CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Deep Learning

Deep Learning

Lecture: Introduction – Part 2

Ted Scully

Contents

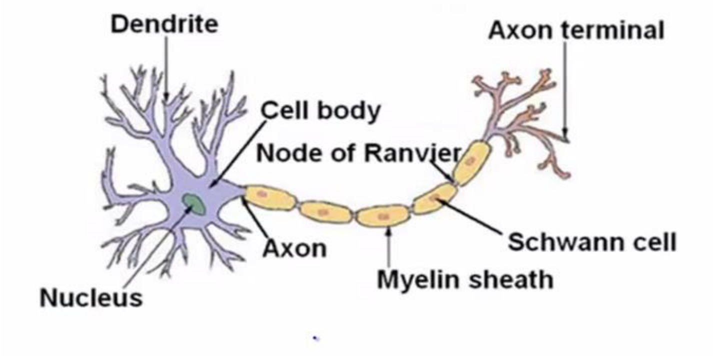
Neural Networks

- Neural networks, are a biologically-inspired approach to machine learning that allows us to build models to map inputs to outputs based on observational data.

- We say biologically-inspired because the original core unit of a neural network (a neuron) was a simplistic model of the neurons in our brains.

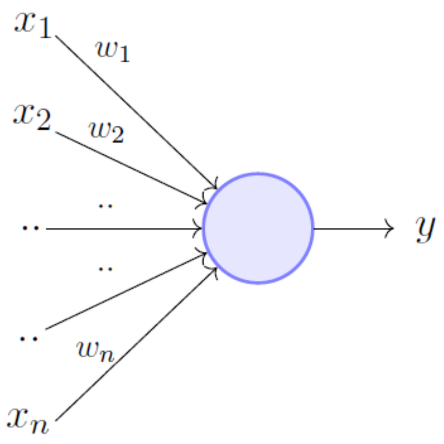
- In each hemisphere of our brain, humans have a primary visual cortex, also known as V1, containing 140 million neurons, with tens of billions of connections between them.

- While the foundation of NN’s are loosely inspired by biology the operation of modern deep learning models is not similar to the operation of our own brains.



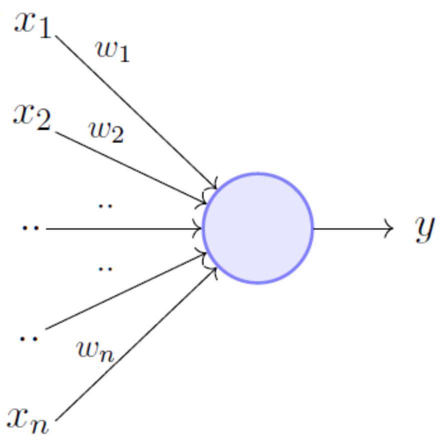
What is a Neuron?

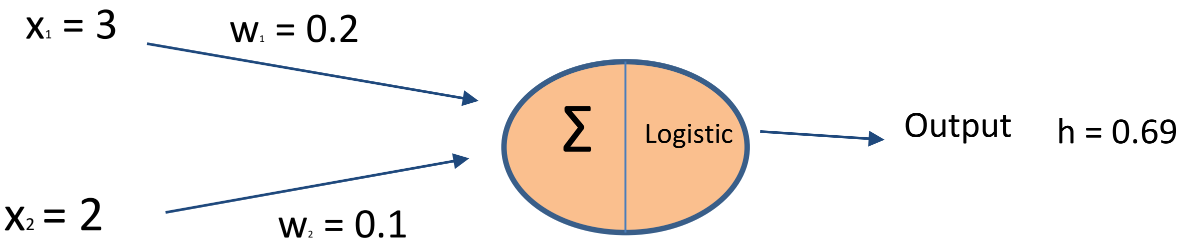
- Artificial neurons are the building blocks of artificial neural networks.
- The neuron receives one or more inputs (which can be feature values) and sum them to produce a prediction.
- More specifically each input is separately weighted. Each weight is multiplied by the input feature value and the resulting sum is passed through a non-linear transformation known as an activation function.



What is a Neuron?

