Exemplo de rede neural MLP e Convolucional com Keras

Importando as bibliotecas

In [1]:
```
import keras
```

```
/home/pedro/anaconda3/envs/Neurais2/lib/python3.5/site-packages/h5py/__init__
.py:36: FutureWarning: Conversion of the second argument of issubdtype from `
float` to `np.floating` is deprecated. In future, it will be treated as `np.f
loat64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
from keras.datasets import mnist
```

Fazendo o download dos dados

In [3]:
```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Analisando os dados

In [4]:
```
import matplotlib.pyplot as plt
```

In [5]:
```
train_images.shape
```

Out[5]: (60000, 28, 28)

In [6]:
```
len(train_labels)
```

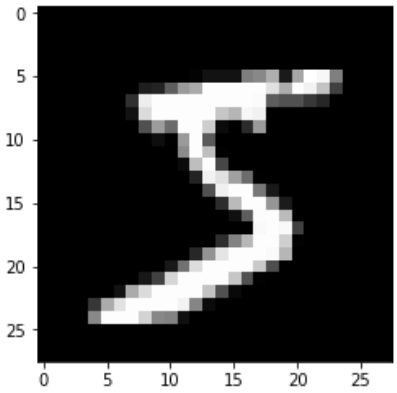
Out[6]: 60000

In [7]:
```
train_images[0]
```

```
Out[7]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
                 18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
                253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
                253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
                253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
                205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
                 90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
                190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
                253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
                241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                 81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
                148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
                253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
                253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
                195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

In [8]:
```
plt.imshow(train_images[0], cmap='gray')
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x7ff143455a58>
```



In [9]:
```
train_labels
```

Out[9]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

In [10]:
```
test_images.shape
```

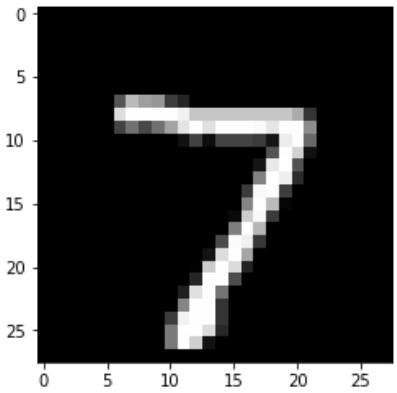
Out[10]:
```
(10000, 28, 28)
```

In [11]:
```
len(test_labels)
```

Out[11]:
```
10000
```

In [12]:
```
plt.imshow(test_images[0], cmap='gray')
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x7ff1434436d8>
```



In [13]:
```
test_labels
```

Out[13]:
```
array([7, 2, 1, ..., 4, 5, 6], dtype=uint8)
```

Normalizando os dados

In [14]:
```
train_images = train_images.astype('float32') / 255
test_images = test_images.astype('float32') / 255
```

In [15]:
```
train_images[0]
```

```
Out[15]: array([[0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
                 0.07058824, 0.49411765, 0.53333336, 0.6862745 , 0.10196079,
                 0.6509804 , 1.        , 0.96862745, 0.49803922, 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.11764706, 0.14117648,
                 0.36862746, 0.6039216 , 0.6666667 , 0.99215686, 0.99215686,
                 0.99215686, 0.99215686, 0.99215686, 0.88235295, 0.6745098 ,
                 0.99215686, 0.9490196 , 0.7647059 , 0.2509804 , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.19215687, 0.93333334, 0.99215686,
                 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
                 0.99215686, 0.99215686, 0.9843137 , 0.3647059 , 0.32156864,
                 0.32156864, 0.21960784, 0.15294118, 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.07058824, 0.85882354, 0.99215686,
                 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.7764706 ,
                 0.7137255 , 0.96862745, 0.94509804, 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.3137255 , 0.6117647 ,
                 0.41960785, 0.99215686, 0.99215686, 0.8039216 , 0.04313726,
                 0.        , 0.16862746, 0.6039216 , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.05490196,
                 0.00392157, 0.6039216 , 0.99215686, 0.3529412 , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        ],
                [0.        , 0.        , 0.        , 0.        , 0.        ,
                 0.        , 0.        , 0.        , 0.        , 0.        ,
```

Entrada da MLP

In [16]:
```
train_images_mlp = train_images.reshape((60000, 28 * 28))
test_images_mlp = test_images.reshape((10000, 28 * 28))
```

In [17]:
```
train_images_mlp.shape
```

Out[17]: (60000, 784)

Entrada da rede convolucional

In [18]:
```
from keras import backend as K
```

In [19]:
```
img_rows = 28
img_cols = 28

if K.image_data_format() == 'channels_first':
    train_images = train_images.reshape(train_images.shape[0], 1, img_rows,
img_cols)
    test_images = test_images.reshape(test_images.shape[0], 1, img_rows, img
_cols)
    input_shape = (1, img_rows, img_cols)
else:
    train_images = train_images.reshape(train_images.shape[0], img_rows, img
_cols, 1)
    test_images = test_images.reshape(test_images.shape[0], img_rows, img_co
ls, 1)
    input_shape = (img_rows, img_cols, 1)
```

In [20]:
```
print('input shape:', input_shape)
```

```
input shape: (28, 28, 1)
```

Transformando rótulos em dados categóricos

In [21]:
```
from keras.utils import to_categorical

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

In [22]:
```
train_labels[0]
```

Out[22]: array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.])

model: https://keras.io/models/model/ (https://keras.io/models/model/)

layers: https://keras.io/layers/about-keras-layers/ (https://keras.io/layers/about-keras-layers/)

In [23]:
```
from keras import models
from keras import layers
```

Rede 1: MLP 2 camadas

In [24]:
```
#definindo a rede
#model: https://keras.io/models/model/
#layers: https://keras.io/layers/about-keras-layers/

network1 = models.Sequential()
network1.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network1.add(layers.Dense(10, activation='softmax'))
```

In [25]:
```
network1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 512)               401920    
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130      
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

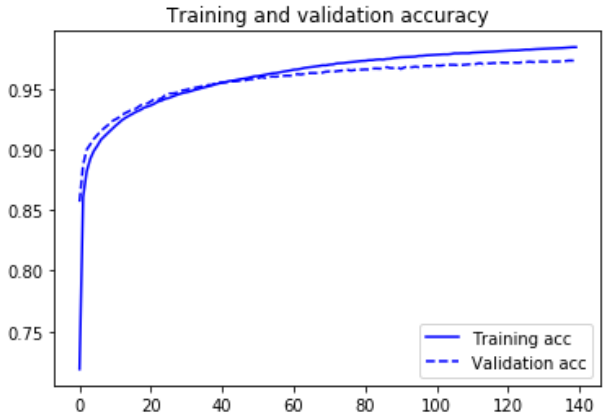
In [26]:
```
#paraleliza o modelo para treinamento em 2 GPUs
network1 = keras.utils.multi_gpu_model(network1,gpus=2)
```

