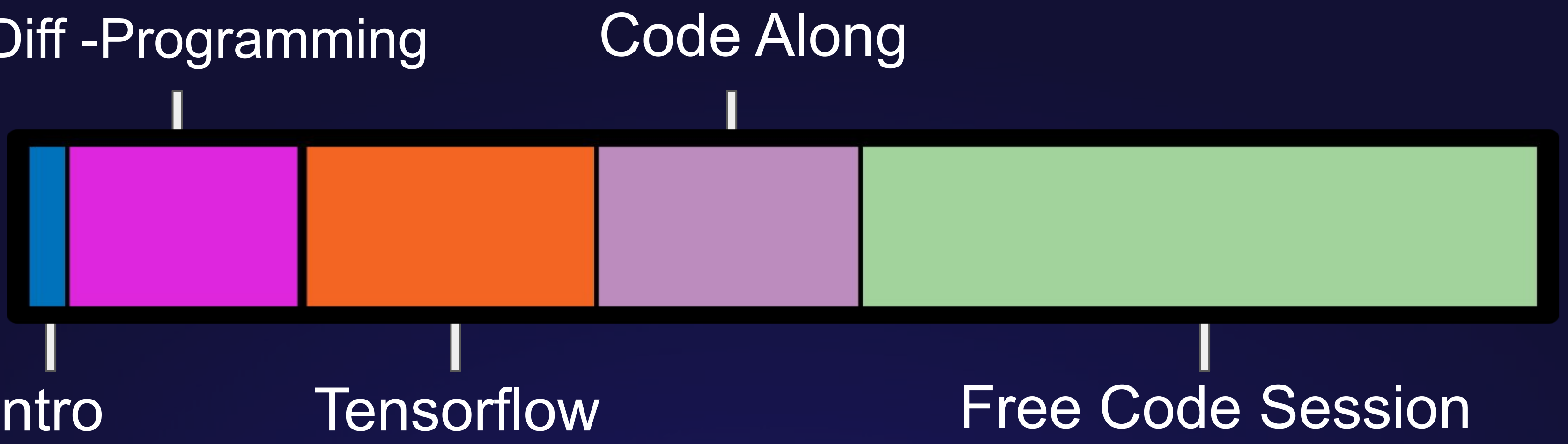
Differentiable Programming With

TensorFlow 2.0

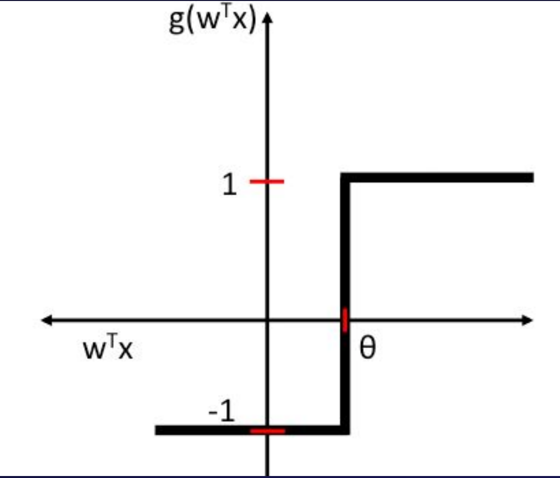
Agenda


Diff -Programming
Code Along
Intro
Tensorflow
Free Code Session

Differentiable Programming

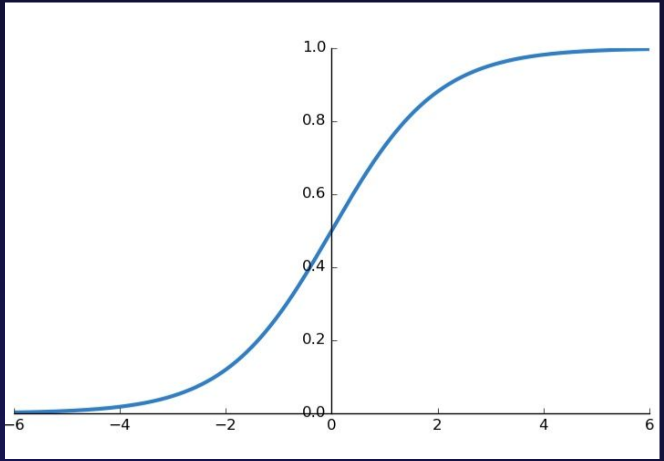
TensorFlow

Discrete



VS

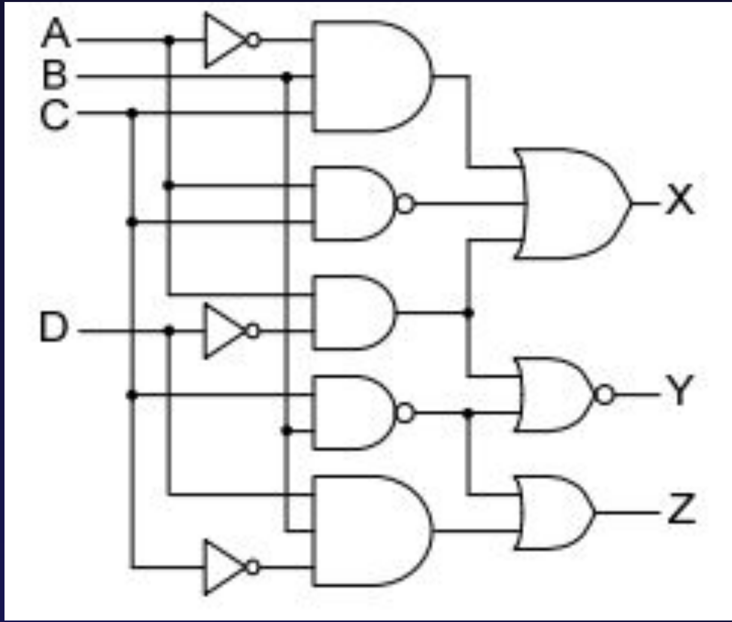
Continuous



Discrete Circuits

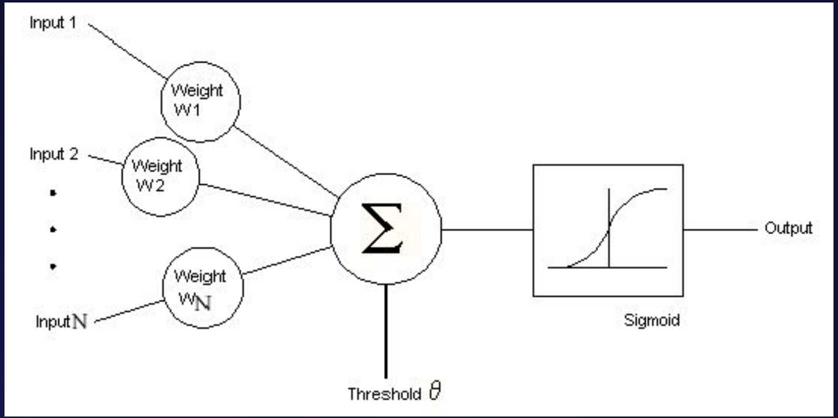
TensorFlow

- Discrete circuits are NOT differentiable
- “Butterfly effect”: Minor weight adjustments can have major output ramifications
- Early Neural Networks used discrete functions, but were hard to train effectively



Continuous Circuits

- Allows for differentiation
- Weight adjustments yield foreseeable changes
- Current training algorithms for continuous circuits are orders of magnitude faster than for discrete circuits



Differentiable Programming

TensorFlow

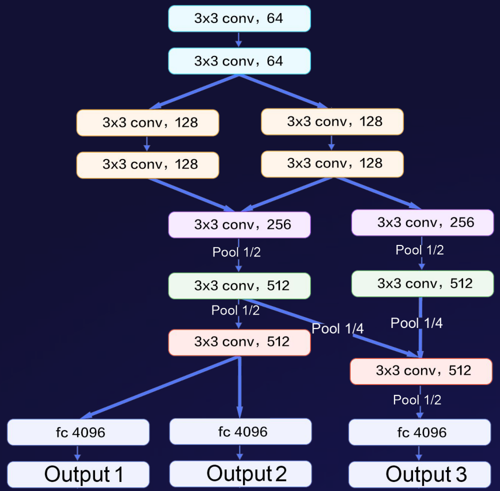
Surprising amount of inherently discrete tasks can be approximately differentiated, for example:

