Miniprojeto 2 - Redes Neurais Convolucionais com MNIST

Alunos

- Ítalo Rodrigo Barbosa Paulino - irbp
- José Nilton de Oliveira Lima Júnior - jnolj

O objetivo deste mini projeto é construir um classificador para a base de dados MNIST utilizando uma Rede Neural Convolucional (CNN). Para isso, utilizaremos a biblioteca tensorflow em conjunto com o keras para modelar a nossa rede neural e só assim depois treiná-la e avaliar o seu desempenho com base no conjunto de teste disponibilizado pelo dataset.

Esse relatório seguirá o modelo de código comentado, feito em Python3 e utilizando Jupyter Notebook.

In [0]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, DepthwiseConv2D
, Conv2D, MaxPool2D, ZeroPadding2D
from tensorflow.keras.layers import BatchNormalization, Activation,
GlobalAveragePooling2D, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical

import numpy as np
import matplotlib.pyplot as plt
import os
from sklearn.model_selection import train_test_split
from collections import Counter
from sklearn.metrics import confusion_matrix

from google.colab import files
```

O primeiro passo feito foi carregar o dataset do MNIST e normalizar os valores correspondentes aos seus pixels. Logo após isso transformamos a imagem em um array tridimensional, mas com apenas um channel (a imagem é em tons de cinza), pois será essa o tipo de entrada esperada pela nossa rede neural.

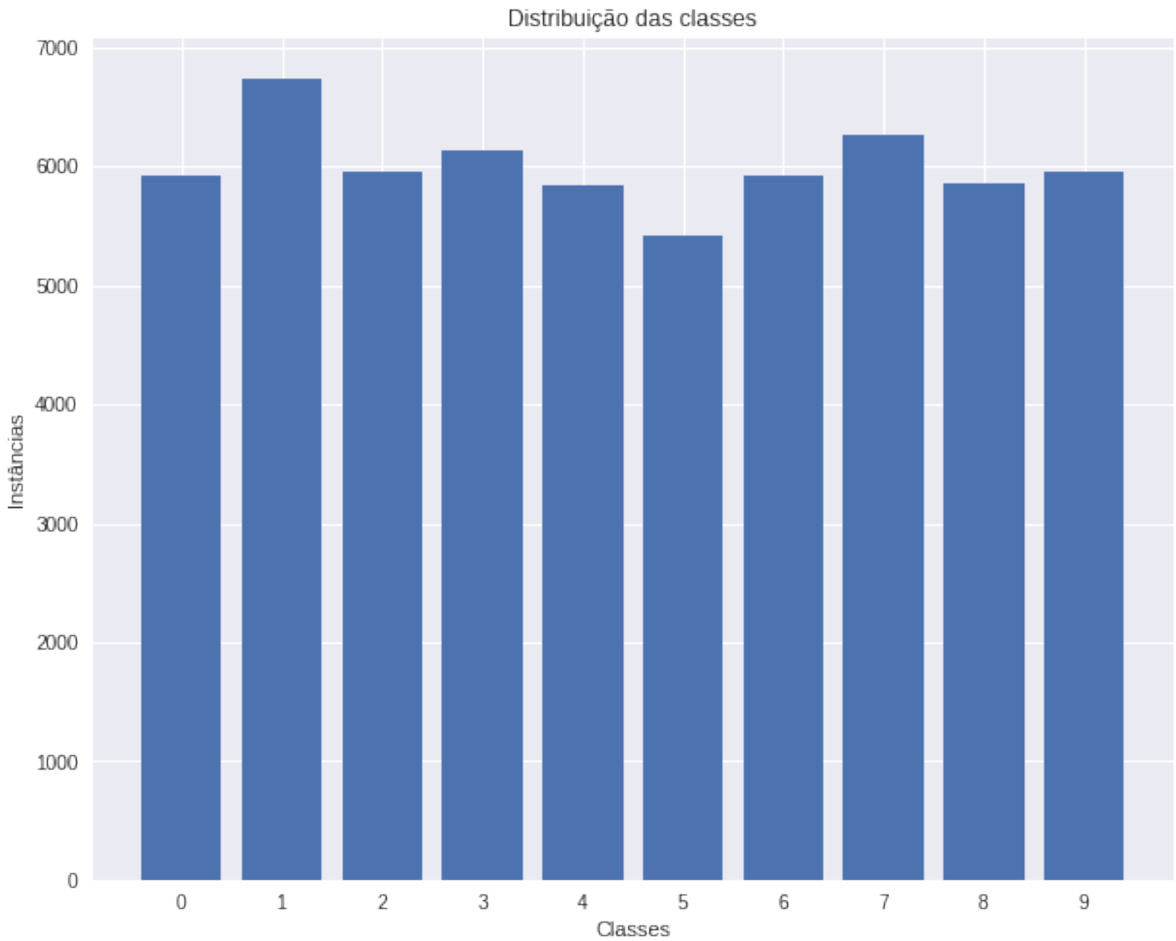
In [2]:
```
mnist = keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train, x_test = x_train.reshape(-1, 28, 28, 1), x_test.reshape(-1
, 28, 28, 1)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf
-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

