



UNIVERSIDADE PAULISTA CAMPINAS
CIÊNCIA DA COMPUTAÇÃO

**ATIVIDADES PRÁTICAS SUPERVISIONADAS
PROCESSAMENTO DE IMAGEM**

PROJETO DE RECONHECIMENTO DE LETRAS DO ALFABETO

**CC5P12
CC6P12
CC5Q12**

**Leandro Do Nascimento Da Silva - Ra: N258667
Leandro Fioravante De Oliveira - Ra :F1928A0
Maria Eduarda Liomerio Goncalves - Ra: F2473H-0
Vinicius Valente Ramirez - Ra: N4898J-5**

Waldevan De Carvalho Ferreira - Ra: N628642

2022

campinas

UNIVERSIDADE PAULISTA CAMPINAS
CIÊNCIA DA COMPUTAÇÃO



PROJETO DE RECONHECIMENTO DE LETRAS DO ALFABETO

Esse relatório foi feito sob a orientação de Vinicius pereira como parte do projeto na unidade curricular de Atividades Práticas Supervisionadas (APS) contendo estudo e aplicação de programação estruturada no tema reciclagem.

2022

campinas

SUMÁRIO

1. RESUMO	
2. INTRODUÇÃO.....	
3. REFERÊNCIA TEÓRICA.....	
4. MÉTODOS...	
5. RESULTADOS.....	
6. CONCLUSÃO.....	
7. TRABALHOS FUTUROS.....	
8. REFERÊNCIAS BIBLIOGRÁFICAS.....	
9. FICHAS	

1. RESUMO

O desenvolvimento alfabético só é possível através do processamento de imagem, que é realizado por nosso cérebro, onde ensinamos por meio de símbolos o que cada letra significa e assim tornando possível a leitura e escrita. Através do processamento de imagem podemos ensinar o computador a reconhecer símbolos, que podem ser ou não letras, e realizar uma tarefa equivalente a cada símbolo identificado. Algumas letras do alfabeto são reconhecidas entre 2 a 3 anos de idade já aos 5 anos de idade já pode se dizer que podem ser reconhecidas completamente. Algumas décadas atrás o processamento de imagem era feito majoritariamente de forma analógica, através de dispositivos ópticos. Apesar disso, devido ao grande aumento de velocidades dos computadores, tais técnicas foram gradualmente substituídas por métodos digitais.

2. INTRODUÇÃO

Inicialmente é válido ressaltar que o processamento de imagem é diferente do processamento de dados, onde o processamento de imagem a entrada e saída são imagens tais como fotografias ou quadros de vídeo. Ao contrário do tratamento de imagens, que se preocupa somente na manipulação de figuras para sua representação final, o processamento de imagens é um estágio para novos processamentos de dados tais como aprendizagem de máquina ou reconhecimento de padrões. A maioria das técnicas envolve o tratamento da imagem como um sinal bidimensional, no qual são aplicados padrões de processamento de sinal.

Além de imagens bidimensionais estáticas, o campo também abrange o processamento de sinais variados pelo tempo, tais como vídeos ou a saída de um equipamento de tomografia. Tais técnicas são especificadas somente para imagens binárias ou em escala de cinza. Há muitas aplicações para os tópicos estudados nessa vertente, como por exemplo fotografia e impressão, imagens de satélite, processamento de imagens médicas, detecção de face ou de objetos, biometria entre outras coisas.

Algumas décadas atrás o processamento de imagem era feito majoritariamente de forma analógica, através de dispositivos ópticos. Apesar disso, devido ao grande aumento de velocidades dos computadores, tais técnicas foram gradualmente substituídas por métodos digitais.

O processamento digital de imagem é geralmente mais versátil, confiável e preciso, além de ser mais fácil de implementar que seus dois analógicos. Hardware especializado ainda é usado para o processamento digital de imagem, contando com arquiteturas de computador paralelas para tal, em sua maioria no processamento de vídeos. O processamento de imagens é, em sua maioria, feito por computadores pessoais.