Searching and selecting files from a file storage

Selecting what move to play in chess

TensorFlow

Differentiable Programming



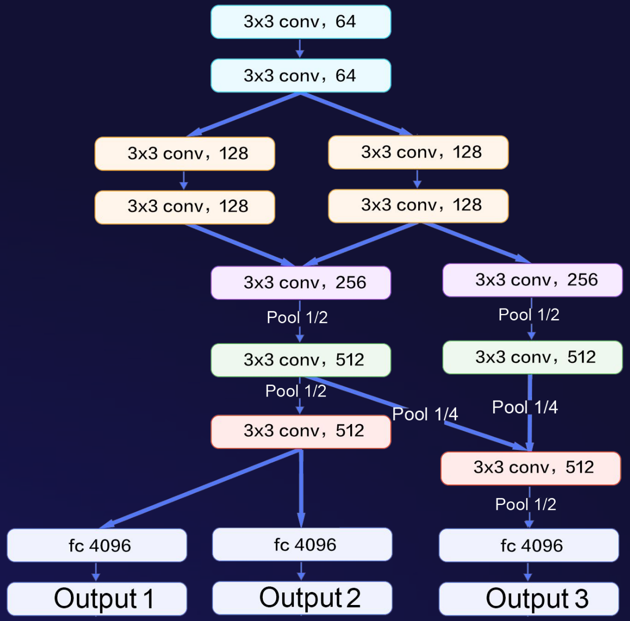
&

Model Architecture

Loss Function

Model Architecture

- Determines the expressibility of the function approximation
- Training and inference speed can vary widely between different model architectures
- Vast amount of different model possibilities can lead to protracted hyperparameter search



Loss Function

- The loss function acts as the learning target
- The loss must be defined so that a function that minimizes the loss also solves the desired problem
- A clever loss function is worth much more than a clever model architecture

TensorFlow

TensorFlow 2.0

TensorFlow

TensorFlow + Keras + PYTORCH

TensorFlow 2.0

Tensorflow

TensorFlow

Inference Engine

- Heavy Optimization
- Auto Differentiation
- Distribution Strategies
- Multi-Platform Support
- *etc...*

Library

- Optimizers
- Layers
- Activations
- Standard Datasets
- Pre-trained Models
- *etc...*

TensorFlow

Inference Engine

Eager Execution

- Express computations in pure Python
- Integrates nicely with your dynamic data structures
- Great for debugging and experimentation

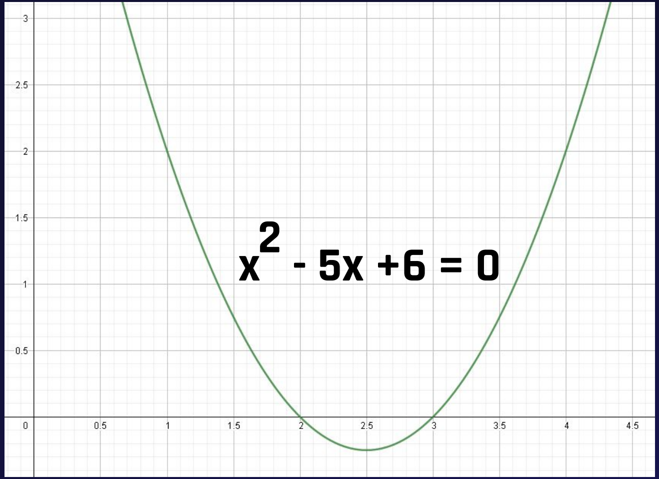
VS

Declarative Graphs

- Pre-define computations in form of a graph
- Allows for heavy optimizations
- Platform independent model structure