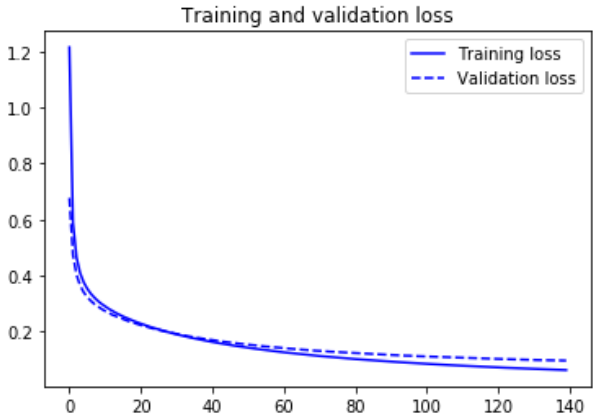
In [27]:
```
#compilando e treinando rede
#optimizers: https://keras.io/optimizers/
#loss funcitons: https://keras.io/losses/
#metrics: https://keras.io/metrics/

network1.compile(optimizer='sgd',
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])
history_mlp = network1.fit(train_images_mlp, train_labels, epochs=140, batch
_size=128, validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 2s 50us/step - loss: 1.2154 -
acc: 0.7188 - val_loss: 0.6776 - val_acc: 0.8569
Epoch 2/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.5892 -
acc: 0.8607 - val_loss: 0.4738 - val_acc: 0.8874
Epoch 3/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.4651 -
acc: 0.8819 - val_loss: 0.4031 - val_acc: 0.8998
Epoch 4/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.4097 -
acc: 0.8923 - val_loss: 0.3651 - val_acc: 0.9042
Epoch 5/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.3766 -
acc: 0.8988 - val_loss: 0.3409 - val_acc: 0.9087
Epoch 6/140
48000/48000 [==============================] - 2s 40us/step - loss: 0.3536 -
acc: 0.9032 - val_loss: 0.3236 - val_acc: 0.9120
Epoch 7/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.3357 -
acc: 0.9081 - val_loss: 0.3101 - val_acc: 0.9153
Epoch 8/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.3216 -
acc: 0.9110 - val_loss: 0.2989 - val_acc: 0.9178
Epoch 9/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.3096 -
acc: 0.9138 - val_loss: 0.2900 - val_acc: 0.9202
Epoch 10/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2994 -
acc: 0.9166 - val_loss: 0.2813 - val_acc: 0.9232
Epoch 11/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2903 -
acc: 0.9194 - val_loss: 0.2737 - val_acc: 0.9247
Epoch 12/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2819 -
acc: 0.9218 - val_loss: 0.2670 - val_acc: 0.9263
Epoch 13/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2742 -
acc: 0.9245 - val_loss: 0.2605 - val_acc: 0.9287
Epoch 14/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2672 -
acc: 0.9264 - val_loss: 0.2554 - val_acc: 0.9303
Epoch 15/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2608 -
acc: 0.9280 - val_loss: 0.2498 - val_acc: 0.9318
Epoch 16/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2545 -
acc: 0.9297 - val_loss: 0.2451 - val_acc: 0.9323
Epoch 17/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2489 -
acc: 0.9313 - val_loss: 0.2396 - val_acc: 0.9341
Epoch 18/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2435 -
acc: 0.9325 - val_loss: 0.2352 - val_acc: 0.9363
Epoch 19/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2383 -
acc: 0.9344 - val_loss: 0.2317 - val_acc: 0.9374
Epoch 20/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2332 -
acc: 0.9355 - val_loss: 0.2276 - val_acc: 0.9379
Epoch 21/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2288 -
acc: 0.9364 - val_loss: 0.2233 - val_acc: 0.9398
Epoch 22/140
48000/48000 [==============================] - 2s 41us/step - loss: 0.2241 -
acc: 0.9379 - val_loss: 0.2200 - val_acc: 0.9416
Epoch 23/140
```

In [28]:
```
#funcao que avalia a rede e retorna seus erros
def Avalia(hist,net,test_img,test_lab, i):#i:usado com early stopping, para
selecionar a rede certa, 1 se nao for usado early stopping
    acc = hist.history['acc']
    val_acc = hist.history['val_acc']
    loss = hist.history['loss']
    val_loss = hist.history['val_loss']

    epochs = range(len(acc))

    plt.plot(epochs, acc, 'b', label='Training acc')
    plt.plot(epochs, val_acc, 'b--', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()

    plt.figure()

    plt.plot(epochs, loss, 'b', label='Training loss')
    plt.plot(epochs, val_loss, 'b--', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()

    plt.show()

    test_loss, test_acc = net.evaluate(test_img, test_lab)
    print('test acc=', test_acc)
    print('training accuracy=',hist.history['acc'][-i])
    print('validation accuracy=',hist.history['val_acc'][-i])
    print('test err=', test_loss)
    print('training err=',hist.history['loss'][-i])
    print('validation err=',hist.history['val_loss'][-i])
```

In [29]:
```
Avalia(history_mlp,network1,test_images_mlp,test_labels,1)
```






```
10000/10000 [==============================] - 1s 61us/step
test acc= 0.9747
training accuracy= 0.9843958333333334
validation accuracy= 0.973
test err= 0.08827327094227076
training err= 0.06306640438735485
validation err= 0.0968948900004228
```

Rede 2: Convolucional 7 camadas

In [31]:
```
#definindo rede
network2 = models.Sequential()
network2.add(layers.Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
network2.add(layers.Conv2D(64, (3, 3), activation='relu'))
network2.add(layers.Conv2D(64, (3, 3), activation='relu'))
network2.add(layers.Conv2D(64, (3, 3), activation='relu'))
network2.add(layers.Flatten())
network2.add(layers.Dense(128, activation='relu'))
network2.add(layers.Dense(64, activation='relu'))
network2.add(layers.Dense(10, activation='softmax'))
```

In [32]:
```
network2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 64)        36928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 20, 20, 64)        36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               3276928
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 3,378,506
Trainable params: 3,378,506
Non-trainable params: 0
_________________________________________________________________
```

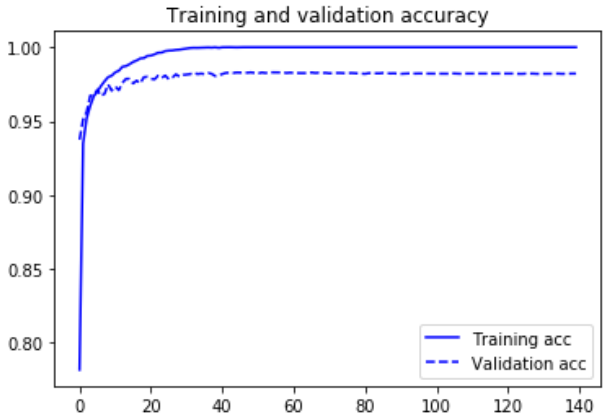
In [33]:
```
network2 = keras.utils.multi_gpu_model(network2,gpus=2) #paralelizando a red
e para treinamento em 2 GPUs
```

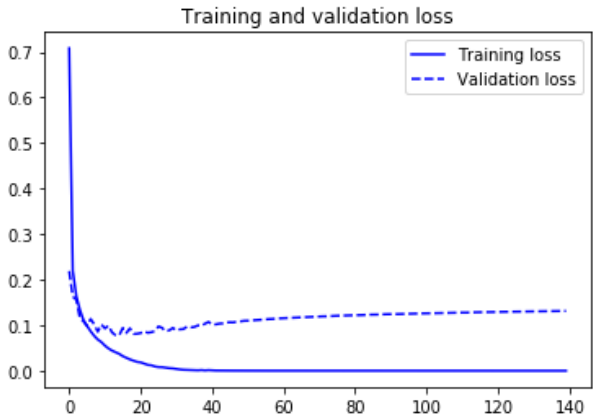
In [34]:
```
#Compilando rede
network2.compile(optimizer='sgd',
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])
```

