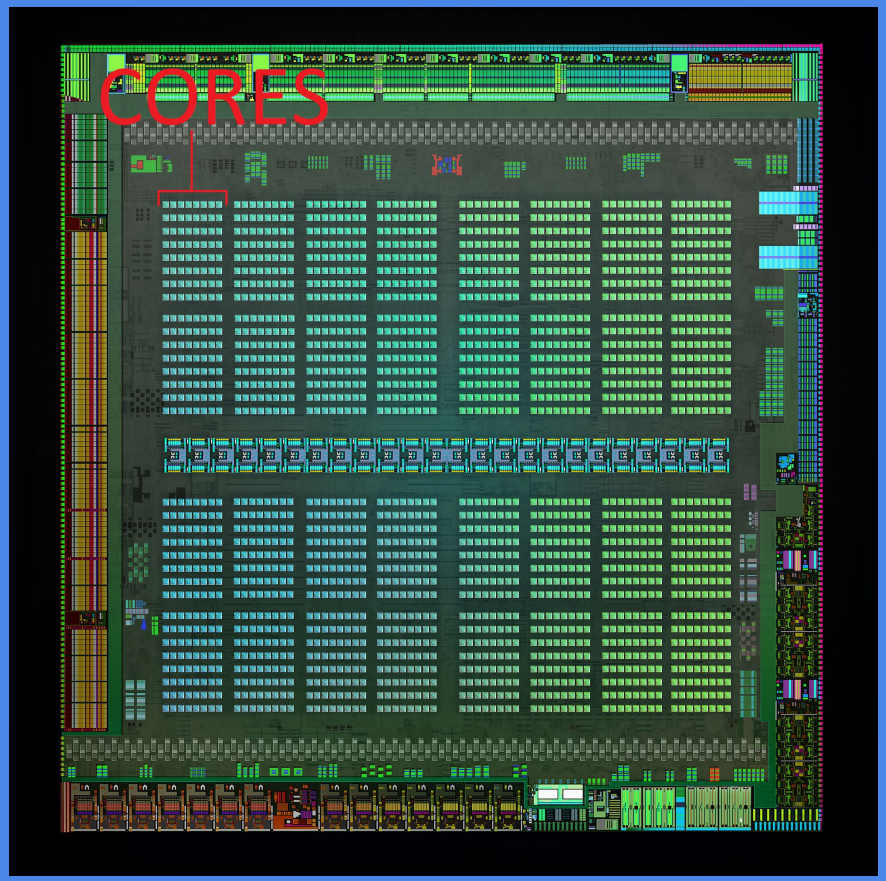

+Y
+X
+Z
Left (X) Axis
Up (Y) Axis
Forward (Z) Axis
Translation
m_0 m_4 m_8 m_{12}
m_1 m_5 m_9 m_{13}
m_2 m_6 m_{10} m_{14}
m_3 m_7 m_{11} m_{15}

World's First GPU

GeForce 256: “A single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that is capable of processing a minimum of 10 million polygons per second”
- NVIDIA 1999



