UNIVERSIDADE FEDERAL DO ESTADO DO RIO GRANDE DO NORTE

Disciplina / Área de conhecimento:

ECT2702 - TÓPICOS AVANÇADOS EM INFORMÁTICA I - T01 (2019.2)

Character Recognition

Luís Fernando Tavares

Natal, 2019

**Sumário**

1 Introdução…………………………………………...…………...…………..............................………......03

## 2 Metodologia.……………...……………………….......................................................….……......04

## 3 Códigos……………………....................................................................................………......05

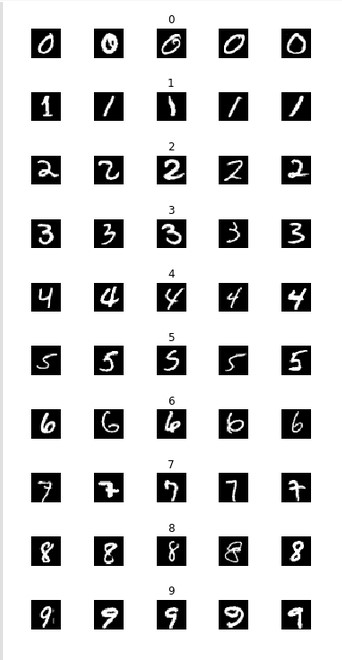
## 4 Experimentos………………..........................................................…....................………......11

02

**1 Introdução**

O problema abordado neste relatório é: uma base de dados que contém um registro de diversas imagens de números escritos a mão, indo de 0 a 9, todas escritas por pessoas diferentes e o objetivo é saber se a partir dessa base podem identificar um numero entre 0 e 9 escrito a mão de outras pessoas diferentes.

A base de dados é composta por números de 0 a 9:



03

**2 Metodologia**

O modelo de Machine Learning utilizado para resolver este problema é o Perceptron Multicamadas (PMC ou MLP — Multi Layer Perceptron), ele é uma rede neural com uma ou mais camadas ocultas com um número indeterminado de neurônios. A camada oculta possui esse nome porque não é possível prever a saída desejada nas camadas intermediárias.

Ele funciona a partir de um estimulo que é propagado da entrada para saída passando por cada camada escondida.

As etapas de treinamento são divididas em três partes:

**1º** Forward pass (Passe avançado):

Nesta etapa do treinamento do modelo, apenas passamos a entrada (atributo) para o modelo e multiplicamos com pesos e adicionamos viés em cada camada e encontramos a saída calculada do modelo.

**2º** Calculate error or loss (Calcular erro ou perda):

Quando passamos a instância de dados (ou um exemplo), obtemos alguma saída do modelo chamado saída prevista e temos o rótulo com os dados que são saída real ou saída esperada. Com base nesses dois, calculamos a perda que precisamos retropropagar (usando o algoritmo Backpropagation). Existem várias funções de perda que usamos com base em nossa produção e exigência.

**3º** Backward pass (Passe para trás):

Após calcular a perda, retropropagamos a perda e atualizamos os pesos do modelo usando gradiente. Nesta etapa, os pesos serão ajustados de acordo com o fluxo do gradiente nessa direção.]

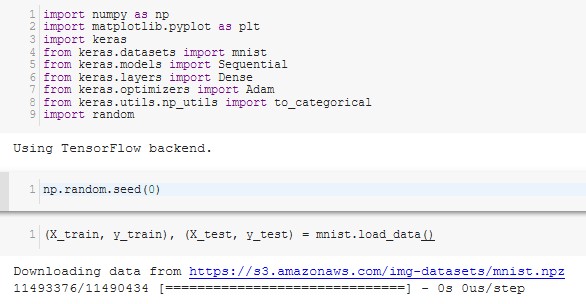
No caso é usado MLP em deep learning para os reconhecimento de números em imagens, e é usado com camadas os quais possuem divesos neuronios para o processamento de imagem.

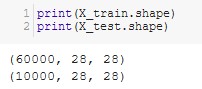
O MNIST dataset é um dos conjuntos de dados mais comuns usados ​​para classificação de imagens e acessíveis a partir de várias fontes diferentes, no caso sera utilizado os numeros de 0 a 9 para serem usados como parametro de comparação de imagens de numeros de outras pessoas escritas a mão, de 0 a 9.