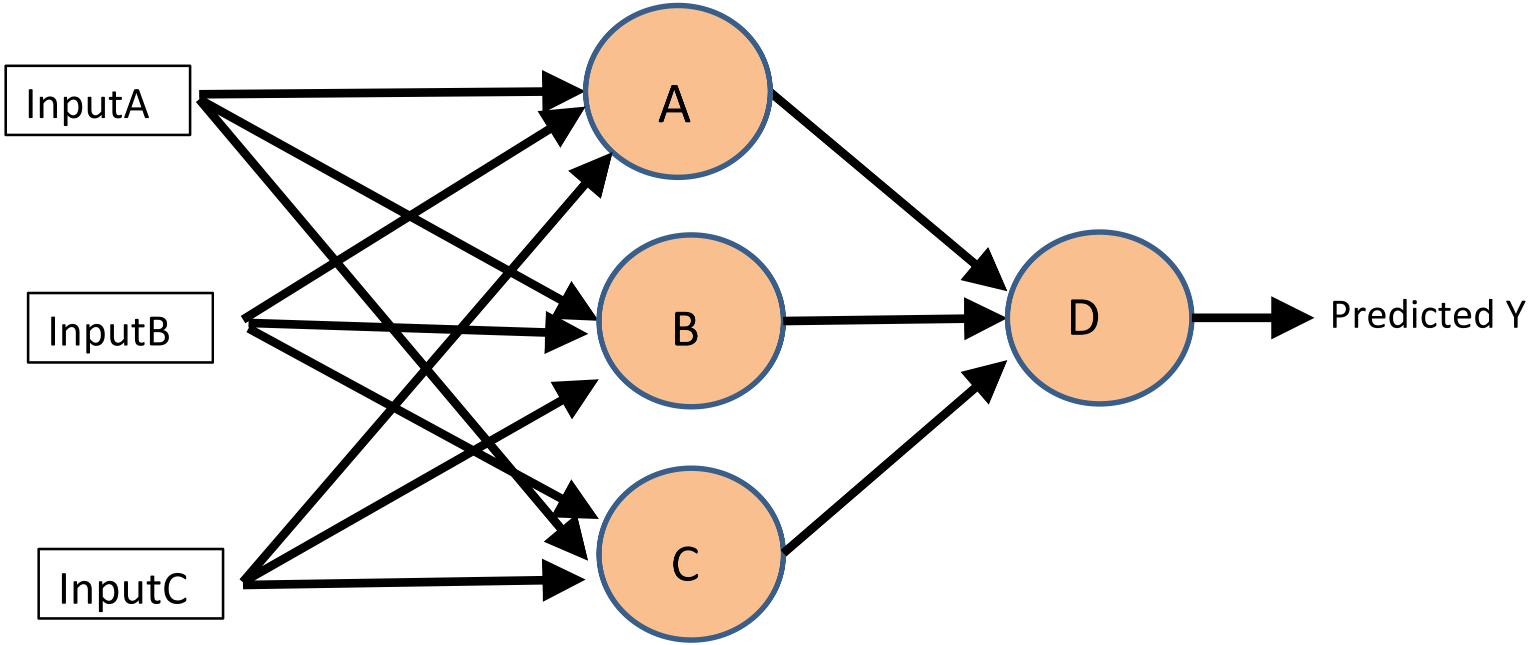
- Artificial neurons are the building blocks of artificial neural networks.
- The neuron receives one or more inputs (which can be feature values) and sum them to produce a prediction.
- More specifically each input is separately weighted. Each weight is multiplied by the input feature value and the resulting sum is passed through a non-linear transformation known as an activation function.