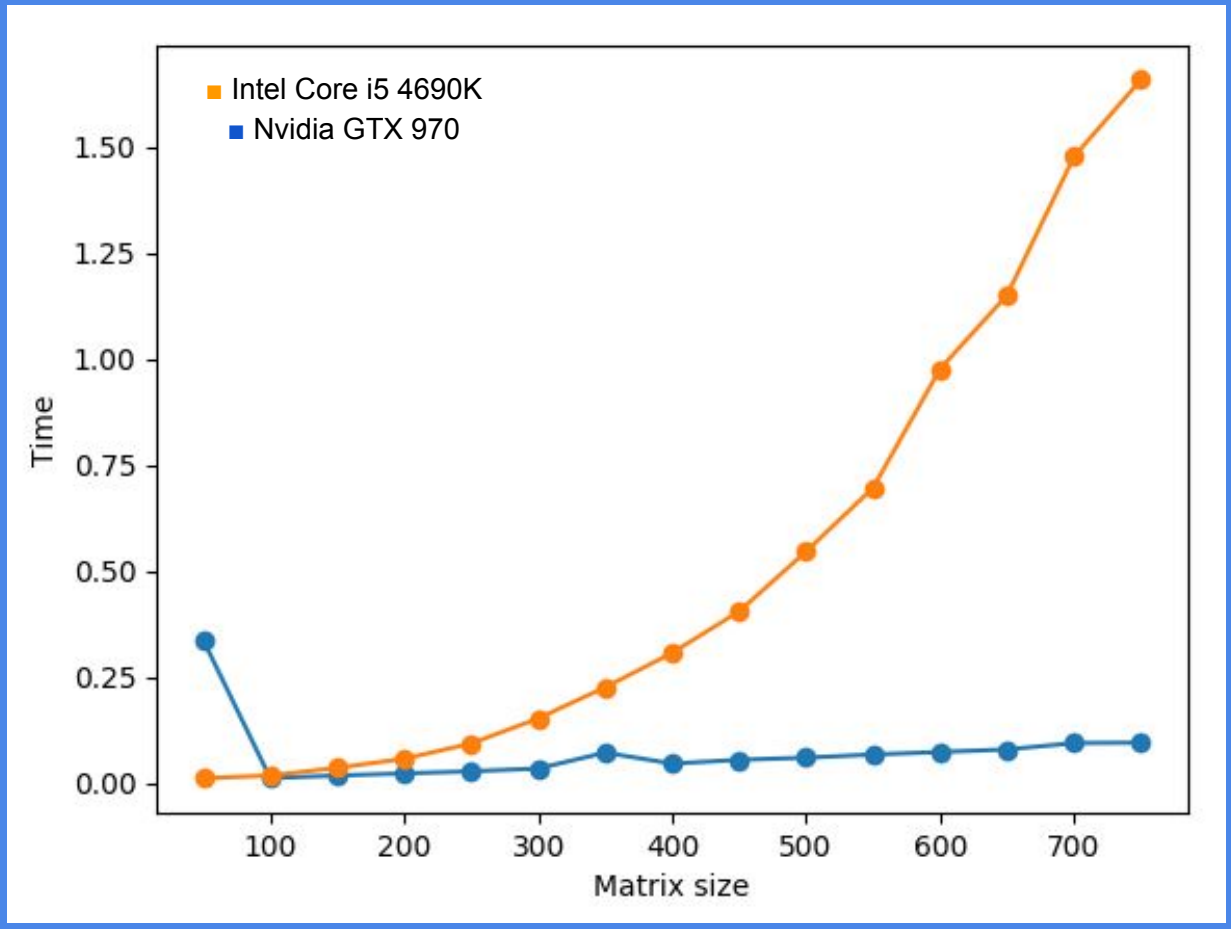
Intel i7. (4 Cores)



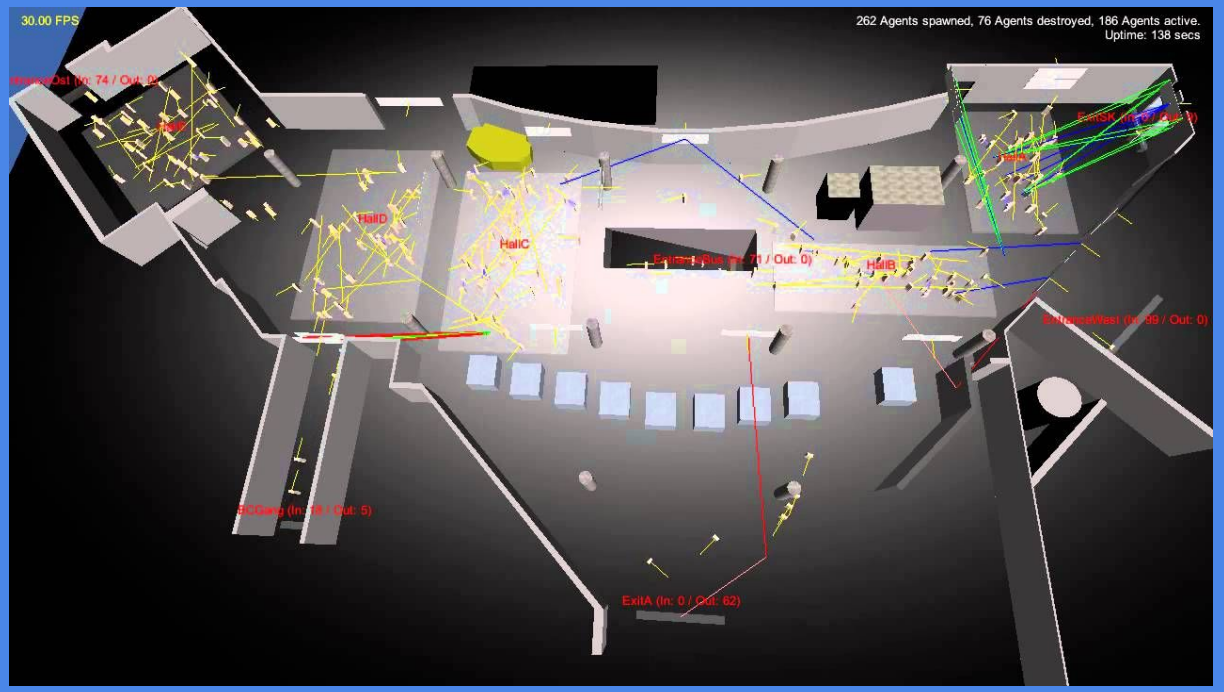
Nvidia GTX 1080 (2560 Cores)

