Universidade Federal de Santa Catarina (UFSC)
Campus Araranguá (ARA)
Disciplina: Projeto de sistemas ubíquos
Professor: Antônio Carlos Sobieranski
Aluno: Ale Chaito

Reconhecimento de Dígitos com Classificador Baseado em Rede Neural

1 Proposta

Identificar dígitos e símbolos matemáticos escritos a mão para construção de expressões regulares capazes de serem traduzidas em linguagem Latex. Para isso utilizaremos uma abordagem com redes neurais classificadoras oferecidas pela biblioteca TensorFlow e a aquisição e processamento de imagem será dada pelo OpenCV.

2 Justificativa

A identificação de dígitos e símbolos escritos a mão abre gama vasta de possibilidades para o desenvolvimento de aplicações, um exemplo é a digitalização de documentos. Nossa problemática envolve traduzir essas expressões matemáticas regulares em Latex. Sendo assim, o escritor da expressão matemática não terá que escrever novamente no computador e não precisará despender tempo para aprender sobre a biblioteca matemática do Latex

3 Conjunto de Dados

As redes neurais necessitam de uma amostragem grande de dados para uma robusta aprendizagem dos padroes provenientes das imagens. O conjunto de dados utilizado para o treino de nossa rede era composto por digitos e simbolos, após diversas pesquisas encontramos apenas 2 conjunto de dados que atingiam os nossos requisitos, o Minist e o Kaggle, porém o segundo conjunto foi descartado devido a nao convergencia da rede durante os treinamentos. Este fato levou a restringirmos a proposta inicial do trabalho apenas para o reconhecimento de dígitos.

3.1 Minist

O banco de dados MNIST de dígitos manuscritos, disponível em MINIST(2018), possui um conjunto de treinamento de 60.000 exemplos e um conjunto de teste de 10.000 exemplos. É um subconjunto de um conjunto maior disponível no NIST. Os dígitos foram normalizados por tamanho ou seja todos valores de pixels estão entre 0 e 1 isto ajuda a rede a convergir mais rápido e centralizados em uma imagem de tamanho fixo 28x28.

4 Desenvolvimento

Após a seleção dos dados optamos por utilizar a linguagem Python para atingir nosso objetivo, devido sua ampla quantidade de bibliotecas para desenvolvimento cientifico e inteligencia artificial.

Começamos por importar os pacotes necessários, utilizamos o Keras integrado ao Tensorflow para criar nossa rede e treina-lá. O Numpy foi utilizado para a estrutura de dados e o Matplotlib para gerar imagens de saída.

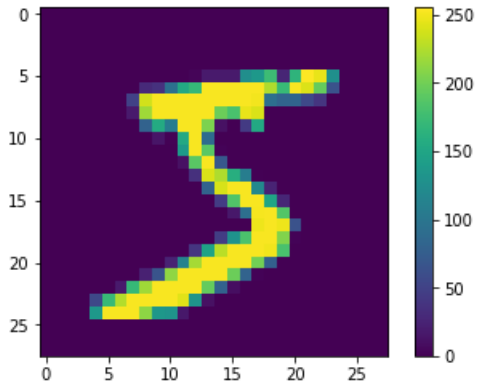
```
from tensorflow import keras
import tensorflow as tf

import numpy as np
import matplotlib.pyplot as plt
```

Conjunto de dados composto por 60.000 images de treinamento e 10.000 para teste.

```
print(train_images.shape)
print(test_images.shape)
(60000, 28, 28) # 60.000 imagens 28x28
(10000, 28, 28) # 10.000 imagens 28x28
```

Plotando a primeira imagem de treino, podemos ver que é o numero 5 ainda em 3 canais RGB.

