Deep Learning by Design

From Zero to Automatic Learning with Tensorflow 2.0

Andrew Ferlitsch, Google Cloud AI

Version January 2020

Topics

Preface

As a Googler, one of my duties is to educate software engineers on how to use machine learning. I already had experience creating online tutorials, meetups, conference presentations, training workshops, and coursework for private coding schools and university graduate studies, but I am always looking for new ways to effectively teach.

Welcome to my latest approach, the idiomatic programmer. My audience are software engineers, machine learning engineers and junior to mid-level data scientists. While for the later, one may assume that the initial chapters would seem redundant - but with my unique approach you will likely find additional insight as a welcomed refresher.

You should know at least the basics of Python. It's okay if you still struggle with what is a compression, what is a generator; you still have some confusion with the weird multi-dimensional array slicing, and this thing about which objects are mutable and non-mutable on the heap. For this tutorial it's okay.

For those software engineers wanting to become a machine learning engineer -- What does that mean? A machine learning engineer (MLE) is an applied engineer. You don't need to know statistics (really you don't!), you don't need to know computational theory. If you fell asleep in your college calculus class on what a derivative is, that's okay, and if somebody asks you to do a matrix multiplication, feel free to ask, “why?”

Your job is to learn the knobs and levers of a framework, and apply your skills and experience to produce solutions for real world problems. That's what I am going to help you with and that's what the composable design pattern using TF.Keras is about.

Tensorflow (TF) is a low-level graph based (symbolic programming) framework, while Keras was an abstraction on top of Tensorflow as a high-level abstraction (imperative programming), which is now fused into Tensorflow 2.0. Composable is an abstraction on top of an abstraction (TF.Keras).

Composable moves beyond building a model to building amalgamations of models that are entire applications for real world production.

The Machine Learning Steps

You've likely seen this before. A successful ML engineer will need to decompose a machine learning solution into the following steps:

1. Identify the Type of Model for the Problem
2. Design the Model
3. Prepare the Data for the Model
4. Train the Model
5. Deploy the Model

That is very 2017. It's now 2020 and a lot of things have progressed. There are now vasts numbers of model types, but with abstractions -- like composable -- we are seeing model types converging and likely by 2022 we will see just a handful of abstractions covering all types of production.

Outside of research, ML practitioners don't design models, they guide the design of the models. Data preparation is becoming more automated, some of which is moving into the models and other cases handled upstream by other models that have learned to prepare data.

We don't train one model anymore, we train a plurality of models, and for each model we train instances of the model in stages from warmup, pre-training and finally full-training.

Models are deployed in a wide variety of manners from the cloud, to mobile devices, to IoT (edge) devices. The deployment may modify the model (e.g., quantization, compression), perform continuous evaluation and be updated and versioned with continuous integration/continuous development (CI/CD).

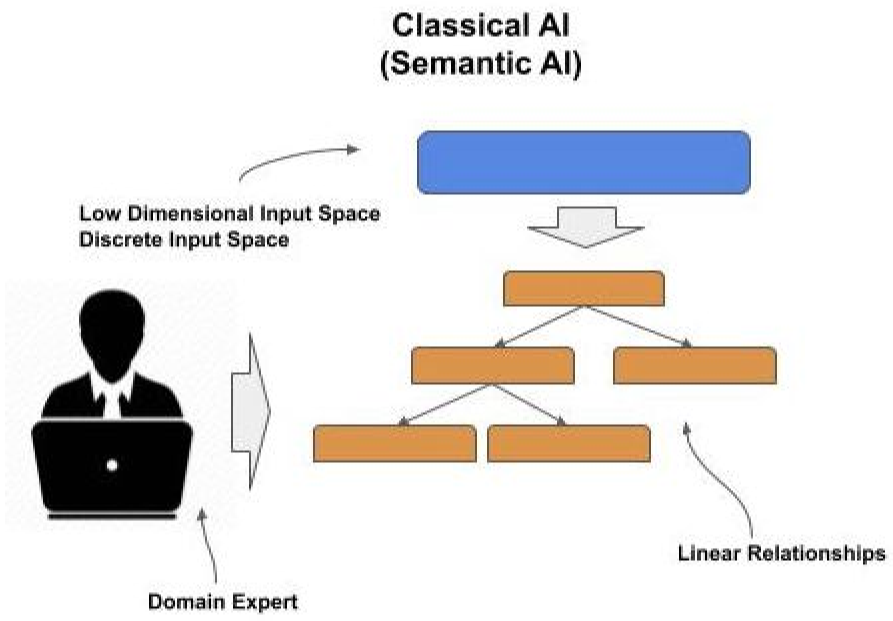
And the above list does not cover other new things we see in production, such as QA, A/B testing, auditing and bias adjustments.