Intel Core i5 4690K a 4 core CPU at 4GHz
vs
NVIDIA GTX 970 a 1664 core GPU at 1,350GHz

```python
from __future__ import print_function
import matplotlib
import matplotlib.pyplot as plt
import tensorflow as tf
import time

def get_times(maximum_time):

    device_times = {
        "/gpu:0":[],
        "/cpu:0":[]
    }
    matrix_sizes = range(500,50000,50)

    for size in matrix_sizes:
        for device_name in device_times.keys():

            print("####### Calculating on the " + device_name + " #######")

            shape = (size,size)
            data_type = tf.float16
            with tf.device(device_name):
                r1 = tf.random_uniform(shape=shape, minval=0, maxval=1, dtype=data_type)
                r2 = tf.random_uniform(shape=shape, minval=0, maxval=1, dtype=data_type)
                dot_operation = tf.matmul(r2, r1)


            with tf.Session(config=tf.ConfigProto(log_device_placement=True)) as session:
                    start_time = time.time()
                    result = session.run(dot_operation)
                    time_taken = time.time() - start_time
                    print(result)
                    device_times[device_name].append(time_taken)

            print(device_times)

            if time_taken > maximum_time:
                return device_times, matrix_sizes


device_times, matrix_sizes = get_times(1.5)
gpu_times = device_times["/gpu:0"]
cpu_times = device_times["/cpu:0"]

plt.plot(matrix_sizes[:len(gpu_times)], gpu_times, 'o-')
plt.plot(matrix_sizes[:len(cpu_times)], cpu_times, 'o-')
plt.ylabel('Time')
plt.xlabel('Matrix size')
plt.show()
```

