THAT®

CONFERENCE

Robert Loesch

Customer Engineer - **Google Cloud**

rloesch@google.com
https://www.linkedin.com/in/robertloesch/
https://github.com/googlecloudlab/

SPECIAL THANKS TO ALL OUR AWESOME CAMP SPONSORS!

TruStage™

HEARTLAND BUSINESS SYSTEMS

Intellibus

aws

Unspecified SOFTWARE CO

Architect.io

ionic

CAMUNDA

alexa

Google Cloud

CORE BTS

GrapeCity

Join Us Next Year!
WI 24
THAT®
CONFERENCE
JULY 29TH - AUG. 1ST

Session Feedback

https://that.land/43k4pkQ

Google Cloud Speaking Sessions

THAT Conference Wisconsin, Wisconsin Dells, July 24 - 27

Day	Start Time	Title	Room	Speaker
Monday	2:00 PM	Introduction to Tensorflow	6	Robert Loesch
Tuesday	2:30 PM	Building your personal brand to represent the authentic YOU	African 50	Morgan Bell
Wednesday	4:00 PM	Generative AI on Google Cloud: Unlocking the Potential for Enterprises	African 80	Sagar Kewalramani
Thursday	1:00 PM	Code Championship - Competitive Computer Coding for Kids	6	Luke Schlangen

Google Cloud Innovators

The Innovators program gives developers and practitioners the latest updates, access to technologies and expertise, and exclusive benefits to build your skills on Google Cloud.

Sign up today!

- Add the Innovator's badge to your developer profile
- Join live discussions and AMAs
- Stay on top of the latest Google Cloud news right in your inbox
- Innovators digital badge and video conference background
- Receive 500 learning credits on us for a limited time

Cloud.google.com/innovators

THAT®

Google Cloud Innovators

Introduction to

TensorFlow

Today's Learning Objectives

01 Understand where ML fits in the field of AI and compare traditional ML models with neural network models

02 Learn about the concepts of Artificial Neural Networks: Layers, Perceptrons, Activation Functions

03 Tinker with Neural Networks in Tensorflow Playground

04 Running cloud-based neural network model Jupyter notebooks with Google Colab

05 Work through a neural network workflow with Keras

AI ML vs DL

Data Science

Know this game?

g.co/quickdraw

- How does it work?
- Could we program this?

How does ML work in "Quick, draw!"?

[Quick, Draw! The Data]

Rule-based approach

```
if object.height > 10:
  do x
if object.color is blue:
  do y
if object.numberOfLegs > 2:
  do z
...
```

Machine Learning

Frame the goal of the 'product'

↓

Train a model using examples

↓

Refine until goals are met

knowyourdata.withgoogle.com

Know Your Data BETA

DOCUMENTATION FAQ STAY UP TO DATE GITHUB

Know Your Data helps researchers, engineers, product teams, and decision makers **understand datasets** with the goal of **improving data quality**, and helping **mitigate fairness and bias issues.**

Explore Know Your Data

Explore 70+ ML datasets

Interactively explore image datasets supported by the TensorFlow Datasets API

Try it →

AI
ML
Artificial Intelligence
Machine Learning

Types of Machine Learning

Machine Learning Process

Kahoot!

ARTIFICIAL
NEURAL NETWORK

$\mathbf{a}_{l+1} = \sigma(W_l \mathbf{a}_l + b_l)$
Machine learning
Neural network

Neural networks are a set of algorithms that are modeled after the human brain. They're designed to recognize patterns and interpret sensory data through a kind of machine perception, labeling, or clustering raw inputs.

What Is a Neural Network?

The layers of neurons are connected and weighed against one another until the neurons reach the final, outer layer. The final layer returns a numerical or encoded categorical result.

