TensorFlow, Keras

Gustavo Gomides

CE229

Sumário

Introdução

Figura 1: TensorFlow, Keras

Fonte: medium.com [1]

Inteligência Artificial

“O estudo das faculdades mentais pelo uso de modelos computacionais.” (Charniak e McDermott, 1985) [2]

“A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas.” (Kurzweil, 1990) [3]

“Inteligência Computacional é o estudo do projeto de agentes inteligentes.” (Poole et al., 1998) [4]

Redes Neurais Artificiais

“As redes neurais artificiais representam uma tecnologia que tem raízes em muitas disciplinas: neurociências, matemática, estatística, física, ciência da computação e engenharia.” [5]

“Uma rede neural pode ser considerada como uma técnica de processamento de dados que mapeia, ou relaciona, algum tipo de fluxo de entrada de informações para um fluxo de saída de dados.” [6]

Processos de Aprendizados

- **Supervisionado**:
  - agente externo indica a resposta desejada para o padrão de entrada

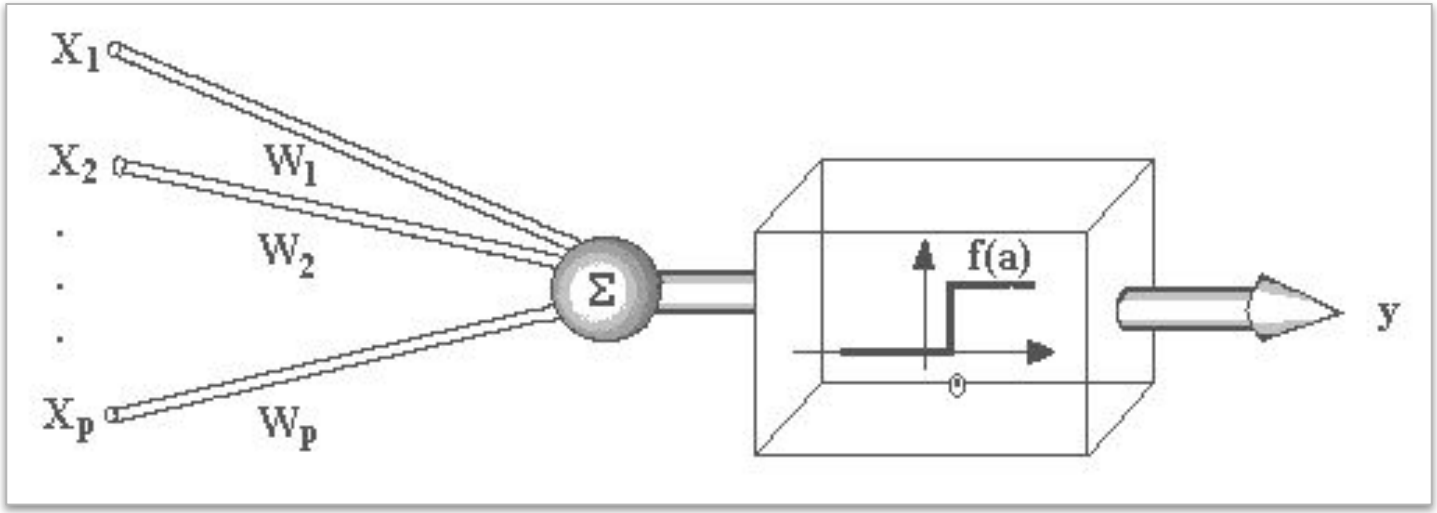
- **Não Supervisionado**:
  - quando não existe um agente externo (auto-organização)

- **Reforço**:
  - quando um crítico externo avalia a saída da rede [7]