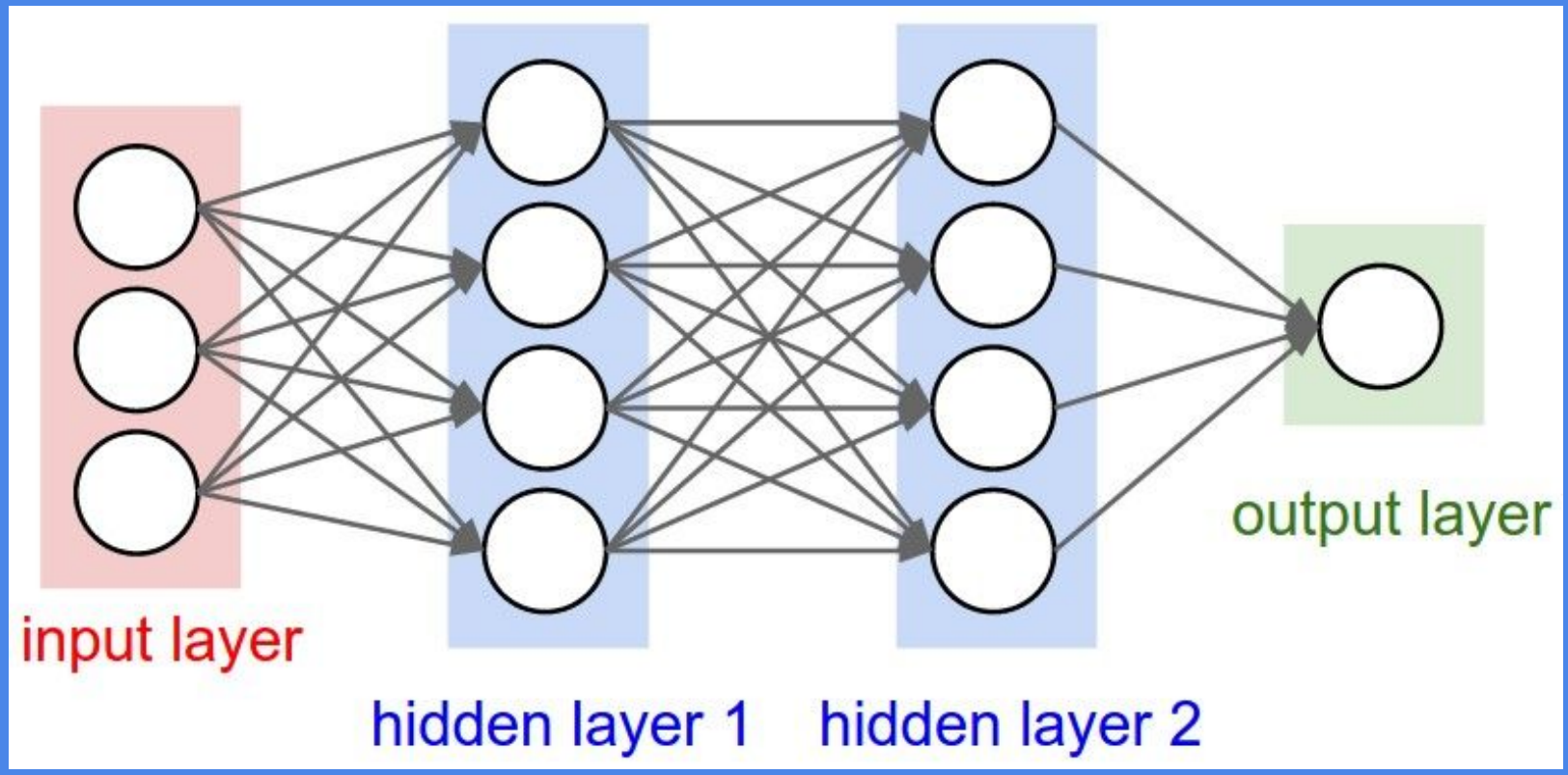


Ok, GPUs are faster for videogames, but, why are so good for AI?