Out[2]:
```
(60000, 28, 28, 1)
```

Em seguida plotamos a distribuição de cada classe, ou seja, quantas instâncias de cada classe está presente no nosso dataset.

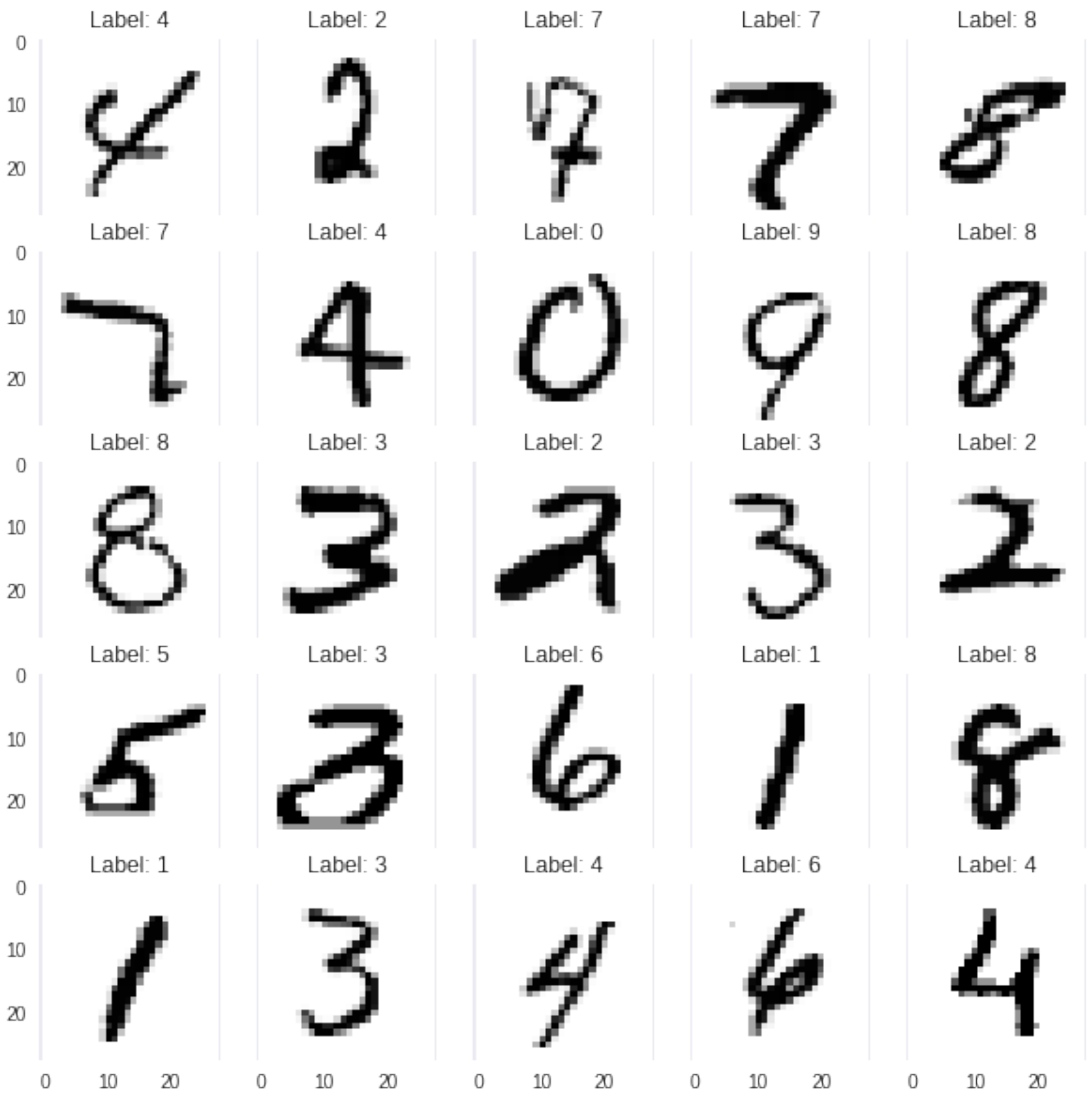
In [3]:
```
unique, count = np.unique(y_train, return_counts=True)
plt.figure(figsize=(10,8))
plt.title("Distribuição das classes")
plt.bar(unique, count)
plt.xticks(unique)
plt.xlabel("Classes")
plt.ylabel("Instâncias")
plt.show()
```



