Using Autoencoders to Detect Errors in Telemetry Events

Andrew Younger

April 15, 2022

Contents

1 Introduction

Telemetry events are essential for accurate reporting on player activity and progress for modern games. They allow the analytics team to dive deeply into topics and analyze important questions from stakeholders in the project. These events can be quite expansive, detailing anything from how a player kills an enemy to what cosmetics a player applies to their character to how quickly players are able to complete the game.

During the development cycle of a game there are frequent code changes which can affect telemetry frequently both accidentally and by design. QC teams work diligently in order to keep up with testing events as they are created but changes that affect older events often produce bugs which impact reporting done by the analytics team for playtests. In addition, telemetry events can be quite large and have many attributes with many different possibilities. A player.kill event, for example, where there are 50 weapons and 50 enemy types tracked yiels 2500 possibilities in just two columns. This further exacerbates the issues that the QC teams face when trying to catch every erroneous scenario that occurs inside telemetry events leading to the questions of: "How do we determine when an event has bugs?" and "How do we know when QC teams should retest old events?".

An immediate proposition to answer these questions is to use some form of automated bug detection. There are two methods of bug detection which could be helpful for QC teams to track down and deal with bugs: checking for some sort of spike in event frequency (if events start transmitting too often or stop transmitting then there may be an issue with the event's trigger mechanism) and statistsically determining if event attribute A is supposed to match with event attribute B. This report outlines a process for the second method by using an autoencoder to learn the possibility space of an event.

1.1 Autoencoders

Autoencoders are type of neural network that fall into the *encoder-decoder* family of neural networks. The general process for an autoencoder is shown in Figure **??**. The encoding section of an autoencoder learns to "compress" the data down into a smaller dimension so that the decoding section of the autoencoder can attempt to recreate the original data from this low-level representation of the data. Autoencoders are used for several purposes including: image denoising, facial recognition, and, importantly for finding bugs, anomaly detection.

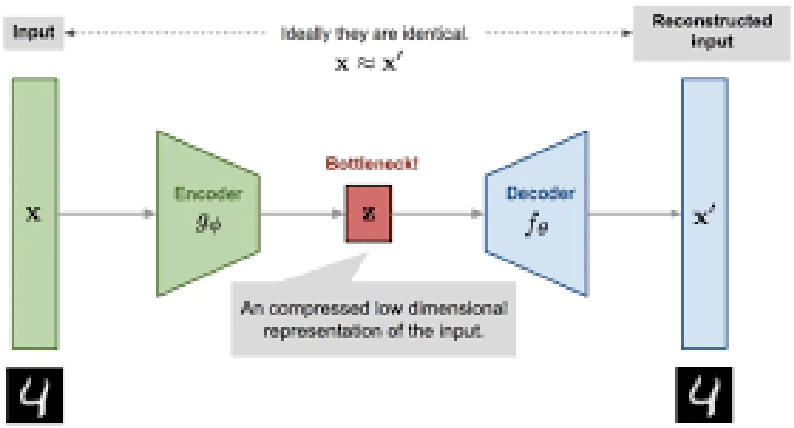

Figure 1: The general layout of an autoencoder network. The network works by learning efficient compression to a low dimension and then reconstructing the data as closely as possible [**?**].

Using an autoencoder to detect anomalies in data is very straightforward. Test

References

[1] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor, "Recursive Autoencoders for Document Layout Generation" `arXiv:1909.00302v4 [cs.CV]`