From Tinkering to Engineering: Measurements in Tensorflow Playground

Axel Harstad
University of California, Berkeley
and NTNU
axeloh@berkeley.edu

Henrik Hoeiness
University of California, Berkeley
and NTNU
henrhoi@berkeley.edu

Gerald Friedland
University of California, Berkeley
fractor@eecs.berkeley.edu

***Abstract*—In this article, we present an extension of the Tensorflow Playground [7], called Tensorflow Meter (short TFMeter). TFMeter is an interactive neural network architecting tool that allows the visual creation of different architecture of neural networks. In addition to its ancestor, the playground, our tool shows information-theoretic measurements while constructing, training, and testing the network. As a result, each change results in a change in at least one of the measurements, providing for a better engineering intuition of what different architectures are able to learn. The measurements are derived from various places in the literature, including recent work by [4]. In this paper, we describe our web application that is available online[8] and argue that in the same way that the original Playground is meant to build an intuition about neural networks, our extension educates users on available measurements, which we hope will ultimately improve experimental design and reproducibility in the field.**

I. Introduction

When performing neural network experiments many details need to be addressed, such as the number of neurons, network topology, activation function, and secondary parameters like learning rate or regularization. Today, the most common way to find these parameters seems to be through a "guess and check" approach. While this can work fine anecdotally, it is certainly not ideal, as parameter search can take exponential time, burns billions of computing cycles, may lead to oversized networks where energy is wasted during both training and testing, and, if the experiments do not work, it is back to the drawing board. As a result, there is currently a movement to improve the reproducibility of machine learning experiments through engineering measurements.

In accordance with this idea, we present our extension to the Tensorflow Playground [7], [6], which we call TFMeter (see Figure 1. In our extension, all functionality of the original Tensorflow Playground is preserved but we introduce information-theoretic measurements, such as Memory Equivalent Capacity and bitwise generalization. These measures were presented in various places in the literature, such as [5], [2], [4], [3]. In addition, TFMeter features the ability to manually remove network links, and add direct connections (to simulate a Residual Network). Furthermore, it is possible to run the architected network on one's own dataset. We also indicate bias by measuring class balance and comparing it to the accuracy for each class. This gives the user a flexible platform where one can obtain an intuition for neural networks and deep learning.

II. Measurements

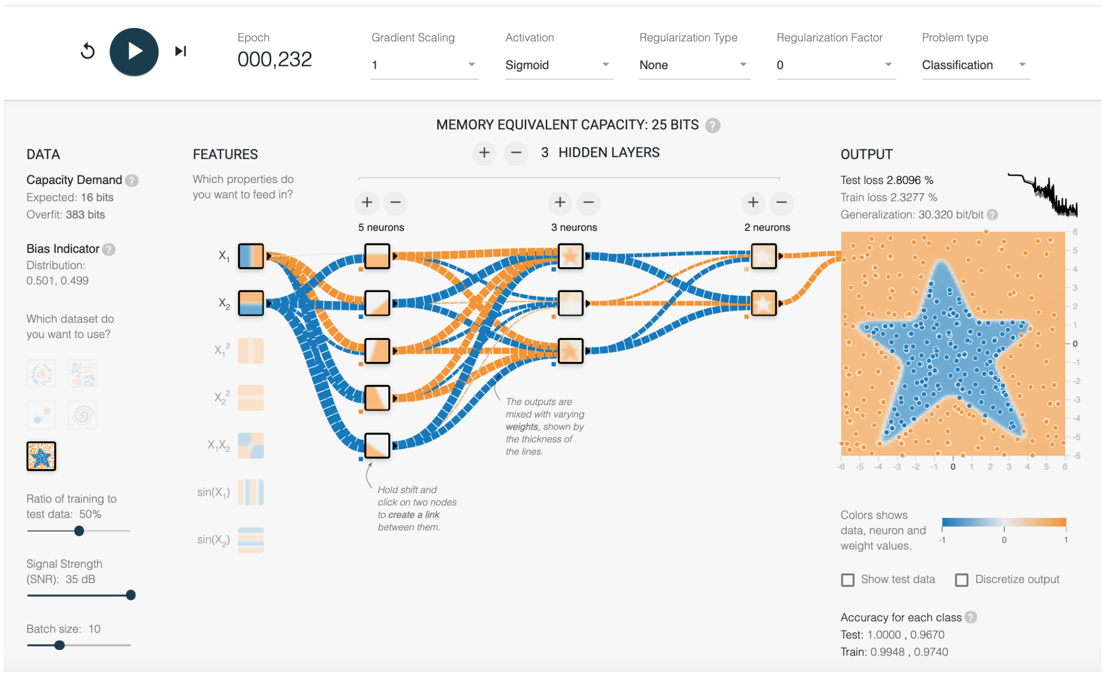
The capacity of a Neural Network is defined as the number of functions a network can implement. TFMeter [8], just like its ancestor, only supports binary classification. This allows us to adopt the definition of capacity presented in [2], [5], where individual neurons are treated as electrical elements and thus a network's memorization capacity (number of points it can overfit) can be calculated using circuit rules [4]. For a given neural network architecture, we use these engineering rules to compute the memorization capacity of a network or Memory Equivalent Capacity (MEC). In TFMeter, the MEC is shown just above the corresponding network, see Figure 1.

