

# Introdução a *Convolutional Neural Networks* com *Keras* API

LUCAS BEZERRA MAIA

Orientador: Geraldo Braz Junior



## SUMÁRIO

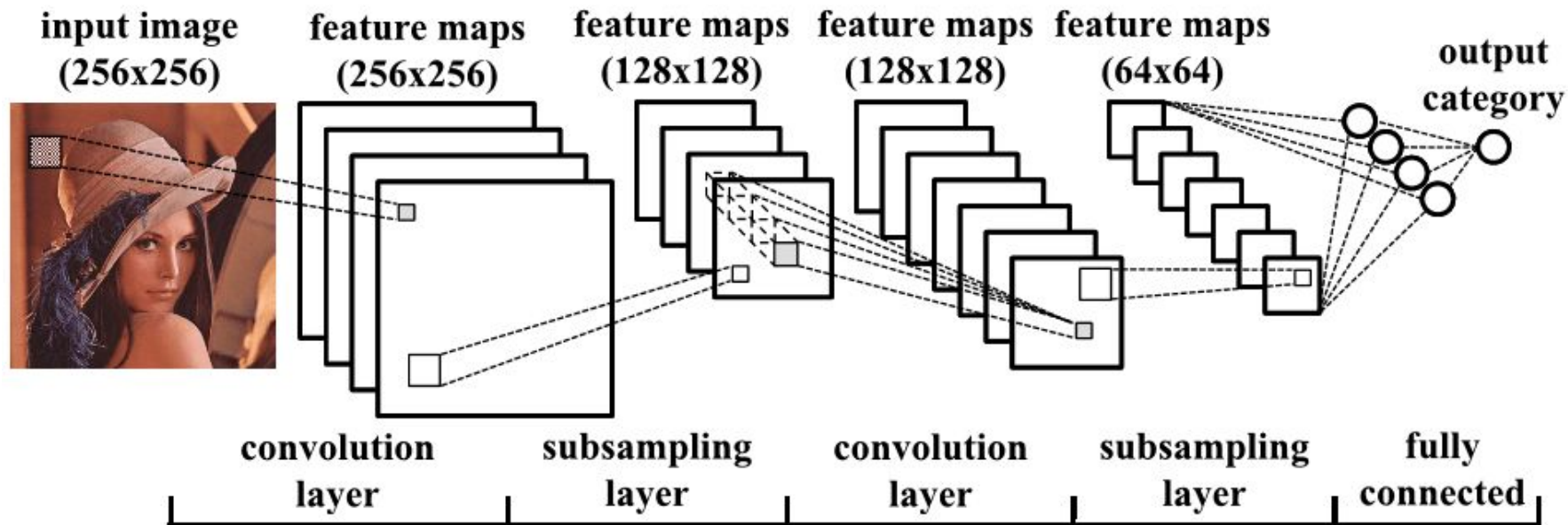
- Introdução;
- *Deep Features*;
- Tipos de Inicialização de Treinamento;
- Implementando CNN com *Keras*