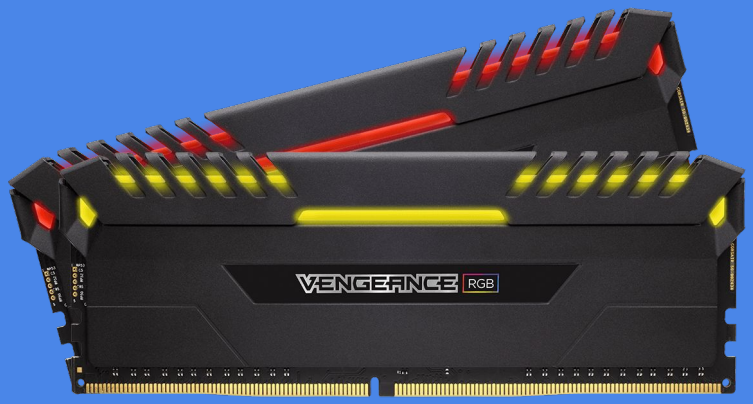


Airport simulation using multi agent systems

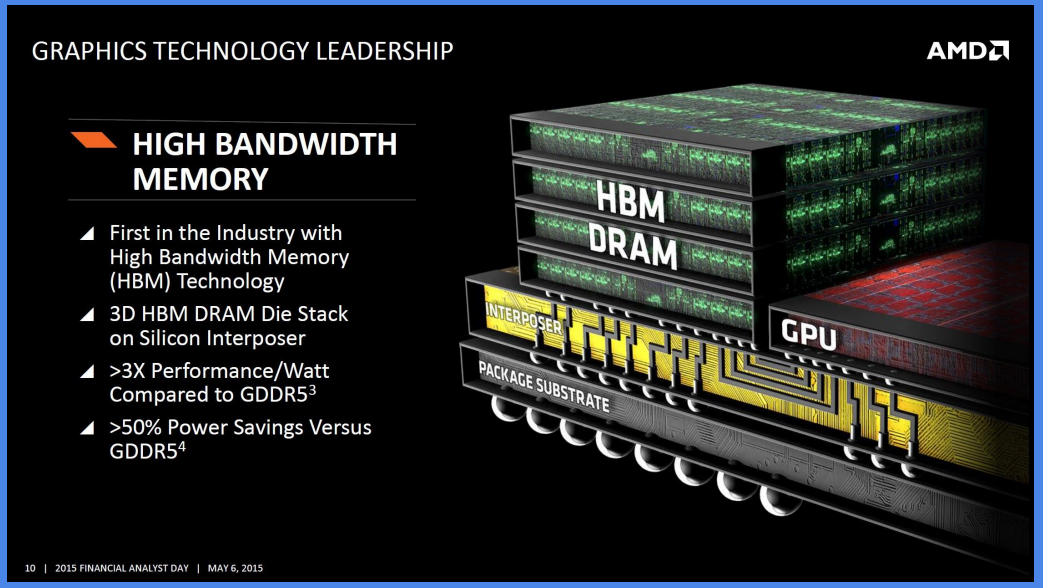
Neural networks, deep learning…



High bandwidth, computer power isn't everything



DDR4
Low latency
50GB/S



GDDR5, GGDR5X, HBM
High bandwidth
900GB/s

NEURAL NETWORK CPU vs GPU

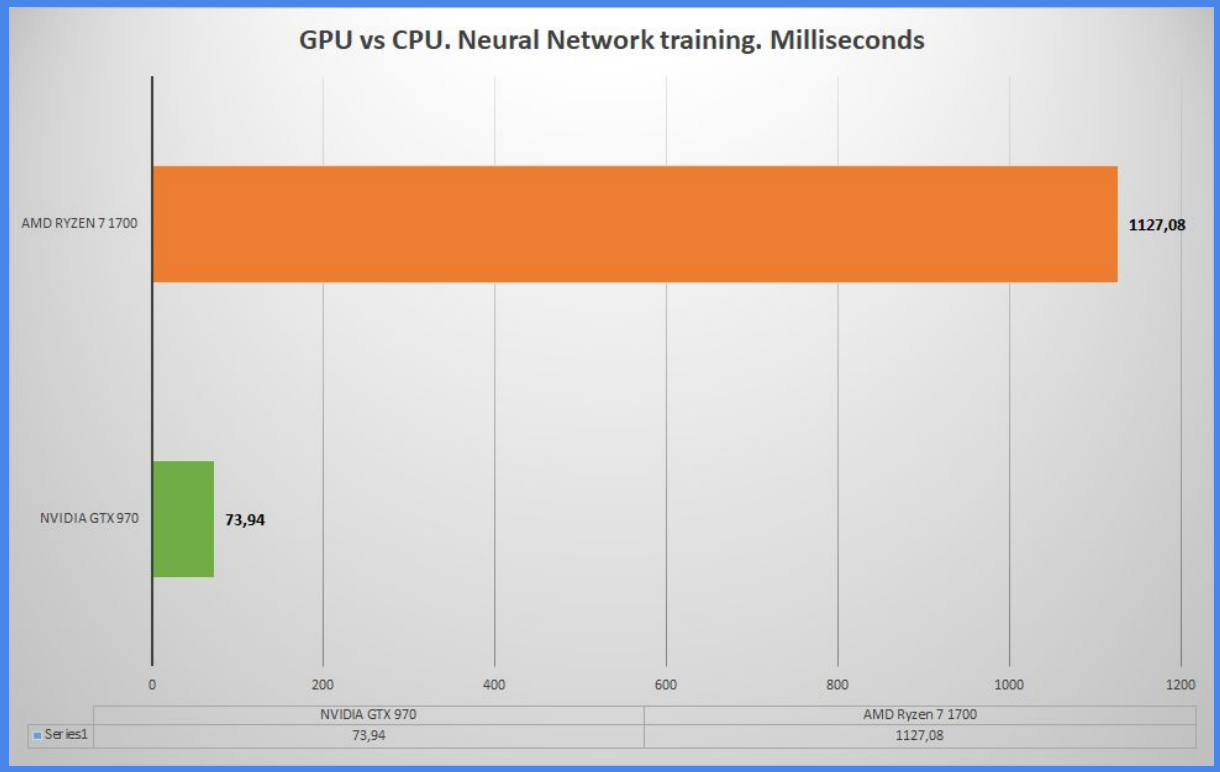
TensorFlow. MNIST convolutional mode.