MEC allows us to compare different network architectures independent of a task (dataset and labels). For example, consider the two networks in Figure 2. The first network has more neurons in the subsequent layers than the second network, but their MEC is the same. This indicates that the second network is more efficient than the first, as using the first network would result in wasting memory and computation resources. Furthermore, as the second network has less parameters, it should have a better chance to avoid overfitting [4].

We have seen that MEC allows the comparison of different architectures independent of the task. However, sizing a network properly to a task requires an estimate of the required capacity. [3] proposes a heuristic method to estimate the required neural network capacity for any given dataset and binary labeling. Experimental results for various datasets show that it works well for estimating the expected required capacity. Therefore, we use this heuristic method to estimate the expected capacity demand for a given dataset. If features are selected, TFMeter takes that into account. The expected capacity demand is shown in the top-left corner, see Figure 1.

The expected capacity requirement of a dataset is useful because it provides an estimate for how large the network needs to be. For example, in Figure 3, we used two different networks for the same binary classification task. The expected required capacity for this dataset was 12 bits. The first network had a MEC equal to the expected required capacity, while the second network had a MEC of 35 bits. Both networks reached 100 % test accuracy. That is, the larger network is oversized. This relation between network capacity and performance is quantifiable and is called generalization.

TFMeter [8] uses the measurement of generalization defined in [4], which is $G = \frac{\#\ correctly\ predicted\ instances}{Memory\ Equivalent\ Capacity}$.



Epoch 000,232

Gradient Scaling: 1

Activation: Sigmoid

Regularization Type: None

Regularization Factor: 0

Problem type: Classification

MEMORY EQUIVALENT CAPACITY: 25 BITS

DATA

Capacity Demand
Expected: 16 bits
Overfit: 383 bits

Bias Indicator
Distribution:
0.501, 0.499

Which dataset do you want to use?

Ratio of training to test data: 50%

Signal Strength (SNR): 35 dB

Batch size: 10

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

5 neurons · 3 neurons · 2 neurons

The outputs are mixed with varying weights, shown by the thickness of the lines.

Hold shift and click on two nodes to create a link between them.

OUTPUT

Test loss 2.8096 %
Train loss 2.3277 %
Generalization: 30.320 bit/bit

Colors shows data, neuron and weight values.

