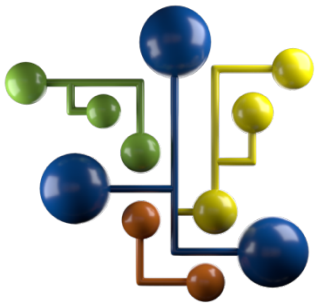

Data Science Academy

www.datascienceacademy.com.br

Deep Learning Frameworks

TensorFlow 1 x TensorFlow 2

O TensorFlow 2.0 é um grande salto em relação ao TensorFlow 1.0. Aqui estão as principais diferenças:

1. **Facilidade de uso**: Muitas bibliotecas antigas (exemplo tf.contrib) foram removidas e algumas consolidadas. Por exemplo, no TensorFlow1.x, o modelo pode ser criado usando Contrib, Layers, Keras ou Estimators; muitas opções para a mesma tarefa confundem novos usuários. O TensorFlow 2.0 promove o TensorFlow Keras para experimentação de modelos e Estimators para servir os modelos em escala, e as duas APIs são muito convenientes de usar.

2. **Eager Execution** ("Execução Ansiosa"): No TensorFlow 1.x o desenvolvimento do código é dividido em duas partes: construção do grafo computacional e posteriormente criação de uma sessão para executá-lo. Isso era um pouco complicado, especialmente se no grande modelo que você projetou, um pequeno erro existia em algum lugar no começo. O TensorFlow 2.0 Eager Execution é implementado por padrão, ou seja, você não precisa mais criar uma sessão para executar o grafo computacional, e pode ver o resultado do seu código diretamente, sem a necessidade de criar a Sessão. No TF 2 a programação é similar ao que você faz em Python.

3. **Construção e Deploy de Modelos Facilmente**: Com o TensorFlow 2.0 fornecendo API de alto nível do TensorFlow Keras, o usuário tem uma maior flexibilidade na criação do modelo. Pode-se definir o modelo usando a API funcional ou sequencial do Keras. A API do TensorFlow Estimator permite executar o modelo em um host local ou em um ambiente multi-servidor distribuído sem alterar seu modelo. Os grafos computacionais são poderosos em termos de desempenho. No TensorFlow 2.0, você pode usar o *decorator* tf.function para que o bloco seguinte de funções seja executado como um único grafo. Isso é feito através do poderoso recurso chamado Autograph no TensorFlow 2.0. Isso permite que os usuários otimizem a função e aumentem a portabilidade. E a melhor parte, você pode escrever a função usando a sintaxe Python natural. Efetivamente, você pode usar o decorator tf.function para transformar código Python simples em grafo. Enquanto o decorator @tf.function se aplica ao bloco de funções imediatamente a seguir, quaisquer funções chamadas por ele também serão executadas no modo grafo. Assim, no TensorFlow 2.0, os usuários devem refatorar seu código em funções menores, chamadas conforme necessário. Em geral, não é necessário decorar

cada uma dessas funções menores com tf.function; use apenas tf.function para decorar cálculos de alto nível - por exemplo, uma etapa do treinamento ou a passagem direta do seu modelo.

Para expandir essa ideia, no TensorFlow 1.x, precisamos construir o grafo computacional. O TensorFlow 2.0 não cria grafo por padrão. No entanto, como todo Engenheiro de Machine Learning sabe, os grafos são bons para velocidade. O TensorFlow 2.0 fornece ao usuário a criação de um grafo que pode ser chamado usando uma função python @tf.function. A função tf.function() criará um grafo separado para cada conjunto exclusivo de formas e tipos de dados de entrada. No exemplo abaixo, teremos três grafos separados criados, um para cada tipo de dados de entrada.

```
@tf.function
def f(x): return tf.add(x, 1.)

scalar = tf.constant(1.0)
vector = tf.constant([1.0, 1.0])
matrix = tf.constant([[3.0]])

print(f(scalar))
print(f(vector))
print(f(matrix))
```

4. **Pipeline de Dados Simplificado**: O TensorFlow 2.0 possui um módulo separado chamado TensorFlow DataSets que pode ser usado para operar com o modelo de maneira mais elegante. Não apenas possui uma grande variedade de conjuntos de dados existentes, facilitando o trabalho de experimentar uma nova arquitetura - também possui uma maneira bem definida de adicionar seus dados a ela.

5. **Não Precisamos Mais de Placehoders**: No TensorFlow 1.x para a construção de um modelo, precisamos primeiro declarar espaços reservados (placeholders). Essas são as variáveis fictícias que mais tarde (na sessão) serão usadas para alimentar os dados no modelo. O TF 2 não tem mais placeholders.

6. **Criação do Modelo**: Havia muitas APIs internas para a construção de camadas como tf.contrib, tf.layers e tf.keras, e também era possível construir camadas definindo as operações matemáticas reais. No TensorFlow 2.0, você pode criar seu modelo definindo suas próprias operações matemáticas, pois antes você pode usar o módulo de matemática (tf.math) e o módulo de álgebra linear (tf.linalg). No entanto, você pode tirar proveito do módulo Keras API e tf.layers de alto nível. A parte importante é que não precisamos mais definir espaços reservados.

O TensorFlow 2.0 traz muitas melhorias, mas ainda tem muitas limitações. Diversos algoritmos e operações avançadas ainda não foram implementadas.

Como o TensorFlow 1.x ainda está disponível e tem suporte do Google, dependendo da operação talvez tenhamos que usar essa versão para resolver determinados problemas.