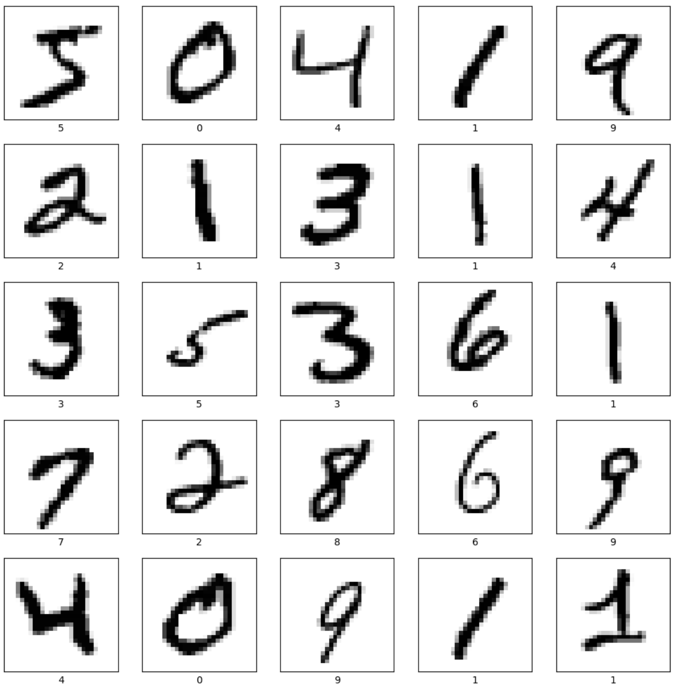
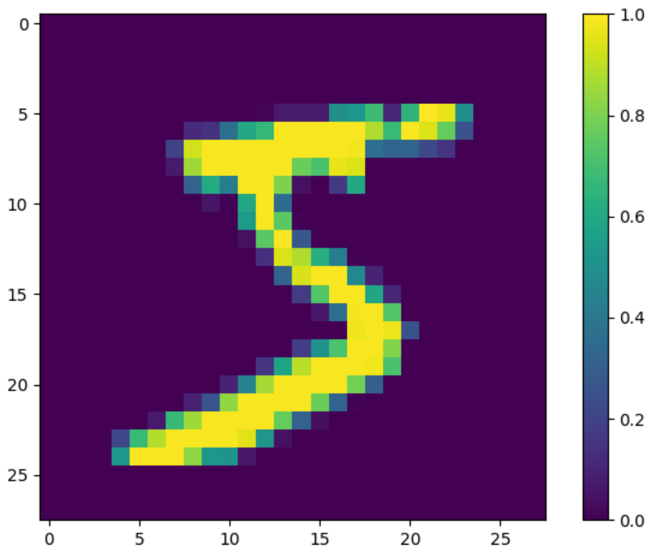
# 我的 Tensor flow 的第一步

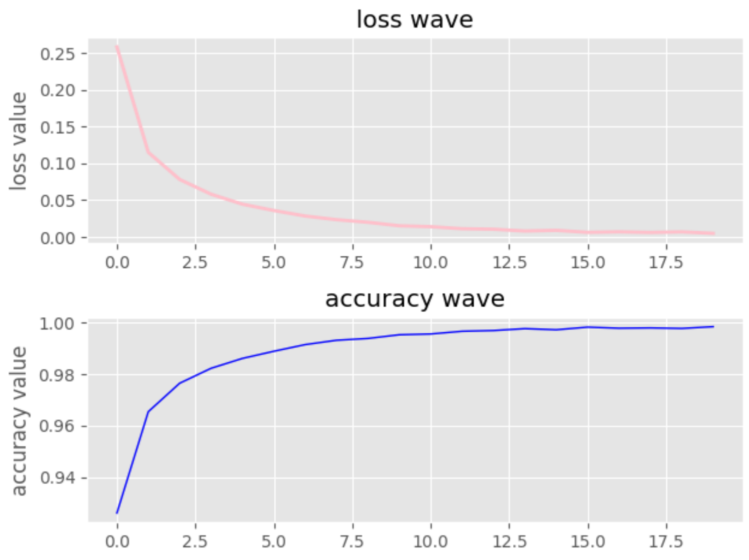
这是我神经网络学习的第一步，首先，在理解了back-propagation neural network之后，通过初次建立`Keras.Dense`，可以直接简化对权重的运算和建立，也免去了链式法则求导的复杂性。Tensor flow利用梯度带，将各个求导得以计算，也就是优化器。下面展示运行结果：



这个图片展示了每个数字下对应的数字，也就是标签和图片的一一对应关系。



通过将每个像素点，降维到0-1，然后喂给kernel层运算。



上图展示了loss函数和accuracy函数的实际训练过程。

```
"""
@Project: tensorflow_neural_network
@Time:2021/6/8 15:32
@Author: LeeRoc
@Description: learning Tensorflow
"""

import tensorflow as tf
import matplotlib.pyplot as plt

class Minist_Neural_Network:
    def __init__(self):
        """
        @note: to initilize the essential parameter of neural network.
        @param: normalize to get the scope between 0-1.
        @param: Image's shape is (6000, 28*28), lable's shape is (6000,), every
lable is mapped every image, o2o.
        @rtype: null
        """
        self.model = None
        self.history = None
        self.normalization = 255.0
        (self.Image_trains, self.Lable_trains), (self.Image_tests,
self.Lable_tests) = tf.keras.datasets.mnist.load_data()
        self.Image_trains = self.Image_trains / self.normalization
        self.Image_tests = self.Image_tests / self.normalization
        self.__Neural_kernal()

    def __Neural_kernal(self):
        """
        @note: this function is to define the layers of the model.
            Flatten's function is to change the (28, 28) to (784, ).
            compile is to specifying a loss, metrics, and an optimizer.
        @rtype: null
        """
        self.model = tf.keras.Sequential([
            tf.keras.layers.Flatten(input_shape=(28, 28)),
            tf.keras.layers.Dense(128, activation='relu'),
            tf.keras.layers.Dense(10)
        ])

        self.model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                           metrics=['accuracy'])

    def Activated(self, epochs=10):
        """
        @note: history is to record loss value and learning accuracy.
        @rtype: tuple shape
        """
        self.history = self.model.fit(self.Image_trains, self.Lable_trains,
epochs=epochs)
        test_loss, test_acc = self.model.evaluate(self.Image_tests,
self.Lable_tests, verbose=2)

        print('\nTest accuracy:', test_acc,
              '\nTest loss:', test_loss)

    def Save(self):
        self.model.save()

    def Specified_Plot(self):
        """
        @note: draw a tensors stream by a frankly specified picture
        @rtype: null
        """
        # draw a picture
        plt.style.use('ggplot')
        fig = plt.figure()

        ax1 = fig.add_subplot(2, 1, 1)
        ax1.set_ylabel('loss value')
        ax1.set_title('loss wave')
        ax1.plot(range(0, 20), self.history['loss'], color='pink', lw=2)

        ax2 = fig.add_subplot(2, 1, 2)
        ax2.set_ylabel('accuracy value')
        ax2.set_title('accuracy wave')
        ax2.plot(range(0, 20), self.history['accuracy'], color='Blue', lw=1)

        plt.show()

if __name__ == "__main__":
    minst = Minist_Neural_Network()
    history = minst.Activated(10)
    minst.Save()
```