04

**03 Códigos**

**1º Importação e seleção dos atributos de teste e treinamento:**

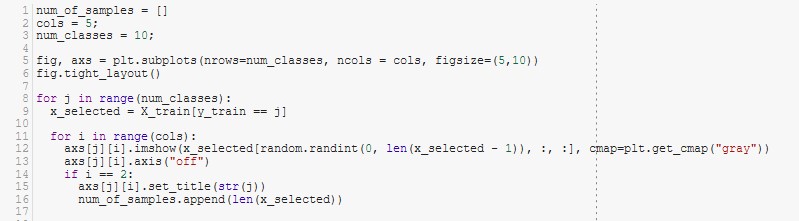




Também faz a importação de API (Application Programming Interface ou Interface de Programação de Aplicativos), que servem para ter acesso ao base de dados, no caso em .npz, denominada de mnist, a modelos para a criação de camadas, a tipos específicos de algoritmos como a Adam (algoritmo de otimização), e a seleção dos atributos (X) que são usados para comparação do parâmetro (Y), separado para o teste e o treinamento, os quais possuem respectivamente 10000 e 60000 números de imagens no conjunto de dados e possui uma matriz que representa o tamanho da imagem: 28 x 28 pixels.

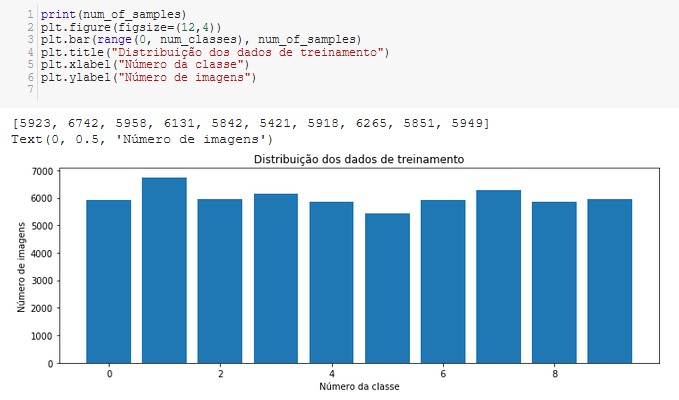
05

**2º Seleção dos atributos:**



Esse código permite a escolha dos atributos, no caso descritos como num\_classes, os quais são escolhidos os números 0 a 9, e eles são separado em 5 imagens (utilizando os atributos e metodos axs, cols, figsize, fig.tight\_layout(), etc) mostrando as imagens de números escritos a mão.

**3º** Dados de Treinamento:



Esse código mostra o gráfico da distribuição da quantidade de números de imagens que um o número da classe possui.

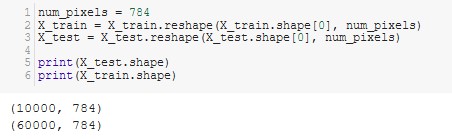
**4º Filtro (Preto e Branco)**:



Esse código normaliza os nossos dados, isso é feito dividindo os códigos RGB (X\_train e X\_test) por 255, isso muda a cor dela para preto e branco, diminuindo a complexidade do processamento da imagem.

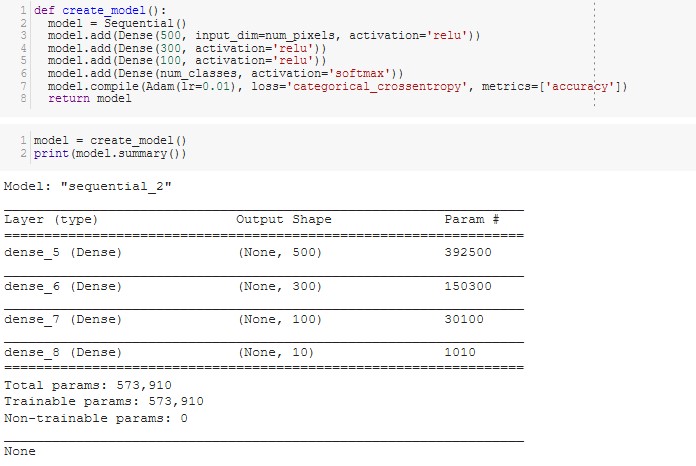
**06**

**5º Remodelagem:**

****

**Ele remodela os tamanhos das imagens para que elas possam processar uma imagem que possua uma grande quantidade de pixels, no caso 784 pixels.**

**6º Camadas:**

****

Esse código cria as camadas (Neurônios) para serem processados os atributos, inicialmente é criado um modelo sequencial e em seguida são adicionadas ao modelo as camadas do tipo “Dense”.

A camada inicial possui quantidade de neurônios presentes (500), a entrada de certo número de atributos (784 pixels) e a função de ativação da camada (Rectifier).

A segunda camada possui quantidade de neurônios presentes (300) e a função de ativação da camada (Rectifier).

A terceira camada possui quantidade de neurônios presentes (100) e a função de ativação da camada (Rectifier).

07

Última camada é adicionada os números das classes (10) e a função de ativação da camada (softmax).