AMD Ryzen 7 1700 a 8 cores / 16 threads CPU at 3,2Ghz.
Released: 2017. Price: US$ 320

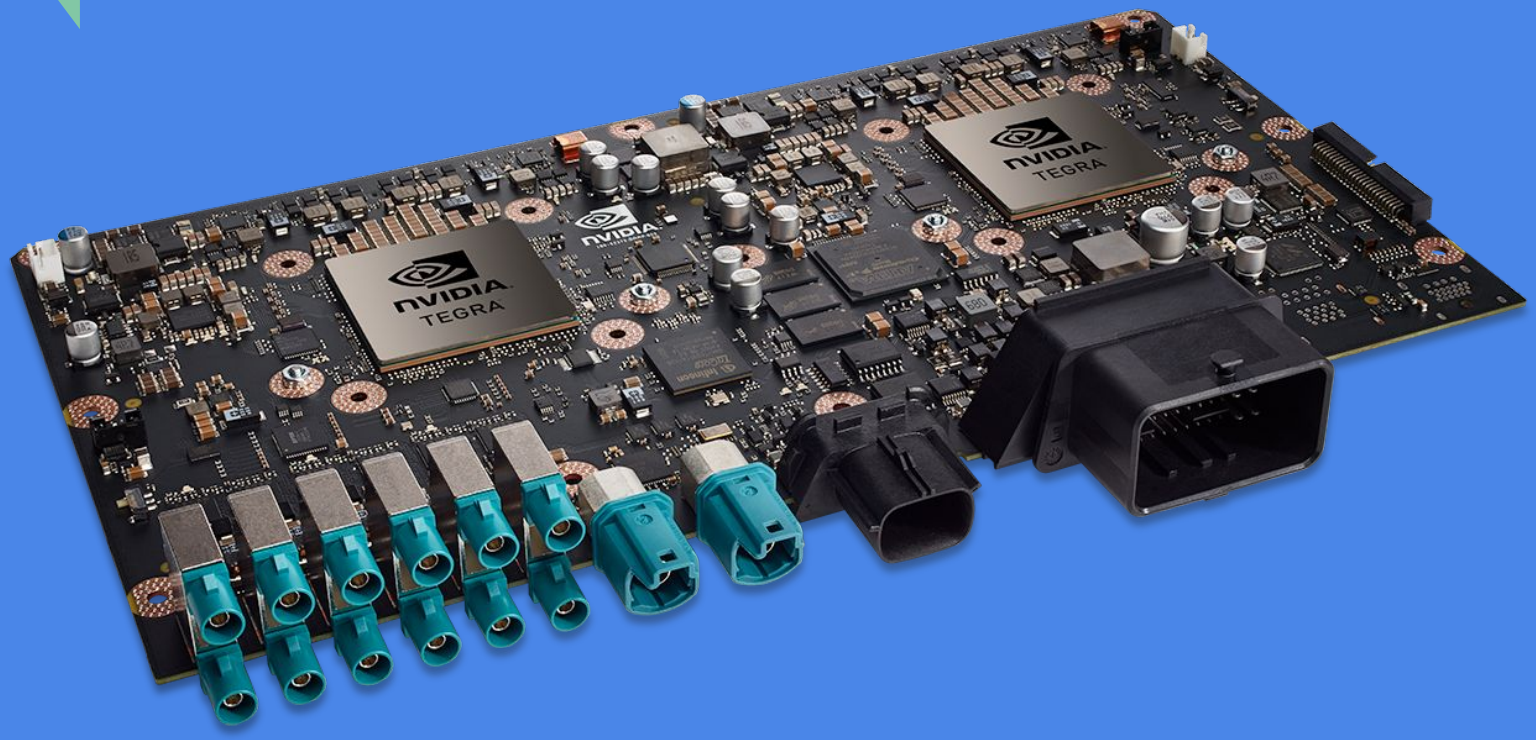
NVIDIA GTX 970 a 1664 core GPU at 1,350GHz
Released: 2014. Price; US$ 329