Neurônio de McCulloch e Pitts

- Primeiro modelo de neurônio artificial desenvolvido - 1943

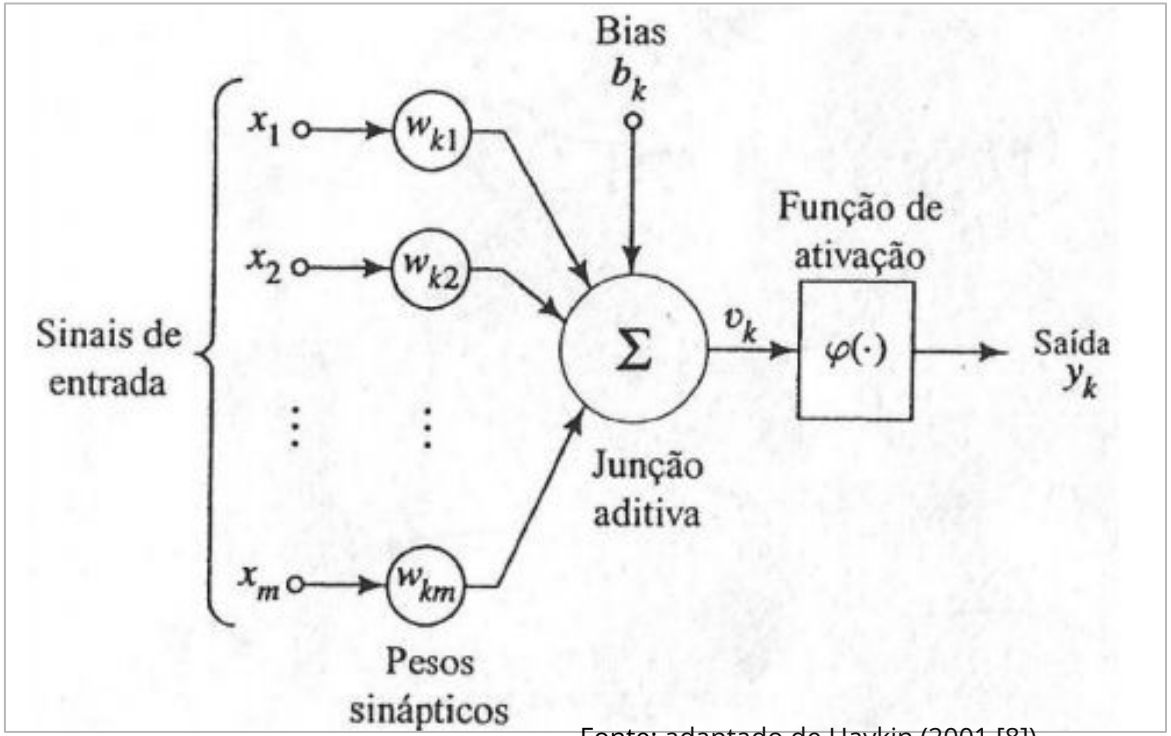
Figura 2: Neurônio de McCulloch e Pitts



Fonte: conteudo.icmc.usp.br [7]