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
```

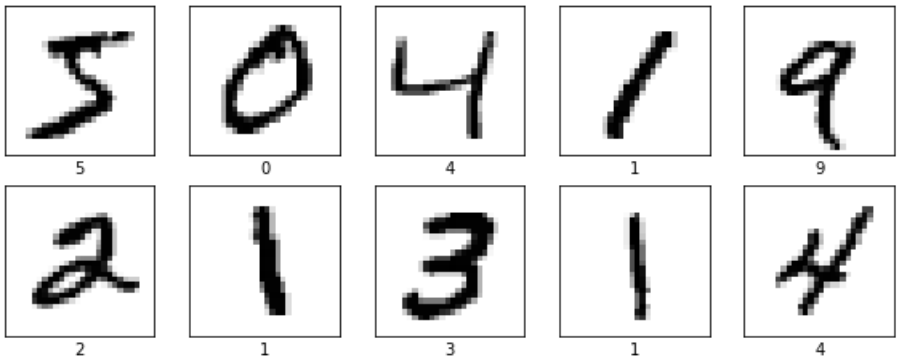


Para ajudar a rede a convergir mais rápido vamos normalizar os dados entre [0,1], dividindo pelo máximo 255.

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Vamos plotar 10 imagens com seus devidos rótulos do nosso conjunto de treino para verificar como elas ficaram após os tratamento.

```
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(train_labels[i])
```



Com os dados devidamente preparados podemos montar a estrutura da nossa rede, que será composta por uma camada de entrada com forma 28 x 28 com uma camada intermediária de 128 neurônios e 10 neurônios de saída.

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Para o modelo treinar é necessário compilar e escolher uma métrica de aprendizagem.

```
model.compile(optimizer=tf.train.AdamOptimizer(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

Agora então estamos prontos para iniciar o treinamento da rede passando como parâmetro as 60.000 imagens e seus rótulos. Podemos observar que com apenas 5 épocas obteve-se 98 porcento de acurácia.

```
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 47us/step - loss: 0.2597 - acc: 0.9272
Epoch 2/5
60000/60000 [==============================] - 3s 43us/step - loss: 0.1153 - acc: 0.9658
Epoch 3/5
60000/60000 [==============================] - 3s 44us/step - loss: 0.0782 - acc: 0.9771
Epoch 4/5
60000/60000 [==============================] - 3s 43us/step - loss: 0.0578 - acc: 0.9826
Epoch 5/5
60000/60000 [==============================] - 3s 44us/step - loss: 0.0452 - acc: 0.9862
```

Vamos testar a robustez da predicao do modelo utilizando os dados de teste.

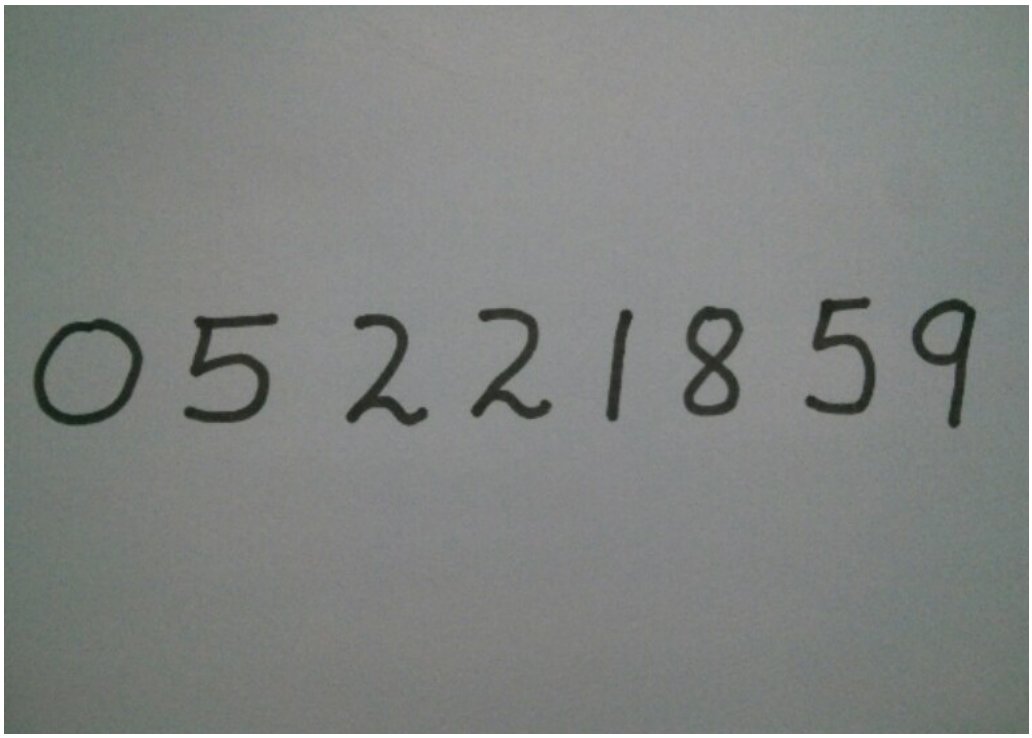
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