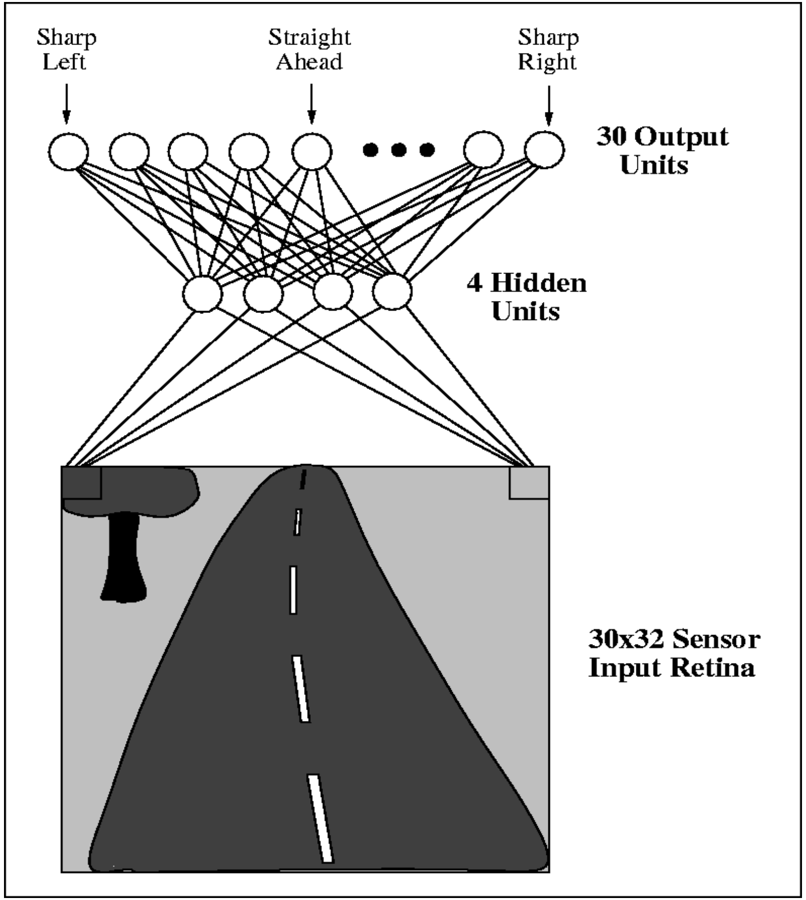
A Neural Network

- The example in the image below shows a two layer neural network with a single hidden layer (notice the inputs are not included as a layer in neural networks). Notice we are stacking together some of the neurons from the previous slide.



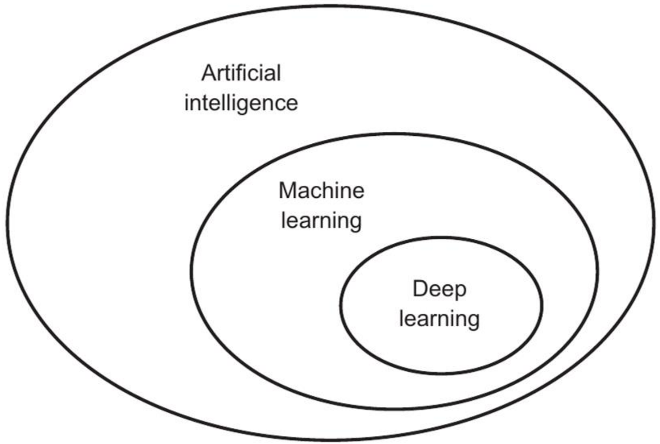
ALVINN

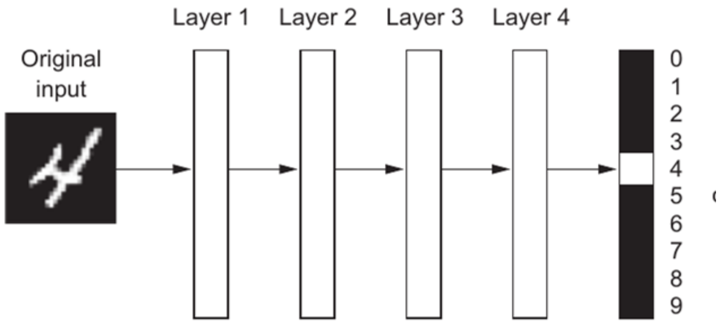
- ALVINN's architecture consists of a **single hidden layer** back-propagation network (the image is a slightly simplified version of the ALVINN architecture).
- The input layer of the network consists of a 30x32 image which receives input from the vehicles video camera.
- Each input unit is fully connected to a layer of **four hidden units** which are in turn fully connected to a layer of 30 output units.
- The output layer is **a linear representation of the direction** the vehicle should travel in order to keep the vehicle on the road



What is Deep Learning

1. So what is Deep Learning?
2. At their core the vast majority of deep learning models are based on artificial neural networks.
3. More specifically, a <u>deep network consists of successive layers of artificial neural networks stacked consecutively</u>.
4. How many layers contribute to a model of the data is called the <u>depth</u> of the model.