In [35]:
```
#treinando rede
history = network2.fit(train_images, train_labels,
                       batch_size=128,
                       epochs=140,
                       validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 7s 149us/step - loss: 0.7083 -
acc: 0.7818 - val_loss: 0.2191 - val_acc: 0.9374
Epoch 2/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.2195 -
acc: 0.9349 - val_loss: 0.1630 - val_acc: 0.9512
Epoch 3/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.1637 -
acc: 0.9520 - val_loss: 0.1560 - val_acc: 0.9563
Epoch 4/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.1337 -
acc: 0.9604 - val_loss: 0.1154 - val_acc: 0.9672
Epoch 5/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.1112 -
acc: 0.9669 - val_loss: 0.1086 - val_acc: 0.9672
Epoch 6/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0983 -
acc: 0.9700 - val_loss: 0.0984 - val_acc: 0.9713
Epoch 7/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0868 -
acc: 0.9738 - val_loss: 0.1138 - val_acc: 0.9671
Epoch 8/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0766 -
acc: 0.9770 - val_loss: 0.1019 - val_acc: 0.9685
Epoch 9/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0686 -
acc: 0.9798 - val_loss: 0.0858 - val_acc: 0.9748
Epoch 10/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0630 -
acc: 0.9808 - val_loss: 0.1016 - val_acc: 0.9706
Epoch 11/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0552 -
acc: 0.9830 - val_loss: 0.0931 - val_acc: 0.9734
Epoch 12/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0490 -
acc: 0.9841 - val_loss: 0.0991 - val_acc: 0.9709
Epoch 13/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0442 -
acc: 0.9868 - val_loss: 0.0840 - val_acc: 0.9759
Epoch 14/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0407 -
acc: 0.9873 - val_loss: 0.0781 - val_acc: 0.9786
Epoch 15/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0370 -
acc: 0.9885 - val_loss: 0.0781 - val_acc: 0.9787
Epoch 16/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0320 -
acc: 0.9901 - val_loss: 0.0941 - val_acc: 0.9755
Epoch 17/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0283 -
acc: 0.9910 - val_loss: 0.0831 - val_acc: 0.9774
Epoch 18/140
48000/48000 [==============================] - 6s 122us/step - loss: 0.0249 -
acc: 0.9924 - val_loss: 0.0932 - val_acc: 0.9755
Epoch 19/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0222 -
acc: 0.9927 - val_loss: 0.0809 - val_acc: 0.9798
Epoch 20/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0194 -
acc: 0.9941 - val_loss: 0.0812 - val_acc: 0.9800
Epoch 21/140
48000/48000 [==============================] - 6s 123us/step - loss: 0.0184 -
acc: 0.9944 - val_loss: 0.0828 - val_acc: 0.9787
Epoch 22/140
48000/48000 [==============================] - 6s 122us/step - loss: 0.0158 -
acc: 0.9952 - val_loss: 0.0865 - val_acc: 0.9782
Epoch 23/140
```

In [36]:
```
#avaliando rede
Avalia(history,network2,test_images,test_labels,1)
```






```
10000/10000 [==============================] - 2s 172us/step
test acc= 0.9822
training accuracy= 1.0
validation accuracy= 0.9821666668256124
test err= 0.1251584162220022
training err= 2.685475903323701e-05
validation err= 0.13133609068408775
```

Rede 3: Convolucional com MaxPooling

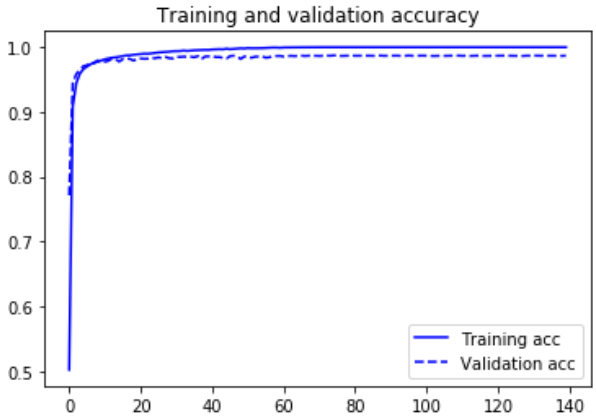
In [38]:
```
#Definindo rede
network3 = models.Sequential()
network3.add(layers.Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
network3.add(layers.Conv2D(64, (3, 3), activation='relu'))
network3.add(layers.Conv2D(64, (3, 3), activation='relu'))
network3.add(layers.MaxPooling2D(pool_size=(2, 2)))
network3.add(layers.Conv2D(64, (3, 3), activation='relu'))
network3.add(layers.MaxPooling2D(pool_size=(2, 2)))
network3.add(layers.Flatten())
network3.add(layers.Dense(128, activation='relu'))
network3.add(layers.Dense(64, activation='relu'))
network3.add(layers.Dense(10, activation='softmax'))

network3.summary()
```

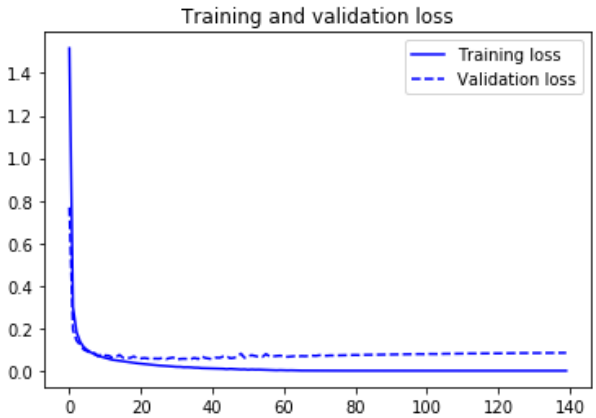
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 24, 24, 64)        18496     
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 22, 22, 64)        36928     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 64)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 9, 9, 64)          36928     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 1024)              0         
_________________________________________________________________
dense_6 (Dense)              (None, 128)               131200    
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_8 (Dense)              (None, 10)                650       
=================================================================
Total params: 232,778
Trainable params: 232,778
Non-trainable params: 0
_________________________________________________________________
```

In [39]:
```
#compilando e treinando
network3 = keras.utils.multi_gpu_model(network3,gpus=2)

network3.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])

history3=network3.fit(train_images, train_labels,
                      batch_size=128,
                      epochs=140,
                      validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 5s 98us/step - loss: 1.5170 -
acc: 0.5021 - val_loss: 0.7682 - val_acc: 0.7715
Epoch 2/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.2989 -
acc: 0.9089 - val_loss: 0.1927 - val_acc: 0.9445
Epoch 3/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.1820 -
acc: 0.9449 - val_loss: 0.1424 - val_acc: 0.9587
Epoch 4/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.1367 -
acc: 0.9593 - val_loss: 0.1222 - val_acc: 0.9649
Epoch 5/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.1113 -
acc: 0.9664 - val_loss: 0.0997 - val_acc: 0.9709
Epoch 6/140
48000/48000 [==============================] - 5s 94us/step - loss: 0.0971 -
acc: 0.9704 - val_loss: 0.0917 - val_acc: 0.9721
Epoch 7/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0867 -
acc: 0.9737 - val_loss: 0.0823 - val_acc: 0.9758
Epoch 8/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0773 -
acc: 0.9766 - val_loss: 0.0833 - val_acc: 0.9751
Epoch 9/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0702 -
acc: 0.9782 - val_loss: 0.0692 - val_acc: 0.9797
Epoch 10/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0657 -
acc: 0.9795 - val_loss: 0.0786 - val_acc: 0.9762
Epoch 11/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0603 -
acc: 0.9815 - val_loss: 0.0719 - val_acc: 0.9777
Epoch 12/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0561 -
acc: 0.9827 - val_loss: 0.0711 - val_acc: 0.9791
Epoch 13/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0517 -
acc: 0.9838 - val_loss: 0.0619 - val_acc: 0.9825
Epoch 14/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0492 -
acc: 0.9844 - val_loss: 0.0632 - val_acc: 0.9813
Epoch 15/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0462 -
acc: 0.9854 - val_loss: 0.0747 - val_acc: 0.9779
Epoch 16/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.0438 -
acc: 0.9861 - val_loss: 0.0612 - val_acc: 0.9816
Epoch 17/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.0411 -
acc: 0.9874 - val_loss: 0.0610 - val_acc: 0.9823
Epoch 18/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0388 -
acc: 0.9875 - val_loss: 0.0604 - val_acc: 0.9821
Epoch 19/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.0364 -
acc: 0.9884 - val_loss: 0.0674 - val_acc: 0.9793
Epoch 20/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.0353 -
acc: 0.9889 - val_loss: 0.0594 - val_acc: 0.9819
Epoch 21/140
48000/48000 [==============================] - 5s 96us/step - loss: 0.0334 -
acc: 0.9893 - val_loss: 0.0580 - val_acc: 0.9826
Epoch 22/140
48000/48000 [==============================] - 5s 95us/step - loss: 0.0308 -
acc: 0.9901 - val_loss: 0.0601 - val_acc: 0.9822
Epoch 23/140
```

In [40]:
```
#avaliando rede
Avalia(history3,network3,test_images,test_labels,1)
```