O processamento de imagem é uma área que vem tendo crescimento em várias áreas científicas, que estabelece relações entre duas disciplinas entre elas que podemos citar são compreensão de imagens, a análise em multiresolução e multi-frequência, a análise estatística, a codificação e a transmissão de imagens e etc.

Os espaços do alfabeto tem como a principal prioridade proporcionar e estabelecer uma estimulação ao desenvolvimento dos códigos, tendo a importância de mostrar as imagens com um trabalho lúdico e participativo nas linguagens de gêneros textuais e na comunicação entre os seres humanos.

O desenvolvimento alfabético normalmente é iniciado na infância, com o aprendizado de identificar, ler e escrever as letras. Este aprendizado pode ser facilitado com brincadeiras, jogos, caça palavras e músicas, que estimulam as crianças a criarem hábitos de leitura e a reconhecer os sons (fonemas) das sílabas, conforme o avanço desse aprendizado é inserindo na aprendizagem os livros e jornais para o desenvolvimento do vocabulário e das figuras de linguagem.

O desenvolvimento alfabético só é possível através do processamento de imagem, que é realizado por nosso cérebro, onde ensinamos por meio de símbolos o que cada letra significa e assim tornando possível a leitura e escrita. Através do processamento de imagem podemos ensinar o computador a reconhecer símbolos, que podem ser ou não letras, e realizar uma tarefa equivalente a cada símbolo identificado. Algumas letras do alfabeto são reconhecidas entre 2 a 3 anos de idade já aos 5 anos de idade já pode se dizer que podem ser reconhecidas completamente.

Estimular atividades que ajudem ao desenvolvimento como leitura, massinhas assistir desenho que estimule o desenvolvimento de aprendizagem e a estimulação dentro da sala de aula dando exercícios interativos para o desenvolvimento da criança muitos locais de ensino fundamental estão tendo um planejamento de ensino melhor do que antigamente, tendo uma intenção de trabalhar muito mais no desenvolvimento das crianças.

palavras Chaves : (imagem,processos,desnvolvimento)

3. REFERÊNCIA TEÓRICA

Processamento de Imagem Uma imagem pode ser definida como uma função bidimensional, $f(x, y)$, onde x e y são coordenadas no espaço, e a amplitude de f em qualquer par de coordenadas (x, y) é denominado intensidade ou “nível de cinza” naquele ponto. Quando os valores das coordenadas e da intensidade são valores finitos e discretos pode haver o processamento digital da imagem

Aprendizado de Máquina Aprendizado de Máquina é o ramo da Ciência da Computação que utiliza experiências passadas para aprender e usa o conhecimento adquirido para tomar decisões. Aprendizado de Máquina é a interseção da Ciência da Computação, Engenharia e Estatística. A finalidade do Aprendizado de Máquina é criar um modelo genérico que detecta padrões e regras não conhecidas a partir de exemplos conhecidos.

Aprendizado de Máquina Supervisionado No Aprendizado de Máquina Supervisionado, dentro do conjunto de dados, tem-se os dados e as classes de cada item. E posteriormente (depois do treinamento), dado um item, o modelo será capaz de colocá-lo em alguma das classes disponíveis. O Aprendizado de Máquina Supervisionado é separado em classificação e regressão. Classificação: Quando o problema analisado tem a resposta em classes bem definidas e discretas. Exemplo: dado o tamanho e idade do tumor, classificar em câncer ou não câncer. Regressão: O problema submetido não tem classes bem definidas e a resposta é sempre um número real que pode estar em um intervalo muito grande, ou seja, problema contínuo. Exemplo: prever o preço de uma casa, dadas as características (bairro, tamanho, etc).

Aprendizado de Máquina Não Supervisionado Com o Aprendizado de Máquina Não Supervisionado, a máquina é treinada com dados sem informações de saída, e o objetivo é agrupar os elementos baseados em características similares ou em características que os tornam únicos. Esses grupos são chamados de clusters. Com o Aprendizado Não Supervisionado, o objetivo não é procurar por uma resposta única, aproximada, específica ou certa. Ao invés disso, é relacionar cada elemento do grupo pela similaridade das características ou do comportamento entre os membros, e também pelas diferenças com os outros grupos.