Neurônio não-linear

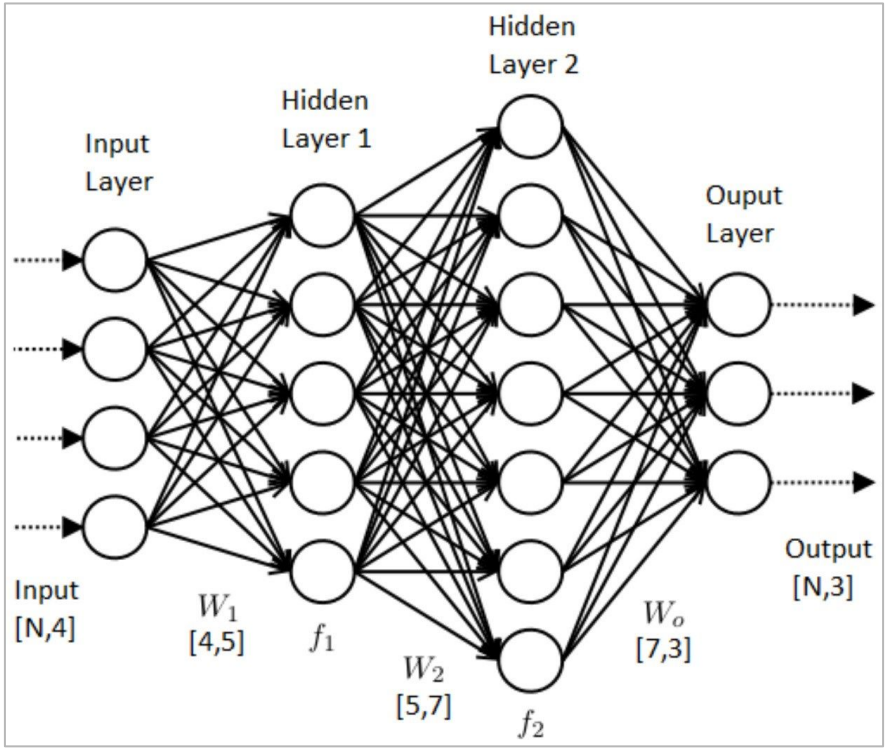
Figura 3: Modelo não-linear de um neurônio



Fonte: adaptado de Haykin (2001 [8])

Rede Neural

Figura 4: Exemplo de rede neural



Fonte: medium.com [9]

Convolutional Neural Network

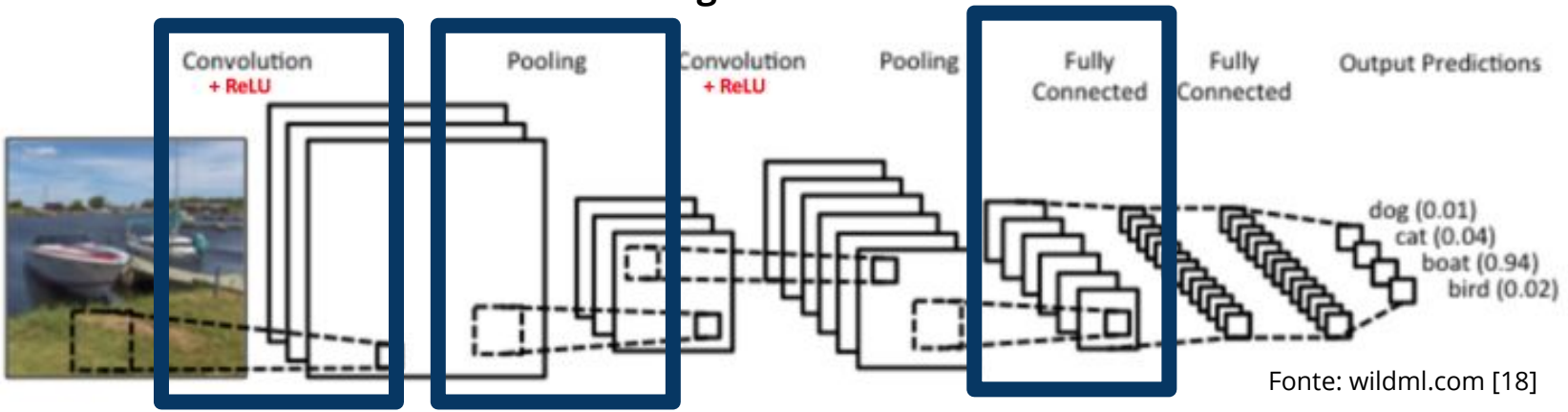
CNN

Convolutional Neural Network (CNN) - Inspirada nos processos biológicos

- Variação de perceptrons multi camada desenvolvidos de modo a demandar o mínimo de pré processamento possível
- Usada principalmente em reconhecimento de imagens e processamento de vídeo[17]