```
10000/10000 [==============================] - 1s 84us/step
test acc= 0.9875
training accuracy= 1.0
validation accuracy= 0.9869166665077209
test err= 0.06799678473660249
training err= 7.754406777651941e-05
validation err= 0.08442384547831004
```

Rede 4: Convolucional com MaxPooling e Dropout

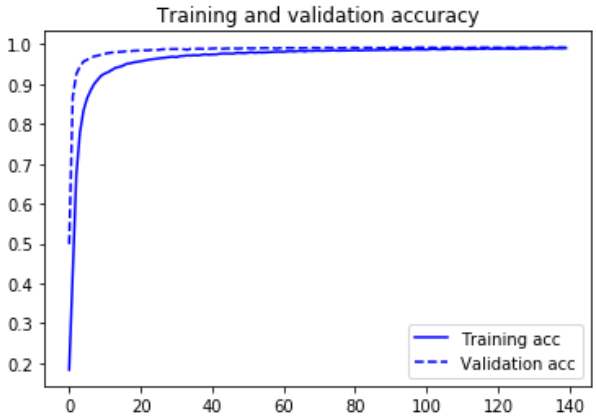
In [42]:
```
#Definindo rede
network4 = models.Sequential()
network4.add(layers.Conv2D(32, kernel_size=(3, 3),
                 activation='relu',input_shape=input_shape))
network4.add(layers.Conv2D(64, (3, 3), activation='relu'))
network4.add(layers.Conv2D(64, (3, 3), activation='relu'))
network4.add(layers.MaxPooling2D(pool_size=(2, 2)))
network4.add(layers.Conv2D(64, (3, 3), activation='relu'))
network4.add(layers.MaxPooling2D(pool_size=(2, 2)))
network4.add(layers.Dropout(0.25))
network4.add(layers.Flatten())
network4.add(layers.Dense(128, activation='relu'))
network4.add(layers.Dropout(0.5))
network4.add(layers.Dense(64, activation='relu'))
network4.add(layers.Dropout(0.5))
network4.add(layers.Dense(10, activation='softmax'))

network4.summary()
```

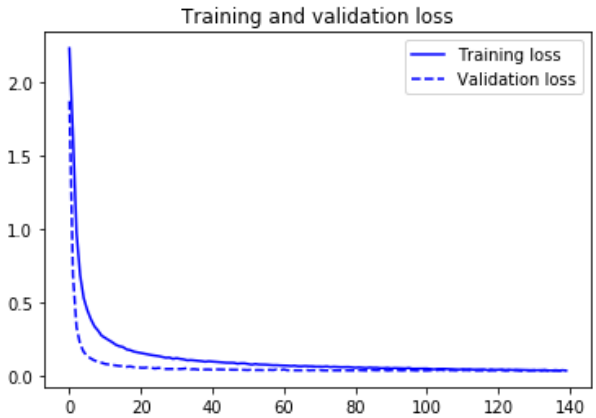
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 22, 22, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 9, 9, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               131200
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_11 (Dense)             (None, 10)                650
=================================================================
Total params: 232,778
Trainable params: 232,778
Non-trainable params: 0
_________________________________________________________________
```

In [43]:
```
#compilando e treinando
network4 = keras.utils.multi_gpu_model(network4,gpus=2)

network4.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])

history4=network4.fit(train_images, train_labels,
                      batch_size=128,
                      epochs=140,
                      validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 5s 107us/step - loss: 2.2324 -
acc: 0.1829 - val_loss: 1.8721 - val_acc: 0.4985
Epoch 2/140
48000/48000 [==============================] - 5s 103us/step - loss: 1.6359 -
acc: 0.4259 - val_loss: 0.6838 - val_acc: 0.8683
Epoch 3/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.9745 -
acc: 0.6725 - val_loss: 0.3309 - val_acc: 0.9255
Epoch 4/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.6860 -
acc: 0.7782 - val_loss: 0.2157 - val_acc: 0.9442
Epoch 5/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.5298 -
acc: 0.8337 - val_loss: 0.1628 - val_acc: 0.9568
Epoch 6/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.4481 -
acc: 0.8643 - val_loss: 0.1346 - val_acc: 0.9615
Epoch 7/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.3863 -
acc: 0.8842 - val_loss: 0.1178 - val_acc: 0.9658
Epoch 8/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.3376 -
acc: 0.9009 - val_loss: 0.1042 - val_acc: 0.9700
Epoch 9/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.3086 -
acc: 0.9117 - val_loss: 0.0975 - val_acc: 0.9716
Epoch 10/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.2738 -
acc: 0.9218 - val_loss: 0.0894 - val_acc: 0.9742
Epoch 11/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.2582 -
acc: 0.9268 - val_loss: 0.0807 - val_acc: 0.9762
Epoch 12/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.2430 -
acc: 0.9308 - val_loss: 0.0771 - val_acc: 0.9782
Epoch 13/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.2286 -
acc: 0.9359 - val_loss: 0.0733 - val_acc: 0.9795
Epoch 14/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.2113 -
acc: 0.9413 - val_loss: 0.0702 - val_acc: 0.9806
Epoch 15/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.2012 -
acc: 0.9432 - val_loss: 0.0667 - val_acc: 0.9811
Epoch 16/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.1962 -
acc: 0.9462 - val_loss: 0.0667 - val_acc: 0.9815
Epoch 17/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.1795 -
acc: 0.9510 - val_loss: 0.0612 - val_acc: 0.9829
Epoch 18/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.1751 -
acc: 0.9521 - val_loss: 0.0636 - val_acc: 0.9827
Epoch 19/140
48000/48000 [==============================] - 5s 104us/step - loss: 0.1649 -
acc: 0.9547 - val_loss: 0.0586 - val_acc: 0.9837
Epoch 20/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.1600 -
acc: 0.9561 - val_loss: 0.0583 - val_acc: 0.9842
Epoch 21/140
48000/48000 [==============================] - 5s 103us/step - loss: 0.1551 -
acc: 0.9574 - val_loss: 0.0548 - val_acc: 0.9847
Epoch 22/140
48000/48000 [==============================] - 5s 105us/step - loss: 0.1492 -
acc: 0.9590 - val_loss: 0.0545 - val_acc: 0.9849
Epoch 23/140
```

In [44]:
```
#avaliando rede
Avalia(history4,network4,test_images,test_labels,1)
```






```
10000/10000 [==============================] - 1s 95us/step
test acc= 0.9946
training accuracy= 0.9907083333333333
validation accuracy= 0.9922499998410543
test err= 0.023490405937680953
training err= 0.035374769183186196
validation err= 0.03634200700823506
```

Rede 5: Convolucional com MaxPooling, Dropout e Weight Decay (norma L2)

In [46]:
```
#Definindo rede
network5 = models.Sequential()
network5.add(layers.Conv2D(32, kernel_size=(3, 3),
                 activation='relu',input_shape=input_shape))
network5.add(layers.Conv2D(64, (3, 3), activation='relu'))
network5.add(layers.Conv2D(64, (3, 3), activation='relu'))
network5.add(layers.MaxPooling2D(pool_size=(2, 2)))
network5.add(layers.Conv2D(64, (3, 3), activation='relu'))
network5.add(layers.MaxPooling2D(pool_size=(2, 2)))
network5.add(layers.Dropout(0.25))
network5.add(layers.Flatten())
network5.add(layers.Dense(128, activation='relu', activity_regularizer=keras
.regularizers.l2(0.0001)))
network5.add(layers.Dropout(0.5))
network5.add(layers.Dense(64, activation='relu', activity_regularizer=keras.
regularizers.l2(0.0001)))
network5.add(layers.Dropout(0.5))
network5.add(layers.Dense(10, activation='softmax'))

network5.summary()
```