4. MÉTODOS

Todo o estudo, desenvolvimento, modelagem contidos neste projeto são semelhantes do que seria um estudo de aprendizado profundo para aplicações práticas reais, além de que, existem muitos conceitos matemáticos e estatísticos estão por trás dos métodos utilizados, e o trabalho contribuiu para a formação do aluno e para a aplicação de conhecimentos obtidos ao longo da graduação.

6. CONCLUSÃO

Este projeto teve como objetivo desenvolver e modelar uma rede neural convolucional para a classificação de caracteres manuscritos. Para o aprendizado, desenvolvimento e modelamento, foi necessário um estudo árduo sobre a matemática, álgebra linear e estatística envolvidas no aprendizado profundo, mais especificamente em redes neurais convolucionais. Para isso, foram usados, livros, artigos, frameworks e bibliotecas mais atuais para o desenvolvimento, como: Keras, QT, Tensor Flow, flask, jupyter notebook, linguagem Python, etc.

Trabalhar com um conjunto de dados complexo foi desafiante, normalmente quando se vai desenvolver para este tipo de problema, o conjunto de dados usados é o MNIST.

As redes neurais convolucionais se mostraram um método eficiente para a classificação de caracteres manuscritos. A melhor acurácia obtida no artigo original do EMNIST, enquanto a implementação com redes neurais convolucionais contidas neste trabalho, teve como acurácia 76,67%.

Comparação entre a acurácia do artigo do EMNIST e a deste trabalho. Método Fonte Acurácia Classificador Linear Artigo EMNIST (COHEN et al., 2017) 69,71%

Redes Neurais Convolucionais Este trabalho 76,67% Fonte: Elaborada pelo autor. Todo o estudo, desenvolvimento, modelagem contidos neste projeto são semelhantes do que seria um estudo de aprendizado profundo para aplicações práticas reais, além de que, existem muitos conceitos matemáticos e estatísticos estão por trás dos métodos utilizados, e o trabalho contribuiu para a formação do aluno e para a aplicação de conhecimentos obtidos ao longo da graduação.

7. TRABALHOS FUTUROS

Como sugestão de trabalhos futuros, pode-se considerar a melhoria dos resultados alcançados por este trabalho. Para isso, é possível estudar o ajuste dos hiperparâmetros e a tentativa de modelar a arquitetura com modelos mais complexos, observando o comportamento da rede. Além da tentativa de combinar este método com outros métodos, aplicar transformações no conjunto de dados. Também, considerar treinar a rede em computadores mais robustos, explorando o paralelismo obtido por meio das GPU's, para otimizar e não ter como problema o tempo de processamento.

8. REFERÊNCIAS BIBLIOGRÁFICAS

Aqui se encontram todas as pesquisas, fontes bibliográficas e todos os sites usados para a realização desse projeto:

PROJETO: CONHECENDO AS LETRAS DO ALFABETO

<<http://ceiivetespeziaschmitt.blogspot.com/2016/09/projeto-conhecendo-as-letas-do-alfabeto.html>>

LER TEXTOS DENTRO DE IMAGENS COM PYTHON

<<https://www.hashtagtreinamentos.com/ler-textos-dentro-de-imagens-com-python>>

COMO EXTRAIR TEXTO DE IMAGENS COM PYTHON?

<<https://acervolima.com/como-extrair-texto-de-imagens-com-python/>>

Reconhecimento de caracteres em imagens com Tesseract-OCR e Python

<<https://medium.com/@genilsonmedeiros/reconhecimento-de-caracteres-em-imagens-com-tesseract-ocr-e-python-parte-2-7da512049987>>

5 Bibliotecas open source para reconhecimentos de objetos e OCR

<<https://micreiros.com/5-bibliotecas-open-source-para-reconhecimentos-de-objetos-e-ocr/>>

<<https://educacaopublica.cecierj.edu.br/artigos/9/36/brincando-com-as-letas-e-as-palavras-projeto-interdisciplinar-entre-educaccedilatildeo-fiacutesica-e-pedagogia>>

<<http://ceiivetespeziaschmitt.blogspot.com/2016/09/projeto-conhecendo-as-letas-do-alfabeto.html>>