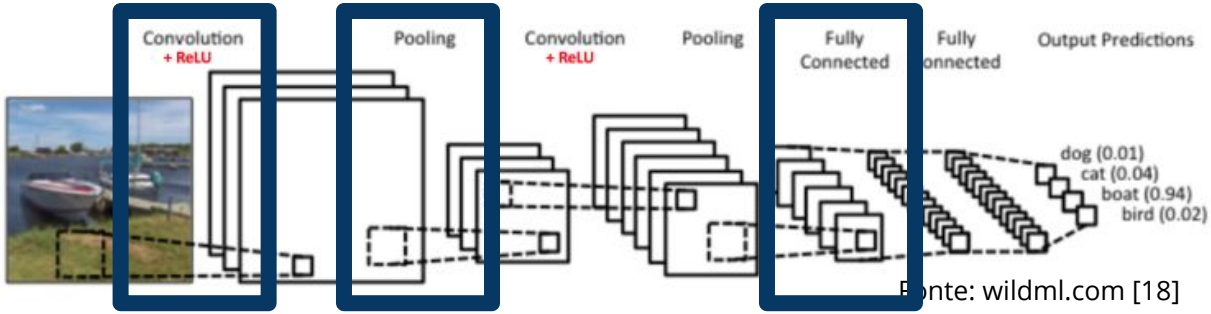
CNNs possuem geralmente 3 hidden layers

Figura 5: CNN



Fonte: wildml.com [18]

Figura 5: CNN



Fonte: wildml.com [18]

- **Convolutional Layers:** aplica operações de convolução nos dados de entrada

- **Pooling Layers:** combina as saídas dos neurônios em uma camada em um único neurônio na próxima camada

- **Fully Connected Layers:** conecta todos os neurônios em uma camada para cada neurônio em outra camada[17]

TensorFlow

Figura 6: TensorFlow

Fonte: www.tensorflow.org [10]