15.24 times faster

```
import numpy
from six.moves import urllib
from six.moves import xrange  # pylint: disable=redefined-builtin
import tensorflow as tf

SOURCE_URL = 'http://yann.lecun.com/exdb/mnist/'
WORK_DIRECTORY = 'data'
IMAGE_SIZE = 28
NUM_CHANNELS = 1
PIXEL_DEPTH = 255
NUM_LABELS = 10
VALIDATION_SIZE = 5000  # Size of the validation set.
SEED = 66478  # Set to None for random seed.
BATCH_SIZE = 64
NUM_EPOCHS = 10
EVAL_BATCH_SIZE = 64
EVAL_FREQUENCY = 100  # Number of steps between evaluations.


FLAGS = None


def data_type():
  """Return the type of the activations, weights, and placeholder variables."""
  if FLAGS.use_fp16:
    return tf.float16
  else:
    return tf.float32


def maybe_download(filename):
  """Download the data from Yann's website, unless it's already here."""
  if not tf.gfile.Exists(WORK_DIRECTORY):
    tf.gfile.MakeDirs(WORK_DIRECTORY)
  filepath = os.path.join(WORK_DIRECTORY, filename)
  if not tf.gfile.Exists(filepath):
    filepath, _ = urllib.request.urlretrieve(SOURCE_URL + filename, filepath)
    with tf.gfile.GFile(filepath) as f:
      size = f.size()
    print('Successfully downloaded', filename, size, 'bytes.')
  return filepath


def extract_data(filename, num_images):
  """Extract the images into a 4D tensor [image index, y, x, channels].

  Values are rescaled from [0, 255] down to [-0.5, 0.5].
  """
  print('Extracting', filename)
  with gzip.open(filename) as bytestream:
    bytestream.read(16)
    buf = bytestream.read(IMAGE_SIZE * IMAGE_SIZE * num_images * NUM_CHANNELS)
    data = numpy.frombuffer(buf, dtype=numpy.uint8).astype(numpy.float32)
    data = (data - (PIXEL_DEPTH / 2.0)) / PIXEL_DEPTH
    data = data.reshape(num_images, IMAGE_SIZE, IMAGE_SIZE, NUM_CHANNELS)
    return data


def extract_labels(filename, num_images):
  """Extract the labels into a vector of int64 label IDs."""
  print('Extracting', filename)
  with gzip.open(filename) as bytestream:
    bytestream.read(8)
    buf = bytestream.read(1 * num_images)
    labels = numpy.frombuffer(buf, dtype=numpy.uint8).astype(numpy.int64)
  return labels


def fake_data(num_images):
  """Generate a fake dataset that matches the dimensions of MNIST."""
  data = numpy.ndarray(
      shape=(num_images, IMAGE_SIZE, IMAGE_SIZE, NUM_CHANNELS),
      dtype=numpy.float32)
  labels = numpy.zeros(shape=(num_images,), dtype=numpy.int64)
  for image in xrange(num_images):
    label = image % 2
```