Agora fazemos a divisão do conjunto de treino em conjunto de treino e conjundo de validação que servirá para avaliar o desempenho da nossa rede enquanto a mesma é treinada. Logo depois foi plotado alguns exemplos do nosso dataset.

In [0]:
```
y_train, y_test = to_categorical(y_train), to_categorical(y_test)
random_seed = 2
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.1, random_state=random_seed)
```

In [65]:
```
fig, ax = plt.subplots(5, 5, sharex=True, sharey=True, figsize=(10,
10))
index = 0
for row in range(5):
    for col in range(5):
        ax[row, col].imshow(x_train[index].reshape(28,28))
        ax[row, col].set_title("Label: {}".format(y_train[index].ar
gmax()))
        ax[row, col].grid(False)
        index += 1
```



Abaixo segue a implementação do nosso modelo de CNN e logo após um sumário mostra a estrutura da rede com o número de parâmetros em cada camada assim como o número total de parâmetros treináveis. Duas camadas de Dropout foram utilizadas para que uma certa proporção de "nós" da nossa rede seja aleatoriamente ignorada, isso força que a rede extraia as features de maneira distribuída ajudando também a reduzir o overfitting.

In [0]:
```
model = Sequential()

model.add(Conv2D(filters=32,
                 kernel_size=(5,5),
                 padding="same",
                 activation="relu",
                 input_shape=(28,28,1)))
model.add(Conv2D(filters=32,
                 kernel_size=(5,5),
                 padding="same",
                 activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters=64,
                 kernel_size=(3,3),
                 padding="same",
                 activation="relu"))
model.add(Conv2D(filters=64,
                 kernel_size=(3,3),
                 padding="same",
                 activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation="softmax"))
```

In [7]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        832
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        25632
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 256)               803072
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 887,530
Trainable params: 887,530
Non-trainable params: 0
_________________________________________________________________
```

In [0]:
```
def check_tpu():
    try:
        tpu_address = 'grpc://' + os.environ['COLAB_TPU_ADDR']
        print ('TPU address is', tpu_address)

        with tf.Session(tpu_address) as session:
          devices = session.list_devices()

        return True, tpu_address
    except Exception:
        return False, ""
```

In [9]:
```
has_tpu, tpu_address = check_tpu()
if has_tpu:
    tpu_model = tf.contrib.tpu.keras_to_tpu_model(
        model,
        strategy=tf.contrib.tpu.TPUDistributionStrategy(
            tf.contrib.cluster_resolver.TPUClusterResolver(tpu_addr
ess)
        )
    )
```