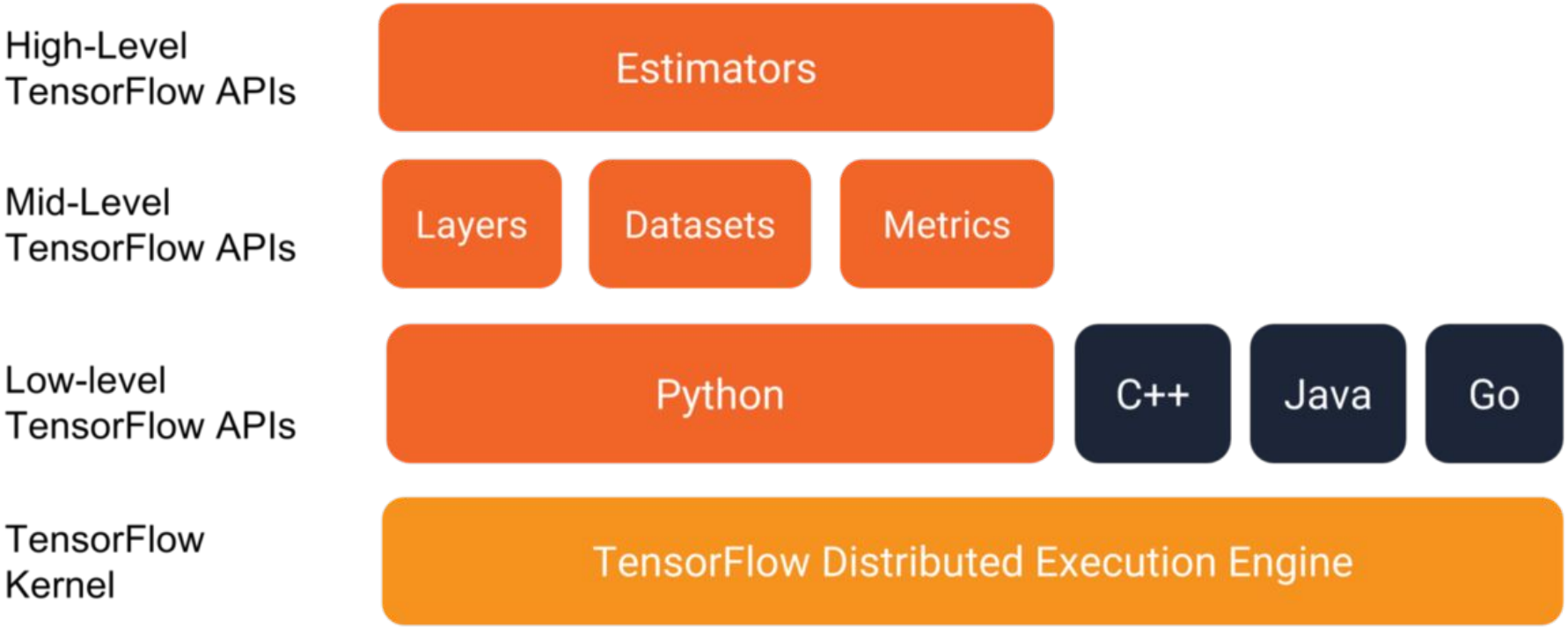
TensorFlow

- Biblioteca open-source para computação numérica usando grafos computacionais.[11]
- Originalmente desenvolvido pela Google Brain Team na organização de pesquisa Machine Intelligence do Google para Deep Learning.[11]
- Lançada em 09 de novembro de 2015, sendo a primeira versão estável (1.0.0) em 11 de fevereiro de 2017.[10]
- Atualmente encontra-se na versão 1.12.2.[10]

Figura 7: Empresas que utilizam TensorFlow

Fonte: www.tensorflow.org [10]

Figura 8: TensorFlow Stack

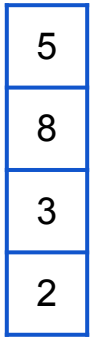


Fonte: medium.com [13]

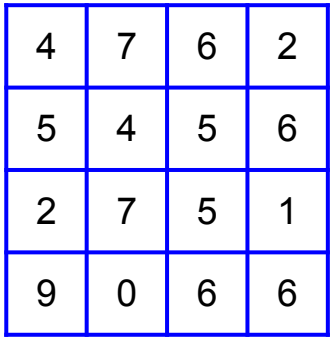
Tensor

- Tensor pode ser visualizado com um array multidimensional de números[12], sendo que:
  - um escalar é um tensor
  - uma matriz é um tensor
  - um vetor é um tensor

Tensor de dimensão [1]



Tensor de dimensão [5]



Tensor de dimensão [4, 4]

Data Flow

- Toda a computação é representada em grafos:
  - Nós são as operações
  - Arestas são os tensores

- Consiste em duas fases:
  - construção do grafo
  - execução do grafo

Figura 9: Exemplo de operação TensorFlow

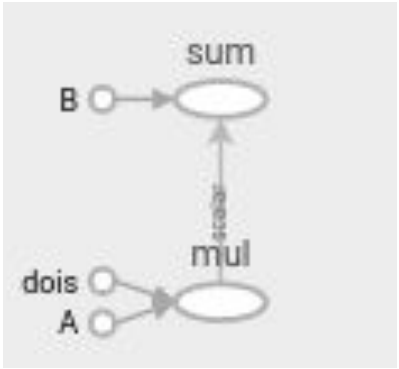
```python
import tensorflow as tf

tf.reset_default_graph()

# y = 2 * A + B
```

construção do grafo

```python
with tf.name_scope("y"):
    dois = tf.constant(2, name="dois")
    A = tf.constant(3, name="A")
    B = tf.constant(5, name="B")
    mul_op = tf.multiply(dois, A, name='mul')
    soma_op = tf.add(mul_op, B, name='sum')
```