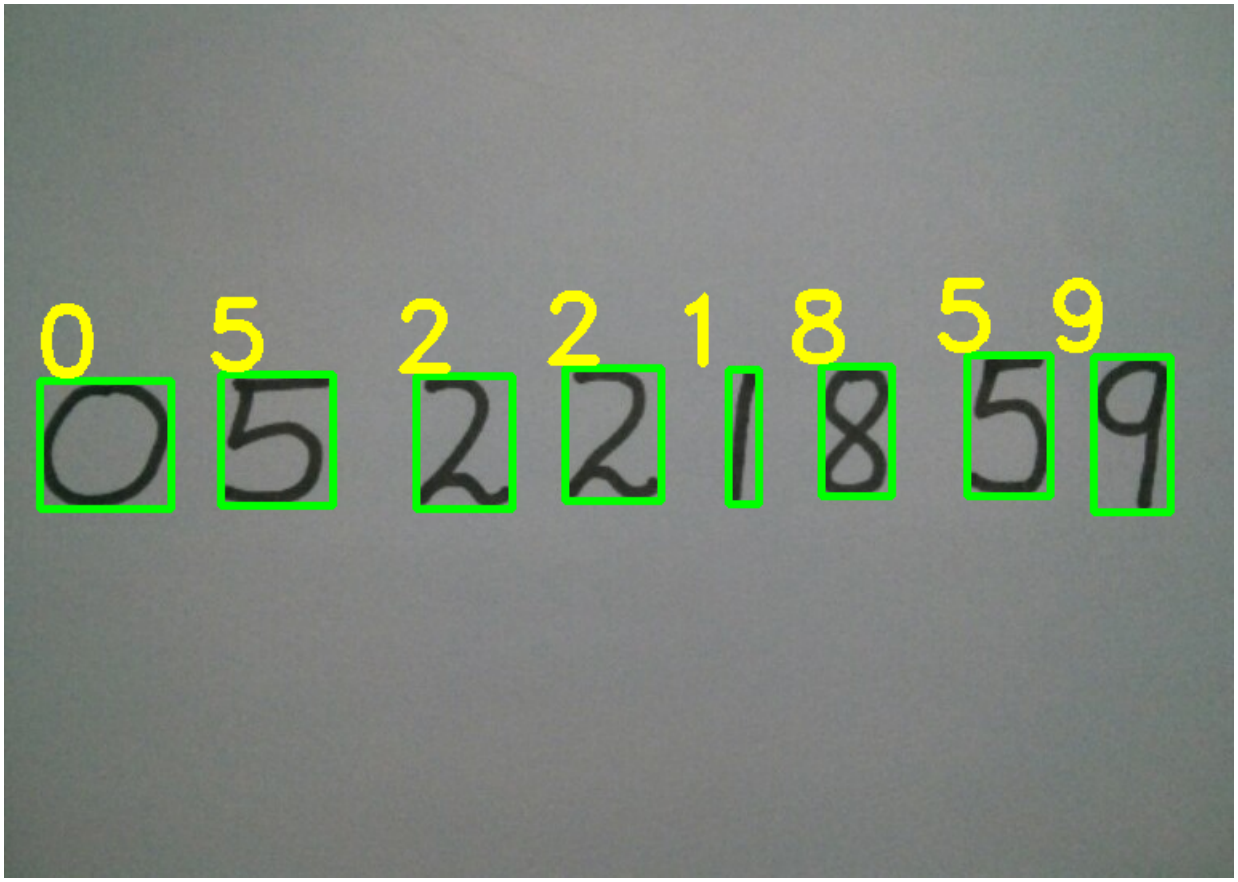
```
10000/10000 [==============================] - 0s 28us/step
Test accuracy: 0.9786
```