```
TPU address is grpc://10.79.236.226:8470
INFO:tensorflow:Querying Tensorflow master (b'grpc://10.79.236.226
:8470') for TPU system metadata.
INFO:tensorflow:Found TPU system:
INFO:tensorflow:*** Num TPU Cores: 8
INFO:tensorflow:*** Num TPU Workers: 1
INFO:tensorflow:*** Num TPU Cores Per Worker: 8
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:CPU:0, CPU, -1, 10286764627793717633)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:XLA_CPU:0, XLA_CPU, 17179869184, 1545492
5982782434438)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:XLA_GPU:0, XLA_GPU, 17179869184, 3839510
994108593343)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:0, TPU, 17179869184, 486880024873964
8966)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:1, TPU, 17179869184, 130900917395422
84063)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:2, TPU, 17179869184, 905905170247984
9442)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:3, TPU, 17179869184, 288107262343864
5419)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:4, TPU, 17179869184, 162890196397625
27406)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:5, TPU, 17179869184, 130324781477984
07067)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:6, TPU, 17179869184, 423676341664824
8849)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU:7, TPU, 17179869184, 149612181805609
4762)
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worke
r/replica:0/task:0/device:TPU_SYSTEM:0, TPU_SYSTEM, 17179869184, 1
0225653847187607492)
WARNING:tensorflow:tpu_model (from tensorflow.contrib.tpu.python.t
pu.keras_support) is experimental and may change or be removed at
any time, and without warning.
INFO:tensorflow:Connecting to: b'grpc://10.79.236.226:8470'
```

Como função otimizadora nós utilizamos o RMSprop, pois ele efetua os ajustes de forma bem simples na tentativa de reduzir a agressividade e diminuindo o learning rate monotonicamente. Poderíamos ter utilizado o Stochastic Gradient Descent (SGD), mas optamos pelo RMSprop por ele apresentar um desempenho melhor.

In [0]:
```
optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
if has_tpu:
    tpu_model.compile(optimizer=optimizer, loss="categorical_crosse
ntropy", metrics=["accuracy"])
else:
    model.compile(optimizer=optimizer, loss="categorical_crossentro
py", metrics=["accuracy"])
```

In [0]:
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)

epochs = 30
batch_size = 1000
```

In [0]:
```
def train_gen(batch_size):
  while True:
    offset = np.random.randint(0, x_train.shape[0] - batch_size)
    yield x_train[offset:offset+batch_size], y_train[offset:offset
+ batch_size]
```

Abaixo segue o histórico de treino para cada uma das 30 épocas que nossa rede foi submetida durante o treino.