The Structure of a Neural Network

An **input layer** of input values (transformed by weight coefficients)

The Structure of a Neural Network

A single **hidden layer** of neurons (single neuron or multiple neurons)

The Structure of a Neural Network

An **output layer** reports the classification or regression value

Neural Networks

Neural networks are particularly useful in data science because they serve multiple purposes:

01

One of the most popular uses for a neural network is as a classification algorithm to determine how to categorize an input.

02

Another popular use for a neural network is as a regression model to predict dependent outputs from independent input variables.

Kahoot!

X0
W0
X1
W1
X2
W2
Σ
Y
PERCEPTRON

A **Perceptron** is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn, and it processes elements in the training set one at a time.

Perceptron Model

The perceptron model has four major components

Input Values (shown as *x*'s)

The **input values**, which are typically labelled as χ or chi. Depending on how many features or variables exist in the dataset, the number of input values will change.

Weight Coefficients (denoted as *w*'s)

The **weight coefficients** are applied to each input value to help the machine learning model identify features of interest. The weight coefficients are typically labelled as w or omega.

A Constant Value Called Bias

Bias adjusts the model to make it either more or less sensitive to the weights.

The **bias term** is an additional input typically labelled as ω. The bias term helps to shift the output of the model, which may be necessary for properly training the model.

Net Summary Function (the Summation)

Summation aggregates all weighted inputs to feed into the activation function.

Perceptron, the Computational Neuron

The perceptron, also known as the **linear binary classifier**, is most commonly used to separate data into two groups.

Perceptron, the Computational Neuron

Consider linearly separable data: the perceptron algorithm separates and classifies the data into two groups using a linear equation

Since we provide the model with our input features and parameters, the perceptron model is a form of **supervised machine learning**.

Perceptron Example

Our model will try to classify values in a two-dimensional space; our perceptron model will use three inputs:

$\chi 1$	the *x* value
$\chi 2$	the *y* value
$\omega 0$	the bias constant

The end result of our two-dimensional perceptron model is the net sum function: $\omega 0 + \chi 1 \omega 1 + \chi 2 \omega 2$.

As with any untrained machine learning model, the weights and coefficients are arbitrary and oftentimes random.

Untrained Perceptron Model on Our Dataset

An untrained model almost classified the two groups perfectly.

It misclassified one blue circle.

Untrained Perceptron Model on our Dataset

The perceptron model will evaluate each data point and determine if the input weights should change.

If a data point is classified correctly, the weights will not change.

If a data point is misclassified, the weights will move the model closer to the missed data point.

Trained Perceptron Model on our Dataset

Each data point is evaluated to determine if the input weights should change.

Since a data point is misclassified, the weights will move the model closer to the missed data point.

Perceptron Model Training

As with other machine learning algorithms and models, perceptron model training continues until one of three conditions is met:

The perceptron model exceeds a predetermined performance threshold set by the designer before training. In machine learning, this is quantified by minimizing the loss metric.	For example, if we are working with noisy data that cannot be preprocessed or excluded, our model may not be able to exceed a certain level of performance without overfitting. Therefore, we would want to set a training cutoff at the point of model convergence.
The perceptron model training performs a set number of iterations determined by the designer before training.	For example, if we know roughly how many iterations it takes for a model to achieve desired performance, we can just "set it and forget it" to train over a specific interval.
The perceptron model is stopped or encounters an error during training.	This will typically be a hardware or power issue as long as our input data goes through cleaning and preprocessing before use. If we set up our model to save itself after a specific number of training iterations, we can resume training immediately.

A simple perceptron model is very similar to our basic statistical models. However, the power of the perceptron model comes from its ability to handle multidimensional data and interact with other perceptron models.

If each neuron has its own output, how does the neural network combine each neuron's output into the model's classification or regression output?

Activation Function (Transformation After Summation)

Neural networks use an **activation function** to transform the output of each neuron to a quantitative value.

Activation Functions Overview

There are a variety of activation functions that can be used for many specific purposes. However, most neural networks will use one of the following activation functions.