<<https://www.institutoclaro.org.br/educacao/para-ensinar/planos-de-aula/as-letas-do-alfabeto-vogais-e-consoantes/>>

<https://diversa.org.br/artigos/tea-educacao-infantil/?gclid=Cj0KCQiAsdKbBhDHARIsANJ6-jdF-Uk2WKoWCjqCVSW3EVn8i2EZDWglY4e-HqFX_QLnO6nJBEXF5SMaAv18EALw_wcB>

<<https://escolakids.uol.com.br/portugues/alfabeto-ensino-fundamental-i.htm>>

DESENVOLVIMENTO DO PROJETO

Este trabalho tem como objetivo colocar em prática as aulas de processamento de imagens, desenvolvendo um projeto em Python para identificar letras do alfabeto por meio de uma imagem processada e exibindo o resultado de qual letra a imagem é correspondente.

Resultado do Projeto

```
In [1]: ## Instalando dependências e configurando

In [2]: import tensorflow as tf ## pip install tensorflow

In [3]: import numpy as np ## pip install numpy

In [4]: import os

In [5]: # Evitando erros de Out of Memory (OOM) setando aumento de memoria de GPU

        gpus = tf.config.experimental.list_physical_devices('GPU')
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)

In [6]: # Limitando a quantidade de extensões

In [7]: import cv2

In [8]: import imghdr

In [9]: data_dir = 'data'

n [10]: image_exts = ['png', 'jpg', 'bmp', 'jpeg']

n [11]: for image_class in os.listdir(data_dir):
        for image in os.listdir(os.path.join(data_dir, image_class)):
```

```
In [11]: for image_class in os.listdir(data_dir):
        for image in os.listdir(os.path.join(data_dir, image_class)):
            image_path = os.path.join(data_dir, image_class, image)
            try:
                img = cv2.imread(image_path)
                tip = imghdr.what(image_path)
                if tip not in image_exts:
                    print('Image not in ext list {}'.format(image_path))
                    os.remove(image_path)
            except Exception as e:
                print('Issue with image {}'.format(image_path))
```

```
Issue with image data\0_u\train_4f
Issue with image data\u_l\train_75
Issue with image data\U_u\train_55
```

```
In [12]: # Carregando dados utilizando Keras

In [13]: from matplotlib import pyplot as plt

In [14]: data = tf.keras.utils.image_dataset_from_directory('data')

        Found 116316 files belonging to 10 classes.

In [15]: data_iterator = data.as_numpy_iterator()

In [16]: # Gerar outro lote do iterador

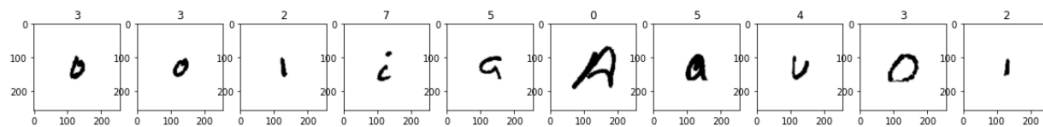
In [17]: batch = data_iterator.next()
```

```
Out[38]: (array([[[[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157]  
...],  
[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157]],  
  
[[[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157],  
...],  
[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157],  
[0.00392157, 0.00392157, 0.00392157]]])
```

```
In [41]: batch[0].max()
```

```
Out[41]: 255.0
```

```
In [42]: fig, ax = plt.subplots(ncols=10, figsize=(20,20))
for idx, img in enumerate(batch[0][:10]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```



```
In [43]: # Repartindo dados
```

```
In [31]: len(data)
```

```
Out[31]: 3635
```

```
In [46]: train_size = int(len(data)*.7)
val_size = int(len(data)*.2)
test_size = int(len(data)*.1)+1
```

```
In [49]: # train_size e val_size utilizado para treinamento da maquina
# test_size utilizado para validação dos dados
```

```
Out[49]: 364
```

```
In [33]: train_size+val_size+test_size
```

```
Out[33]: 3635
```

```
In [33]: train_size+val_size+test_size
```

```
Out[33]: 3635
```

```
In [ ]: # Função "take" define a quantidade de dados que vamos levar nessa partição em particular
# Função "skip" pula os lotes que já foram alocados na partição de treinamento
```