execução

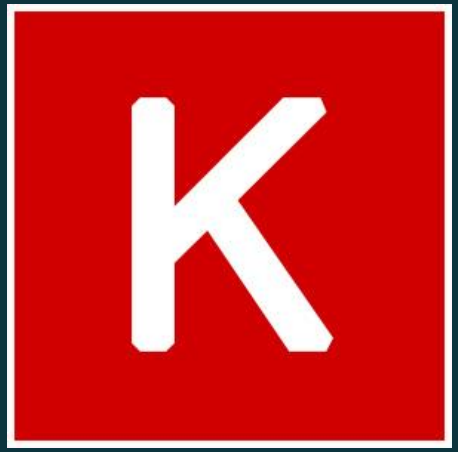
```python
with tf.Session() as sess:
    y = sess.run(soma_op)
    print("y = {}".format(y))
    writer = tf.summary.FileWriter('exemplos/ex1', sess.graph)
```



Fonte: próprio autor

Keras

Figura 10: Keras

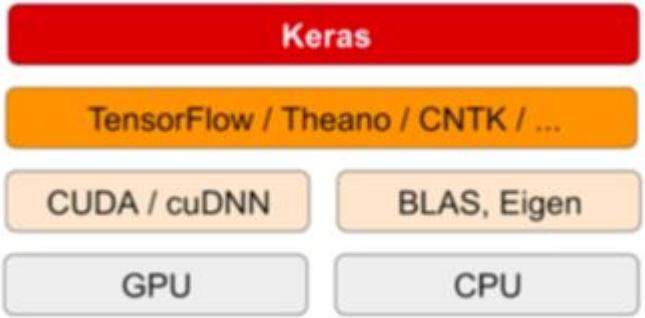


Fonte: keras.io [14]

Keras

- High-level neural networks API
- Escrito em Python
- Foco principal em propiciar o rápido desenvolvimento das redes neurais

Figura 11: Keras level



Fonte: junyelee [15]

TensorFlow vs Keras[16]

	High-level	Velocidade	Arquitetura	Debugging	Dataset	Popularidade	TOTAL
TensorFlow							2
Keras							4

TensorFlow

- Alta performance
- Maiores funcionalidades
- Execução mais rápida em grandes datasets

Keras

- Rápida prototipação
- Suporte a múltiplos back-end
- Código mais fácil

Aplicação Prática

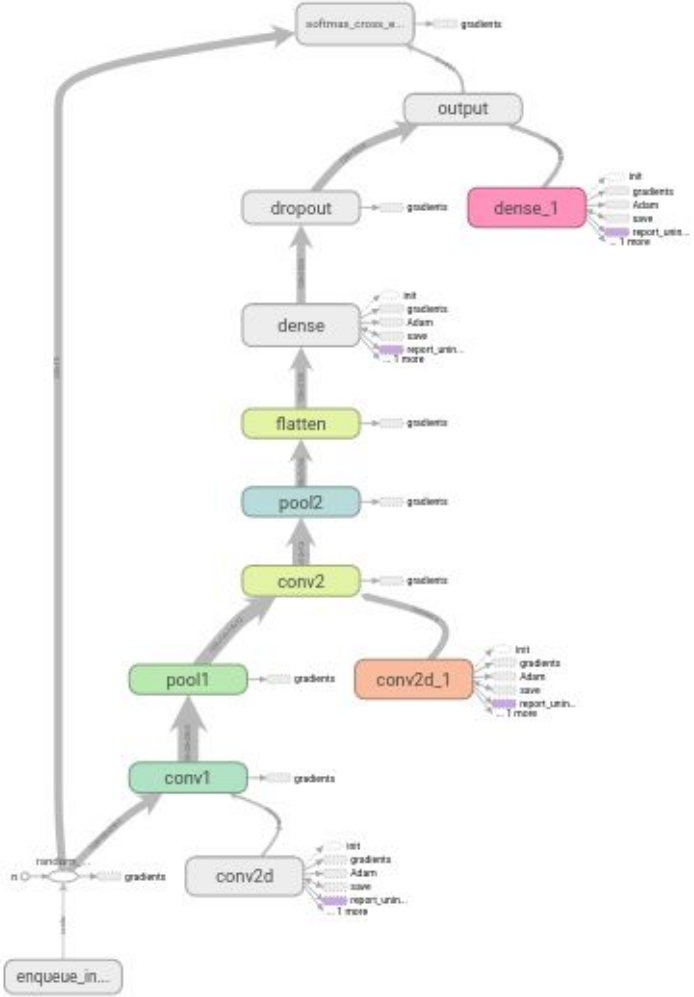
MNIST[19]