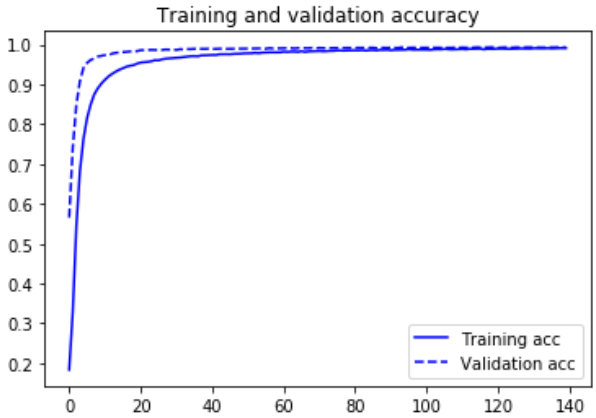
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 22, 22, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 9, 9, 64)          36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               131200
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                650
=================================================================
Total params: 232,778
Trainable params: 232,778
Non-trainable params: 0
_________________________________________________________________
```

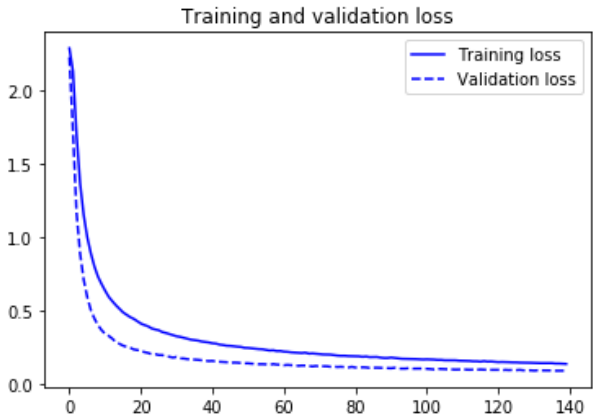
In [47]:
```
#COMPILANDO
network5 = keras.utils.multi_gpu_model(network5,gpus=2)
network5.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
```

In [48]:
```
#TREINANDO

history5=network5.fit(train_images, train_labels,
                      batch_size=128,
                      epochs=140,
                      validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 5s 104us/step - loss: 2.2820 -
acc: 0.1836 - val_loss: 2.2215 - val_acc: 0.5657
Epoch 2/140
48000/48000 [==============================] - 5s 98us/step - loss: 2.1258 -
acc: 0.3287 - val_loss: 1.7159 - val_acc: 0.7405
Epoch 3/140
48000/48000 [==============================] - 5s 98us/step - loss: 1.6976 -
acc: 0.5433 - val_loss: 1.1640 - val_acc: 0.8487
Epoch 4/140
48000/48000 [==============================] - 5s 98us/step - loss: 1.3627 -
acc: 0.6817 - val_loss: 0.8856 - val_acc: 0.9082
Epoch 5/140
48000/48000 [==============================] - 5s 98us/step - loss: 1.1489 -
acc: 0.7634 - val_loss: 0.7096 - val_acc: 0.9414
Epoch 6/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.9985 -
acc: 0.8151 - val_loss: 0.5907 - val_acc: 0.9540
Epoch 7/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.8892 -
acc: 0.8491 - val_loss: 0.5027 - val_acc: 0.9613
Epoch 8/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.7998 -
acc: 0.8747 - val_loss: 0.4456 - val_acc: 0.9648
Epoch 9/140
48000/48000 [==============================] - 5s 97us/step - loss: 0.7301 -
acc: 0.8900 - val_loss: 0.4007 - val_acc: 0.9698
Epoch 10/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.6801 -
acc: 0.9019 - val_loss: 0.3668 - val_acc: 0.9715
Epoch 11/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.6349 -
acc: 0.9115 - val_loss: 0.3417 - val_acc: 0.9734
Epoch 12/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.5934 -
acc: 0.9199 - val_loss: 0.3277 - val_acc: 0.9736
Epoch 13/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.5636 -
acc: 0.9264 - val_loss: 0.3089 - val_acc: 0.9771
Epoch 14/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.5360 -
acc: 0.9319 - val_loss: 0.2867 - val_acc: 0.9792
Epoch 15/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.5117 -
acc: 0.9368 - val_loss: 0.2746 - val_acc: 0.9802
Epoch 16/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.4877 -
acc: 0.9403 - val_loss: 0.2614 - val_acc: 0.9813
Epoch 17/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.4722 -
acc: 0.9444 - val_loss: 0.2533 - val_acc: 0.9822
Epoch 18/140
48000/48000 [==============================] - 5s 97us/step - loss: 0.4553 -
acc: 0.9467 - val_loss: 0.2465 - val_acc: 0.9826
Epoch 19/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.4434 -
acc: 0.9482 - val_loss: 0.2357 - val_acc: 0.9843
Epoch 20/140
48000/48000 [==============================] - 5s 97us/step - loss: 0.4271 -
acc: 0.9521 - val_loss: 0.2316 - val_acc: 0.9832
Epoch 21/140
48000/48000 [==============================] - 5s 97us/step - loss: 0.4114 -
acc: 0.9549 - val_loss: 0.2263 - val_acc: 0.9859
Epoch 22/140
48000/48000 [==============================] - 5s 98us/step - loss: 0.4013 -
acc: 0.9558 - val_loss: 0.2173 - val_acc: 0.9857
Epoch 23/140
```

In [49]:
```
#avaliando
Avalia(history5,network5,test_images,test_labels,1)
```






```
10000/10000 [==============================] - 1s 103us/step
test acc= 0.9947
training accuracy= 0.9911875
validation accuracy= 0.9928333331743876
test err= 0.03683618170768022
training err= 0.13717680968840917
validation err= 0.09076368876298269
```

Rede 6: Convolucional com MaxPooling, Dropout e Weight Decay (norma L2) maior que na rede anterior

In [51]:
```
#Definindo rede
network6 = models.Sequential()
network6.add(layers.Conv2D(32, kernel_size=(3, 3),
                 activation='relu',input_shape=input_shape))
network6.add(layers.Conv2D(64, (3, 3), activation='relu'))
network6.add(layers.Conv2D(64, (3, 3), activation='relu'))
network6.add(layers.MaxPooling2D(pool_size=(2, 2)))
network6.add(layers.Conv2D(64, (3, 3), activation='relu'))
network6.add(layers.MaxPooling2D(pool_size=(2, 2)))
network6.add(layers.Dropout(0.25))
network6.add(layers.Flatten())
network6.add(layers.Dense(128, activation='relu', activity_regularizer=keras
.regularizers.l2(0.005)))
network6.add(layers.Dropout(0.5))
network6.add(layers.Dense(64, activation='relu', activity_regularizer=keras.
regularizers.l2(0.005)))
network6.add(layers.Dropout(0.5))
network6.add(layers.Dense(10, activation='softmax'))

network6.summary()
```

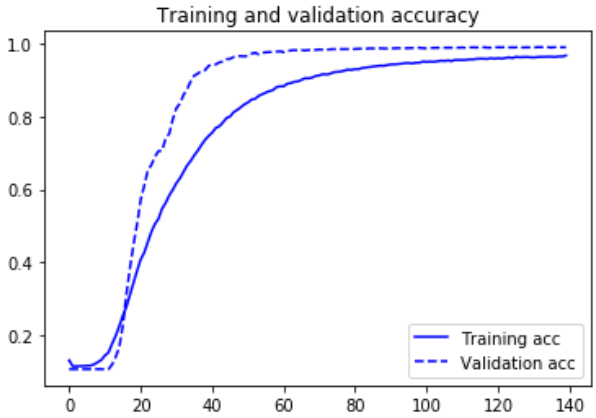
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 22, 22, 64)        36928
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 9, 9, 64)          36928
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_7 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               131200
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                650
=================================================================
Total params: 232,778
Trainable params: 232,778
Non-trainable params: 0
_________________________________________________________________
```

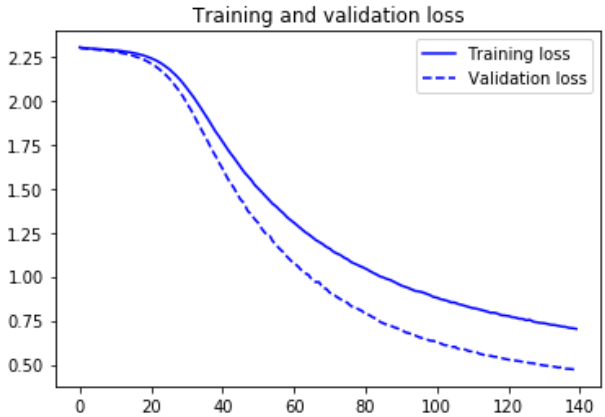
In [52]:
```
#COMPILANDO
network6 = keras.utils.multi_gpu_model(network6,gpus=2)
network6.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
```