Eager Execution Example

TensorFlow



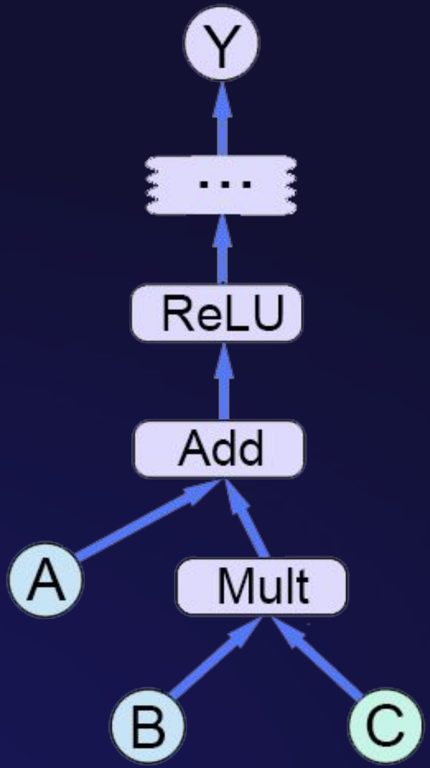

```python
def func(x):
  return x**2 - 5*x + 6

x = tf.Variable(1.0)
with tf.GradientTape() as tape:
  loss = tf.abs(func(x))

grad = tape.gradient(loss, x)
print(grad.numpy(), x.numpy())
```

Computation Graphs

- Predefine computation in the form of graph
- Gives compiler apriori information, allowing for optimization: *common subexpression elimination, constant folding, etc...*
- Hardware agnostic, allowing for easy deployment
- Intuitive for large models



TensorFlow

Graph API Overview

Keras

Easy to use

Sequential

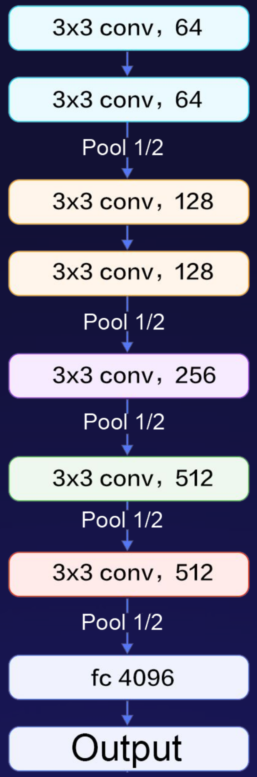
Functional

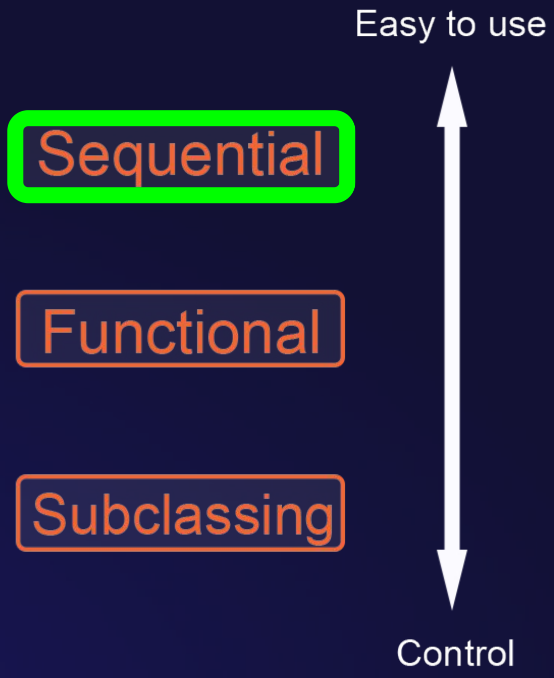
Subclassing

Control

Graph API Overview - Sequential

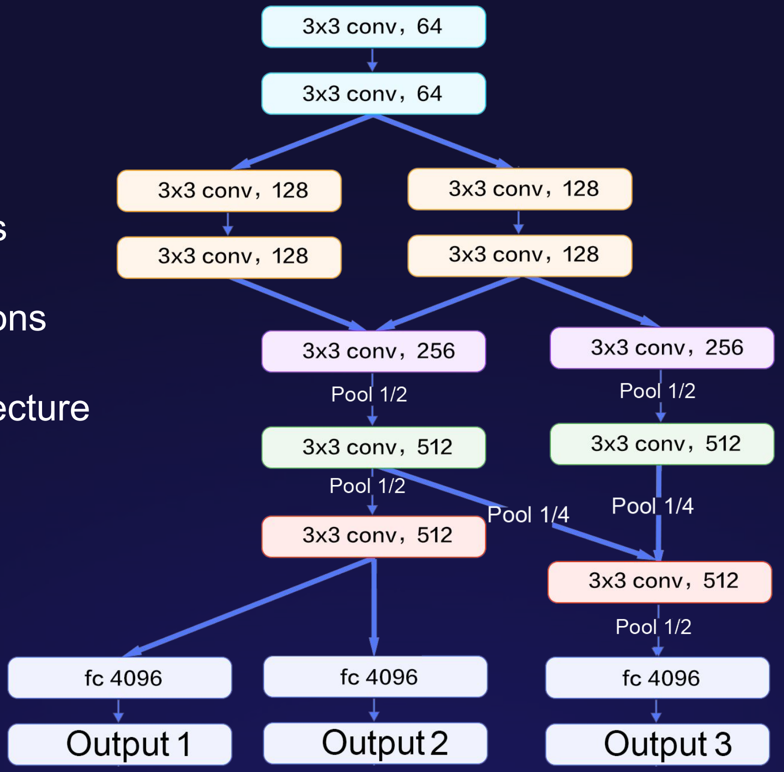
- Define sequentially stacked models
- Minimal code
- Great Overview