- Dataset com dígitos manuscritos:
  - 60000 dados de treinamento
  - 10000 dados de teste
- Imagens 28 x 28

Figura 11: MNIST

Fonte: yashk2810.github.io [20]

CNN


output
dropout
dense_1
dense
flatten
pool2
conv2
pool1
conv2d_1
conv1
conv2d
enqueue_in...

TensorFlow

mnist_tf.py

```
# 5x5 conv, 32 inputs, 64 outputs
wc2 = tf.Variable(tf.random_normal([5, 5, 32, 64]))
bc2 = tf.Variable(tf.random_normal([64]))
# fully connected, 7*7*64 inputs, 1024 outputs
wd1 = tf.Variable(tf.random_normal([7*7*64, 1024]))
# 1024 inputs, 10 outputs (class prediction)
wout = tf.Variable(tf.random_normal([1024, n_classes]))
bd1 = tf.Variable(tf.random_normal([1024]))

bout = tf.Variable(tf.random_normal([n_classes]))
# Construct model
# _X = tf.reshape(x, shape=[-1, 28, 28, 1])
# Convolution Layer
conv1 = conv2d(x, wc1, bc1)
# Max Pooling (down-sampling)
conv1 = max_pool(conv1, k=2)
# Convolution Layer
conv2 = conv2d(conv1, wc2, bc2)
# Max Pooling (down-sampling)
conv2 = max_pool(conv2, k=2)
# Fully connected layer
# Reshape conv2 output to fit dense layer input
dense1 = tf.reshape(conv2, [-1, wd1.get_shape().as_list()[0]])
# Relu activation
dense1 = tf.nn.relu(tf.add(tf.matmul(dense1, wd1), bd1))
# Apply Dropout
dense1 = tf.nn.dropout(dense1, keep_prob)
# Output, class prediction
pred = tf.add(tf.matmul(dense1, wout), bout)
# Define loss and optimizer
cost = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y, logits=pred))
optimizer =\
    tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
# Evaluate model
correct_pred = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
# Initializing the variables
init = tf.initialize_all_variables()
# Launch the graph
```

TensorFlow Estimator

mnist_tf_estimator.py

```python
def build_cnn(features, labels, mode):
    input_layer = features['X']

    with tf.name_scope("conv1"):
        conv1 = tf.layers.conv2d(inputs = input_layer, filters = 32, kernel_size=[5,5], activation = tf.nn.relu,
                                 padding = 'same')

    with tf.name_scope("pool1"):
        pooling1 = tf.layers.max_pooling2d(inputs = conv1, pool_size = [2,2], strides = 2)


    with tf.name_scope("conv2"):
        conv2 = tf.layers.conv2d(inputs = pooling1, filters = 64, kernel_size = [5,5], activation = tf.nn.relu,
                                 padding = 'same')

    with tf.name_scope("pool2"):
        pooling2 = tf.layers.max_pooling2d(inputs = conv2, pool_size = [2,2], strides = 2)

    with tf.name_scope("flatten"):
        flattening = tf.reshape(pooling2, [-1, 7 * 7 * 64])

    with tf.name_scope("dense"):
        dense = tf.layers.dense(inputs = flattening, units = 1024, activation = tf.nn.relu)

    with tf.name_scope("dropout"):
        dropout = tf.layers.dropout(inputs = dense, rate = 0.2, training=mode == tf.estimator.ModeKeys.TRAIN)

    with tf.name_scope("output"):
        output_layer = tf.layers.dense(inputs = dropout, units = 10)

        predicts = tf.argmax(output_layer, axis = 1)

    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode = mode, predictions = predicts)

    error = tf.losses.softmax_cross_entropy(onehot_labels=labels, logits=output_layer)

    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.train.AdamOptimizer(learning_rate = 0.001)
        train = optimizer.minimize(error, global_step = tf.train.get_global_step())
        return tf.estimator.EstimatorSpec(mode = mode, loss = error, train_op = train)

    if mode == tf.estimator.ModeKeys.EVAL:
          eval_metric_ops = {
              'acc': tf.metrics.accuracy(
                 tf.argmax(input=output_layer, axis=1),
                 tf.argmax(input=labels, axis=1))
          }

          return tf.estimator.EstimatorSpec(mode=mode, predictions=predicts,
                                            loss=error, eval_metric_ops=eval_metric_ops)
```