In [53]:
```
#TREINANDO
history6=network6.fit(train_images, train_labels,
                      batch_size=128,
                      epochs=140,
                      validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 5s 109us/step - loss: 2.3032 -
acc: 0.1305 - val_loss: 2.2995 - val_acc: 0.1060
Epoch 2/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2993 -
acc: 0.1141 - val_loss: 2.2973 - val_acc: 0.1060
Epoch 3/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2978 -
acc: 0.1141 - val_loss: 2.2955 - val_acc: 0.1060
Epoch 4/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2966 -
acc: 0.1143 - val_loss: 2.2939 - val_acc: 0.1060
Epoch 5/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2954 -
acc: 0.1148 - val_loss: 2.2920 - val_acc: 0.1060
Epoch 6/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2943 -
acc: 0.1154 - val_loss: 2.2900 - val_acc: 0.1060
Epoch 7/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2929 -
acc: 0.1161 - val_loss: 2.2878 - val_acc: 0.1060
Epoch 8/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2913 -
acc: 0.1195 - val_loss: 2.2855 - val_acc: 0.1060
Epoch 9/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2898 -
acc: 0.1253 - val_loss: 2.2832 - val_acc: 0.1060
Epoch 10/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2880 -
acc: 0.1317 - val_loss: 2.2805 - val_acc: 0.1060
Epoch 11/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2860 -
acc: 0.1436 - val_loss: 2.2775 - val_acc: 0.1060
Epoch 12/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2840 -
acc: 0.1519 - val_loss: 2.2740 - val_acc: 0.1062
Epoch 13/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2812 -
acc: 0.1758 - val_loss: 2.2700 - val_acc: 0.1188
Epoch 14/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2784 -
acc: 0.1975 - val_loss: 2.2654 - val_acc: 0.1392
Epoch 15/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2749 -
acc: 0.2242 - val_loss: 2.2603 - val_acc: 0.1663
Epoch 16/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2709 -
acc: 0.2529 - val_loss: 2.2547 - val_acc: 0.2252
Epoch 17/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2667 -
acc: 0.2775 - val_loss: 2.2481 - val_acc: 0.3114
Epoch 18/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2616 -
acc: 0.3108 - val_loss: 2.2407 - val_acc: 0.3841
Epoch 19/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2558 -
acc: 0.3437 - val_loss: 2.2321 - val_acc: 0.4486
Epoch 20/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2488 -
acc: 0.3770 - val_loss: 2.2223 - val_acc: 0.5059
Epoch 21/140
48000/48000 [==============================] - 5s 100us/step - loss: 2.2406 -
acc: 0.4078 - val_loss: 2.2108 - val_acc: 0.5721
Epoch 22/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.2323 -
acc: 0.4269 - val_loss: 2.1981 - val_acc: 0.6106
Epoch 23/140
```

In [54]:
```
Avalia(history6,network6,test_images,test_labels,1)
```






```
10000/10000 [==============================] - 1s 101us/step
test acc= 0.9927
training accuracy= 0.9684375
validation accuracy= 0.9910833331743876
test err= 0.2668167231321335
training err= 0.7030133541425069
validation err= 0.468558078845342
```

Rede 7: MLP de 4 camadas

In [75]:
```
#Observa-se que o MLP que foi feito (network1) teve underfitting, portanto,
sera criado um MLP com mais capacidade

network7 = models.Sequential()
network7.add(layers.Dense(256, activation='relu', input_shape=(28 * 28,)))
network7.add(layers.Dense(256, activation='relu'))
network7.add(layers.Dense(256, activation='relu'))
network7.add(layers.Dense(256, activation='relu'))
network7.add(layers.Dense(256, activation='relu'))
network7.add(layers.Dense(256, activation='relu'))
network7.add(layers.Dense(10, activation='softmax'))
```

In [76]:
```
network7.summary()
```

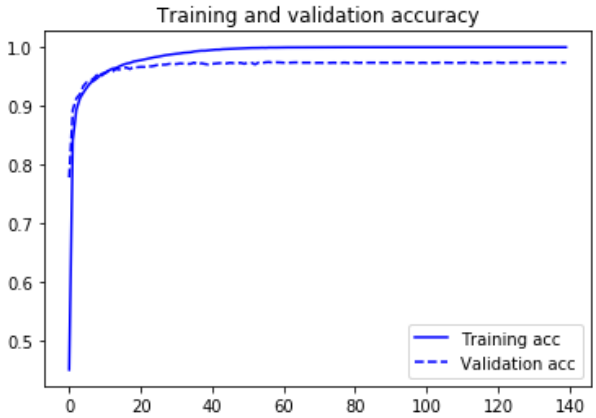
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_37 (Dense)             (None, 256)               200960
_________________________________________________________________
dense_38 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_39 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_40 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_41 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_42 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_43 (Dense)             (None, 10)                2570
=================================================================
Total params: 532,490
Trainable params: 532,490
Non-trainable params: 0
_________________________________________________________________
```

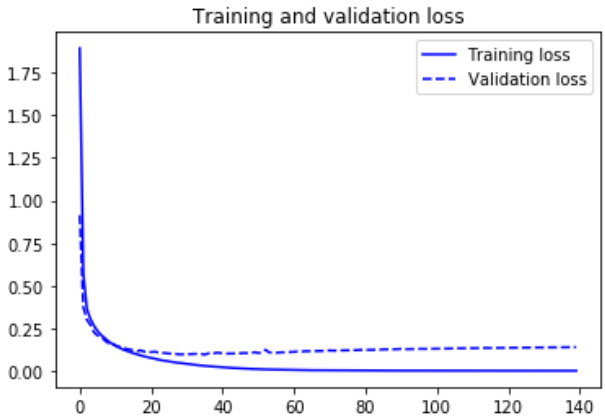
In [77]:
```
#COMPILANDO
network7 = keras.utils.multi_gpu_model(network7,gpus=2)
network7.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
```

In [78]:
```
#TREINANDO

history7 = network7.fit(train_images_mlp, train_labels, epochs=140, batch_si
ze=128, validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 3s 68us/step - loss: 1.8938 -
acc: 0.4495 - val_loss: 0.9134 - val_acc: 0.7777
Epoch 2/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.5712 -
acc: 0.8409 - val_loss: 0.3740 - val_acc: 0.8928
Epoch 3/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.3640 -
acc: 0.8931 - val_loss: 0.3001 - val_acc: 0.9126
Epoch 4/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.3034 -
acc: 0.9127 - val_loss: 0.2700 - val_acc: 0.9197
Epoch 5/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.2643 -
acc: 0.9229 - val_loss: 0.2261 - val_acc: 0.9341
Epoch 6/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.2335 -
acc: 0.9318 - val_loss: 0.2071 - val_acc: 0.9406
Epoch 7/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.2100 -
acc: 0.9394 - val_loss: 0.2043 - val_acc: 0.9426
Epoch 8/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1915 -
acc: 0.9433 - val_loss: 0.1786 - val_acc: 0.9477
Epoch 9/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1741 -
acc: 0.9481 - val_loss: 0.1628 - val_acc: 0.9539
Epoch 10/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1591 -
acc: 0.9542 - val_loss: 0.1644 - val_acc: 0.9513
Epoch 11/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1484 -
acc: 0.9566 - val_loss: 0.1484 - val_acc: 0.9555
Epoch 12/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1369 -
acc: 0.9600 - val_loss: 0.1399 - val_acc: 0.9590
Epoch 13/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1265 -
acc: 0.9633 - val_loss: 0.1409 - val_acc: 0.9577
Epoch 14/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1188 -
acc: 0.9643 - val_loss: 0.1292 - val_acc: 0.9628
Epoch 15/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1114 -
acc: 0.9671 - val_loss: 0.1257 - val_acc: 0.9618
Epoch 16/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.1035 -
acc: 0.9700 - val_loss: 0.1207 - val_acc: 0.9638
Epoch 17/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.0971 -
acc: 0.9720 - val_loss: 0.1152 - val_acc: 0.9653
Epoch 18/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.0915 -
acc: 0.9731 - val_loss: 0.1205 - val_acc: 0.9626
Epoch 19/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.0853 -
acc: 0.9756 - val_loss: 0.1129 - val_acc: 0.9664
Epoch 20/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.0804 -
acc: 0.9766 - val_loss: 0.1133 - val_acc: 0.9658
Epoch 21/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.0756 -
acc: 0.9778 - val_loss: 0.1101 - val_acc: 0.9664
Epoch 22/140
48000/48000 [==============================] - 3s 55us/step - loss: 0.0717 -
acc: 0.9792 - val_loss: 0.1126 - val_acc: 0.9663
Epoch 23/140
```

In [79]:
```
#avaliando
Avalia(history7,network7,test_images_mlp,test_labels,1)
```