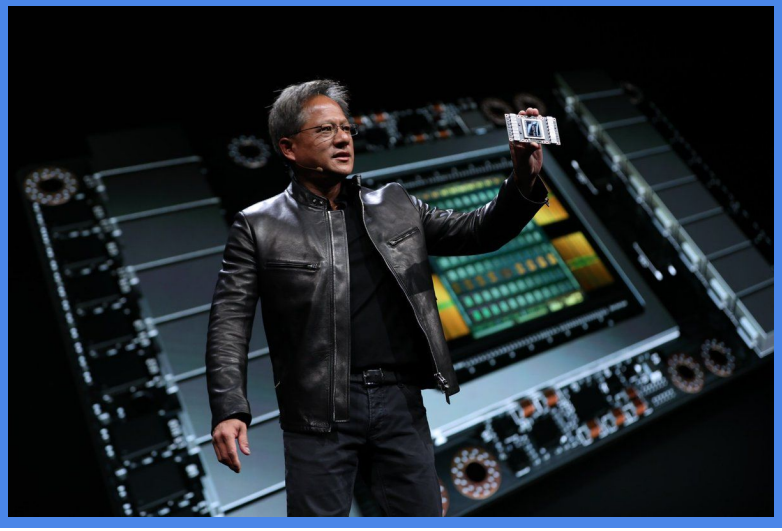


A world of possibilities
$$ Real applications for AI $$



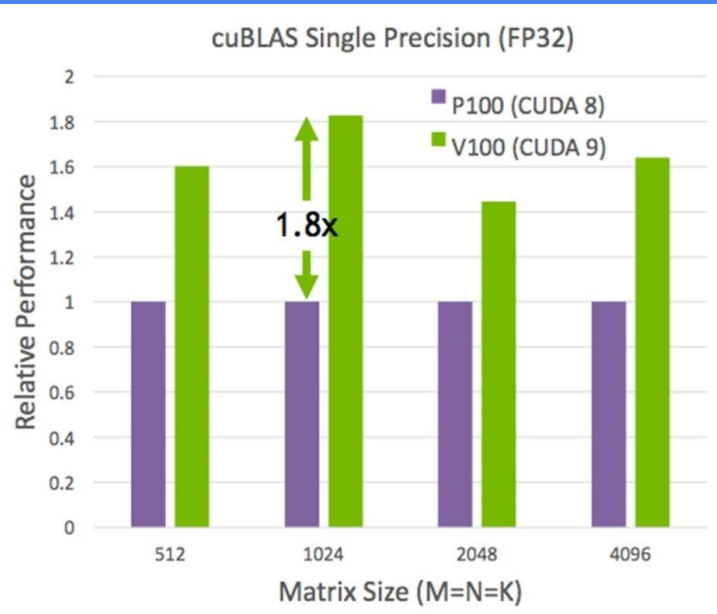
NVIDIA parker, a computer for autonomous cars with a 256 core GPU

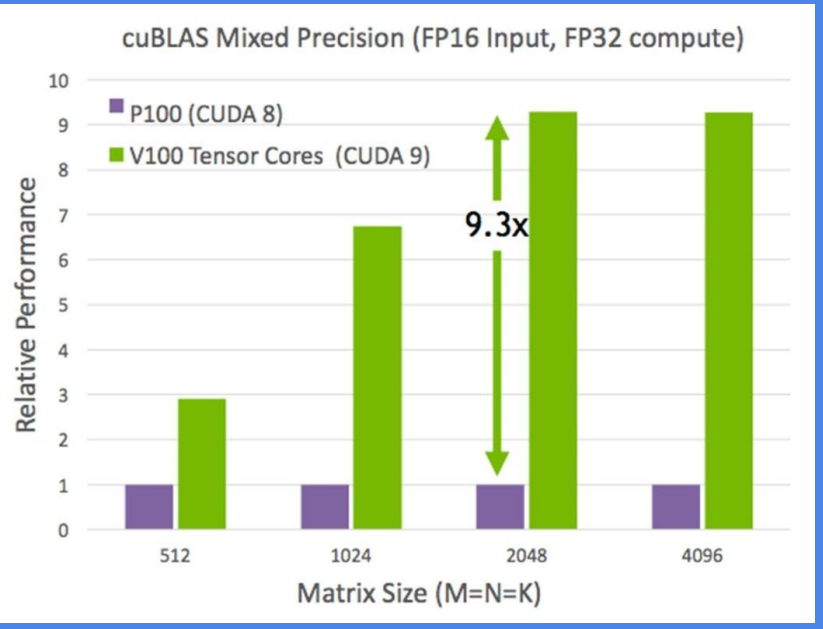
A vision of the future



GPU PERFORMANCE COMPARISON

	P100	V100	Ratio
DL Training	10 TFLOPS	120 TFLOPS	12x
DL Inferencing	21 TFLOPS	120 TFLOPS	6x
FP64/FP32	5/10 TFLOPS	7.5/15 TFLOPS	1.5x
HBM2 Bandwidth	720 GB/s	900 GB/s	1.2x
STREAM Triad Perf	557 GB/s	855 GB/s	1.5x
NVLink Bandwidth	160 GB/s	300 GB/s	1.9x
L2 Cache	4 MB	6 MB	1.5x
L1 Caches	1.3 MB	10 MB	7.7x





New generation of supercomputers for AI

Nvidia DGX-1

IBM Minsky

Questions