```
In [44]: train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

```
In [51]: len(train)
```

```
Out[51]: 2544
```

```
In [54]: from tensorflow.keras.models import Sequential
```

```
In [55]: # Essa camada cria um núcleo de convolução que se convolve com a camada de entrada para produzir um "tensor" de saídas
```

```
In [56]: from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

```
In [58]: model = Sequential()
```

```
In [59]: model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
model.add(MaxPooling2D())
```

```
model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
```

```
model.add(Conv2D(16, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
```

```
model.add(Flatten())
```

```
model.add(Dense(100, activation='relu'))
```

```
Total params: 3,696,625
Trainable params: 3,696,625
Non-trainable params: 0
```

```
In [63]: # Treinando a maquina
```

```
In [64]: logdir = 'logs'
```

```
In [65]: tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

```
In [*]: hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])
```

```
Epoch 1/20
2544/2544 [=====] - 1608s 632ms/step - loss: -21866269900800.0000 - accuracy: 0.0470 - val_loss: -1181
98129131520.0000 - val_accuracy: 0.0463
Epoch 2/20
2544/2544 [=====] - 1657s 651ms/step - loss: -743588627480576.0000 - accuracy: 0.0471 - val_loss: -190
8522270851072.0000 - val_accuracy: 0.0460
Epoch 3/20
2544/2544 [=====] - 1611s 633ms/step - loss: -4771952405250048.0000 - accuracy: 0.0470 - val_loss: -89
91719655735296.0000 - val_accuracy: 0.0462
Epoch 4/20
552/2544 [=====>.....] - ETA: 19:14 - loss: -10354601362456576.0000 - accuracy: 0.0444
```

```
In [*]: # Gráfico de performance
```

```
In [*]: fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

```
model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
In [61]: model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

```
In [62]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 125, 125, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_2 (Conv2D)	(None, 60, 60, 16)	4624
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten (Flatten)	(None, 14400)	0
dense (Dense)	(None, 256)	3686656
dense_1 (Dense)	(None, 1)	257

```
=====
Total params: 3,696,625
Trainable params: 3,696,625
```



```
In [*]: # Gráfico de performance
```

```
In [*]: fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

```
In [*]: # Avaliar desempenho
```

```
In [*]: from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

```
In [*]: pre = precision()
re = recall()
acc = BinaryAccuracy()
```

```
In [*]: for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X) #Efetuando previsões
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)
```

```
In [*]: print(f'Precision:{pre.result().numpy()}, Recall:{re.result().numpy()}, Accuracy:{acc.result().numpy()})
```

9. Funcionamento de projeto

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
## Instalando dependências e configurando
```

```
# In[2]:
```

```
import tensorflow as tf ## pip install tensorflow
```

```
# In[3]:
```

```
import numpy as np ## pip install numpy
```

```
# In[4]:
```

```
import os
```

```
# In[5]:
```

```
# Evitando erros de Out of Memory (OOM) setando aumento de memoria de GPU
```

```
gpus = tf.config.experimental.list_physical_devices('GPU')
```

```
for gpu in gpus:
```

```
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
# In[6]:
```

```
# Limitando a quantidade de extensões
```

```
# In[7]:
```

```
import cv2
```

```
# In[8]:
```

```
import imghdr
```

```
# In[9]:
```

```
data_dir = 'data'
```

```
# In[10]:
```

```
image_exts = ['png', 'jpg', 'bmp', 'jpeg']
```

```
# In[11]:
```

```
for image_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir, image_class)):
        image_path = os.path.join(data_dir, image_class, image)
        try:
            img = cv2.imread(image_path)
            tip = imghdr.what(image_path)
            if tip not in image_exts:
                print('Image not in ext list {}'.format(image_path))
                os.remove(image_path)
        except Exception as e:
            print('Issue with image {}'.format(image_path))
```

```
# In[12]:
```

```
# Carregando dados utilizando Keras
```

```
# In[13]:
```

```
from matplotlib import pyplot as plt
```

```
# In[14]:
```

```
data = tf.keras.utils.image_dataset_from_directory('data')
```

```
# In[15]:
```

```
data_iterator = data.as_numpy_iterator()
```

```
# In[16]:
```

```
# Gerar outro lote do iterador
```

```
# In[17]:
```

```
batch = data_iterator.next()
```

```
# In[18]:
```

```
# Imagens representadas como arrays do numpy
```

```
# In[19]:
```

```
batch[0].shape
```

```
# In[21]:
```

```
batch[1]
```

```
# In[22]:
```

```
# Utilizando matplotlib e a subfunção subplots para traçar 10 imagens de cada vez
```

```
# In[23]:
```

```
fig, ax = plt.subplots(ncols=10, figsize=(20,20))
for idx, img in enumerate(batch[0][:10]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```