```
10000/10000 [==============================] - 1s 77us/step
test acc= 0.973
training accuracy= 0.9999791666666666
validation accuracy= 0.9737499998410543
test err= 0.14596334818315704
training err= 0.0007363217900274321
validation err= 0.13942046153172852
```

Rede 8: MLP de 4 camadas com dropout

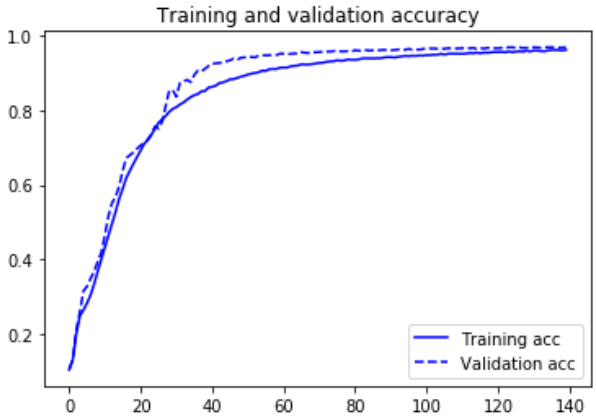
In [80]:
```
#definindo rede
network8 = models.Sequential()
network8.add(layers.Dense(256, activation='relu', input_shape=(28 * 28,)))
network8.add(layers.Dropout(0.5))
network8.add(layers.Dense(256, activation='relu'))
network8.add(layers.Dropout(0.5))
network8.add(layers.Dense(256, activation='relu'))
network8.add(layers.Dropout(0.5))
network8.add(layers.Dense(256, activation='relu'))
network8.add(layers.Dropout(0.5))
network8.add(layers.Dense(256, activation='relu'))
network8.add(layers.Dropout(0.5))
network8.add(layers.Dense(256, activation='relu'))
network8.add(layers.Dropout(0.5))
network8.add(layers.Dense(10, activation='softmax'))

network8.summary()
```

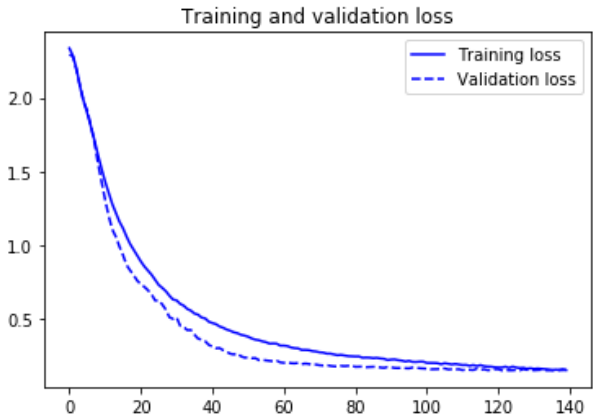
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_44 (Dense)             (None, 256)               200960
_________________________________________________________________
dropout_18 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_45 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_19 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_46 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_20 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_47 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_21 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_48 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_22 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_49 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_23 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_50 (Dense)             (None, 10)                2570
=================================================================
Total params: 532,490
Trainable params: 532,490
Non-trainable params: 0
_________________________________________________________________
```

In [81]:
```
#paralelizando, compilando e treinando
network8 = keras.utils.multi_gpu_model(network8,gpus=2)
network8.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
history8 = network8.fit(train_images_mlp, train_labels, epochs=140, batch_si
ze=128, validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 4s 84us/step - loss: 2.3374 -
acc: 0.1039 - val_loss: 2.2942 - val_acc: 0.1060
Epoch 2/140
48000/48000 [==============================] - 3s 72us/step - loss: 2.2851 -
acc: 0.1303 - val_loss: 2.2808 - val_acc: 0.1311
Epoch 3/140
48000/48000 [==============================] - 3s 71us/step - loss: 2.1972 -
acc: 0.2009 - val_loss: 2.1862 - val_acc: 0.2112
Epoch 4/140
48000/48000 [==============================] - 3s 72us/step - loss: 2.0744 -
acc: 0.2480 - val_loss: 2.0769 - val_acc: 0.2598
Epoch 5/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.9775 -
acc: 0.2644 - val_loss: 1.9703 - val_acc: 0.3155
Epoch 6/140
48000/48000 [==============================] - 3s 73us/step - loss: 1.8959 -
acc: 0.2825 - val_loss: 1.9142 - val_acc: 0.3275
Epoch 7/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.8068 -
acc: 0.3048 - val_loss: 1.8114 - val_acc: 0.3488
Epoch 8/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.7121 -
acc: 0.3342 - val_loss: 1.6999 - val_acc: 0.3678
Epoch 9/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.6121 -
acc: 0.3683 - val_loss: 1.5544 - val_acc: 0.3944
Epoch 10/140
48000/48000 [==============================] - 3s 73us/step - loss: 1.5182 -
acc: 0.4001 - val_loss: 1.4318 - val_acc: 0.4187
Epoch 11/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.4305 -
acc: 0.4307 - val_loss: 1.3068 - val_acc: 0.4649
Epoch 12/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.3541 -
acc: 0.4634 - val_loss: 1.1960 - val_acc: 0.5172
Epoch 13/140
48000/48000 [==============================] - 3s 73us/step - loss: 1.2780 -
acc: 0.4968 - val_loss: 1.1051 - val_acc: 0.5520
Epoch 14/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.2175 -
acc: 0.5284 - val_loss: 1.0551 - val_acc: 0.5668
Epoch 15/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.1579 -
acc: 0.5631 - val_loss: 0.9890 - val_acc: 0.5972
Epoch 16/140
48000/48000 [==============================] - 3s 73us/step - loss: 1.1133 -
acc: 0.5900 - val_loss: 0.9309 - val_acc: 0.6391
Epoch 17/140
48000/48000 [==============================] - 3s 72us/step - loss: 1.0563 -
acc: 0.6193 - val_loss: 0.8620 - val_acc: 0.6722
Epoch 18/140
48000/48000 [==============================] - 4s 73us/step - loss: 1.0057 -
acc: 0.6384 - val_loss: 0.8262 - val_acc: 0.6796
Epoch 19/140
48000/48000 [==============================] - 4s 73us/step - loss: 0.9669 -
acc: 0.6572 - val_loss: 0.7933 - val_acc: 0.6872
Epoch 20/140
48000/48000 [==============================] - 3s 72us/step - loss: 0.9267 -
acc: 0.6745 - val_loss: 0.7562 - val_acc: 0.6972
Epoch 21/140
48000/48000 [==============================] - 3s 72us/step - loss: 0.8872 -
acc: 0.6919 - val_loss: 0.7319 - val_acc: 0.7058
Epoch 22/140
48000/48000 [==============================] - 3s 73us/step - loss: 0.8527 -
acc: 0.7101 - val_loss: 0.7144 - val_acc: 0.7121
Epoch 23/140
```

In [82]:
```
#avaliando
Avalia(history8,network8,test_images_mlp,test_labels,1)
```






```
10000/10000 [==============================] - 1s 86us/step
test acc= 0.9683
training accuracy= 0.9620833333333333
validation accuracy= 0.9689166668256124
test err= 0.15233140578726306
training err= 0.148334436049064
validation err= 0.15050022852793335
```