Por ultimo é adicionado uma modelo de otimização (Adam) com uma taxa de aprendizado de 0.01, e uma função para calcular a perda (categorical\_crossentropy).

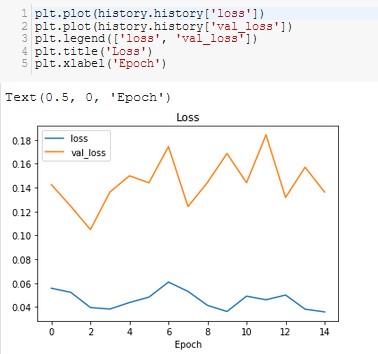
7**º Treinamento:**

****

Esse código inicia o treinamento que permite a estimativa do gradiente, o batch\_size é um hiperparâmetro de descida de gradiente que controla o número de amostras de treinamento a serem trabalhadas antes que os parâmetros internos do modelo sejam atualizados, quanto menor o batch\_size, menor é acurácia do gradiente, porem utilizara menos memória e será mais rápido, e vai ditar quantas interações levara para completar um epoch é um hiperparâmetro de descida de gradiente que controla o número de passagens completas pelo conjunto de dados de treinamento*.*

O validation\_split são dados nos quais avalia a perda e quaisquer métricas de modelo no final de cada época, o verbose simboliza como voce quer ver o processo de treinamento de cada epoch, 0 = silencioso, 1 = barra de progresso, 2 = uma linha por época e o shuffle é uma variavel Booleana (embaralha os dados de treinamento antes de cada época) ou str (para 'lote').

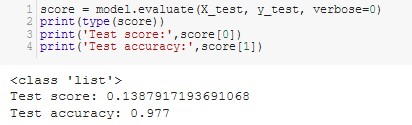
8**º Gráfico de Perdas:**



Apresenta o grafico que compara a perda do treinamento com a perda do modelo de otimização.

08

9**º Taxa de** Acurácia**:**

****

Calcula e mostra a taxa de acurácia do treinamento para reconhecer um numero de 0 a 9 escrito a mão.

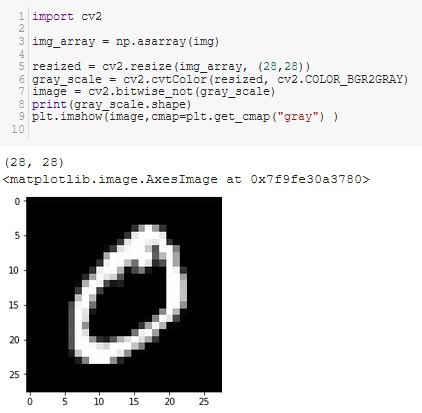
10**º Importação da Imagem:**

****

Faz a importação da imagem contida no url e a imprime na tela.

09

11**º Remodelagem da Imagem:**

****

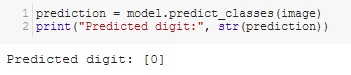
**A imagem é colocada em um array e remodelada para que fique de um tamanho de 28X28 pixels e em seguida passa por um filtro de grayscale, ou seja cada pixels é uma única amostra de um espaço de cores.**

12**º Filtro (Preto e Branco):**

****

**A imagem é** normaliza, diminuindo a complexidade do processamento da imagem.

13**º Resultado:**



Demonstra por fim a identificação da imagem selecionada para passar pelo código.

10

**4 Experimentos**

Os tipos de testes executados foram para, de acordo com os dados fornecidos, ter um gral de certeza de é possível identificar um número feito à mão, entre 0 e 9, utilizando uma base desses números escritos de formas diferentes, por pessoas diferentes, utilizando quatro camadas escondidas e uma de saída, as quatro primeiras com a função de ativação chamada Rectifier e a última com a softmax e uma taxa de aprendizado de 0.01, e para descobrir a estimativa do gradiente, o batch\_size = 200, os epochs = 15, validation\_split = 0.1, shuffle =1 e verbose=1, e em seguida foi inserido uma imagem de um numero e foi remodelada, para uma matriz 28X28 e passada por um “filtro” para deixar preto e branco depois remodelada para 784 pixels e compara da com a base de dados.

Parâmetro avaliado foi a imagem de um numero escrito a mão, no caso 0, com a base de dados (mnist) de numeros escritos a mão (0 a 9) de diversas pessoas diferentes.

Os resultados obtidos foram, de acordo com os testes e treinamento, uma taxa de 97,7% de acurácia na identificação do numeros escrito a mão no intervalo de 0 a 9, o que foi confirmado ser veridico quando a imagem do numero 0 escrito a mão foi passado para ser comparado com o treinamento, foi identificado com exito, assim este programa pode ser usado para identificar qual é o número, entre 0 a 9, escrito a mão foi inserido no programa.

11