Keras

mnist_keras.py

```
def build_cnn():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(5, 5), activation='relu', input_shape=(28, 28, 1), padding = 'same'))
    model.add(MaxPooling2D(pool_size = [2,2], strides = 2))
    model.add(Conv2D(64, (5, 5), activation='relu', padding = 'same'))
    model.add(MaxPooling2D(pool_size = [2,2], strides = 2))
    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adam(lr = 0.001),
                  metrics=['accuracy'])

    model.summary()

    return model
```

Comparativo

	TensorFlow	TensorFlow Estimator	Keras
Tempo Desenvolvimento	5h	3h	30m
Tempo Execução	~200s	~140s	~366s
Linhas Código	154	92	55
Acurácia	~98%	~97%	~99%

Conclusão

Conclusão

- Prototipação e desenvolvimento mais rápido utilizando Keras;
- Alta performance utilizando TensorFlow;
- Não existe biblioteca “bala-prata”.

Referências

[1] https://medium.com/tensorflow/training-and-serving-ml-models-with-tf-keras-fd975cc0fa27

[2] Charniak, E. and McDermott, D. (1985). Introduction to Artificial Intelligence. Addison-Wesley.

[3] Kurzweil, R. (1990). The Age of Intelligent Machines. MIT Press.

[4] Poole, D., Mackworth, A. K., and Goebel, R. (1998). Computational intelligence: A logical approach. Oxford University Press.

[5] Grupo de Pesquisa em Engenharia de Software. Inteligência Artificial Aprendizagem Supervisionada.

[6] Azoff E. M. (1994). Neural network time series forecasting of financial markets. Chichester: John Wiley and Sons

[7] http://conteudo.icmc.usp.br/pessoas/andre/research/neural/

[8] Redes Neurais - 2ed., SIMON S. HAYKIN, 2001

[9] https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4

[10] www.tensorflow.org

[11] http://datascienceacademy.com.br/blog/o-que-e-o-tensorflow-machine-intelligence-platform/

[12] https://pt.slideshare.net/matthiasfeys/introduction-to-tensorflow-66591270

[13] https://medium.com/@navinkb/beginners-guide-to-tensorflow-by-a-beginner-ebb93e521ece

[14] https://keras.io

[15] http://junyelee.blogspot.com/2018/01/deep-learning-with-python.html

[16] https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/

[17] Convolutional neural network. Disponível em: <https://en.wikipedia.org/wiki/Convolutional_neural_network>. Acesso em: 10 nov. 2018.

[18] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[19] http://yann.lecun.com/exdb/mnist/

[20] https://yashk2810.github.io/Applying-Convolutional-Neural-Network-on-the-MNIST-dataset/

Fim!

Gustavo Gomides
gustavo.gomides7@gmail.com

https://github.com/gustavogomides/tensorflow_ce229