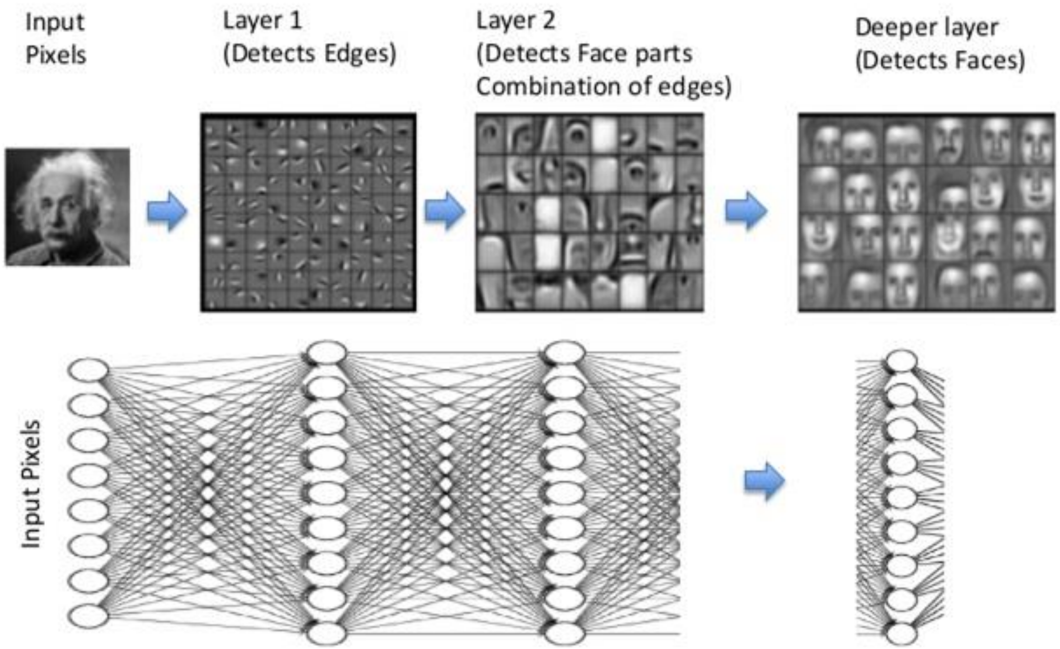
Deep Learning

1. In deep learning, each layer **transforms** its input data into a more abstract and composite representation.

2. This is easier to conceptualize with images. The first layer will take in the raw pixels and may learn to recognize edges within the original image.

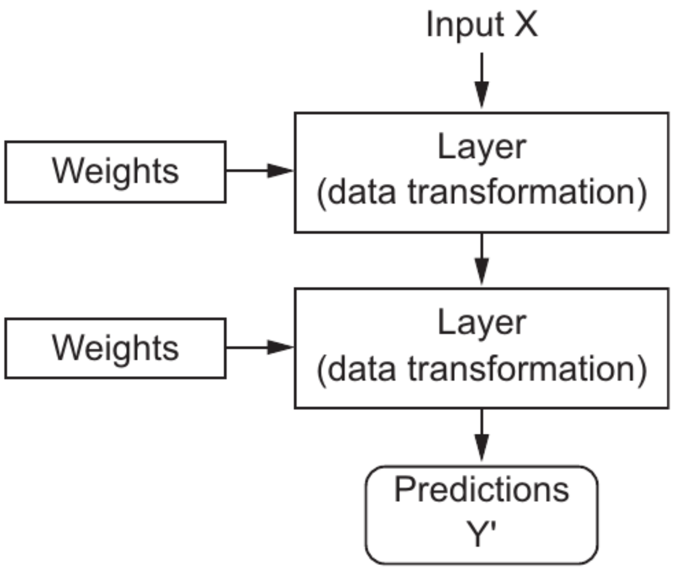
3. The next layer may learn to identify collections of specific edges.