# 1

## Introdução



# REDE NEURAL CONVOLUCIONAL (CNN)

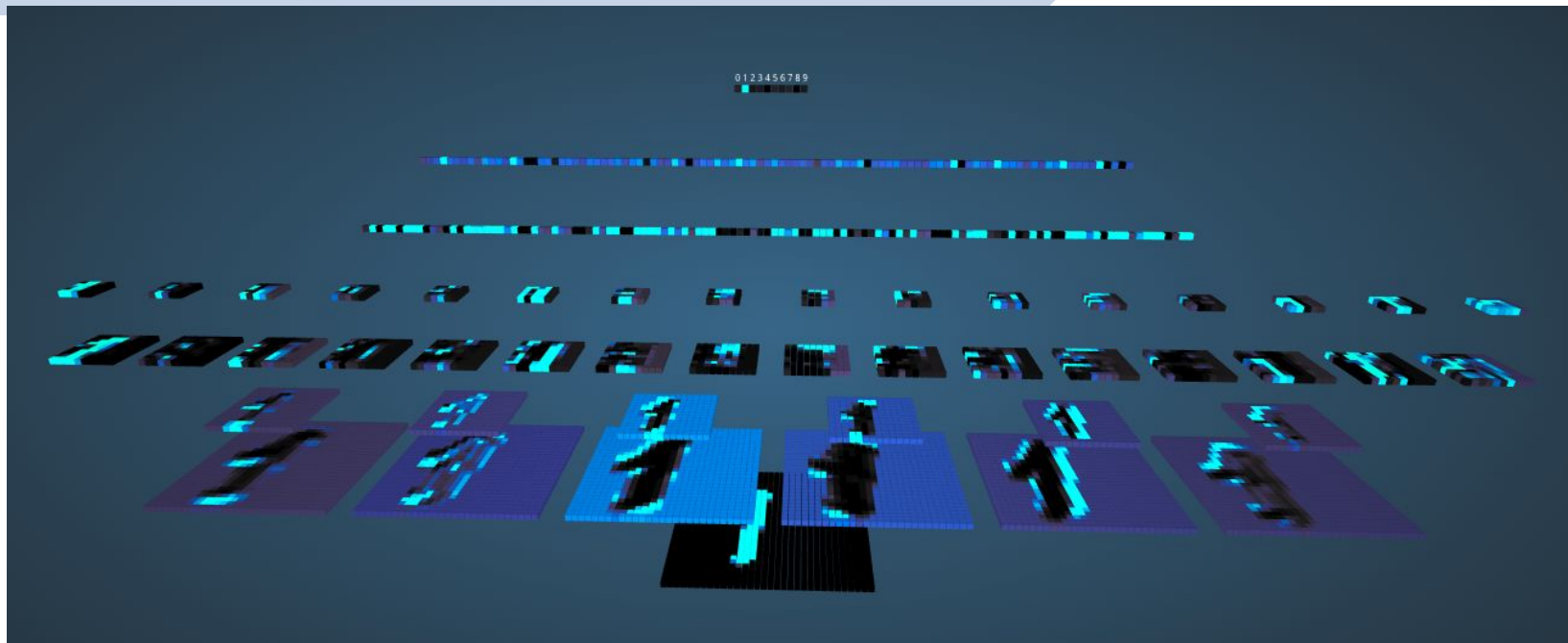


# 2

## *Deep Features*

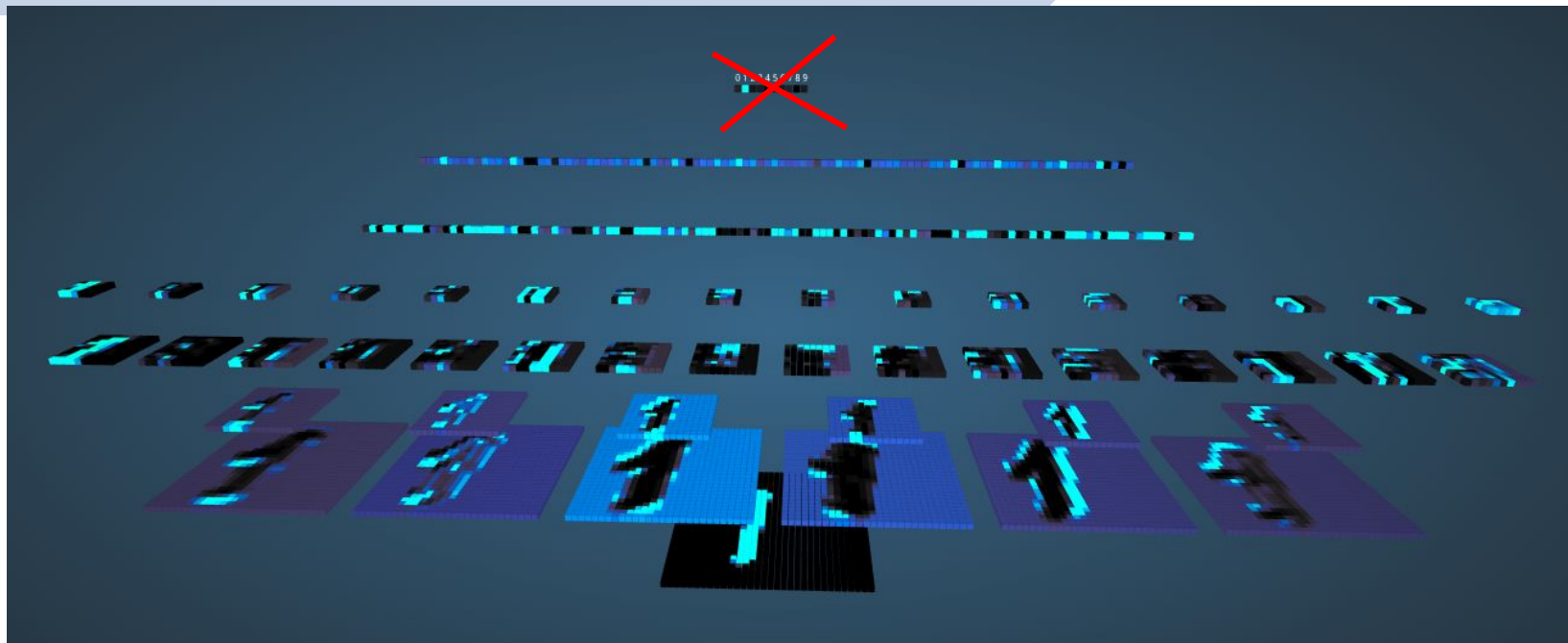


## DEEP FEATURES





## DEEP FEATURES



# 3

## Tipos de Inicialização de Treino





## TREINO “DO ZERO”

- Inicia Rede com pesos randômicos;
- Efetua os ajustes dos pesos do zero;
- Pode demorar encontrar modelo ótimo;
- Precisa de mais amostras de treinamento;



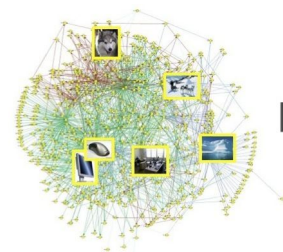
## TREINO “FINE TUNING”

- Inicia Rede com pesos já treinados em outras bases;
  - ▷ Conceito de *Transfer Learning*
- Efetua os ajustes dos pesos com a nova base;
- Pode encontrar modelo ótimo de forma mais rápida;
- Precisa de menos amostras de treinamento;



## TREINO “FINE TUNING” - IMAGENET CHALLENGE

- Inicia Rede com pesos já treinados em outras bases;
  - Conceito de *Transfer Learning*
- Efetua os ajustes dos pesos com a nova base;
- Pode encontrar modelo ótimo de forma mais rápida;
- Precisa de menos amostras de treinamento;