In [14]:
```
if has_tpu:
    history = tpu_model.fit_generator(train_gen(batch_size),
                                      epochs=epochs,
                                      validation_data=(x_val, y_val
),
                                      steps_per_epoch=x_train.shape
[0]//batch_size,
                                      callbacks=[learning_rate_redu
ction])
else:
    history = model.fit_generator(train_gen(batch_size),
                                      epochs=epochs,
                                      validation_data=(x_val, y_val
),
                                      steps_per_epoch=x_train.shape
[0]//batch_size,
                                      callbacks=[learning_rate_redu
ction])
```

```
Epoch 1/30
INFO:tensorflow:New input shapes; (re-)compiling: mode=train, [Ten
sorSpec(shape=(125, 28, 28, 1), dtype=tf.float32, name='conv2d_inp
ut0'), TensorSpec(shape=(125, 10), dtype=tf.float32, name='dense_1
_target0')]
INFO:tensorflow:Overriding default placeholder.
INFO:tensorflow:Remapping placeholder for conv2d_input
INFO:tensorflow:Cloning RMSprop {'lr': 0.0010000000474974513, 'rho
': 0.8999999761581421, 'decay': 0.0, 'epsilon': 1e-08}
```

```
INFO:tensorflow:Get updates: Tensor("loss/mul:0", shape=(), dtype=
float32)
INFO:tensorflow:Started compiling
INFO:tensorflow:Finished compiling. Time elapsed: 2.26594495773315
43 secs
INFO:tensorflow:Setting weights on TPU model.
53/54 [============================>.] - ETA: 0s - loss: 0.6942 -
acc: 0.7777INFO:tensorflow:New input shapes; (re-)compiling: mode=
eval, [TensorSpec(shape=(125, 28, 28, 1), dtype=tf.float32, name='
conv2d_input0'), TensorSpec(shape=(125, 10), dtype=tf.float32, nam
e='dense_1_target0')]
INFO:tensorflow:Overriding default placeholder.
INFO:tensorflow:Remapping placeholder for conv2d_input
INFO:tensorflow:Cloning RMSprop {'lr': 0.0010000000474974513, 'rho
': 0.8999999761581421, 'decay': 0.0, 'epsilon': 1e-08}
INFO:tensorflow:Started compiling
INFO:tensorflow:Finished compiling. Time elapsed: 1.30892777442932
13 secs
54/54 [==============================] - 8s 149ms/step - loss: 0.6
862 - acc: 0.7804 - val_loss: 0.1524 - val_acc: 0.9520
Epoch 2/30
54/54 [==============================] - 2s 34ms/step - loss: 0.15
15 - acc: 0.9538 - val_loss: 0.0340 - val_acc: 0.9907
Epoch 3/30
54/54 [==============================] - 2s 34ms/step - loss: 0.09
45 - acc: 0.9702 - val_loss: 0.0215 - val_acc: 0.9947
Epoch 4/30
54/54 [==============================] - 2s 35ms/step - loss: 0.05
29 - acc: 0.9831 - val_loss: 0.0111 - val_acc: 0.9947
Epoch 5/30
54/54 [==============================] - 2s 37ms/step - loss: 0.05
70 - acc: 0.9830 - val_loss: 0.0168 - val_acc: 0.9960
Epoch 6/30
54/54 [==============================] - 2s 37ms/step - loss: 0.04
41 - acc: 0.9877 - val_loss: 0.0149 - val_acc: 0.9960
Epoch 7/30
54/54 [==============================] - 2s 39ms/step - loss: 0.03
53 - acc: 0.9886 - val_loss: 0.0092 - val_acc: 0.9973
Epoch 8/30
54/54 [==============================] - 2s 37ms/step - loss: 0.03
07 - acc: 0.9901 - val_loss: 0.0116 - val_acc: 0.9973
Epoch 9/30
54/54 [==============================] - 2s 38ms/step - loss: 0.03
12 - acc: 0.9904 - val_loss: 0.0118 - val_acc: 0.9973
Epoch 10/30
54/54 [==============================] - 2s 36ms/step - loss: 0.02
53 - acc: 0.9926 - val_loss: 0.0044 - val_acc: 0.9987
Epoch 11/30
54/54 [==============================] - 2s 34ms/step - loss: 0.01
90 - acc: 0.9941 - val_loss: 0.0064 - val_acc: 0.9987
Epoch 12/30
54/54 [==============================] - 2s 38ms/step - loss: 0.02
75 - acc: 0.9917 - val_loss: 0.0100 - val_acc: 0.9973
Epoch 13/30
52/54 [===========================>..] - ETA: 0s - loss: 0.0200 -
acc: 0.9945
Epoch 00013: ReduceLROnPlateau reducing learning rate to 0.0005000
```

```
000237487257.
54/54 [==============================] - 2s 38ms/step - loss: 0.01
93 - acc: 0.9947 - val_loss: 0.0048 - val_acc: 0.9987
Epoch 14/30
54/54 [==============================] - 2s 42ms/step - loss: 0.02
12 - acc: 0.9930 - val_loss: 0.0057 - val_acc: 0.9973
Epoch 15/30
54/54 [==============================] - 2s 42ms/step - loss: 0.01
95 - acc: 0.9936 - val_loss: 0.0082 - val_acc: 0.9973
Epoch 16/30
53/54 [============================>.] - ETA: 0s - loss: 0.0149 -
acc: 0.9962
Epoch 00016: ReduceLROnPlateau reducing learning rate to 0.0002500
000118743628.
54/54 [==============================] - 2s 43ms/step - loss: 0.01
47 - acc: 0.9963 - val_loss: 0.0098 - val_acc: 0.9987
Epoch 17/30
54/54 [==============================] - 2s 41ms/step - loss: 0.01
38 - acc: 0.9960 - val_loss: 0.0121 - val_acc: 0.9960
Epoch 18/30
54/54 [==============================] - 2s 41ms/step - loss: 0.01
66 - acc: 0.9951 - val_loss: 0.0124 - val_acc: 0.9973
Epoch 19/30
52/54 [===========================>..] - ETA: 0s - loss: 0.0181 -
acc: 0.9952
Epoch 00019: ReduceLROnPlateau reducing learning rate to 0.0001250
000059371814.
54/54 [==============================] - 2s 39ms/step - loss: 0.01
77 - acc: 0.9954 - val_loss: 0.0097 - val_acc: 0.9987
Epoch 20/30
54/54 [==============================] - 2s 42ms/step - loss: 0.00
56 - acc: 0.9979 - val_loss: 0.0021 - val_acc: 0.9987
Epoch 21/30
54/54 [==============================] - 2s 39ms/step - loss: 0.01
26 - acc: 0.9959 - val_loss: 0.0085 - val_acc: 0.9973
Epoch 22/30
52/54 [===========================>..] - ETA: 0s - loss: 0.0107 -
acc: 0.9971
Epoch 00022: ReduceLROnPlateau reducing learning rate to 6.2500002
9685907e-05.
54/54 [==============================] - 2s 37ms/step - loss: 0.01
10 - acc: 0.9969 - val_loss: 0.0100 - val_acc: 0.9973
Epoch 23/30
54/54 [==============================] - 2s 38ms/step - loss: 0.01
58 - acc: 0.9960 - val_loss: 0.0049 - val_acc: 0.9973
Epoch 24/30
54/54 [==============================] - 2s 38ms/step - loss: 0.01
45 - acc: 0.9967 - val_loss: 0.0070 - val_acc: 0.9973
Epoch 25/30
53/54 [============================>.] - ETA: 0s - loss: 0.0127 -
acc: 0.9962
Epoch 00025: ReduceLROnPlateau reducing learning rate to 3.1250001
48429535e-05.
54/54 [==============================] - 2s 37ms/step - loss: 0.01
25 - acc: 0.9963 - val_loss: 0.0054 - val_acc: 0.9987
Epoch 26/30
54/54 [==============================] - 2s 36ms/step - loss: 0.00
```

```
86 - acc: 0.9975 - val_loss: 0.0108 - val_acc: 0.9973
Epoch 27/30
54/54 [==============================] - 2s 35ms/step - loss: 0.01
06 - acc: 0.9964 - val_loss: 0.0111 - val_acc: 0.9947
Epoch 28/30
53/54 [============================>.] - ETA: 0s - loss: 0.0075 -
acc: 0.9973
Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.5625000
742147677e-05.
54/54 [==============================] - 2s 36ms/step - loss: 0.00
85 - acc: 0.9970 - val_loss: 0.0051 - val_acc: 0.9987
Epoch 29/30
54/54 [==============================] - 2s 38ms/step - loss: 0.00
60 - acc: 0.9978 - val_loss: 0.0018 - val_acc: 1.0000
Epoch 30/30
54/54 [==============================] - 2s 38ms/step - loss: 0.00
80 - acc: 0.9972 - val_loss: 0.0039 - val_acc: 0.9987
```