4. The next layer may learn to recognize small component of the image such as ears, mouth and so on.



How Neural Networks Work

- The <u>transformation implemented by a layer is parameterized by its weights</u>
- Learning means finding a set of values for the weights of all layers in a network, such that the network will correctly map example inputs to their associated targets.
- A deep neural network can contain tens of millions of parameters!

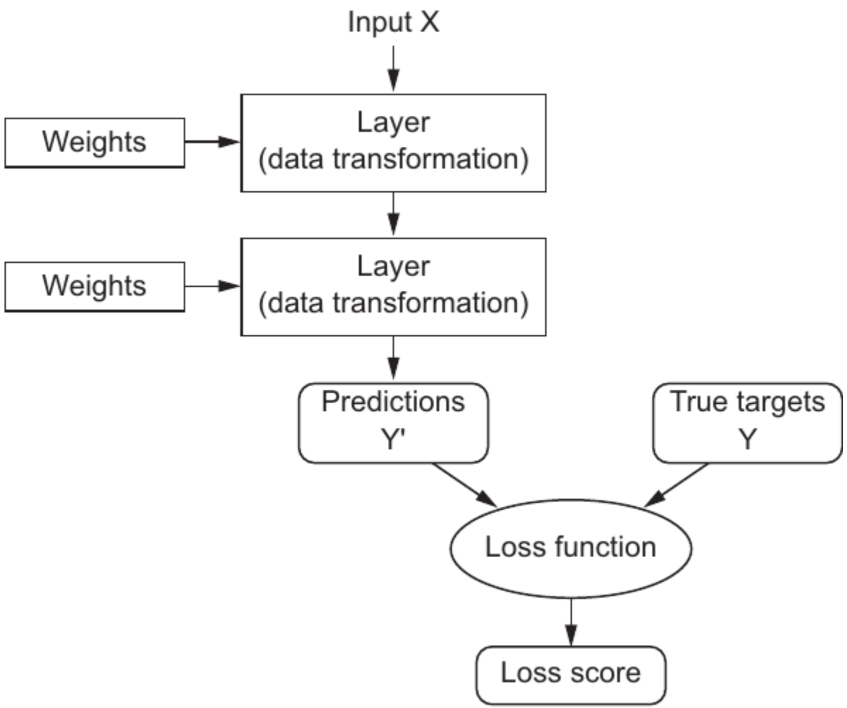


How Neural Networks Work – Loss Function

- We need to be able to measure how far the predicted output of a neural network is from what you expected.

- This is the purpose of the loss function of the network, also called the objective function.

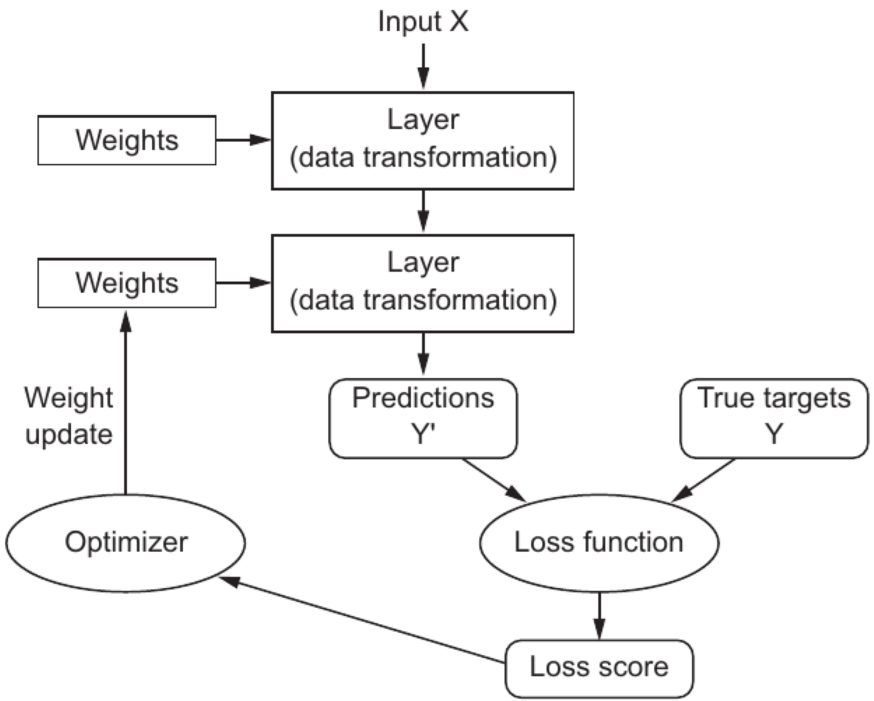
- The loss function takes the predictions of the network and the true target
  - (what you wanted the network to output) and computes a distance score, capturing how well the network has done on specific example(s).