IMAGENET

# Prática

Usando *Keras* API para  
treinar uma CNN simples

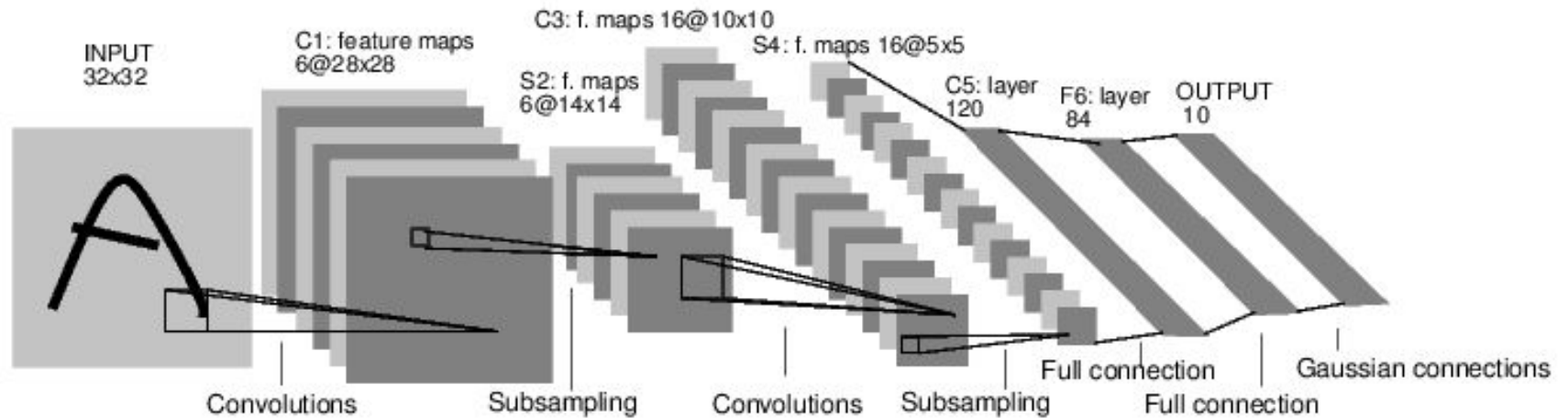


## PRÁTICA - Objetivo

- Objetivo:
  - ▷ Aprender fundamentos de *Deep Learning*, usando *keras*



# PRÁTICA - CNN





## PRÁTICA - Git Repository

- git clone

[https://gitlab.com/lucasmaia1202/vc\\_cnn.git](https://gitlab.com/lucasmaia1202/vc_cnn.git)



## PRÁTICA - Libs

- #Verificar se todas a libs estão instaladas
- terminal\$: `python checkup.py`





## PRÁTICA - Libs

- `import setup`
- `import ip`
- `import keras`



- `from keras.models import Sequential`
- `from keras.layers import Dense, Dropout, Flatten`
- `from keras.layers import Conv2D, MaxPooling2D`



## PRÁTICA - Validação





## PRÁTICA - Separar Conjuntos

- # Declarar lista de dicionários informando as bases
- `database = [{"url": "Digitos/", "img_type": "jpg"}]`



## PRÁTICA - Separar Conjuntos

- # Declarar sets
- `(train_list, y_train), (test_list, y_test), (valid_list, y_valid) = setup.config_base(database=database, test_prop=0.3, valid_prop=0.1)`



## PRÁTICA - Variáveis Necessárias

- `img_rows = 32`
- `img_cols = 32`
- `channels = 3`



## PRÁTICA - Carregar Imagens em Arrays

- `X_train = ip.list_to_array(train_list, (img_rows, img_cols), channels)`
- `X_test = ip.list_to_array(test_list, (img_rows, img_cols), channels)`
- `X_valid = ip.list_to_array(valid_list, (img_rows, img_cols), channels)`



## PRÁTICA - Imprimir *Shapes*

- # Salvar arquivo e roda
- `print(X_train.shape)`
- `print(X_test.shape)`
- `print(X_valid.shape)`





## PRÁTICA - Transformar Saídas

- **num\_classes** = 10
- `y_train = keras.utils.to_categorical(y_train, num_classes)`
- `y_test = keras.utils.to_categorical(y_test, num_classes)`
- `y_valid = keras.utils.to_categorical(y_valid, num_classes)`



## PRÁTICA - Agora SIM a CNN

- **cnn** = Sequential()
- **cnn.add**(Conv2D(  
    **filters**=6, **kernel\_size**=(3, 3),  
    **activation**='relu',  
    **input\_shape**=(img\_rows, img\_cols, channels)  
))



## PRÁTICA - Agora SIM a CNN

- `cnn.add(MaxPooling2D(pool_size=(2, 2)))`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(MaxPooling2D(pool_size=(2, 2)))`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(Dropout(0.25))`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(Flatten())`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(Dense(units=120, activation="relu"))`





## PRÁTICA - Agora SIM a CNN

- `cnn.add(Dense(units=84, activation="relu"))`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(Dropout(0.5))`



## PRÁTICA - Agora SIM a CNN

- `cnn.add(Dense(units=num_classes, activation="softmax"))`



## PRÁTICA - Só falta compilar a CNN criada

```
■ cnn.compile(  
    loss=keras.losses.categorical_crossentropy,  
    optimizer=keras.optimizers.SGD(),  
    metrics=['accuracy']  
)
```



## PRÁTICA - Treino da CNN

```
■ cnn.fit(X_train, y_train,  
         batch_size=128,  
         epochs=12,  
         verbose=1,  
         validation_data=(X_valid, y_valid)  
         )
```



## PRÁTICA - Avaliação da CNN

- `score = cnn.evaluate(X_test, y_test)`
- `print("Acuracia:" score[1])`



# OBIGADO!

Dúvidas?

lucasmaia1202@gmail.com