Linear Function

The **linear function** transforms the output into the coefficients of a linear model (the equation of a line).

Sigmoid Function

The **sigmoid function** transforms the neuron's output to a range between 0 and 1, which is especially useful for predicting probabilities. A neural network that uses the sigmoid function will output a model with a characteristic S-curve.

Tanh Function

The **tanh function** transforms the output to a range between −1 and 1.
The output for a model using a tanh function also forms a characteristic S-curve.
It's primary use is classifying data into one of two classes.

Rectified Linear Unit (ReLU)

The **rectified linear unit (ReLU)** function returns a value from 0 to infinity. This activation function transforms any negative input to 0. It is the most commonly used activation function in neural networks due to its faster learning and simplified output. However, it is not always appropriate for simpler models.

Leaky ReLU Function

The **leaky ReLU function** is an alternative to the ReLU function, which can sometimes work better. Instead of transforming negative input values to 0, it transforms them into much smaller negative values.

TensorFlow Playground

We are going to use the TensorFlow Playground to build our Neural Networks.

TensorFlow Playground is an application developed by TensorFlow as a teaching tool to “*demystify the black box*” of neural networks.

TensorFlow Playground

The application provides a working simulation of a neural network as it trains on a variety of different datasets and conditions.

Time to Code

Neural Networks: Playground Exercises

https://bit.ly/3POMLmy

Suggested Time:

30 minutes

Google Colab

Google Colab

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with:

- Zero configuration required
- Free access to GPUs
- Easy sharing

Google Colab

Advantages of Google Colab

01

Google Colab is a cloud-based notebook, different from Jupyter Notebook, which runs locally on our machine.

02

Cloud-based notebooks are user friendly and won’t require any type of module installation.

03

TensorFlow was also developed by Google, which makes running in Colab fairly seamless.

Time to Code

101 Exercises for Python Fundamentals

https://bit.ly/44Rv7TT

Suggested Time:

30 minutes

Understanding the TensorFlow Neural Network Structure

There are a number of smaller modules within the TensorFlow 2.0 library that make it even easier to build machine learning models.

For our purposes, we'll use the Keras module to help build our neural networks

Keras

Layers in a Neural Network explained
Watch later
Share
deep learning
Watch on YouTube

Keras contains multiple classes and objects that can be combined to design a variety of neural network types.

Keras

For a basic neural network, we will use two Keras classes:

Sequential class

The Sequential class is a linear stack of neural network layers where data flows from one layer to the next. This class is what we simulated in the TensorFlow Playground.

Dense class

The generalized Dense class allows us to add layers within the neural network.

Keras

For the **Sequential model**, we will add multiple Dense layers that will act as our input, hidden, and output layers.

For each **Dense layer**, we'll define the number of neurons and activation functions.

Keras

Finally, once we have completed the Sequential model design, we will apply the scikit-learn model -> fit -> predict/transform workflow

Time to Code

Work through a Neural Network Workflow

https://bit.ly/3pJw6py

Suggested Time:

30 minutes

Debrief on Neural Networks

Debrief on Neural Networks

Regardless of what machine learning model or technology we use, we follow the same general workflow:

01 Decide on a model and create a model instance.

02 Split into training and testing sets.

03 Preprocess the data

04 Train/fit the training data to the model.

05 Evaluate the model for predictions and transformations.

Debrief on Neural Networks

When creating a neural network, we use two Keras classes:

Sequential class

`Sequential` is a linear stack of neural network layers where data flows from one layer to the next.

Dense class

`Dense` allows us to add layers within the neural network.

In the dense class, we add *at least one* input layer where we define the activation function, and we add an output layer that uses a probability activation function.

Dense Class

Then, we:

- Create a summary to get the structure of the `Sequential` model.
- Compile the `Sequential` model.
- Fit the model to the training data with a defined number of epochs.
- Evaluate the model using the test data.

Questions?

Session Feedback

https://that.land/43k4pkQ