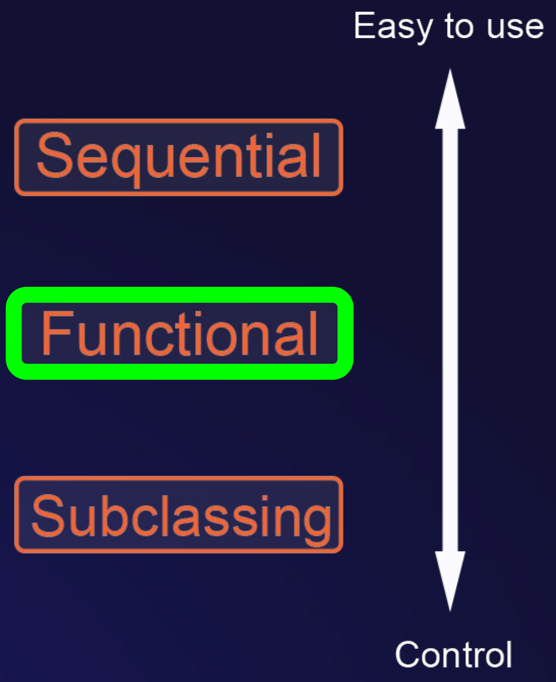




Graph API Overview - Functional

- Non Sequential Models
- Layer-based Connections
- Good for simple architecture experimentation



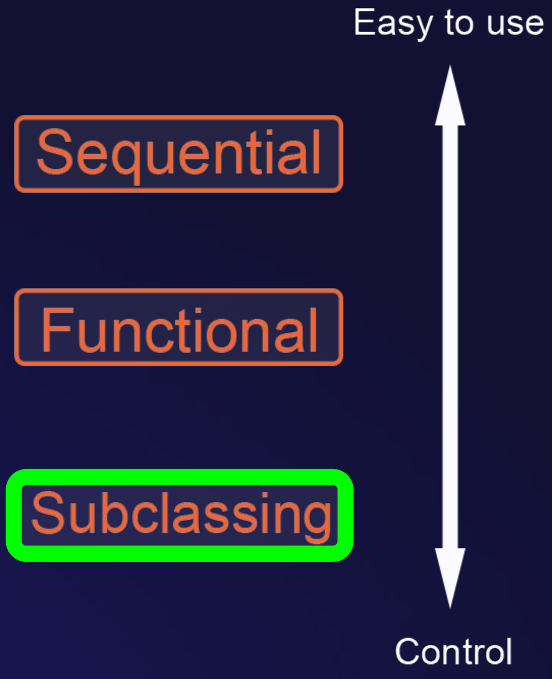


Graph API Overview - Subclassing

- Write everything from scratch
- Custom Optimizers
- Custom Losses
- Custom Activations

```python
class MyModel(tf.keras.Model):
  def __init__(self, num_classes=10):
    super(MyModel, self).__init__(name='my_model')
    self.dense_1 = layers.Dense(32, activation='relu')
    self.dense_2 = layers.Dense(num_classes,activation='softmax')

  def call(self, inputs):
    # Define your forward pass here
    x = self.dense_1(inputs)
    return self.dense_2(x)
```



TensorFlow

Code Session

Banknote Authentication Classification

- Binary classification problem
- 1372 data samples with 4 features
- *** Download Link ***

Words of Wisdom