In [70]:

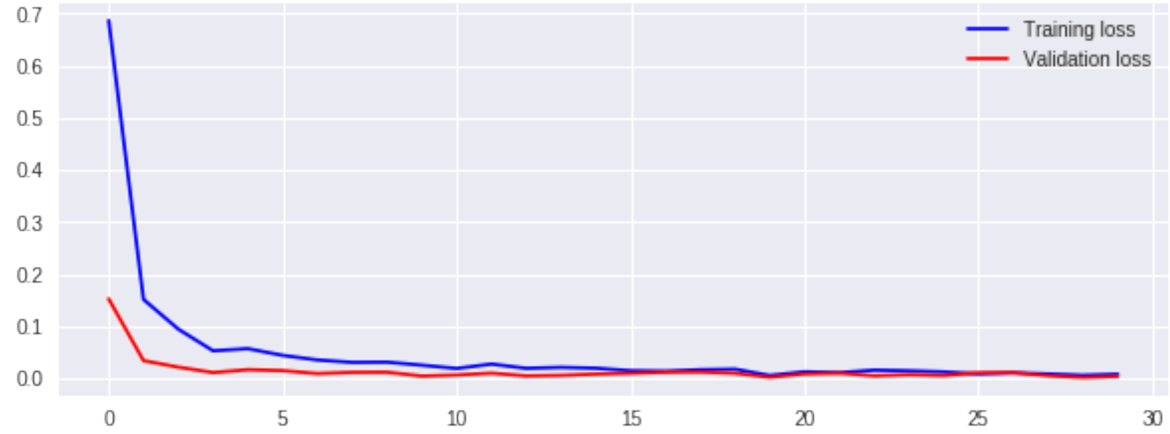
```python
if has_tpu:
    loss, acc = tpu_model.evaluate(x_test, y_test)
    cpu_model = tpu_model.sync_to_cpu()
    predictions = cpu_model.predict(x_test)
else:
    loss, acc = model.evaluate(x_test, y_test)
    predictions = model.predict(x_test)

pred_labels = np.argmax(predictions, axis=1)
```