Everyone is specializing, and likely you will too, pick a speciality that fits your skills and passion.

Welcome to AI in 2020.

Classical vs Narrow AI

Let’s briefly cover the difference between classical AI (also known as semantic AI) and today’s modern narrow AI (also known as statistical AI). In classical AI, models were designed as rule-based systems. These systems were used to solve problems that could not be solved by a mathematical equation. Instead the system is designed to mimic a subject matter expert (also known as domain expert).

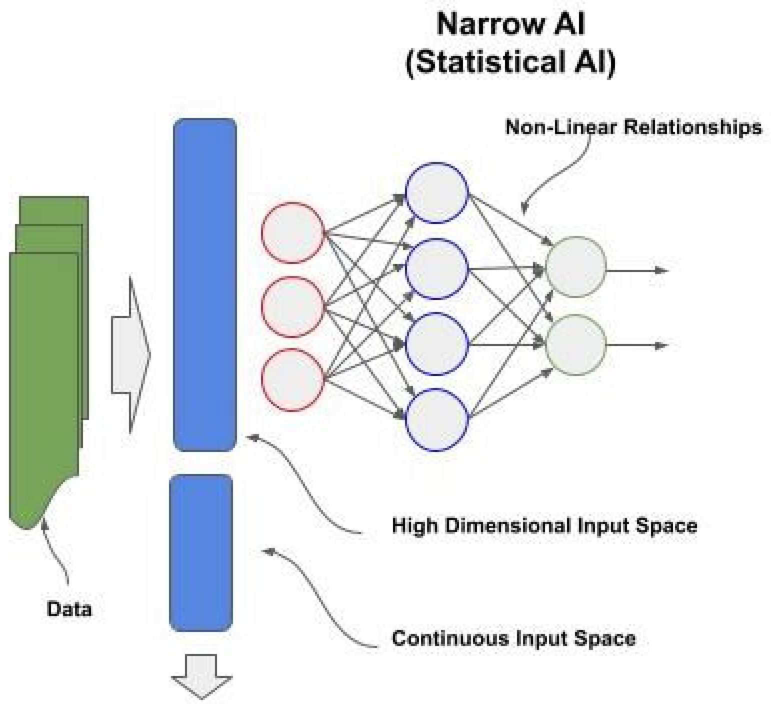


Classical AI works well in input spaces that are low-dimensionality (i.e., low number of distinct inputs), the input space can be broken into discrete segments (e.g., categorical, bins), and there is a strong linear relationship between the discrete space and the output. Systems like this were well suited for problems like predicting the quality of a wine. But they failed to scale to larger problems, where accuracy would drop dramatically and need for continuous refinement of the rules; as well as inconsistencies between domain experts in designing the rules.

In narrow AI, one trains a model on a large amount of data, alleviating the need for a domain expert. A model is constructed using principles of statistics for sampling distributions to learn patterns in the distribution that can then be applied with high accuracy to samples not seen by training (i.e.., population distribution). When trained with large amounts of data which would be representative (sampling distribution) of the population distribution, they can model problems that have substantially higher dimensionality in the input space (i.e., large number of distinct inputs), and inputs which can be a mix of discrete and continuous.

These models work well with input space that has a high level of non-linearity to the outputs (i.e., predictions), by learning the boundaries to segment up the input space, where within the segments there is a high level of linear relationship to the output. Finally, these types of models based on statistics and large amounts of data are referred to as narrow AI in that they are good at solving narrow problems, but is still challenging to generalize them to problems of a wide scope.

In addition to covering deep learning for narrow AI, in later chapters we will cover today's technique of generalizing models as we move into an era of pre-AGI (artificial general intelligence).


Narrow AI
(Statistical AI)
Non-Linear Relationships
High Dimensional Input Space
Continuous Input Space
Data