```
# In[24]:
```

```
# Classe 0 = A maiúsculo
# Classe 1 = E maiúsculo
# Classe 2 = I maiúsculo
# Classe 3 = O maiúsculo
# Classe 4 = U maiúsculo
#####
# Classe 5 = a minúsculo
# Classe 6 = e minúsculo
# Classe 7 = i minúsculo
# Classe 8 = o minúsculo
# Classe 9 = u minúsculo
```

```
# In[25]:
```

```
# Escalando datos
```

```
# In[37]:
```

```
data = data.map(lambda x,y: (x/255, y))
```

```
# In[38]:
```

```
data.as_numpy_iterator().next()
```

```
# In[41]:
```

```
batch[0].max()
```

```
# In[42]:
```

```
fig, ax = plt.subplots(ncols=10, figsize=(20,20))  
for idx, img in enumerate(batch[0][:10]):  
    ax[idx].imshow(img.astype(int))  
    ax[idx].title.set_text(batch[1][idx])
```

```
# In[43]:
```

```
# Repartiendo datos
```

```
# In[31]:
```

```
len(data)
```

```
# In[46]:
```

```
train_size = int(len(data)*.7)
val_size = int(len(data)*.2)
test_size = int(len(data)*.1)+1
```

```
# In[49]:
```

```
# train_size e val_size utilizado para treinamento da máquina
# test_size utilizado para validação dos dados
```

```
# In[33]:
```

```
train_size+val_size+test_size
```

```
# In[ ]:
```

```
# Função "take" define a quantidade de dados que vamos levar nessa partição em particular
# Função "skip" pula os lotes que já foram alocado na partição de treinamento
```

```
# In[44]:
```

```
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

```
# In[51]:
```

```
len(train)
```

```
# In[54]:
```

```
from tensorflow.keras.models import Sequential
```

```
# In[55]:
```

```
# Essa camada cria um núcleo de convolução que se envolve com a camada de entrada  
para produzir um "tensor" de saídas
```

```
# In[56]:
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

```
# In[58]:
```

```
model = Sequential()
```

```
# In[59]:
```

```
model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))  
model.add(MaxPooling2D())
```

```
model.add(Conv2D(32, (3,3), 1, activation='relu'))  
model.add(MaxPooling2D())
```

```
model.add(Conv2D(16, (3,3), 1, activation='relu'))  
model.add(MaxPooling2D())
```

```
model.add(Flatten())
```

```
model.add(Dense(256, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

```
# In[61]:
```



```
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

```
# In[62]:
```

```
model.summary()
```

```
# In[63]:
```

```
# Treinando a máquina
```

```
# In[64]:
```

```
logdir = 'logs'
```

```
# In[65]:
```

```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

```
# In[ ]:
```

```
hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])
```

```
# In[ ]:
```

```
# Gráfico de performance
```

```
# In[ ]:
```

```
fig = plt.figure()  
plt.plot(hist.history['loss'], color='teal', label='loss')
```

```
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