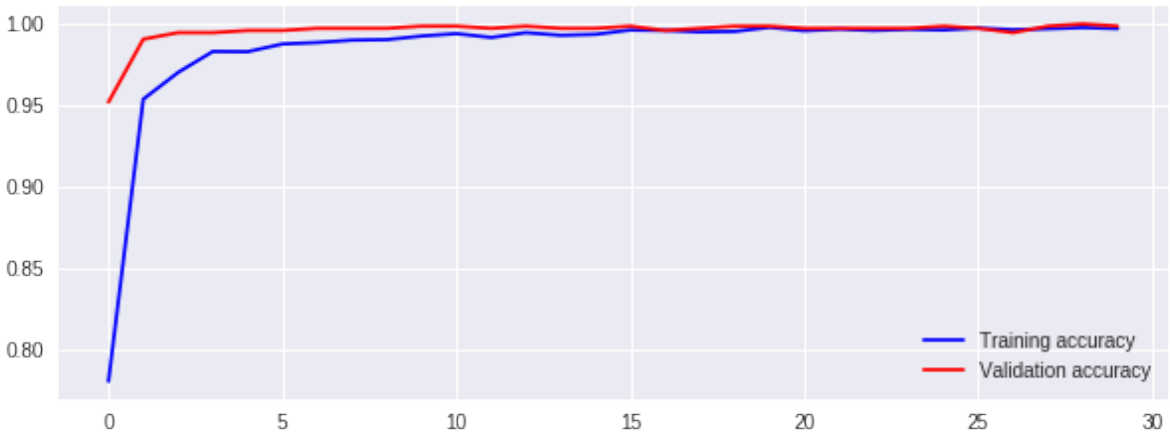
```
10000/10000 [==============================] - 2s 217us/step
INFO:tensorflow:Copying TPU weights to the CPU
Acurácia conjunto de teste: 99.6%
```

Como podemos ver nos gráficos plotados abaixo, a nossa rede obteve um excelente resultado com uma acurácia alta e com uma taxa de erro bastante baixa. Podemos observar também que na maioria das épocas a acurácia do conjunto de validação é maior que a acurácia do conjunto de treino, isso é bastante importante pois indica que não está acontecendo um overfit sobre o conjunto de treino.