How Neural Networks Work

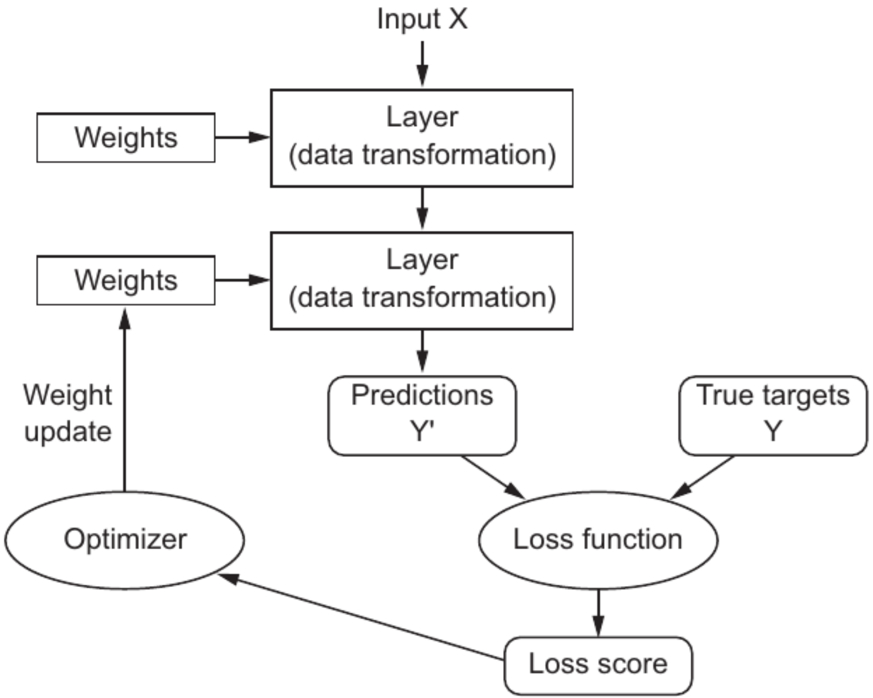
- The fundamental component of neural networks and deep learning systems is to use the score calculated by our loss function as a **feedback signal** to adjust the value of the weights a little, in a direction that will lower the loss score for the current example.

- This adjustment is the job of the optimizer, which implements what's called the Backpropagation algorithm: the central algorithm in deep learning.



How Neural Networks Work

1. Initially, the weights of the network may be assigned random values, so the network merely implements a series of random transformations.
2. As you would expect the initial outputs are far from what it should ideally be, and the loss score is accordingly very high.
3. But with every example the network processes, the weights are adjusted a little in the correct direction, and the loss score decreases.
4. This is the training loop, which repeated a sufficient number of times, yields weight values that minimize the loss function.



Contents

- Recap of Important Concepts from Practical ML
- Overview of Deep Learning Module
- A Brief History of AI and the Emergence of Deep Learning
- Introduction to Deep Learning
- Reasons for Success

Why is Deep Learning Successful Now?

- The two key ideas of deep learning for computer vision—**convolutional neural networks** and **backpropagation**—were already well understood in **1989**.

- The Long Short-Term Memory (**LSTM**) algorithm, which is fundamental to deep learning for time series, was developed in **1997** and has barely changed since.

- So why did deep learning only take off after 2012? What changed in these two decades?

- In general, four factors are driving advances in deep learning:
    - Hardware
    - Datasets and benchmarks
    - Algorithmic advances
    - The Frameworks

Hardware

- The speed of off the shelf CPUs increased by a factor of **5000** between 1990 and 2010. As a result, it is now possible to run small deep-learning models on your laptop.