Show test data · Discretize output

Accuracy for each class
Test: 1.0000 , 0.9670
Train: 0.9948 , 0.9740

Fig. 1: TFMeter [8] in action. This network is classifying data from a dataset uploaded by the user. Neurons biases and weights are visualized with heatmaps and boldness of connections.The top-right corner shows a graph plotting the loss over time along with the network's generalization ratio. The network's Memory Equivalent Capacity is found in the top-middle section, and the dataset's expected capacity demand in the top-left corner, along with a bias indicator. Classification accuracies for each class are shown bottom-right.

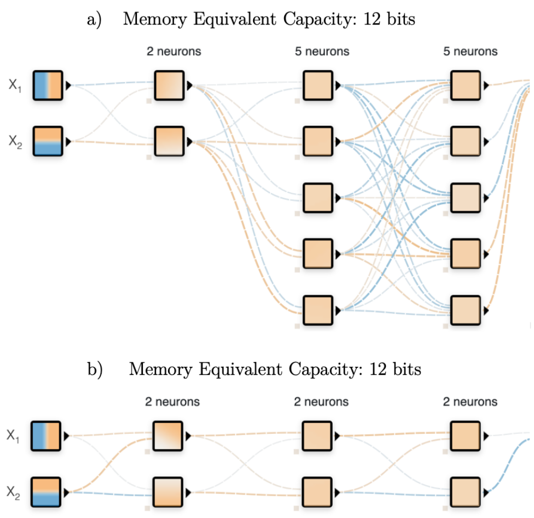

a) Memory Equivalent Capacity: 12 bits
2 neurons
5 neurons
5 neurons
X1
X2
b) Memory Equivalent Capacity: 12 bits
2 neurons
2 neurons
2 neurons
X1
X2

Fig. 2: Two networks illustrating Memory Equivalent Capacity: In both networks, the second layer is dependent on the first hidden layer. The greater number of neurons in a) makes no difference to the overall capacity of the network, so both networks end up with equal capacity.

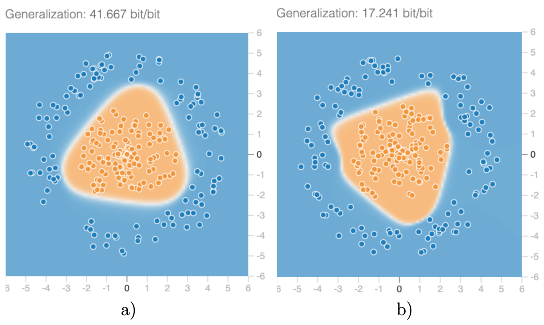


Fig. 3: Classifications of two networks. The Memory Equivalent Capacity of network a) is set to equal the dataset's expected capacity demand and b) is above it. Network a) yields a generalization ratio of $G = 41.67$, while b) yields $G = 17.24$. Both networks reach $100\,\%$ accuracy.

A machine learner with generalization $G \leq 1$ implies no generalization, while $G > 1$ implies successful generalization.

The machine learner with the lowest capacity and highest accuracy is the one that uses the representation function(s) most effectively. Therefore, it has the lowest chance of failing when applied to unseen data. This is consistent with Occam's razor [1], which dictates that for two competing correct hypotheses one should follow the one with the least assumptions.

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K.Warmuth, *Occam's razor*, Information processing letters 24: 377–380, 1987

[2] T. M. Cover, *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, 1965.

[3] G. Friedland, *Reproducibility and Experimental Design for Machine Learning on Audio and Multimedia Data*, MM '19: Proceedings of the 27th ACM International Conference on Multimedia, pp. 2709–2710, October 2019.

[4] G. Friedland, and M. Krell, *A Capacity Scaling Law for Artificial Neural Networks*, 2018

[5] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, 2003.

[6] D. Smilkov and S. Carter, *TensorFlow Playground, http://playground.tensorflow.org/*, 2017

[7] D. Smilkov, S. Carter, D. Sculley, F. Viégas, M. Wattenberg, *Direct-Manipulation Visualization of Deep Networks*, 2017.

[8] Tensorflow Meter, *http://tfmeter.icsi.berkeley.edu*, March 16th, 2020.