In [48]:

```
fig, ax = plt.subplots(2, 1, figsize=(10,8))
ax[0].plot(history.history['loss'], color='b', label="Training loss
")
ax[0].plot(history.history['val_loss'], color='r', label="Validatio
n loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['acc'], color='b', label="Training accur
acy")
ax[1].plot(history.history['val_acc'], color='r',label="Validation
accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```



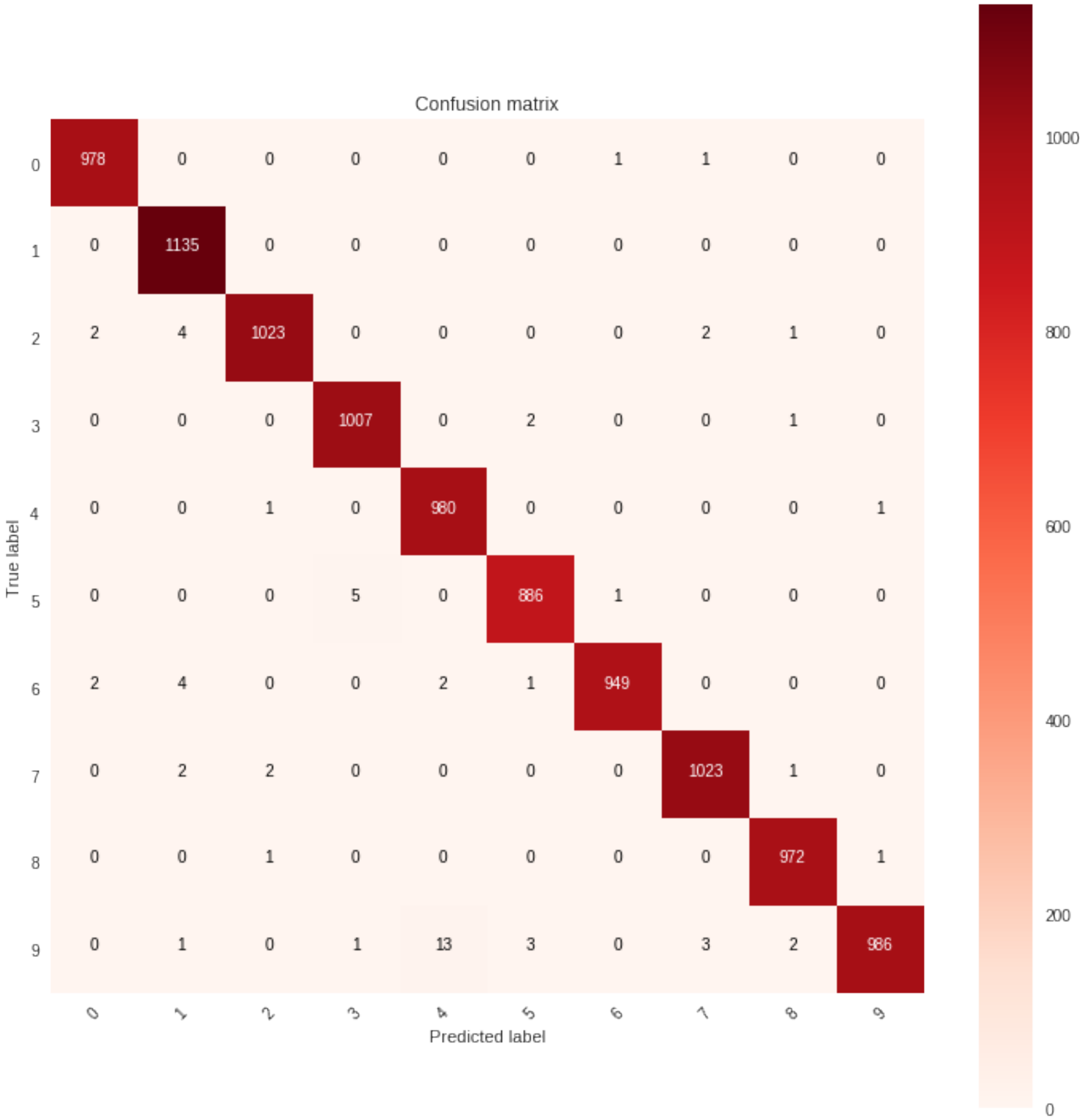


In [0]:

```
def plot_confusion_matrix(cm, classes):
    cmap = plt.get_cmap('Reds')
    tick_marks = classes

#     cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.figure(figsize=(10, 10))
    plt.grid(False)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title("Confusion matrix")
    plt.colorbar()
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, format(cm[i, j]),
                     horizontalalignment='center',
                     color='white' if cm[i , j] > thresh else 'blac
k')

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [46]:

```
cm = confusion_matrix(y_test.argmax(axis=1), pred_labels)
classes = np.arange(predictions.shape[1])
plot_confusion_matrix(cm, classes)
```


Confusion matrix
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
978 0 0 0 0 0 1 1 0 0
0 1135 0 0 0 0 0 0 0 0
2 4 1023 0 0 0 0 2 1 0
0 0 0 1007 0 2 0 0 1 0
0 0 1 0 980 0 0 0 0 1
0 0 0 5 0 886 1 0 0 0
2 4 0 0 2 1 949 0 0 0
0 2 2 0 0 0 0 1023 1 0
0 0 1 0 0 0 0 0 972 1
0 1 0 1 13 3 0 3 2 986
1000
800
600
400
200
0

In [67]:
```python
fig, ax = plt.subplots(5, 5, sharex=True, sharey=True, figsize=(10, 10))
index = 0
for row in range(5):
    for col in range(5):
        ax[row, col].imshow(x_test[index].reshape(28,28))
        ax[row, col].set_title("Label: {}\nPredicted: {}".format(y_test[index].argmax(),
                                                                   pred_labels[index]))
        ax[row, col].grid(False)
        index += 1
```