- However, training specialized deep-learning models for image or language processing often requires much more computational power.

- Throughout the 2000s, companies like NVIDIA and AMD have been developing fast, massively parallel chips (graphical processing units [GPUs]) to power the graphics of increasingly photorealistic video games. For example, the NVIDIA GTX 1080 (a very modest GPU) is capable of 10 TeraFlops. To put this in perspective this is 10 trillion float32 operations a second

- In 2007, NVIDIA launched CUDA (a programming interface for its line of GPUs).
- Deep neural networks, as will see mainly perform matrix multiplication, which are parallelizable.
- Around 2011, some researchers began to write CUDA implementations of neural nets— Alex Krizhevsky were among the first.

The Explosion of Data

- Data is the fuel for deep learning machines (Deep models typically excel with large amounts of data).

- Deep learning is <u>data hungry</u> and the rate at which we collect and generate data is unprecedented.

- For example:
  - Over the last two years alone 90 percent of the data in the world was generated.
  - More than 3.7 billion humans use the internet (that’s a growth rate of 7.5 percent over 2016).
  - Instagram (600M) users post 46,740 photos every minute
  - Emails sent every minute estimated to be 156 million.
  - The number of tweets sent every minute is 456,000.
  - Facebook has over 2 billion active users, more than 300 million photos get uploaded per day on facebook.

Algorithmic Advances

- In addition to hardware and data, until the late 2000s, we were missing a reliable way to train very deep neural networks.

- A number of really important algorithmic advances occurred around around 2009–2010:
  - Better <u>activation functions</u> for neural layers
  - Better <u>weight-initialization</u> schemes
  - Better <u>optimization schemes</u>, such as RMSProp and Adam

- Only when these improvements arrived did it allows us to training models with 10 or more layers.

The Frameworks

- Another key driving factor in the success of deep learning is the wide availability of a range of excellent deep learning frameworks.
- In the early days, deep learning required significant C++ and CUDA expertise, which few people possessed.
- Now, Python programming skills allow you to implement advanced deep learning networks.
- TensorFlow was originally developed by the Google Brain team for internal use in Google. In 2015 it was released and open-sourced.
- TensorFlow 2 now uses Keras it’s high level API.

```python
import tensorflow as tf

mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu, input_shape=(784,)),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

results = model.evaluate(x_test, y_test)
```

How many learnable parameters in the first layer of this network.

```
import tensorflow as tf

class ShallowVGGNet:
    @staticmethod
    def build(width, height, depth, classes):
        model = tf.keras.models.Sequential()
        inputShape = (height, width, depth)
        chanDim = -1

        # first CONV => RELU => CONV => RELU => POOL layer set
        model.add(tf.keras.layers.Conv2D(32, (3, 3), padding="same",
            input_shape=inputShape, activation='relu'))
        model.add(tf.keras.layers.BatchNormalization(axis=chanDim))
        model.add(tf.keras.layers.Conv2D(32, (3, 3), padding="same",activation='relu'))
        model.add(tf.keras.layers.BatchNormalization(axis=chanDim))
        model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(tf.keras.layers.Dropout(0.25))

        # second CONV => RELU => CONV => RELU => POOL layer set
        model.add(tf.keras.layers.Conv2D(64, (3, 3), padding="same",activation='relu'))
        model.add(tf.keras.layers.BatchNormalization(axis=chanDim))
        model.add(tf.keras.layers.Conv2D(64, (3, 3), padding="same",activation='relu'))
        model.add(tf.keras.layers.BatchNormalization(axis=chanDim))
        model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(tf.keras.layers.Dropout(0.25))

        # first (and only) set of FC => RELU layers
        model.add(tf.keras.layers.Flatten())
        model.add(tf.keras.layers.Dense(512,activation='relu'))
        model.add(tf.keras.layers.BatchNormalization())
        model.add(tf.keras.layers.Dropout(0.5))

        # softmax classifier
        model.add(tf.keras.layers.Dense(classes, activation='softmax'))

        # return the constructed network architecture
        return model
```