Rede 9: MLP de 4 camadas com weight decay (norma L2)

In [84]:

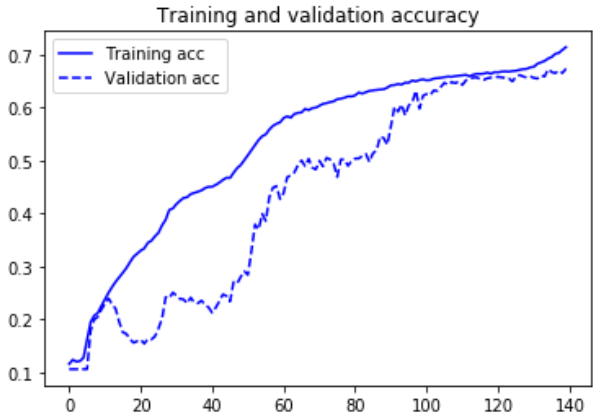
```
#definindo rede

network9 = models.Sequential()
network9.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,),
                          activity_regularizer=keras.regularizers.l2(0.0001)
))
network9.add(layers.Dropout(0.5))
network9.add(layers.Dense(512, activation='relu', activity_regularizer=keras
.regularizers.l2(0.0001)))
network9.add(layers.Dropout(0.5))
network9.add(layers.Dense(512, activation='relu', activity_regularizer=keras
.regularizers.l2(0.0001)))
network9.add(layers.Dropout(0.5))
network9.add(layers.Dense(512, activation='relu', activity_regularizer=keras
.regularizers.l2(0.0001)))
network9.add(layers.Dropout(0.5))
network9.add(layers.Dense(512, activation='relu', activity_regularizer=keras
.regularizers.l2(0.0001)))
network9.add(layers.Dropout(0.5))
network9.add(layers.Dense(512, activation='relu', activity_regularizer=keras
.regularizers.l2(0.0001)))
network9.add(layers.Dropout(0.5))
network9.add(layers.Dense(10, activation='softmax'))

network9.summary()
```

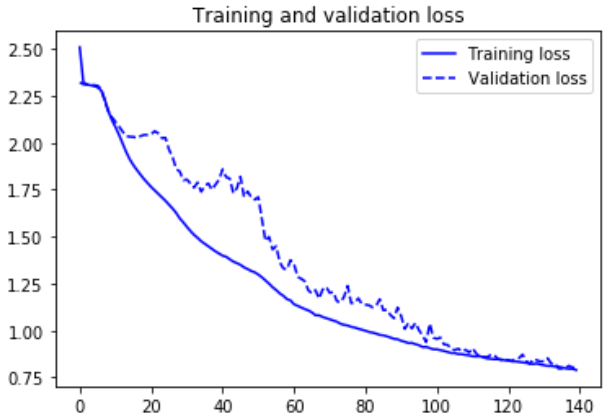
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_51 (Dense)             (None, 512)               401920    
_________________________________________________________________
dropout_24 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_52 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_25 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_53 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_26 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_54 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_27 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_55 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_28 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_56 (Dense)             (None, 512)               262656    
_________________________________________________________________
dropout_29 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_57 (Dense)             (None, 10)                5130      
=================================================================
Total params: 1,720,330
Trainable params: 1,720,330
Non-trainable params: 0
_________________________________________________________________
```

In [85]:
```
#paralelizando, compilando e treinando
network9 = keras.utils.multi_gpu_model(network9,gpus=2)
network9.compile(optimizer='sgd',
                 loss=keras.losses.categorical_crossentropy,
                 metrics=['accuracy'])
history9 = network9.fit(train_images_mlp, train_labels, epochs=140, batch_si
ze=128, validation_split=0.2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/140
48000/48000 [==============================] - 5s 101us/step - loss: 2.5081 -
acc: 0.1159 - val_loss: 2.3215 - val_acc: 0.1060
Epoch 2/140
48000/48000 [==============================] - 4s 86us/step - loss: 2.3223 -
acc: 0.1239 - val_loss: 2.3113 - val_acc: 0.1060
Epoch 3/140
48000/48000 [==============================] - 4s 87us/step - loss: 2.3102 -
acc: 0.1202 - val_loss: 2.3085 - val_acc: 0.1060
Epoch 4/140
48000/48000 [==============================] - 4s 87us/step - loss: 2.3055 -
acc: 0.1213 - val_loss: 2.3073 - val_acc: 0.1060
Epoch 5/140
48000/48000 [==============================] - 4s 86us/step - loss: 2.3018 -
acc: 0.1275 - val_loss: 2.3066 - val_acc: 0.1060
Epoch 6/140
48000/48000 [==============================] - 4s 87us/step - loss: 2.2951 -
acc: 0.1615 - val_loss: 2.3034 - val_acc: 0.1061
Epoch 7/140
48000/48000 [==============================] - 4s 86us/step - loss: 2.2788 -
acc: 0.1949 - val_loss: 2.2868 - val_acc: 0.1788
Epoch 8/140
48000/48000 [==============================] - 4s 86us/step - loss: 2.2303 -
acc: 0.2071 - val_loss: 2.2241 - val_acc: 0.1997
Epoch 9/140
48000/48000 [==============================] - 4s 86us/step - loss: 2.1678 -
acc: 0.2119 - val_loss: 2.1688 - val_acc: 0.2047
Epoch 10/140
48000/48000 [==============================] - 4s 85us/step - loss: 2.1212 -
acc: 0.2253 - val_loss: 2.1357 - val_acc: 0.2188
Epoch 11/140
48000/48000 [==============================] - 4s 85us/step - loss: 2.0839 -
acc: 0.2376 - val_loss: 2.1117 - val_acc: 0.2291
Epoch 12/140
48000/48000 [==============================] - 4s 86us/step - loss: 2.0423 -
acc: 0.2502 - val_loss: 2.0862 - val_acc: 0.2397
Epoch 13/140
48000/48000 [==============================] - 4s 86us/step - loss: 1.9969 -
acc: 0.2612 - val_loss: 2.0594 - val_acc: 0.2284
Epoch 14/140
48000/48000 [==============================] - 4s 86us/step - loss: 1.9508 -
acc: 0.2711 - val_loss: 2.0351 - val_acc: 0.2200
Epoch 15/140
48000/48000 [==============================] - 4s 86us/step - loss: 1.9120 -
acc: 0.2795 - val_loss: 2.0320 - val_acc: 0.1950
Epoch 16/140
48000/48000 [==============================] - 4s 87us/step - loss: 1.8814 -
acc: 0.2880 - val_loss: 2.0310 - val_acc: 0.1756
Epoch 17/140
48000/48000 [==============================] - 4s 87us/step - loss: 1.8548 -
acc: 0.2970 - val_loss: 2.0257 - val_acc: 0.1727
Epoch 18/140
48000/48000 [==============================] - 4s 87us/step - loss: 1.8301 -
acc: 0.3080 - val_loss: 2.0350 - val_acc: 0.1627
Epoch 19/140
48000/48000 [==============================] - 4s 88us/step - loss: 1.8060 -
acc: 0.3181 - val_loss: 2.0413 - val_acc: 0.1560
Epoch 20/140
48000/48000 [==============================] - 4s 87us/step - loss: 1.7854 -
acc: 0.3241 - val_loss: 2.0420 - val_acc: 0.1587
Epoch 21/140
48000/48000 [==============================] - 4s 87us/step - loss: 1.7644 -
acc: 0.3299 - val_loss: 2.0451 - val_acc: 0.1618
Epoch 22/140
48000/48000 [==============================] - 4s 87us/step - loss: 1.7459 -
acc: 0.3340 - val_loss: 2.0621 - val_acc: 0.1543
Epoch 23/140
```

In [86]:
```
Avalia(history9,network9,test_images_mlp,test_labels,1)
```






```
10000/10000 [==============================] - 1s 119us/step
test acc= 0.6688
training accuracy= 0.7138958333333333
validation accuracy= 0.6733333333333333
test err= 0.7766052742004395
training err= 0.7893838154474894
validation err= 0.8091879213651021
```