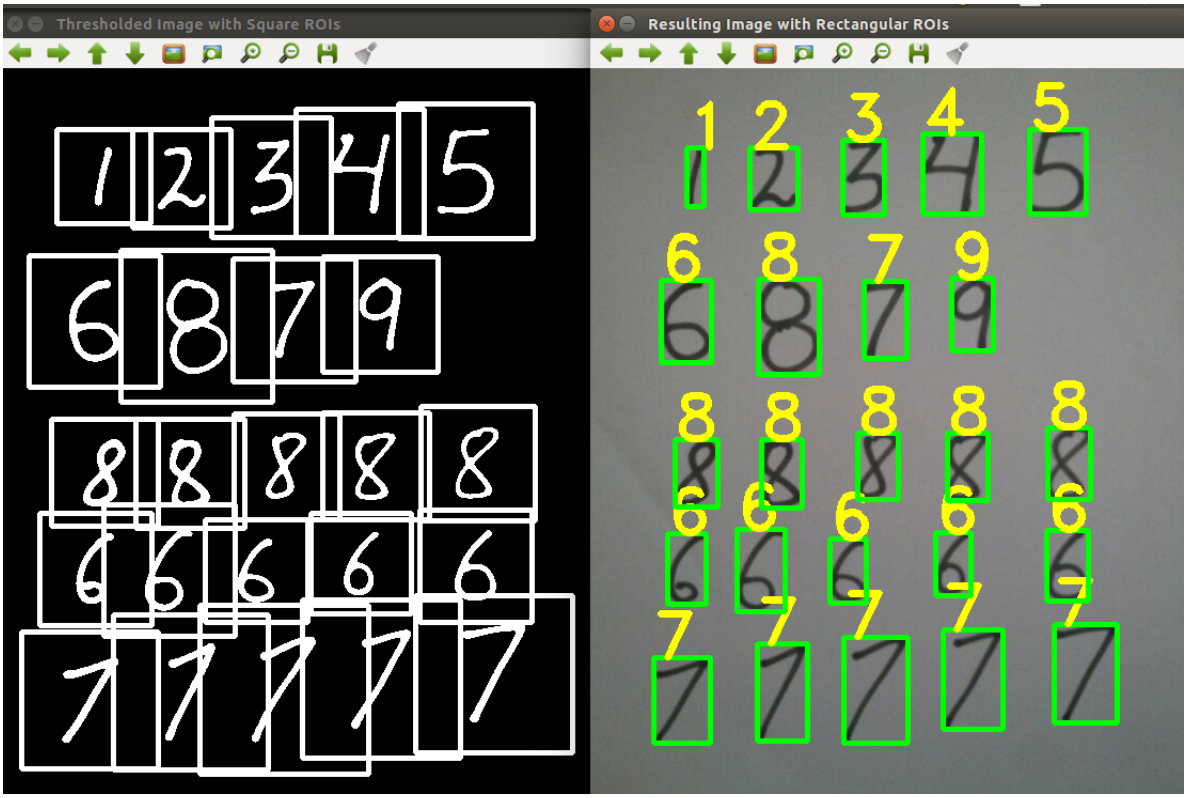
Para finalizar e utilizar os pesos da nossa rede já treinada, executaremos o comando para salvar o modelo em disco.

```
model.save("model.h5")
```

Agora utilizando o modelo salvo, vamos submeter uma imagem com dígitos escritos a mão.


05221859


0 5 2 2 1 8 5 9



5 Conclusão

Através deste trabalho podemos ver o poder das redes neurais no reconhecimento de objetos, uma rede com apenas 128 neurônios na camada intermediaria foi capaz de reconhecer 10 classes com precisão incrível, para futuros trabalhos é interessante buscar uma forma de filtrar os ruídos provenientes dos frames e adicionar símbolos matemáticos ao conjunto de dados.

6 Referencias

MNIST. Database of handwritten digits. Disponível em: <http://yann.lecun.com/exdb/mnist/> Acesso em: 26 nov. 2018.

Kaggle. Database of handwritten symbols and digits. Disponível em: <https://www.kaggle.com/xair Acesso em: 26 nov. 2018.

TensorFlow. TensorFlow is an open-source machine learning library for research and production.. Disponível em: <https://www.tensorflow.org/guide/keras> Acesso em: 26 nov. 2018.