```
# In[ ]:
```

```
# Avaliar desempenho
```

```
# In[ ]:
```

```
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

```
# In[ ]:
```

```
pre = precision()
re = recall()
acc = BinaryAccuracy()
```

```
# In[ ]:
```

```
for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X) #Efetuando previsões
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)
```

```
# In[ ]:
```

```
print(f'Precision:{pre.result().numpy()}, Recall:{re.result().numpy()},
Accuracy:{acc.result().numpy()})
```

```
# In[ ]:
```

```
# Teste
```


```
# In[ ]:
```

```
img = cv2.imread('train_69_00020.png')  
plt.imshow(img)  
plt.show()
```

```
# In[ ]:
```

FICHA

[illegible]



UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Waldemar de Carvalho Ferreira TURMA: CCSP12 RA: 11628642

CURSO: Administração de Empresas CAMPUS: Guaratinguetá SEMESTRE: 05 TURNO: Noite

CÓDIGO DA ATIVIDADE: _____ SEMESTRE: _____ ANO GRADE: _____

DATA	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
01/out	Discussão para divisão de funções	4 horas	Waldemar de Carvalho		
03/out	DESCRIÇÃO DA ATIVIDADE	9 horas	Waldemar de Carvalho		
05/out	DESCRIÇÃO DA ATIVIDADE	9 horas	Waldemar de Carvalho		
06/out	Leitura/ vídeos para desenvolvimento	15 horas	Waldemar de Carvalho		
09/out	Leitura/ vídeos para desenvolvimento	15 horas	Waldemar de Carvalho		
07/out	Montagem de rascunho para desenvolvimento	8 horas	Waldemar de Carvalho		
12/out	Montagem de rascunho para desenvolvimento	8 horas	Waldemar de Carvalho		
13/out	Alimentação das partes de desenvolvimento	10 horas	Waldemar de Carvalho		
24/out	Alimentação das partes de desenvolvimento	10 horas	Waldemar de Carvalho		
13/out	Desenvolvimento e teste	15 horas	Waldemar de Carvalho		
24/out	Desenvolvimento e teste	15 horas	Waldemar de Carvalho		
28/out	Conclusões finais	10 horas	Waldemar de Carvalho		
29/out	Conclusões finais	10 horas	Waldemar de Carvalho		
04/nov	Montagem do Trabalho	4 horas	Waldemar de Carvalho		
05/nov	Montagem do Trabalho	4 horas	Waldemar de Carvalho		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso

TOTAL DE HORAS ATRIBUÍDAS: 75 Horas

AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: 1/1

NOME: Marcia Eduarda Lameria Gomes TURMA: CC6P12 RA: F2423H-0
CURSO: Curso de Computação CAMPUS: SLUP SEMESTRE: 6 TURNO: noite
CÓDIGO DA ATIVIDADE: _____ SEMESTRE: _____ ANO GRADE: _____

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
01/10/2022	discussão para divisão de funções	4 horas	Marcia E. L. Gomes		
03/10/2022	DESCRIÇÃO DA ATIVIDADE	9 horas	Marcia E. L. Gomes		
06/10/2022	DESCRIÇÃO DA ATIVIDADE	9 horas	Marcia E. L. Gomes		
06/10/2022	leitura/ vídeos para desenvolvimento	15 horas	Marcia E. L. Gomes		
09/10/2022	leitura/ vídeos para desenvolvimento	15 horas	Marcia E. L. Gomes		
07/10/2022	montagem de resumo para desenvolvimento	8 horas	Marcia E. L. Gomes		
12/10/2022	montagem de resumo para desenvolvimento	8 horas	Marcia E. L. Gomes		
23/10/2022	alinhamento das partes de desenvolvimento	20 horas	Marcia E. L. Gomes		
24/10/2022	alinhamento das partes de desenvolvimento	10 horas	Marcia E. L. Gomes		
13/10/2022	Desenvolvimento e teste	15 horas	Marcia E. L. Gomes		
24/10/2022	Desenvolvimento e teste	15 horas	Marcia E. L. Gomes		
28/10/2022	conclusões finais	10 horas	Marcia E. L. Gomes		
29/10/2022	conclusões finais	10 horas	Marcia E. L. Gomes		
04/11/2022	montagem do Trabalho	4 horas	Marcia E. L. Gomes		
05/11/2022	montagem do Trabalho	4 horas	Marcia E. L. Gomes		

[1] Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75

AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO