Aula 7

Bibliotecas de dados

Bibliotecas de modelos pré-treinados

Eduardo Lobo Lustosa Cabral

1. Objetivos

Apresentar bilbiotecas de conjunto de dados: TensorFlow Data Service (TFDS), Kaggle, Keras.

Apresentar bibliotecas de redes neurais pré-treinadas: TensorFlow Hub e Keras.

2. Biblitecas de conjuntos de dados

Existe um número surpreendente de conjuntos de dados abertos na internet e vários sites excelentes que indicam onde encontrar dados.

As coleções de conjuntos de dados mais úteis são as seguintes:

- Repositório de dados de aprendizado de máquina da UCI (University of California Irvine) mantém uma coleção de conjuntos de dados para aprendizado de máquina marcados por tipo de tarefa.
- O Kaggle possui uma coleção de conjuntos de dados muito grande. O Kaggle também tem concursos para mineração de dados e informações sobre empregos em ciência de dados.
- O KDnuggets é outra coleção de conjuntos de dados, sendo a maioria gratuita. Dentro do KDnuggets, existem links para dados governamentais, APIs de dados e competições.
- Além desses sites, existem muitos outros locais onde encontrar dados, como por exemplo, dados de governos e organizações não govenamentais, tais como, US Government Data, UK Government Data, Canada's Open Data Exchange, World Health Organization e o World Bank.
- O TensorFlow Data Service é uma coleção de dados com conjuntos de dados prontos para serem usados de forma muito fácil.
- O Keras também possui uma pequena coleção de conjuntos de dados muito fácil de ser usada.

2.1 Kaggle

O Kaggle é a maior comunidade de machine learning do mundo (www.kaggle.com).

O Kaggle promove competições onde o modelo com melhor desempenho na tarefa determinada ganha prêmio em dinheiro.

O Kaggle possui a maior coleção de conjunto de dados do mundo e esses dados estão disponíveis livremente (https://www.kaggle.com/datasets).

Os conjuntos de dados do Kaggle cobrem todas as áreas possíveis e são muito bons para testar novos algoritmos.

O Kaggle também oferece cursos livres nas áreas de Machine Learning e Ciências de Dados.

2.2 Repositório de dados da UCI

A UCI matém uma coleção de cerca de 560 conjuntos de dados de diversos tipos para diferentes tipos de tarefas https://archive.ics.uci.edu/ml/datasets.php.

Em geral os conjuntos de dados desse repositório são pequenos e um pouco limtados, porém são úteis para "começar".

2.3 TensorFlow DataService (TFDS)

O TFDS consiste de uma biblioteca de conjuntos de dados de vários tipos de problemas que podem ser usados livremente.

Para usar essa bilbioteca ela deve ser importada junto com o TensorFlow.

```
import tensorflow as tf
import tensorflow_datasets as tfds
print("Using TensorFlow Version:", tf.__version__)
```

```
Using TensorFlow Version: 2.17.0
```

Os conjuntos de dados existem podem ser vistos em https://www.tensorflow.org/datasets/catalog/overview. Nesse link tem uma descrição detalhada de todos os conjuntos de dados.

Para ver a lista dos conjuntos de dados disponíveis no TFDS pode-se também usar:

```
lista = tfds.list_builders()
print("Número de datasets =", len(lista))
print(' ')
print('Lista de conjuntos de dados do TFDS')
for name in lista:
    print(name)
```

```
Número de datasets = 1291
```

```
Lista de conjuntos de dados do TFDS
abstract_reasoning
accentdb
aeslc
aflw2k3d
ag_news_subset
ai2_arc
ai2_arc_with_ir
aloha_mobile
amazon_us_reviews
anli
answer_equivalence
arc
asqa
asset
assin2
asu_table_top_converted_externally_to_rlds
austin_buds_dataset_converted_externally_to_rlds
austin_sailor_dataset_converted_externally_to_rlds
austin_sirius_dataset_converted_externally_to_rlds
bair_robot_pushing_small
bc_z
bccd
beans
bee_dataset
beir
berkeley_autolab_ur5
berkeley_cable_routing
berkeley_fanuc_manipulation
berkeley_gnm_cory_hall
berkeley_gnm_recon
berkeley_gnm_sac_son
berkeley_mvp_converted_externally_to_rlds
berkeley_rpt_converted_externally_to_rlds
big_patent
bigearthnet
billsum
binarized_mnist
binary_alpha_digits
ble_wind_field
blimp
booksum
bool_q
bot_adversarial_dialogue
bridge
bridge_data_msr
bucc
c4
c4_wsrs
```

```
caltech101
caltech_birds2010
caltech_birds2011
cardiotox
cars196
cassava
cats_vs_dogs
celeb_a
celeb_a_hq
cfq
cherry_blossoms
chexpert
cifar10
cifar100
cifar100_n
cifar10_1
cifar10_corrupted
cifar10_h
cifar10_n
citrus_leaves
cityscapes
civil_comments
clevr
clic
clinc_oos
cmaterdb
cmu_franka_exploration_dataset_converted_externally_to_rlds
cmu_play_fusion
cmu_stretch
cnn_dailymail
coco
coco_captions
coil100
colorectal_histology
colorectal_histology_large
columbia_cairlab_pusht_real
common_voice
conll2002
conll2003
conq_hose_manipulation
controlled_noisy_web_labels
coqa
corr2cause
cos_e
cosmos_qa
covid19
covid19sum
crema_d
criteo
```

```
cs_restaurants
curated_breast_imaging_ddsm
cycle_gan
d4rl_adroit_door
d4rl_adroit_hammer
d4rl_adroit_pen
d4rl_adroit_relocate
d4rl_antmaze
d4rl_mujoco_ant
d4rl_mujoco_halfcheetah
d4rl_mujoco_hopper
d4rl_mujoco_walker2d
dart
databricks_dolly
davis
deep1b
deep_weeds
definite_pronoun_resolution
dementiabank
diabetic_retinopathy_detection
diamonds
dices
div2k
dlr_edan_shared_control_converted_externally_to_rlds
dlr_sara_grid_clamp_converted_externally_to_rlds
dlr_sara_pour_converted_externally_to_rlds
dmlab
dobbe
doc_nli
dolphin_number_word
domainnet
downsampled_imagenet
drop
dsprites
dtd
duke_ultrasound
e2e_cleaned
efron_morris75
emnist
eraser_multi_rc
esnli
eth_agent_affordances
eurosat
fashion_mnist
flic
flores
fmb
food101
forest_fires
```

```
fractal20220817_data
fuss
gap
geirhos_conflict_stimuli
gem
genomics_ood
german_credit_numeric
gigaword
glove100_angular
glue
goemotions
gov_report
gpt3
gref
groove
grounded_scan
gsm8k
gtzan
gtzan_music_speech
hellaswag
higgs
hillstrom
horses_or_humans
howell
i_naturalist2017
i_naturalist2018
i_naturalist2021
iamlab_cmu_pickup_insert_converted_externally_to_rlds
imagenet2012
imagenet2012_corrupted
imagenet2012_fewshot
imagenet2012_multilabel
imagenet2012_real
imagenet2012_subset
imagenet_a
imagenet_lt
imagenet_pi
imagenet_r
imagenet_resized
imagenet_sketch
imagenet_v2
imagenette
imagewang
imdb_reviews
imperialcollege_sawyer_wrist_cam
io_ai_tech
irc_disentanglement
iris
istella
```

```
jaco_play
kaist_nonprehensile_converted_externally_to_rlds
kddcup99
kitti
kmnist
kuka
laion400m
lambada
lfw
librispeech
librispeech_lm
libritts
ljspeech
lm1b
locomotion
lost_and_found
lsun
lvis
malaria
maniskill_dataset_converted_externally_to_rlds
math_dataset
math_qa
mctaco
media_sum
mimic_play
mlqa
mnist
mnist_corrupted
movie_lens
movie_rationales
movielens
moving_mnist
mrqa
mslr_web
mt_opt
mtnt
multi_news
multi_nli
multi_nli_mismatch
natural_instructions
natural_questions
natural_questions_open
newsroom
nsynth
nyu_depth_v2
nyu_door_opening_surprising_effectiveness
nyu_franka_play_dataset_converted_externally_to_rlds
nyu_rot_dataset_converted_externally_to_rlds
ogbg_molpcba
```

```
omniglot
open_images_challenge2019_detection
open_images_v4
openbookqa
opinion_abstracts
opinosis
opus
oxford_flowers102
oxford_iiit_pet
para_crawl
pass
patch_camelyon
paws_wiki
paws_x_wiki
penguins
pet_finder
pg19
piqa
places365_small
placesfull
plant_leaves
plant_village
plantae_k
plex_robosuite
protein_net
q_re_cc
qa4mre
qasc
qm9
quac
quality
quickdraw_bitmap
race
radon
real_toxicity_prompts
reddit
reddit_disentanglement
reddit_tifu
ref_coco
resisc45
rlu_atari
rlu_atari_checkpoints
rlu_atari_checkpoints_ordered
rlu_control_suite
rlu_dmlab_explore_object_rewards_few
rlu_dmlab_explore_object_rewards_many
rlu_dmlab_rooms_select_nonmatching_object
rlu_dmlab_rooms_watermaze
rlu_dmlab_seekavoid_arena01
```

```
rlu_locomotion
rlu_rwrl
robo_set
robomimic_mg
robomimic_mh
robomimic_ph
robonet
robosuite_panda_pick_place_can
roboturk
rock_paper_scissors
rock_you
s3o4d
salient_span_wikipedia
samsum
savee
scan
scene_parse150
schema_guided_dialogue
sci_tail
scicite
scientific_papers
scrolls
segment_anything
sentiment140
shapes3d
sift1m
simpte
siscore
smallnorb
smartwatch_gestures
snli
so2sat
speech_commands
spoc_robot
spoken_digit
squad
squad_question_generation
stanford_dogs
stanford_hydra_dataset_converted_externally_to_rlds
stanford_kuka_multimodal_dataset_converted_externally_to_rlds
stanford_mask_vit_converted_externally_to_rlds
stanford_online_products
stanford_robocook_converted_externally_to_rlds
star_cfq
starcraft_video
stl10
story_cloze
summscreen
sun397
```

```
super_glue
svhn_cropped
symmetric_solids
taco_play
tao
tatoeba
ted_hrlr_translate
ted_multi_translate
tedlium
tf_flowers
the300w_lp
tidybot
tiny_shakespeare
titanic
tokyo_u_lsmo_converted_externally_to_rlds
toto
trec
trivia_qa
tydi_qa
uc_merced
ucf101
ucsd_kitchen_dataset_converted_externally_to_rlds
ucsd_pick_and_place_dataset_converted_externally_to_rlds
uiuc_d3field
unified_qa
universal_dependencies
unnatural_instructions
usc_cloth_sim_converted_externally_to_rlds
user_libri_audio
user_libri_text
utaustin_mutex
utokyo_pr2_opening_fridge_converted_externally_to_rlds
utokyo_pr2_tabletop_manipulation_converted_externally_to_rlds
utokyo_saytap_converted_externally_to_rlds
utokyo_xarm_bimanual_converted_externally_to_rlds
utokyo_xarm_pick_and_place_converted_externally_to_rlds
vctk
vima_converted_externally_to_rlds
viola
visual_domain_decathlon
voc
voxceleb
voxforge
waymo_open_dataset
web_graph
web_nlg
web_questions
webvid
wider_face
```

```
wiki40b
wiki_auto
wiki_bio
wiki_dialog
wiki_table_questions
wiki_table_text
wikiann
wikihow
wikipedia
wikipedia_toxicity_subtypes
wine_quality
winogrande
wit
wit_kaggle
wmt13_translate
wmt14_translate
wmt15_translate
wmt16_translate
wmt17_translate
wmt18_translate
wmt19_translate
wmt_t2t_translate
wmt_translate
wordnet
wsc273
xnli
xquad
xsum
xtreme_pawsx
xtreme_pos
xtreme_s
xtreme_xnli
yahoo_ltrc
yelp_polarity_reviews
yes_no
youtube_vis
huggingface:acronym_identification
huggingface:ade_corpus_v2
huggingface:adv_glue
huggingface:adversarial_qa
huggingface:aeslc
huggingface:afrikaans_ner_corpus
huggingface:ag_news
huggingface:ai2_arc
huggingface:air_dialogue
huggingface:ajgt_twitter_ar
huggingface:allegro_reviews
huggingface:allocine
huggingface:alt
```

```
huggingface:amazon_polarity
huggingface:amazon_reviews_multi
huggingface:amazon_us_reviews
huggingface:ambig_qa
huggingface:americas_nli
huggingface:ami
huggingface:amttl
huggingface:anli
huggingface:app_reviews
huggingface:aqua_rat
huggingface:aquamuse
huggingface:ar_cov19
huggingface:ar_res_reviews
huggingface:ar_sarcasm
huggingface:arabic_billion_words
huggingface:arabic_pos_dialect
huggingface:arabic_speech_corpus
huggingface:arcd
huggingface:arsentd_lev
huggingface:art
huggingface:arxiv_dataset
huggingface:ascent_kb
huggingface:aslg_pc12
huggingface:asnq
huggingface:asset
huggingface:assin
huggingface:assin2
huggingface:atomic
huggingface:autshumato
huggingface:babi_qa
huggingface:banking77
huggingface:bbaw_egyptian
huggingface:bbc_hindi_nli
huggingface:bc2gm_corpus
huggingface:beans
huggingface:best2009
huggingface:bianet
huggingface:bible_para
huggingface:big_patent
huggingface:bigbench
huggingface:billsum
huggingface:bing_coronavirus_query_set
huggingface:biomrc
huggingface:biosses
huggingface:biwi_kinect_head_pose
huggingface:blbooks
huggingface:blbooksgenre
huggingface:blended_skill_talk
huggingface:blimp
huggingface:blog_authorship_corpus
```

```
huggingface:bn_hate_speech
huggingface:bnl_newspapers
huggingface:bookcorpus
huggingface:bookcorpusopen
huggingface:boolq
huggingface:bprec
huggingface:break_data
huggingface:brwac
huggingface:bsd_ja_en
huggingface:bswac
huggingface:c3
huggingface:c4
huggingface:cail2018
huggingface:caner
huggingface:capes
huggingface:casino
huggingface:catalonia_independence
huggingface:cats_vs_dogs
huggingface:cawac
huggingface:cbt
huggingface:cc100
huggingface:cc_news
huggingface:ccaligned_multilingual
huggingface:cdsc
huggingface:cdt
huggingface:cedr
huggingface:cfq
huggingface:chr_en
huggingface:cifar10
huggingface:cifar100
huggingface:circa
huggingface:civil_comments
huggingface:clickbait_news_bg
huggingface:climate_fever
huggingface:clinc_oos
huggingface:clue
huggingface:cmrc2018
huggingface:cmu_hinglish_dog
huggingface:cnn_dailymail
huggingface:coached_conv_pref
huggingface:coarse_discourse
huggingface:codah
huggingface:code_search_net
huggingface:code_x_glue_cc_clone_detection_big_clone_bench
huggingface:code_x_glue_cc_clone_detection_poj104
huggingface:code_x_glue_cc_cloze_testing_all
huggingface:code_x_glue_cc_cloze_testing_maxmin
huggingface:code_x_glue_cc_code_completion_line
huggingface:code_x_glue_cc_code_completion_token
```

```
huggingface:code_x_glue_cc_code_refinement
huggingface:code_x_glue_cc_code_to_code_trans
huggingface:code_x_glue_cc_defect_detection
huggingface:code_x_glue_ct_code_to_text
huggingface:code_x_glue_tc_nl_code_search_adv
huggingface:code_x_glue_tc_text_to_code
huggingface:code_x_glue_tt_text_to_text
huggingface:com_qa
huggingface:common_gen
huggingface:common_language
huggingface:common_voice
huggingface:commonsense_qa
huggingface:competition_math
huggingface:compguesswhat
huggingface:conceptnet5
huggingface:conceptual_12m
huggingface:conceptual_captions
huggingface:conll2000
huggingface:conll2002
huggingface:conll2003
huggingface:conll2012_ontonotesv5
huggingface:conllpp
huggingface:consumer-finance-complaints
huggingface:conv_ai
huggingface:conv_ai_2
huggingface:conv_ai_3
huggingface:conv_questions
huggingface:coqa
huggingface:cord19
huggingface:cornell_movie_dialog
huggingface:cos_e
huggingface:cosmos_qa
huggingface:counter
huggingface:covid_qa_castorini
huggingface:covid_qa_deepset
huggingface:covid_qa_ucsd
huggingface:covid_tweets_japanese
huggingface:covost2
huggingface:cppe-5
huggingface:craigslist_bargains
huggingface:crawl_domain
huggingface:crd3
huggingface:crime_and_punish
huggingface:crows_pairs
huggingface:cryptonite
huggingface:cs_restaurants
huggingface:cuad
huggingface:curiosity_dialogs
huggingface:daily_dialog
```

```
huggingface:dane
huggingface:danish_political_comments
huggingface:dart
huggingface:datacommons_factcheck
huggingface:dbpedia_14
huggingface:dbrd
huggingface:deal_or_no_dialog
huggingface:definite_pronoun_resolution
huggingface:dengue_filipino
huggingface:dialog_re
huggingface:diplomacy_detection
huggingface:disaster_response_messages
huggingface:discofuse
huggingface:discovery
huggingface:disfl_qa
huggingface:doc2dial
huggingface:docred
huggingface:doqa
huggingface:dream
huggingface:drop
huggingface:duorc
huggingface:dutch_social
huggingface:dyk
huggingface:e2e_nlg
huggingface:e2e_nlg_cleaned
huggingface:ecb
huggingface:ecthr_cases
huggingface:eduge
huggingface:ehealth_kd
huggingface:eitb_parcc
huggingface:electricity_load_diagrams
huggingface:eli5
huggingface:eli5_category
huggingface:elkarhizketak
huggingface:emea
huggingface:emo
huggingface:emotion
huggingface:emotone_ar
huggingface:empathetic_dialogues
huggingface:enriched_web_nlg
huggingface:enwik8
huggingface:eraser_multi_rc
huggingface:esnli
huggingface:eth_py150_open
huggingface:ethos
huggingface:ett
huggingface:eu_regulatory_ir
huggingface:eurlex
huggingface:euronews
```

```
huggingface:europa_eac_tm
huggingface:europa_ecdc_tm
huggingface:europarl_bilingual
huggingface:event2Mind
huggingface:evidence_infer_treatment
huggingface:exams
huggingface:factckbr
huggingface:fake_news_english
huggingface:fake_news_filipino
huggingface:farsi_news
huggingface:fashion_mnist
huggingface:fever
huggingface:few_rel
huggingface:financial_phrasebank
huggingface:finer
huggingface:flores
huggingface:flue
huggingface:food101
huggingface:fquad
huggingface:freebase_qa
huggingface:gap
huggingface:gem
huggingface:generated_reviews_enth
huggingface:generics_kb
huggingface:german_legal_entity_recognition
huggingface:germaner
huggingface:germeval_14
huggingface:giga_fren
huggingface:gigaword
huggingface:glucose
huggingface:glue
huggingface:gnad10
huggingface:go_emotions
huggingface:gooaq
huggingface:google_wellformed_query
huggingface:grail_qa
huggingface:great_code
huggingface:greek_legal_code
huggingface:gsm8k
huggingface:guardian_authorship
huggingface:gutenberg_time
huggingface:hans
huggingface:hansards
huggingface:hard
huggingface:harem
huggingface:has_part
huggingface:hate_offensive
huggingface:hate_speech18
huggingface:hate_speech_filipino
```

```
huggingface:hate_speech_offensive
huggingface:hate_speech_pl
huggingface:hate_speech_portuguese
huggingface:hatexplain
huggingface:hausa_voa_ner
huggingface:hausa_voa_topics
huggingface:hda_nli_hindi
huggingface:head_qa
huggingface:health_fact
huggingface:hebrew_projectbenyehuda
huggingface:hebrew_sentiment
huggingface:hebrew_this_world
huggingface:hellaswag
huggingface:hendrycks_test
huggingface:hind_encorp
huggingface:hindi_discourse
huggingface:hippocorpus
huggingface:hkcancor
huggingface:hlgd
huggingface:hope_edi
huggingface:hotpot_qa
huggingface:hover
huggingface:hrenwac_para
huggingface:hrwac
huggingface:humicroedit
huggingface:hybrid_qa
huggingface:hyperpartisan_news_detection
huggingface:iapp_wiki_qa_squad
huggingface:id_clickbait
huggingface:id_liputan6
huggingface:id_nergrit_corpus
huggingface:id_newspapers_2018
huggingface:id_panl_bppt
huggingface:id_puisi
huggingface:igbo_english_machine_translation
huggingface:igbo_monolingual
huggingface:igbo_ner
huggingface:ilist
huggingface:imagenet-1k
huggingface:imagenet_sketch
huggingface:imdb
huggingface:imdb_urdu_reviews
huggingface:imppres
huggingface:indic_glue
huggingface:indonli
huggingface:indonlu
huggingface:inquisitive_qg
huggingface:interpress_news_category_tr
huggingface:interpress_news_category_tr_lite
```

```
huggingface:irc_disentangle
huggingface:isixhosa_ner_corpus
huggingface:isizulu_ner_corpus
huggingface:iwslt2017
huggingface:jeopardy
huggingface:jfleg
huggingface:jigsaw_toxicity_pred
huggingface:jigsaw_unintended_bias
huggingface:jnlpba
huggingface:journalists_questions
huggingface:kan_hope
huggingface:kannada_news
huggingface:kd_conv
huggingface:kde4
huggingface:kelm
huggingface:kilt_tasks
huggingface:kilt_wikipedia
huggingface:kinnews_kirnews
huggingface:klue
huggingface:kor_3i4k
huggingface:kor_hate
huggingface:kor_ner
huggingface:kor_nli
huggingface:kor_nlu
huggingface:kor_qpair
huggingface:kor_sae
huggingface:kor_sarcasm
huggingface:labr
huggingface:lama
huggingface:lambada
huggingface:large_spanish_corpus
huggingface:laroseda
huggingface:lc_quad
huggingface:lccc
huggingface:lener_br
huggingface:lex_glue
huggingface:liar
huggingface:librispeech_asr
huggingface:librispeech_lm
huggingface:limit
huggingface:lince
huggingface:linnaeus
huggingface:liveqa
huggingface:lj_speech
huggingface:lm1b
huggingface:lst20
huggingface:m_lama
huggingface:mac_morpho
huggingface:makhzan
```

```
huggingface:masakhaner
huggingface:math_dataset
huggingface:math_qa
huggingface:matinf
huggingface:mbpp
huggingface:mc4
huggingface:mc_taco
huggingface:md_gender_bias
huggingface:mdd
huggingface:med_hop
huggingface:medal
huggingface:medical_dialog
huggingface:medical_questions_pairs
huggingface:medmcqa
huggingface:menyo20k_mt
huggingface:meta_woz
huggingface:metashift
huggingface:metooma
huggingface:metrec
huggingface:miam
huggingface:mkb
huggingface:mkqa
huggingface:mlqa
huggingface:mlsum
huggingface:mnist
huggingface:mocha
huggingface:monash_tsf
huggingface:moroco
huggingface:movie_rationales
huggingface:mrqa
huggingface:ms_marco
huggingface:ms_terms
huggingface:msr_genomics_kbcomp
huggingface:msr_sqa
huggingface:msr_text_compression
huggingface:msr_zhen_translation_parity
huggingface:msra_ner
huggingface:mt_eng_vietnamese
huggingface:muchocine
huggingface:multi_booked
huggingface:multi_eurlex
huggingface:multi_news
huggingface:multi_nli
huggingface:multi_nli_mismatch
huggingface:multi_para_crawl
huggingface:multi_re_qa
huggingface:multi_woz_v22
huggingface:multi_x_science_sum
huggingface:multidoc2dial
```

```
huggingface:multilingual_librispeech
huggingface:mutual_friends
huggingface:mwsc
huggingface:myanmar_news
huggingface:narrativeqa
huggingface:narrativeqa_manual
huggingface:natural_questions
huggingface:ncbi_disease
huggingface:nchlt
huggingface:ncslgr
huggingface:nell
huggingface:neural_code_search
huggingface:news_commentary
huggingface:newsgroup
huggingface:newsph
huggingface:newsph_nli
huggingface:newspop
huggingface:newsqa
huggingface:newsroom
huggingface:nkjp-ner
huggingface:nli_tr
huggingface:nlu_evaluation_data
huggingface:norec
huggingface:norne
huggingface:norwegian_ner
huggingface:nq_open
huggingface:nsmc
huggingface:numer_sense
huggingface:numeric_fused_head
huggingface:oclar
huggingface:offcombr
huggingface:offenseval2020_tr
huggingface:offenseval_dravidian
huggingface:ofis_publik
huggingface:ohsumed
huggingface:ollie
huggingface:omp
huggingface:onestop_english
huggingface:onestop_qa
huggingface:open_subtitles
huggingface:openai_humaneval
huggingface:openbookqa
huggingface:openslr
huggingface:openwebtext
huggingface:opinosis
huggingface:opus100
huggingface:opus_books
huggingface:opus_dgt
huggingface:opus_dogc
```

```
huggingface:opus_elhuyar
huggingface:opus_euconst
huggingface:opus_finlex
huggingface:opus_fiskmo
huggingface:opus_gnome
huggingface:opus_infopankki
huggingface:opus_memat
huggingface:opus_montenegrinsubs
huggingface:opus_openoffice
huggingface:opus_paracrawl
huggingface:opus_rf
huggingface:opus_tedtalks
huggingface:opus_ubuntu
huggingface:opus_wikipedia
huggingface:opus_xhosanavy
huggingface:orange_sum
huggingface:oscar
huggingface:para_crawl
huggingface:para_pat
huggingface:parsinlu_reading_comprehension
huggingface:pass
huggingface:paws
huggingface:paws-x
huggingface:pec
huggingface:peer_read
huggingface:peoples_daily_ner
huggingface:per_sent
huggingface:persian_ner
huggingface:pg19
huggingface:php
huggingface:piaf
huggingface:pib
huggingface:piqa
huggingface:pn_summary
huggingface:poem_sentiment
huggingface:polemo2
huggingface:poleval2019_cyberbullying
huggingface:poleval2019_mt
huggingface:polsum
huggingface:polyglot_ner
huggingface:prachathai67k
huggingface:pragmeval
huggingface:proto_qa
huggingface:psc
huggingface:ptb_text_only
huggingface:pubmed
huggingface:pubmed_qa
huggingface:py_ast
huggingface:qa4mre
```

```
huggingface:qa_srl
huggingface:qa_zre
huggingface:qangaroo
huggingface:qanta
huggingface:qasc
huggingface:qasper
huggingface:qed
huggingface:qed_amara
huggingface:quac
huggingface:quail
huggingface:quarel
huggingface:quartz
huggingface:quickdraw
huggingface:quora
huggingface:quoref
huggingface:race
huggingface:re_dial
huggingface:reasoning_bg
huggingface:recipe_nlg
huggingface:reclor
huggingface:red_caps
huggingface:reddit
huggingface:reddit_tifu
huggingface:refresd
huggingface:reuters21578
huggingface:riddle_sense
huggingface:ro_sent
huggingface:ro_sts
huggingface:ro_sts_parallel
huggingface:roman_urdu
huggingface:roman_urdu_hate_speech
huggingface:ronec
huggingface:ropes
huggingface:rotten_tomatoes
huggingface:russian_super_glue
huggingface:rvl_cdip
huggingface:s2orc
huggingface:samsum
huggingface:sanskrit_classic
huggingface:saudinewsnet
huggingface:sberquad
huggingface:sbu_captions
huggingface:scan
huggingface:scb_mt_enth_2020
huggingface:scene_parse_150
huggingface:schema_guided_dstc8
huggingface:scicite
huggingface:scielo
huggingface:scientific_papers
```

```
huggingface:scifact
huggingface:sciq
huggingface:scitail
huggingface:scitldr
huggingface:search_qa
huggingface:sede
huggingface:selqa
huggingface:sem_eval_2010_task_8
huggingface:sem_eval_2014_task_1
huggingface:sem_eval_2018_task_1
huggingface:sem_eval_2020_task_11
huggingface:sent_comp
huggingface:senti_lex
huggingface:senti_ws
huggingface:sentiment140
huggingface:sepedi_ner
huggingface:sesotho_ner_corpus
huggingface:setimes
huggingface:setswana_ner_corpus
huggingface:sharc
huggingface:sharc_modified
huggingface:sick
huggingface:silicone
huggingface:simple_questions_v2
huggingface:siswati_ner_corpus
huggingface:smartdata
huggingface:sms_spam
huggingface:snips_built_in_intents
huggingface:snli
huggingface:snow_simplified_japanese_corpus
huggingface:so_stacksample
huggingface:social_bias_frames
huggingface:social_i_qa
huggingface:sofc_materials_articles
huggingface:sogou_news
huggingface:spanish_billion_words
huggingface:spc
huggingface:species_800
huggingface:speech_commands
huggingface:spider
huggingface:squad
huggingface:squad_adversarial
huggingface:squad_es
huggingface:squad_it
huggingface:squad_kor_v1
huggingface:squad_kor_v2
huggingface:squad_v1_pt
huggingface:squad_v2
huggingface:squadshifts
huggingface:srwac
```

```
huggingface:sst
huggingface:stereoset
huggingface:story_cloze
huggingface:stsb_mt_sv
huggingface:stsb_multi_mt
huggingface:style_change_detection
huggingface:subjqa
huggingface:super_glue
huggingface:superb
huggingface:svhn
huggingface:swag
huggingface:swahili
huggingface:swahili_news
huggingface:swda
huggingface:swedish_medical_ner
huggingface:swedish_ner_corpus
huggingface:swedish_reviews
huggingface:swiss_judgment_prediction
huggingface:tab_fact
huggingface:tamilmixsentiment
huggingface:tanzil
huggingface:tapaco
huggingface:tashkeela
huggingface:taskmaster1
huggingface:taskmaster2
huggingface:taskmaster3
huggingface:tatoeba
huggingface:ted_hrlr
huggingface:ted_iwlst2013
huggingface:ted_multi
huggingface:ted_talks_iwslt
huggingface:telugu_books
huggingface:telugu_news
huggingface:tep_en_fa_para
huggingface:text2log
huggingface:textvqa
huggingface:thai_toxicity_tweet
huggingface:thainer
huggingface:thaiqa_squad
huggingface:thaisum
huggingface:the_pile
huggingface:the_pile_books3
huggingface:the_pile_openwebtext2
huggingface:the_pile_stack_exchange
huggingface:tilde_model
huggingface:time_dial
huggingface:times_of_india_news_headlines
huggingface:timit_asr
huggingface:tiny_shakespeare
```

```
huggingface:tlc
huggingface:tmu_gfm_dataset
huggingface:tne
huggingface:told-br
huggingface:totto
huggingface:trec
huggingface:trivia_qa
huggingface:truthful_qa
huggingface:tsac
huggingface:ttc4900
huggingface:tunizi
huggingface:tuple_ie
huggingface:turk
huggingface:turkic_xwmt
huggingface:turkish_movie_sentiment
huggingface:turkish_ner
huggingface:turkish_product_reviews
huggingface:turkish_shrinked_ner
huggingface:turku_ner_corpus
huggingface:tweet_eval
huggingface:tweet_qa
huggingface:tweets_ar_en_parallel
huggingface:tweets_hate_speech_detection
huggingface:twi_text_c3
huggingface:twi_wordsim353
huggingface:tydiqa
huggingface:ubuntu_dialogs_corpus
huggingface:udhr
huggingface:um005
huggingface:un_ga
huggingface:un_multi
huggingface:un_pc
huggingface:universal_dependencies
huggingface:universal_morphologies
huggingface:urdu_fake_news
huggingface:urdu_sentiment_corpus
huggingface:vctk
huggingface:visual_genome
huggingface:vivos
huggingface:web_nlg
huggingface:web_of_science
huggingface:web_questions
huggingface:weibo_ner
huggingface:wi_locness
huggingface:wider_face
huggingface:wiki40b
huggingface:wiki_asp
huggingface:wiki_atomic_edits
huggingface:wiki_auto
```

```
huggingface:wiki_bio
huggingface:wiki_dpr
huggingface:wiki_hop
huggingface:wiki_lingua
huggingface:wiki_movies
huggingface:wiki_qa
huggingface:wiki_qa_ar
huggingface:wiki_snippets
huggingface:wiki_source
huggingface:wiki_split
huggingface:wiki_summary
huggingface:wikiann
huggingface:wikicorpus
huggingface:wikihow
huggingface:wikipedia
huggingface:wikisql
huggingface:wikitablequestions
huggingface:wikitext
huggingface:wikitext_tl39
huggingface:wili_2018
huggingface:wino_bias
huggingface:winograd_wsc
huggingface:winogrande
huggingface:wiqa
huggingface:wisesight1000
huggingface:wisesight_sentiment
huggingface:wmt14
huggingface:wmt15
huggingface:wmt16
huggingface:wmt17
huggingface:wmt18
huggingface:wmt19
huggingface:wmt20_mlqe_task1
huggingface:wmt20_mlqe_task2
huggingface:wmt20_mlqe_task3
huggingface:wmt_t2t
huggingface:wnut_17
huggingface:wongnai_reviews
huggingface:woz_dialogue
huggingface:wrbsc
huggingface:x_stance
huggingface:xcopa
huggingface:xcsr
huggingface:xed_en_fi
huggingface:xglue
huggingface:xnli
huggingface:xor_tydi_qa
huggingface:xquad
huggingface:xquad_r
```

```
huggingface:xsum
huggingface:xsum_factuality
huggingface:xtreme
huggingface:yahoo_answers_qa
huggingface:yahoo_answers_topics
huggingface:yelp_polarity
huggingface:yelp_review_full
huggingface:yoruba_bbc_topics
huggingface:yoruba_gv_ner
huggingface:yoruba_text_c3
huggingface:yoruba_wordsim353
huggingface:youtube_caption_corrections
huggingface:zest
kubric:kubric_frames
kubric:movi_a
kubric:movi_b
kubric:movi_c
kubric:movi_d
kubric:movi_e
kubric:movi_f
kubric:msn_easy
kubric:multi_shapenet_frames
kubric:nerf_synthetic_frames
kubric:nerf_synthetic_scenes
kubric:shapenet_pretraining
robotics:agent_aware_affordances
robotics:aloha_mobile
robotics:asu_table_top_converted_externally_to_rlds
robotics:austin_buds_dataset_converted_externally_to_rlds
robotics:austin_sailor_dataset_converted_externally_to_rlds
robotics:austin_sirius_dataset_converted_externally_to_rlds
robotics:bc_z
robotics:berkeley_autolab_ur5
robotics:berkeley_cable_routing
robotics:berkeley_fanuc_manipulation
robotics:berkeley_gnm_cory_hall
robotics:berkeley_gnm_recon
robotics:berkeley_gnm_sac_son
robotics:berkeley_mvp_converted_externally_to_rlds
robotics:berkeley_rpt_converted_externally_to_rlds
robotics:bridge
robotics:bridge_data_msr
robotics:bridge_data_v2
robotics:cmu_franka_exploration_dataset_converted_externally_to_rlds
robotics:cmu_play_fusion
robotics:cmu_playing_with_food
robotics:cmu_stretch
robotics:columbia_cairlab_pusht_real
robotics:conq_hose_manipulation
```

```
robotics:dlr_edan_shared_control_converted_externally_to_rlds
robotics:dlr_sara_grid_clamp_converted_externally_to_rlds
robotics:dlr_sara_pour_converted_externally_to_rlds
robotics:dobbe
robotics:droid
robotics:droid_100
robotics:droid_raw
robotics:eth_agent_affordances
robotics:fanuc_manipulation_v2
robotics:fmb
robotics:fractal20220817_data
robotics:furniture_bench_dataset_converted_externally_to_rlds
robotics:iamlab_cmu_pickup_insert_converted_externally_to_rlds
robotics:imperial_wrist_dataset
robotics:imperialcollege_sawyer_wrist_cam
robotics:io_ai_tech
robotics:jaco_play
robotics:kaist_nonprehensile_converted_externally_to_rlds
robotics:kuka
robotics:language_table
robotics:language_table_blocktoabsolute_oracle_sim
robotics:language_table_blocktoblock_4block_sim
robotics:language_table_blocktoblock_oracle_sim
robotics:language_table_blocktoblock_sim
robotics:language_table_blocktoblockrelative_oracle_sim
robotics:language_table_blocktorelative_oracle_sim
robotics:language_table_checkpoints
robotics:language_table_separate_oracle_sim
robotics:language_table_sim
robotics:maniskill_dataset_converted_externally_to_rlds
robotics:mimic_play
robotics:mt_opt_rlds
robotics:mt_opt_sd
robotics:mutex_dataset
robotics:nyu_door_opening_surprising_effectiveness
robotics:nyu_franka_play_dataset_converted_externally_to_rlds
robotics:nyu_rot_dataset_converted_externally_to_rlds
robotics:open_x_embodiment_and_rt_x_oss
robotics:plex_robosuite
robotics:qut_dexterous_manpulation
robotics:robo_net
robotics:robo_set
robotics:robot_vqa
robotics:roboturk
robotics:spoc
robotics:stanford_hydra_dataset_converted_externally_to_rlds
robotics:stanford_kuka_multimodal_dataset_converted_externally_to_rlds
robotics:stanford_mask_vit_converted_externally_to_rlds
robotics:stanford_robocook_converted_externally_to_rlds
```

```
robotics:taco_play
robotics:tidybot
robotics:tokyo_u_lsmo_converted_externally_to_rlds
robotics:toto
robotics:ucsd_kitchen_dataset_converted_externally_to_rlds
robotics:ucsd_pick_and_place_dataset_converted_externally_to_rlds
robotics:uiuc_d3field
robotics:usc_cloth_sim_converted_externally_to_rlds
robotics:utaustin_mutex
robotics:utokyo_pr2_opening_fridge_converted_externally_to_rlds
robotics:utokyo_pr2_tabletop_manipulation_converted_externally_to_rlds
robotics:utokyo_saytap_converted_externally_to_rlds
robotics:utokyo_xarm_bimanual_converted_externally_to_rlds
robotics:utokyo_xarm_pick_and_place_converted_externally_to_rlds
robotics:vima_converted_externally_to_rlds
robotics:viola
```

O TFDS também possui alguns métodos para carregar e visualizar conjuntos de dados. Uma descrição desses métodos e como usá-los pode ser vista em https://www.tensorflow.org/datasets/overview

Método `load()`

Para carregar um conjunto de dados usa-se o método `load()`.

No exemplo abaixo é carrega a biblioteca `tf_flowers`.

```
train_data, test_data = tfds.load('TFFlowers', shuffle_files=True,
split=['train[:80%]', 'train[-20%:]'], as_supervised=True,
with_info=False)
```

```
Downloading and preparing dataset 218.21 MiB (download: 218.21 MiB,
generated: 221.83 MiB, total: 440.05 MiB) to
/root/tensorflow_datasets/tf_flowers/3.0.1...
```

```
{"model_id":"b321799ad5e94253bfb49aafd63ca566","version_major":2,"vers
ion_minor":0}
```

```
Dataset tf_flowers downloaded and prepared to
/root/tensorflow_datasets/tf_flowers/3.0.1. Subsequent calls will
reuse this data.
```

Os argumentos do método `load()` mais comuns são:

- `split`: define que parte dos dados se quer carregar, por exemplo, 'train', 'test', 'all', ['train', 'test'], 'train[80%:]' etc. Existem muitas variações de como usar esse argumento, para mais detalhes consultar documentação.

- Como esse dataset foi criado originalmente somente com o conjunto de dados de treinamento, dividimos ele em dados de treinamento com 80% dos exemplos e dados de teste com 20% dos exemplos.
- `shuffle_files`: controla se quer embaralhar ou não os dados.
- `with_info=True`: retorna os dados e os metadados do conjunto com as informações básicas.
- `as_supervised=True`: carrega os dados na forma de uma tupple de dados de entrada e saídas desejdas.

Esse conjunto de dados não tem a variável `info`.

Se quiser confirmar se os dados foram carregados pode-se verificar o número de exemplos da seguinte forma:

```
print('Número exemplos de treinamento =', len(list(train_data)))
print('Número exemplos de teste =', len(list(test_data)))
```

```
Número exemplos de treinamento = 2936
Número exemplos de teste = 734
```

Esses comandos acima retornariam "2" se os dados tivessem sido carregados com `as_supervised=False`.

Usando `as_supervised=False` os dados são retornados na forma de um dicionário com duas chaves: (1) os dados de entrada e(2) as saídas. Nesse caso deve-se separar as entradas das saídas antes de serem usadas.

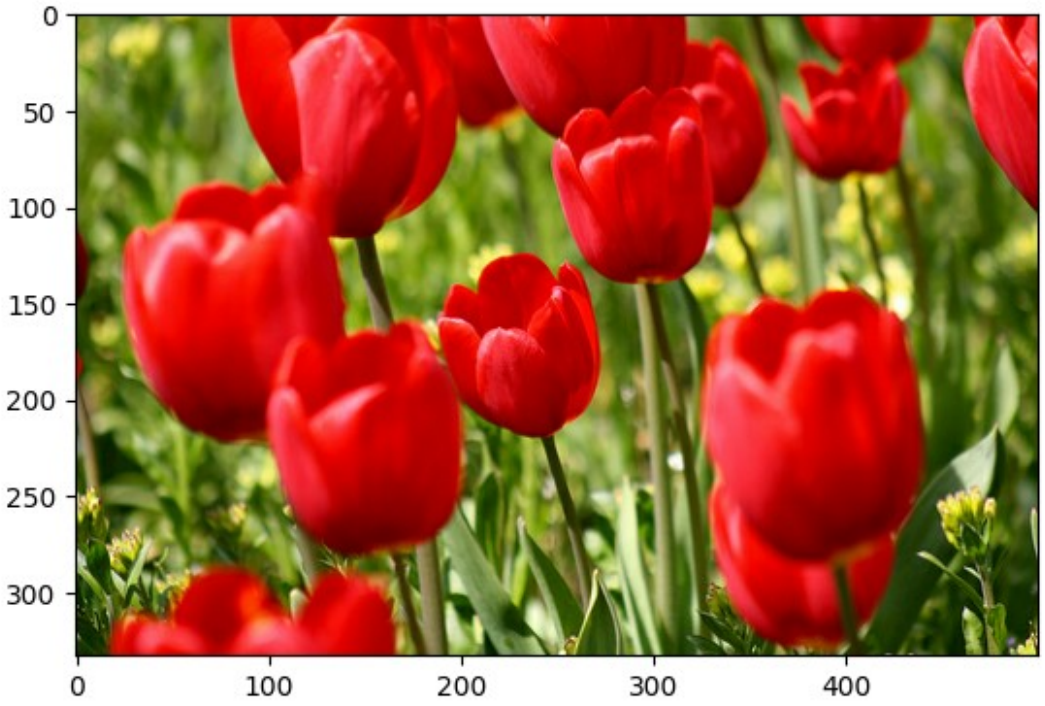
Método `take()`

O método `take()` serve para iteragir sobre um conjunto de dados pegando exemplos.

Por exemplo vamos visualizar as primeiras 5 imagens do conjunto de dados de treinamento (`train_data`).

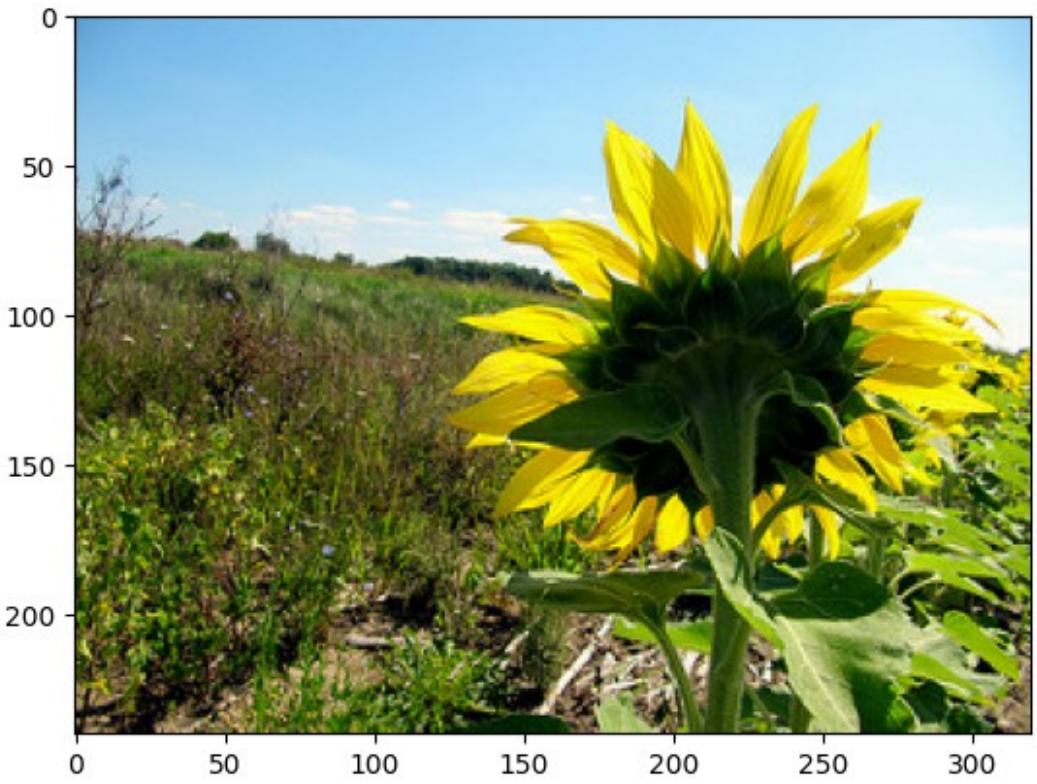
```
# Importa biblioteca MatplotLib para mostrar figura
import matplotlib.pyplot as plt

# Itera no conjutno de dados pegando exemplos
for data in train_data.take(3):
    image, label = data

    print("Classe: {}".format(label))
    plt.imshow(image)
    plt.show()
```

```
Classe: 2
```




```
Classe: 3
```


```
Classe: 3
```



- A forma como os dados de entrada e de saída são extraídos depende de como foram importados. No caso foram importados como uma tupple.
- Se os dados tivessem sido importados como dicionário teríamos que proceder da seguinte forma para extrair a imagem e a classe:
  - `image = data["image"].numpy()`
  - `label = data["label"].numpy()`
- As imagens são carregas como sendo tensores 3D, mesmo se forem em tons de cinza. Nesse caso antes de mostrar a imagem temos que eliminar o 3o eixo fazendo:
  - `image = image.numpy().squeeze()`

2.4 Keras

O Keras possui uma coleção de conjunto de dados pequena, mas é interssante porque já está pronta para ser usado após ser carregada.

Os conjuntos de dados disponíveis pode ser vistos em https://www.tensorflow.org/api_docs/python/tf/keras/datasets.

Para carregar um conjunto dados do Keras é muito simples → basta seguir as recomendações no link do conjutno de dados..

Por exemplo, para carregar o conjunto de dados de regressão de preços de habitação em Boston (https://www.tensorflow.org/api_docs/python/tf/keras/datasets/boston_housing), pode-se fazer o seguinte:

```
from tensorflow.keras.datasets import boston_housing

(x_train, y_train), (x_test, y_test) = boston_housing.load_data()

print('Dimensão do dados de entrada =', x_train.shape, x_test.shape)
print('Dimensão do dados de saída =', y_train.shape, y_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz
57026/57026 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Dimensão do dados de entrada = (404, 13) (102, 13)
Dimensão do dados de saída = (404,) (102,)
```

Se quiser importar os de uma forma mais sofisticada controlando a divisão dos dados nos conjuntos de dados de treinamento e teste, pode-se usar o comando recomendado no link.

```
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.boston_housing.load_data(
                                        path='boston_housing.npz',
test_split=0.1, seed=113)

print('Dimensão do dados de entrada =', x_train.shape, x_test.shape)
print('Dimensão do dados de saída =', y_train.shape, y_test.shape)
```

```
Dimensão do dados de entrada = (455, 13) (51, 13)
Dimensão do dados de saída = (455,) (51,)
```

- `test_split`: fração dos dados para reservar como conjunto de teste.
- `seed`: semente aleatória para embaralhar os dados antes de calcular a divisão de teste.

3. Bilotecas de modelos pré-treinados

É muito difícil sintonizar uma RNA "deep learning", ou seja, é difícil ajustar os hiperparâmetros de uma RNA para obter o desempenho desejado.

Uma abordagem comum e altamente eficiente na área de RNAs é usar uma rede previamente treinada para iniciar o desenvolvimento de uma nova aplicação → isso é usado principalmente se o banco de dados disponível para o problema for pequeno.

O uso de redes pré-treinadas para desenvolver uma nova RNA para realizar uma nova tarefa é chamado de transferência de aprendizado.

A técnica de transferência de aprendizado é muito potente e deve ser sempre avaliada como uma opção ao se desenvolver uma RNA para resolver um novo problema.

A transferência de aprendizado pode ser usada em 3 tipos de problemas:

1. Processamento de textos;
2. Processamento de imagens;
3. Processamento de vídeos.

Na internet existem inúmeras RNAs pré-treinadas disponíveis para resolver diferentes tipos de problemas. Essas RNAs podem ser usadas da forma como foram desenvolvidas e também para serem usadas como base para resolver novos problemas.

O TensorFlow e o Keras disponibilizam diversas RNAs que podem ser facilmente importadas para o nosso ambiente de programação e serem utilizadas da forma que desejamos.

4. Biblioteca de RNAS do Keras

O Keras fornece muitas RNAs já treinadas que podem ser importadas facilmente usando o módulo `tf. keras.applications`.

Uma observação importante é que todas as RNAs pré-treinadas disponíveis no Keras são redes convolucionais desenvolvidas para processar imagens e todas elas foram treinadas para classificação de objetos usando a base de dados ImageNet (http://www.image-net.org/).

As RNAs disponíveis atualmente no Keras e as suas características são as seguintes:

Modelo	Tamanho	Exatidão	Parâmetros	Profundidade
Xception	88 MB	0.945	22,910,480	126
VGG16	528 MB	0.901	138,357,544	23
VGG19	549 MB	0.900	143,667,240	26
ResNet50	98 MB	0.921	25,636,712	-
ResNet101	171 MB	0.928	44,707,176	-
ResNet152	232 MB	0.931	60,419,944	-
ResNet50V2	98 MB	0.930	25,613,800	-
ResNet101V2	171 MB	0.938	44,675,560	-
ResNet152V2	232 MB	0.942	60,380,648	-
ResNeXt50	96 MB	0.938	25,097,128	-
ResNeXt101	170 MB	0.943	44,315,560	-
InceptionV3	92 MB	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.953	55,873,736	572
MobileNet	16 MB	0.895	4,253,864	88
MobileNetV2	14 MB	0.901	3,538,984	88
DenseNet121	33 MB	0.923	8,062,504	121
DenseNet169	57 MB	0.932	14,307,880	169
DenseNet201	80 MB	0.936	20,242,984	201
NASNetMobile	23 MB	0.919	5,326,716	-
NASNetLarge	343 MB	0.960	88,949,818	-

("RNAs Keras.png") A exatidão se refere ao desempenho da RNA no banco de dados ImageNet.

4.1 Carregar modelos do Keras

Para carregar uma RNA pré-treinada do Keras é muito simples → basta seguir as recomendações no link do módulo `tf.keras.apllications` (https://www.tensorflow.org/api_docs/python/tf/keras/applications).

Por exemplo, para carregar a rede VGG16, que é uma das redes mais utilizadas para transferência de aprendizado na área de redes convolucionais, faz-se o seguinte:

```
import tensorflow as tf

vgg16 = tf.keras.applications.VGG16(include_top=True,
weights='imagenet', input_shape=None)
vgg16.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 12s 0us/step

Model: "vgg16"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)             │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block1_conv1 (Conv2D)                │ (None, 224, 224, 64)        │
1,792 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block1_conv2 (Conv2D)                │ (None, 224, 224, 64)        │
36,928 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block1_pool (MaxPooling2D)           │ (None, 112, 112, 64)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block2_conv1 (Conv2D)                │ (None, 112, 112, 128)       │
73,856 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block2_conv2 (Conv2D)                │ (None, 112, 112, 128)       │
147,584 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block2_pool (MaxPooling2D)           │ (None, 56, 56, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block3_conv1 (Conv2D)                │ (None, 56, 56, 256)         │
295,168 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block3_conv2 (Conv2D)                │ (None, 56, 56, 256)         │
590,080 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ block3_conv3 (Conv2D)                │ (None, 56, 56, 256)         │
590,080 │
```

```
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block3_pool (MaxPooling2D)          │ (None, 28, 28, 256)        │
0 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block4_conv1 (Conv2D)               │ (None, 28, 28, 512)        │
1,180,160 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block4_conv2 (Conv2D)               │ (None, 28, 28, 512)        │
2,359,808 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block4_conv3 (Conv2D)               │ (None, 28, 28, 512)        │
2,359,808 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block4_pool (MaxPooling2D)          │ (None, 14, 14, 512)        │
0 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block5_conv1 (Conv2D)               │ (None, 14, 14, 512)        │
2,359,808 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block5_conv2 (Conv2D)               │ (None, 14, 14, 512)        │
2,359,808 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block5_conv3 (Conv2D)               │ (None, 14, 14, 512)        │
2,359,808 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ block5_pool (MaxPooling2D)          │ (None, 7, 7, 512)          │
0 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ flatten (Flatten)                   │ (None, 25088)              │
0 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ fc1 (Dense)                         │ (None, 4096)               │
102,764,544 │
├─────────────────────────────────────┼────────────────────────────┼
───────────────┤
│ fc2 (Dense)                         │ (None, 4096)               │
16,781,312 │
├─────────────────────────────────────┼────────────────────────────┼
```

```
──────────────────┤
│ predictions (Dense)                  │ (None, 1000)                │
4,097,000 │
└──────────────────────────────────────┴─────────────────────────────┴
──────────────────┘

 Total params: 138,357,544 (527.79 MB)

 Trainable params: 138,357,544 (527.79 MB)

 Non-trainable params: 0 (0.00 B)
```

O `tf.keras.applications` possui diversos argumentos. Os mais importantes e mais usados são:

- `weights`: especifica os parâmetros desejados para inicializar a RNA;
- `include_top`: refere a incluir, ou não, as camadas densas que ficam no final da RNA (essas camadas densas correspondem ao classificador das 1.000 classes da ImageNet);
- `input_shape`: é a dimensão das imagens que se quer usar; esse argumento é opcional e só pode ser incluído com valor diferente de `None` se `include_top=False`.

Importante:

Se for desejado utilizar uma rede pré-treinada para realizar transferência de aprendizado deve-se importar a RNA convouliconal sem as camadas densas finais, ou seja, deve-se usar `include_top=False`.

4.2 Utilizar modelo do Keras

Vamos verificar se importamos corretamente a VGG16 processando uma imagem

Para usar a VGG16 temos que fornecer uma imagem colorida com as seguintes características:

- Dimensão (1, 224, 224, 3), assim, além de ser necessário redimensionar a imagem, deve-se acrescentar o primeiro eixo (eixo dos exemplos) com dimensão 1
- A imagem deve ser pré-processada para os valores dos seus pixels estarem de acordo como o que a VGG16 espera receber.

O código da célula abaixo carrega uma imagem usando a função `imread()` da biblioteca Matplotlib e apresenta a imagem.

```
import matplotlib.pyplot as plt

# Carrega imagem
img = plt.imread('panda.jpg')

# Mostra image
```

```
plt.imshow(img)
plt.show()

#from keras.preprocessing.image import load_img
#load an image from file
#image = load_img('cat.jpg', target_size=(224, 224))

# Mostra image
#plt.imshow(image)
#plt.show
```


```
# Mostra dimensão original da imagem
print('Dimensão da imagem original =', img.shape)

# Redimensiona imagem
img = tf.image.resize(img,[224, 224])

# Inclui eixo com dimensão 1
img = tf.expand_dims(img, axis=0)

# Apresenta dimensão da imagem pré-processada e valores de alguns
pixels
print('Dimensão da imagem redimensionada =', img.shape)
print(img[0,1,1,:5])
```

```
Dimensão da imagem original = (360, 480, 3)
Dimensão da imagem redimensionada = (1, 224, 224, 3)
tf.Tensor([125.67855 166.89284 111.23723], shape=(3,), dtype=float32)
```

Para normalizar os valores dos pixles da imagem para o que a VGG16 espera é melhor utilizar o método `tf.keras.applications.vgg16.preprocess_input(img)` fornecido na descrição da VGG16 (https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16).

```
# Normalização dos pixels da imagem
imgp = tf.keras.applications.vgg16.preprocess_input(img)

# Apresenta os valores de alguns pixels
print(imgp[0,1,1,:5])
```

```
tf.Tensor([ 7.2982254 50.11384    1.9985504], shape=(3,),
dtype=float32)
```

No código a seguir a imagem é processada pela VGG16 e o resultado consiste de um vetor de probabilidades da imagem mostrar cada uma das classes da Image-net.

```
# Processa imagem com a VGG16
prob = vgg16.predict(imgp)

# Mostra dimensão do vetor de probabilidades
print('Dimensão do vetor de probabilidades =', prob.shape)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Dimensão do vetor de probabilidades = (1, 1000)
```

A VGG16 classifica objetos em imagens dentro de um universo de 1000 classes. Assim, a sua saída é um vetor de 1000 elementos, onde cada elemento representa a probabilidade da imagem mostrar um determinado objeto.

Para sabermos qual objeto a VGG16 detectou na imagem temos que pegar o índice do elemento do vetor de probabiliaddes `prob` de maior valor. Para fazer isso de forma fácil e obtermos o nome da classe do objeto indetificado, o Keras fornece o método `tf.keras.applications.vgg16.decode_predictions()`, que já possui a lista dos nomes das 1000 classes.

```
# Determina as 5 classes com maior probabilidade
classes =tf.keras.applications.vgg16.decode_predictions(prob, top=5)
classes
```

```
Downloading data from
https://storage.googleapis.com/download.tensorflow.org/data/imagenet_c
lass_index.json
35363/35363 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
[[('n02510455', 'giant_panda', 0.9998727),
  ('n02133161', 'American_black_bear', 4.561956e-05),
  ('n02132136', 'brown_bear', 2.4375793e-05),
  ('n02134418', 'sloth_bear', 2.0417941e-05),
  ('n02488702', 'colobus', 1.1019664e-05)]]
```

Observa-se que a classe 'giant_panda' é a de maior probabilidade. Para selecionar somente o resultado de maior probabilidade pode-se fazer:

```
# Seleciona primeiro resultado
label = classes[0][0]

# Apresenta resultado da classificação
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
giant_panda (99.99%)
```

5. Biblioteca de módulos do TensorFlow Hub

O TensorFlow Hub é uma biblioteca criada para a publicação e compartilhamento de partes reutilizáveis de modelos de "machine learning" TensorFlow Hub.

Existem centenas de modelos e partes de modelos disponíveis no TensorFlow Hub para resolver diversos tipos de problema de processamento de imagem, video, texto e áudio.

A lista de modelos disponíveis pode ser pesquisado no site do TensorFlow Hub.

Os modelos estão em diversos formatos:

- Versão 1 do TF;
- Versão 2 do TF;
- Formato java-script (Tf.js) para serem utilizados páginas da web;
- Formato TF-Lite, para serem utilizados em dispositivos móveis;
- Formato Coral, placa do Google para embarcar modelos de "machine-learning".

Tutoriais do Tensorflow → https://www.tensorflow.org/hub/tutorials

Importar TensorFlow Hub

Para usarmos o TensorFlow Hub devemos importá-lo para o nosso ambiente de programação da seguinte forma:

```
#!pip install tensorflow_hub==0.16.1

import tensorflow_hub as hub
```

5.1 Carregar um módulo do TensorFlow Hub

Para carregar um módulo do TensorFlow Hub é necessário obter a URL (link) de onde está o módulo. Para obter essa URL temos que pesquisar o catálogo de módulos no site TensorFlow Hub.

Por exemplo, queremos o modelo completo da RNA **MobileNet**. Se formos na página da Mobilenet do TensorFlow Hub MobileNet's webpage , veremos que a URL para acessar o modelo é a seguinte:

```
'https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4'
```

A RNA MobiliNet é uma rede convolucional utilizada para processar imagens. Existem várias versões dessa rede, que são utilizadas para tarefas de classificação e também para detecção e localização de objetos.

A versão da MobiliNet que iremos utilizar serve para classificação de objetos em imagens e foi treinada com as imagens da Image-Net.

Para carregar a MobiliNet usa-se o método `hub.load()`, conforme mostrado no código da célula abaixo.

```
MODULE_HANDLE =
'https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4'
modulo = hub.load(MODULE_HANDLE)
```

5.2 Utilizar modelo do TensorFlow Hub

Após carregar o modelo, podemos usá-lo para realizar cálculos. Para isso temos temos que realizar os seguintes passos:

1. Obter uma ou mais imagens;
2. Pré-processar as imagens;
3. Obter os nomes das classes da ImageNet;
4. Processar as imagens com a RNA;
5. Determinar o objeto identificado na imagem.

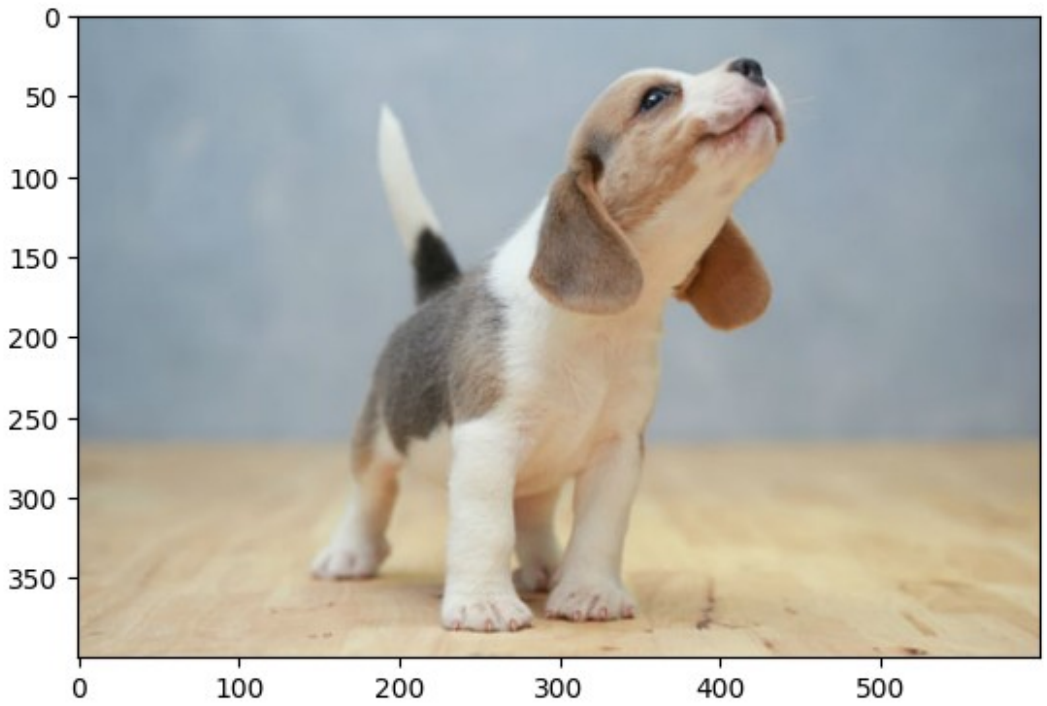
Carregar e processar imagem

Na célula abaixo, carregamos uma imagem de um arquivo e a processamos para colocar na dimensão esperada pela MobiliNet e para transformar os valores dos pixels entre 0 e 1.

```
# Carrega imagem
img = plt.imread('dog.jpg')
print('Dimensão da imagem original =', img.shape)

# Mostra image
plt.imshow(img)
plt.show
```

```
# Redimensiona imagem
img = tf.image.resize(img,[224, 224])/255.
print('Dimensão da imagem =', img.shape)
```

```
Dimensão da imagem original = (400, 600, 3)
Dimensão da imagem = (224, 224, 3)
```



Importar lista de classes da ImageNet

Numa versão mais atual das classes de objetos identificcados pela Image-Net foi adicionada uma nova classe que representa o "fundo".

Observa-se que essa classe não existe nas classes utilizadas pela rede VGG16 que importamos do Keras. Assim, não é possível utilizar o método `tf.keras.applications.vgg16.decode_predictions()` que foi utilizado quando usamos a rede VGG16 importada do Keras.

As classes de objetos estão no arquivo 'imagenet1001_labels.txt'. Assim, vamos ler esse arquivo e salvar as classes na lista `labels`.

```
# Abre arquivo com as classes e coloca na lista "labels"
with open('imagenet1001_labels.txt', 'r') as f:
    labels = [l.strip() for l in f.readlines()]

# Verifica número de classes
num_classes = len(labels)
print('Existe um total de {0} classes.'.format(num_classes))
```

```
Existe um total de 1001 classes.
```

Vamos verificar quais são as primeiras 5 classes da lista.

```
for label in labels[0:5]:
    print(label)
```

```
background
tench
goldfish
great white shark
tiger shark
```

Usando a MobileNet

Nessa versão da MobiliNet do TensorFlow Hub, a última camada não tem nenhuma função de ativação. Portanto, temos que aplicar a função de ativação `softmax`, utilizada em problemas de classifcação multiclasse, para calcular as probabilidades das imagens mostrarem as diversas classes de objetos.

A saída da rede MobiliNet é um vetor de dimensão `(m, 1001)`, onde a primeira dimensão é o número de imagens passadas para a rede. Nesse exemplo usamos somente uma imagem.

O código da célula abaixo passa a imagem para ser processada pela MobiliNet, aplica a função de ativação `softmax` e seleciona as 5 classes de maior probabilidade.

```
# Processamento da imagem pela RNA MobiliNet
yprev = modulo([img])

# Aplicação da função de ativação softmax
prob = tf.nn.softmax(yprev)[0]

# Seleciona as classes com as 5 maiores probabilidades
top_prob, top_indices = tf.math.top_k(prob, k=5)

# Tranforma objetos top_prrb e top_indices em valores numéricos
top_prob_values = top_prob.numpy()
top_indices_values = top_indices.numpy()

# Apresenta as classes selecionadas
for value, i in zip(top_prob_values, top_indices_values):
    print('%s (%.2f%%)' % (labels[i], value*100))
```

```
beagle (89.72%)
basset (2.19%)
English foxhound (1.47%)
Walker hound (1.07%)
bluetick (0.13%)
```

- A função `tf.nn.softmax()` do tensorFlow aplica a função de ativação `softmax`.

- A função `tf.math.top_k()` do TensorFlow acha os k maiores valores e seu índices entre os elementos do último eixo de um tensor.

5.3 Usando um módulo do TensorFlow Hub com o Keras

É possível integrar módulos do TensorFlow Hub em modelos do Keras para criar um novo modelo ou mesmo para retreinar um modelo existente.

Para integrar um módulo do TF-Hub em uma RNA é utilizada a função `hub.KerasLayer`. Podemos adicionar um módulo do TF-Hub em uma RNA sequencial do Keras e adicionar quaisquer outras camadas que desejarmos. Uma vez criado o modelo do Keras, todos os métodos do Keras podem ser utilizados normalmente.

Como exemplo, vamos criar uma rede do Keras usando a RNA MobiliNet importada do TensorFlow Hub. Nesse caso adicionamos a função de ativação na rede. O código da célula abaixo mostra essa operação.

```
input_shape = (224, 224, 3)

# Importa modulo do tensorflow hub
feature_extractor = hub.KerasLayer(MODULE_HANDLE,
input_shape=input_shape)

# Envolver o modelo em uma camada Lambda
probs = tf.keras.layers.Lambda(lambda x: feature_extractor(x))

# Cria RNA
rna = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=input_shape),
    probs,
    tf.keras.layers.Activation('softmax')
])

rna.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
input_layer.py:26: UserWarning: Argument `input_shape` is deprecated.
Use `shape` instead.
  warnings.warn(

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape               ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━┩
│ lambda (Lambda)                      │ (None, 1001)               │
0 │
├──────────────────────────────────────┼────────────────────────────┼
```

```
──────────────────┤
| activation (Activation)               | (None, 1001)                |
0 |
└──────────────────────────────────────┴─────────────────────────────┘
──────────────────┘

 Total params: 0 (0.00 B)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 0 (0.00 B)
```

Cálculo de previsões

Para calcular previsões de um modelo Keras temos que adicionar uma dimensão à nossa imagem para considerar o eixo dos exemplos. Lembre que uma RNA do Keras espera recebe rcomo entrada um tensor de dimensão `(batch_size, image_size)`, onde `image_size` inclui as dimensões dos 3 eixos da imagem.

```
import numpy as np

# Adiciona eixo dos exemplos na iamgem
img_exp = np.expand_dims(img, axis=0)
print('Nova dimensão da imagem =', img_exp.shape)

Nova dimensão da imagem = (1, 224, 224, 3)
```

Da mesma forma que fizemos anteriormente no item 5.2, no código da célula abaixo a imagem é processada pela RNA e depois são selecionadas as 5 classes de maior probabilidade.

```
# Precessamento da imagem
yprev = rna.predict(img_exp)[0]

# Seleciona as 5 maiores probabilidades
top_prob, top_indices = tf.math.top_k(yprev, k=5)

# Transfroma objetos do TF em números
top_prob_values = top_prob.numpy()
top_indices_values = top_indices.numpy()

# Apresenta as classes de maior probabilidade
for value, i in zip(top_prob_values, top_indices_values):
    print('%s (%.2f%%)' % (labels[i], value*100))

1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
beagle (89.72%)
basset (2.19%)
English foxhound (1.47%)
Walker hound (1.07%)
bluetick (0.13%)
```

6. Uso de módulos de vetores de características com o Keras

Além de poder completar modelos do TF-Hub, como fizemos nas seções anteriores, a parte mais importante do TF-Hub são os módulos de **geradores de caraterísticas**, que podem ser usados para criar novas RNAs para realizar novas tarefas utilizando o processo de transferência de aprendizado.

Gerador de características são modelos completos pré-treinados com as camadas finais removidas. As camadas finais de uma RNA convolucional ou recorrente são especializadas na tarefa para a qual foi treinada, mas as camadas anteriores são extratores potentes de características.

Não é objetivo dessa aula apresentar o método de transferência de aprendizado, mas vamos ver como criar e treinar uma nova RNA para realizar uma nova tarefa, usando um módulo do TF-Hub de gerador de caraterísticas.

Observa-se que isso também pode ser realizado com as RNAs importadas do Keras, simplesmente importando-as com a opção `include_top=False`.

Para exemplificar como utilizar um gerador de características importado do TF-Hub e usá-lo para criar uma RNA para realizar uma nova tarefa usaremos o gerador de caraterísticas da MobiliNet.

A nova tarefa será identificar os sinais de mão utilizados no jogo "pedra-papel-tesoura". Assim, veremos também como usar um conjunto de dados importados do TF Data Services para treinar uma RNA, que no caso é o conjunto de imagens "rock_paper_scissors", carregado na Secão 2.2.

6.1 Processamento dos dados

A primeira tarefa para treinar uma RNA é processar os dados utlizados no treinamento. Na Seção 2.3 os dados foram carregados em dois conjuntos: dados de treinamento (train_data) e dados de validação/teste (val_data)

Os dados de entrada da MobiliNet são imagens de dimensão (224, 224, 3) e pixels com valores entre 0 e 1. Como as imagens do conjunto "rock-paper-scissor" tem dimensão (300, 300, 3) e estão no formato RGB, então temos que redimensionar as imagens e dividir os pixels por 255.

Ao importar os dados do TF Data Services, os dados são armazenados em objetos e para podermos usar esses dados de forma eficiente temos que usar os métodos fornecidos para essa classe.

Para redimensionar e normalizar as imagens usamos o método `map()`, que chama uma função criada para essa finalidade.

```
# Dimensão das imagens usadas pela MobiliNet
IMAGE_SIZE = (224, 224)

# Função usada para redimensionar e normalizar as imagens
def format_image(image, label):
```

```
    image = tf.image.resize(image, IMAGE_SIZE) / 255.0
    return  image, label

# Define tamanho do lote de dados de treinamento e validação
BATCH_SIZE = 128

# Cria lotes de dados usando o método map() para chamar a função
format_image()
train_batches = train_data.map(format_image).batch(BATCH_SIZE)
test_batches = test_data.map(format_image).batch(BATCH_SIZE)
```

- A funçao `tf.image.resize()` do TensorFlow serve para redimensionar uma imagem.

O código da célula acima prepara os dados para serem usados no treinamento da RNA

6.2 Criação da RNA

Como mencionado, a nova RNA vai ser criada usando o módulo de vetores de caraterísticas da **MobileNet** importado do TF-Hub. Se formos na página da Mobilenet do TensorFlow Hub MobileNet's webpage, veremos que a URL para acessar esse módulo é a seguinte:

```
https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4
```

O código da célula abaixo importa do TF-Hub o módulo de vetores de características da MobiliNet e incorpora esse módulo em uma RNA do tipo `sequencial` do Keras que possui além desse módulo uma camada densa para classificação de 3 classes.

```
# Número de classes da nova RNA
NUM_CLASSES = 5

# Dimensão das imagens aceitas pela MobiliNet
IMAGE_DIM = (224, 224, 3)

# Carrega vetores de características com a URL do módulo
MODULE_HANDLE
="https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
feature_extractor = hub.KerasLayer(MODULE_HANDLE,
input_shape=IMAGE_DIM)

# Envolver o modelo em uma camada Lambda
features = tf.keras.layers.Lambda(lambda x: feature_extractor(x))

# Cria modelo seguencial do Keras para problema de classificação com
10 classes
rna2 = tf.keras.Sequential([
       features,
       tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])
```

```
# Apresenta configuração da RNA
rna2.summary()
```

```
Model: "sequential_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ lambda_1 (Lambda)                    │ ?                           │               0 (unbuilt) │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ ?                           │               0 (unbuilt) │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
```

```
 Total params: 0 (0.00 B)
```

```
 Trainable params: 0 (0.00 B)
```

```
 Non-trainable params: 0 (0.00 B)
```

- Observa-se que nesse caso o módulo de vetores de caracteríticas da MobiliNet foi importado para uma rede com nome "feature_extractor", que posteriormente é inserido na RNA sequencial como se fosse uma camada.
- Como a rede "feature_extractor" é a primeira camada da RNA ao ser criada tem que definir a dimensão dos dados de entrada.

Com essa nova RNA podemos utilizar todos os métodos do tf.Keras, como se fosse uma rede criada totalmente com o Keras.

6.3 Compilação e treinamento da RNA

O treinamento da RNA deve ser realizado de forma que somente os parâmetros da camada densa, adicionada ao extrator de características da MobiliNet, sejam alterados durante o treinamento. Isso é necessário para não destruirmos a parte da RNA que corresponde à MobiliNet, que já foi previamente treinada com um conjunto de centenas de milhares de imagens. Assim, temos que "congelar" os parâmetros da MobiliNet.

O código da célula abaixo mostra com "congelar" os parâmetros da MobiliNet durante o treinamento e compilar a nova RNA usando o método Adam de otimização.

As classes de flores estão descritas por números inteiros, codificadas da seguinte forma:

- "dente de leão": 0
- "margarida": 1

- "tulipa": 2
- "girassol": 3
- "rosa": 4

Para usar as classes descritas por números inteiros, sem a necessidade de transformar em vetores "one-hot", temos que usar a função de custo `sparse_categorical_crossentropy`, que realiza essa transformação internamente de forma automática.

```
# Congela parâmetros da MobiliNet
feature_extractor.trainable = False

# Define método de otimização
optimizer = 'adam'

# Compila RNA
rna2.compile(optimizer=optimizer,
             loss='sparse_categorical_crossentropy',
             metrics=['accuracy'])
```

O código da célula abaixo realiza o treinameto da RNA usando somente 10 épocas de treinamento.

Observa-se que o treinamento é rápido em razão de estarmos treinando somente a última camada da RNA.

O treinamento da RNA é realizado com o método `fit` e os dados de treinamento e validação são fornecidos por meio do objeto `train_batches`. Esse objeto foi preparado na etapa de processamento (item 6.1), onde se definiu o tamanho do lote (128 exemplos) e o pré-processamento das imagens (redimensionamento e normalização).

Observa-se que o conjunto de validação será fornecido pelo atributo `validation_split` do método `fit`.

Observa-se que o processamento das imagens é realizado em tempo real durante o treinamento para cada lote, na medida em que são utilizados.

```
# Define número de épocas de treinamento
EPOCHS = 10

# Realiz ao treinamento usando os dados de treinamento e validação
history = rna2.fit(train_batches,
                   epochs=EPOCHS,
                   validation_data=test_batches)
```

```
Epoch 1/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 62s 2s/step - accuracy: 0.4538 - loss:
1.3433 - val_accuracy: 0.7943 - val_loss: 0.6226
Epoch 2/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 7s 287ms/step - accuracy: 0.8121 - loss:
0.5640 - val_accuracy: 0.8515 - val_loss: 0.4456
```

```
Epoch 3/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 10s 252ms/step - accuracy: 0.8570 - loss:
0.4202 - val_accuracy: 0.8569 - val_loss: 0.3856
Epoch 4/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 5s 230ms/step - accuracy: 0.8850 - loss:
0.3569 - val_accuracy: 0.8678 - val_loss: 0.3559
Epoch 5/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 6s 273ms/step - accuracy: 0.9008 - loss:
0.3167 - val_accuracy: 0.8719 - val_loss: 0.3361
Epoch 6/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 10s 283ms/step - accuracy: 0.9096 - loss:
0.2881 - val_accuracy: 0.8774 - val_loss: 0.3208
Epoch 7/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 5s 231ms/step - accuracy: 0.9212 - loss:
0.2640 - val_accuracy: 0.8842 - val_loss: 0.3082
Epoch 8/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 11s 251ms/step - accuracy: 0.9267 - loss:
0.2447 - val_accuracy: 0.8883 - val_loss: 0.2974
Epoch 9/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 5s 232ms/step - accuracy: 0.9322 - loss:
0.2268 - val_accuracy: 0.8965 - val_loss: 0.2888
Epoch 10/10
23/23 ━━━━━━━━━━━━━━━━━━━━ 7s 304ms/step - accuracy: 0.9385 - loss:
0.2122 - val_accuracy: 0.8992 - val_loss: 0.2816
```

6.4 Teste da RNA

Após o treinamento é necessário verificar o desempenho da RNA.

O código da célula abaixo calcula o resultado da função de custo e da exatidão para os exemplos validação usando o método `evaluate`.

```
# Avalia desempenho da RNA para os dados de validação
eval_results = rna2.evaluate(test_batches, verbose=0)

# Apresenta os resultados
for metric, value in zip(rna2.metrics_names, eval_results):
    print(metric + ': {:.4}'.format(value))
```

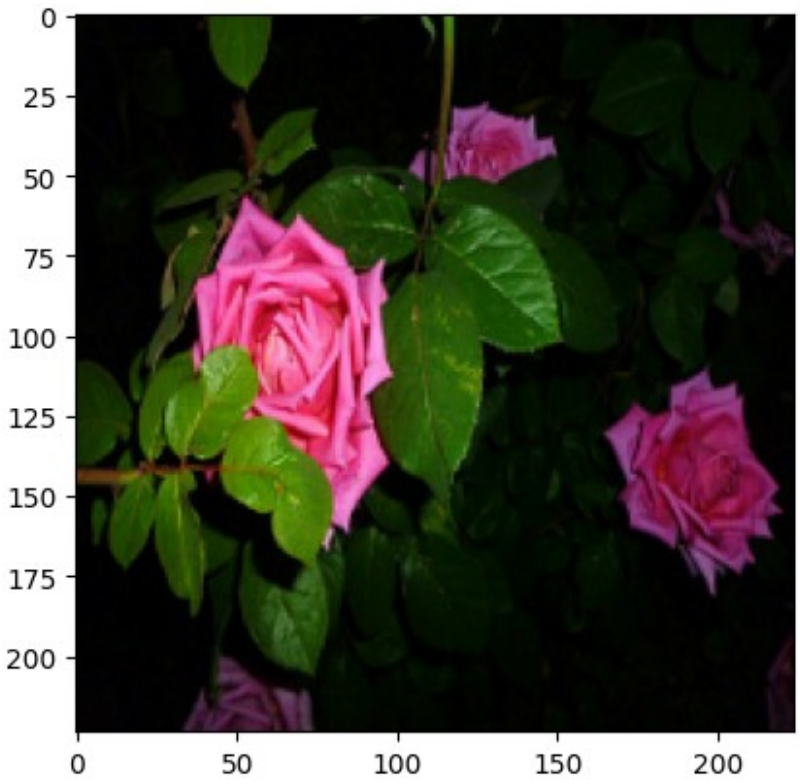
```
loss: 0.2816
compile_metrics: 0.8992
```

O código da célula abaixo calcula as classes previstas para os 5 primeiros exemplos do conjunto de validação usando o método `predict`.

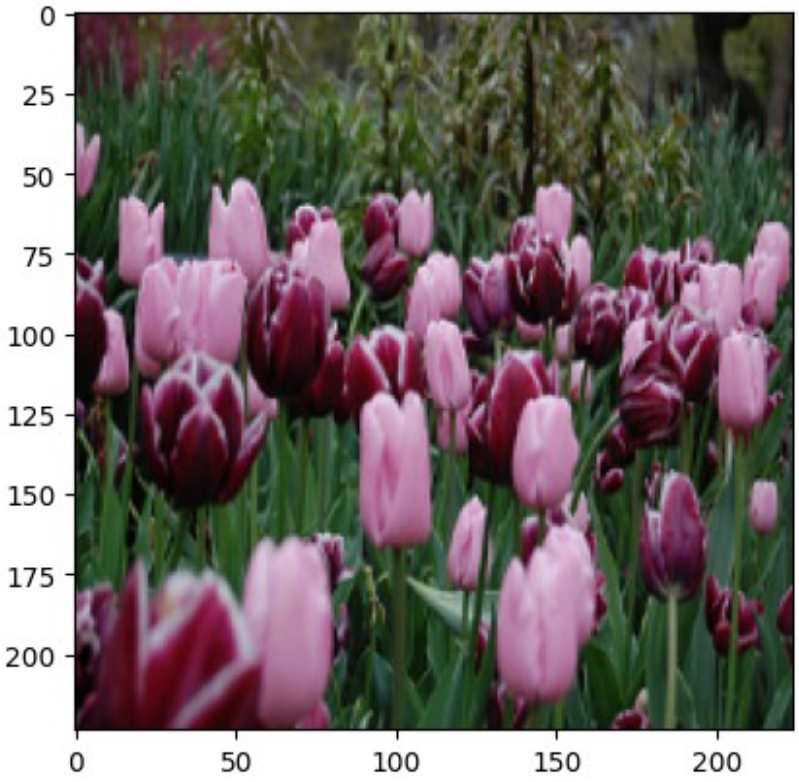
Para podermos fornecer as imagens para a RNA usando o método `predict` precisamos extrai-las do objeto `val_data`, processadas pela função `format_image`, que por sua vez é chamada pelo método `map()`. Além disso, temos que incluir o eixo dos exemplos na imagem de acordo com o esperado por uma RNA do Keras.

```python
for data in test_data.map(format_image).take(5):
    # Extrai imagem e classe
    image, label = data

    # Adiciona eixo dos exemplos
    image = np.expand_dims(image, axis=0)

    # Calcula probabilidades previstas pela RNA
    yprev = rna2.predict(image)

    # Determina classe prevista
    classe_prev = np.argmax(yprev)

    # Apresenta resultados e mostra imagem
    print('\nProbabilidades =', yprev)
    print("Classe real: {}".format(label))
    print("Classe prevista: {}".format(classe_prev))
    plt.imshow(image[0])
    plt.show()
```

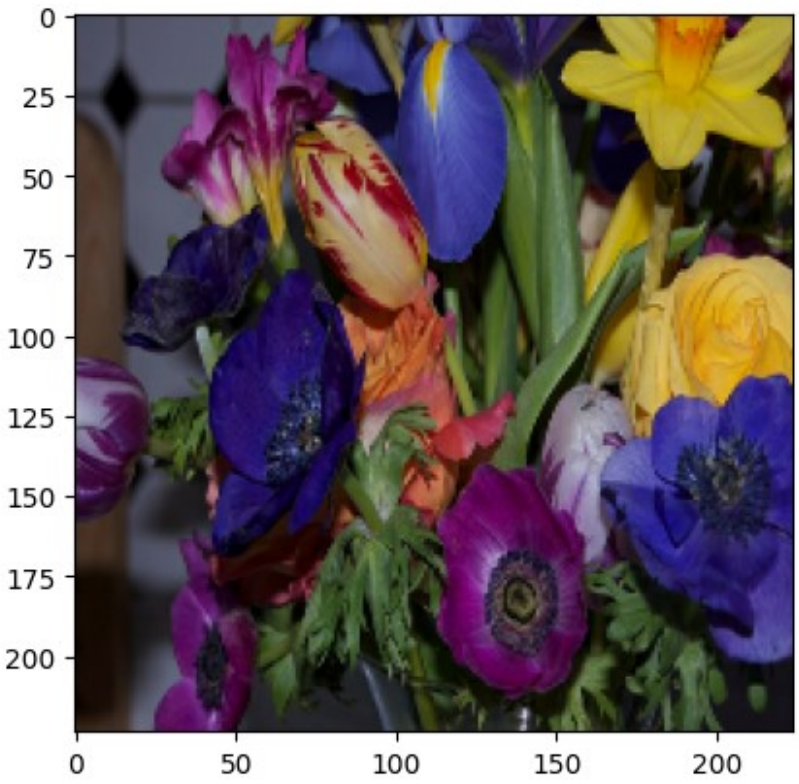
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step

Probabilidades = [[2.8521529e-07 3.9518121e-04 1.4702275e-02
5.4265535e-04 9.8435956e-01]]
Classe real: 4
Classe prevista: 4
```



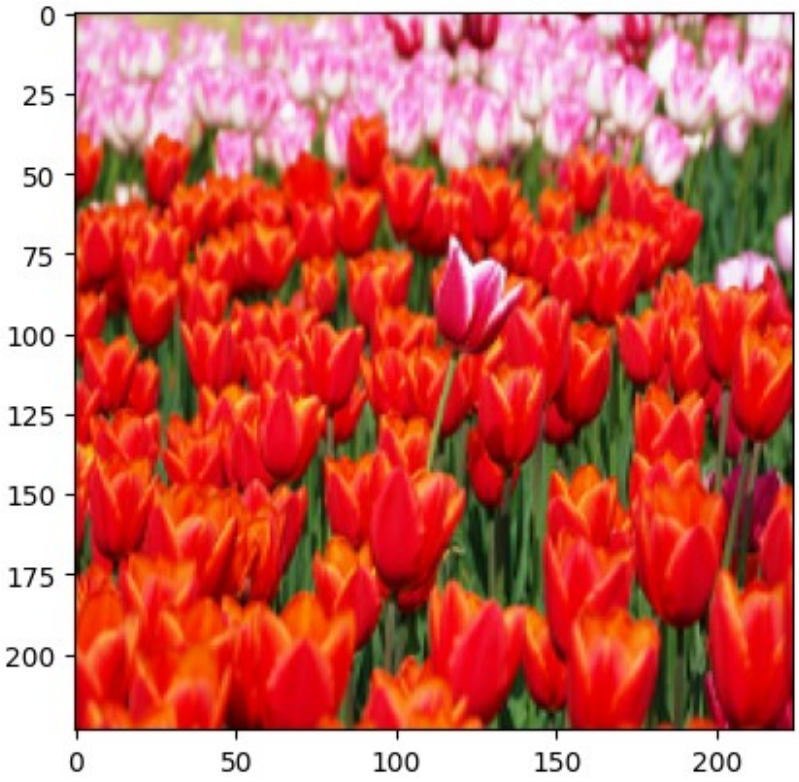

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step

Probabilidades = [[6.7714090e-04 2.8632011e-02 8.9004374e-01
1.2194407e-03 7.9427697e-02]]
Classe real: 2
Classe prevista: 2
```



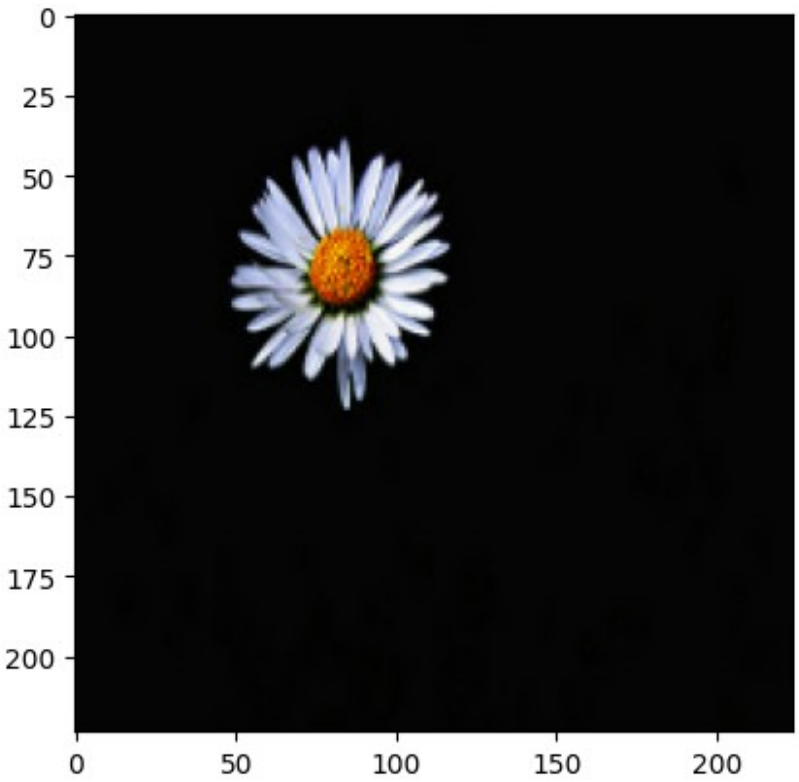

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step

Probabilidades = [[3.2394342e-05 1.3831382e-03 4.7101086e-01
3.0803159e-02 4.9677038e-01]]
Classe real: 2
Classe prevista: 4
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step

Probabilidades = [[4.2297502e-06 4.9554533e-04 9.9826932e-01
4.9526709e-05 1.1812887e-03]]
Classe real: 2
Classe prevista: 2
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step

Probabilidades = [[4.1323109e-03 9.9202418e-01 5.5376506e-05
3.5020732e-03 2.8602965e-04]]
Classe real: 1
Classe prevista: 1
```



7. Biblioteca de módulos do Keras Hub

O Keras Hub (Keras Hub) é uma biblioteca de modelos similar ao TensorFlow Hub .

Da mesma forma que no TensortFlow Hub, existem diversos modelos e partes de modelos disponíveis no para resolver diversos tipos de problema de processamento de imagem, video, texto e áudio. Contudo, o Keras Hub possui menos modelos do que o TensorFlow Hub.

A lista de modelos e camadas disponíveis pode ser pesquisado no manual do keras 3 no link https://keras.io/api/keras_hub/.

A forma de usar esses modelos é similar à forma de usar os modelos do TensorFlow Hub e mais informações podem ser obtidas em https://keras.io/keras_hub/#quickstart.

O Keras Hub precisa ser instalado para poder ser usado.

```
!pip install --upgrade keras-hub
```

```
Requirement already satisfied: keras-hub in
/usr/local/lib/python3.10/dist-packages (0.16.0.dev0)
Requirement already satisfied: absl-py in
/usr/local/lib/python3.10/dist-packages (from keras-hub) (1.4.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from keras-hub) (1.26.4)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from keras-hub) (24.1)
```

```
Requirement already satisfied: regex in
/usr/local/lib/python3.10/dist-packages (from keras-hub) (2024.9.11)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from keras-hub) (13.9.2)
Requirement already satisfied: kagglehub in
/usr/local/lib/python3.10/dist-packages (from keras-hub) (0.3.1)
Requirement already satisfied: tensorflow-text in
/usr/local/lib/python3.10/dist-packages (from keras-hub) (2.17.0)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from kagglehub->keras-hub)
(2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from kagglehub->keras-hub) (4.66.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras-hub) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras-hub)
(2.18.0)
Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras-hub)
(4.12.2)
Requirement already satisfied: tensorflow<2.18,>=2.17.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow-text->keras-
hub) (2.17.0)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0-
>rich->keras-hub) (0.1.2)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
```

```
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (3.4.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (71.0.4)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (2.5.0)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.37.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub-
>keras-hub) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub-
>keras-hub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub-
>keras-hub) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub-
>keras-hub) (2024.8.30)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.44.0)
Requirement already satisfied: namex in
/usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
>tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.0.8)
Requirement already satisfied: optree in
```

```
/usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
>tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (0.13.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow<2.18,>=2.17.0->tensorflow-text-
>keras-hub) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow<2.18,>=2.17.0->tensorflow-text->keras-hub) (3.0.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.18,>=2.17->tensorflow<2.18,>=2.17.0->tensorflow-text-
>keras-hub) (2.1.5)
```

Como exemplo vamos usar a RNA ResNet-50 para calssificar uma imagem.

```
# Importa keras e keras_hub
import keras
import keras_hub

# Carrega RNA ResNet-50 treinada com as imagens da IMageNet
classificador = keras_hub.models.ImageClassifier.from_preset(
    "resnet_50_imagenet",
    activation="softmax")

# Mostra modelo
classificador.summary()
```

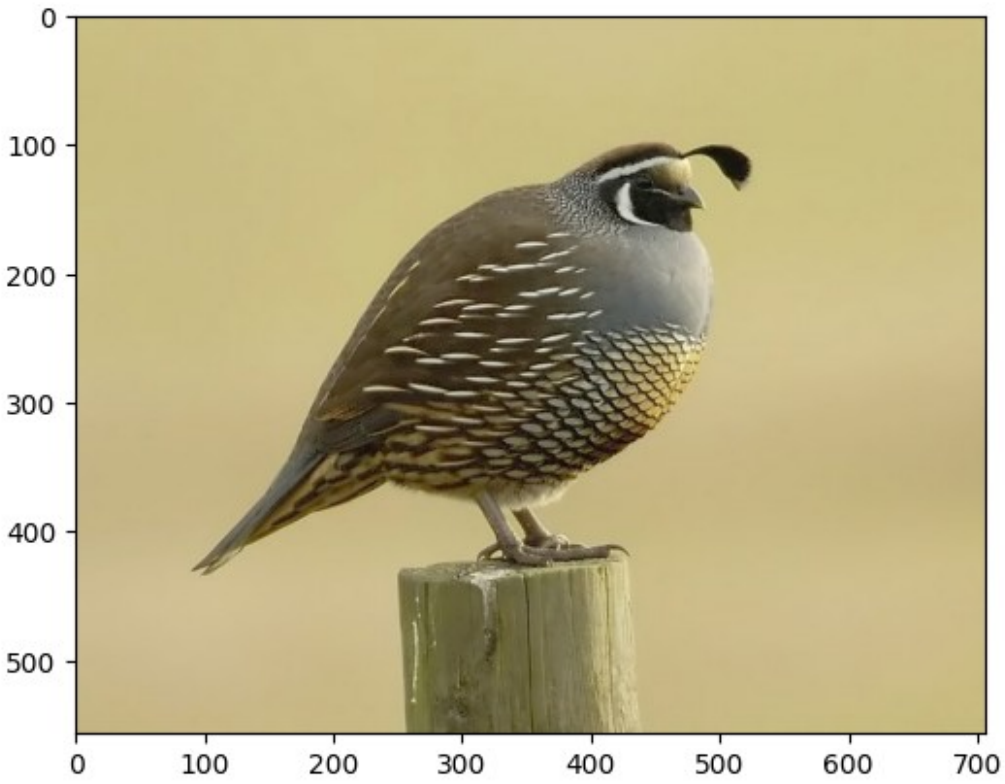
```
Preprocessor: "res_net_image_classifier_preprocessor"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                                                    ┃
Config ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│ res_net_image_converter (ResNetImageConverter)                  │
Image size: (224, 224) │
└────────────────────────────────────────────────────────────────┴─────
─────────────────────────────────────┘

Model: "res_net_image_classifier"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                                 ┃ Output Shape
            Param # ┃
```

```
┃━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫
│ input_layer (InputLayer)                         │ (None, None, None,
3)                 │                        0 │
├──────────────────────────────────────────────────┼────────────────────
───────────────┼─────────────────────────┤
│ res_net_backbone (ResNetBackbone)                │ (None, None, None,
2048)              │               23,561,152 │
├──────────────────────────────────────────────────┼────────────────────
───────────────┼─────────────────────────┤
│ global_average_pooling2d                         │ (None, 2048)
                    0 │
│ (GlobalAveragePooling2D)                         │
                      │
├──────────────────────────────────────────────────┼────────────────────
───────────────┼─────────────────────────┤
│ predictions (Dense)                              │ (None, 1000)
            2,049,000 │
└──────────────────────────────────────────────────┴────────────────────
───────────────┴─────────────────────────┘

 Total params: 25,610,152 (97.69 MB)

 Trainable params: 25,557,032 (97.49 MB)

 Non-trainable params: 53,120 (207.50 KB)
```

Vamos carregar uma imagem para testar a RNA.

```
# Predict a label for a single image.
image_url =
"https://upload.wikimedia.org/wikipedia/commons/a/aa/California_quail.
jpg"
image_path = keras.utils.get_file(origin=image_url)
image = keras.utils.load_img(image_path)

# Mostra imagem
plt.imshow(image)
plt.show()
```

```
Downloading data from
https://upload.wikimedia.org/wikipedia/commons/a/aa/California_quail.j
pg
67120/67120 ━━━━━━━━━━━━━━━━━━━━ 0s 1us/step
```



O modelo criado é usado com o método `predict` como qualquer modelo do Keras.

```
# Inclui eixo dos exemplos
batch = np.array([image])

# Executa modelo
preds = classificador.predict(batch)

# Apresenta as 5 classes de maior probabilidade
for classe, prob in keras_hub.utils.decode_imagenet_predictions(preds)
[0]:
    print('%s (%.2f%%)' % (classe, prob*100))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step
quail (99.97%)
prairie_chicken (0.01%)
partridge (0.00%)
bullet_train (0.00%)
ruffed_grouse (0.00%)
```

- A função `keras_hub.utils.decode_imagenet_predictions` codifica a saída prevista pela ResNet50 treinada com as imagens da ImageNet.

Outro exemplo, é retreinar o modelo BERT do Google de linguagemm pré-treinado para classificar revisões de filmes para verificar se a avaliação do filme é boa ou ruim.

Nesse caso o conjunto de dados ustilizado é o "IMDB_reviews", que é bastante conhecido.

```
# Carrega modelo BERT
classificador = keras_hub.models.BertClassifier.from_preset(
    "bert_base_en_uncased",
    activation="softmax",
    num_classes=2)

# Mostra modelo
classificador.summary()
```

```
Preprocessor: "bert_text_classifier_preprocessor"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                                                                                      ┃
Config ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
┃ bert_tokenizer (BertTokenizer)                                                                    ┃
Vocab size: 30,522 ┃
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
```

```
Model: "bert_text_classifier"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                  ┃ Output Shape                                                                                      ┃
Param # ┃ Connected to                                                                                      ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
┃ padding_mask (InputLayer)     ┃ (None, None)                                                                                      ┃
0 ┃ -                                                                                                 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
┃ segment_ids (InputLayer)      ┃ (None, None)                                                                                      ┃
0 ┃ -                                                                                                 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
┃ token_ids (InputLayer)        ┃ (None, None)                                                                                      ┃
0 ┃ -                                                                                                 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
┃ bert_backbone (BertBackbone)  ┃ [(None, 768), (None,                                                                              ┃
109,482,240 ┃ padding_mask[0][0],                                                                               ┃
┃                               ┃ None, 768)]                                                                                       ┃
┃ segment_ids[0][0],                                                                                ┃
┃                               ┃                                                                                                   ┃
┃ token_ids[0][0]                                                                                   ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
```

```
──────────┼──────────────────────────────────┼──────────────────────┼──────────
│ classifier_dropout (Dropout)  │ (None, 768)                  │
0 │ bert_backbone[0][0]           │
├──────────────────────────────────┼──────────────────────┼──────────
│ logits (Dense)                │ (None, 2)                    │
1,538 │ classifier_dropout[0][0]      │
└──────────────────────────────────┴──────────────────────┴──────────
──────────┴──────────────────────────────────┘

 Total params: 109,483,778 (417.65 MB)

 Trainable params: 109,483,778 (417.65 MB)

 Non-trainable params: 0 (0.00 B)
```

```
# Carrega conjunto de dados IMDB movie reviews.
imdb_train, imdb_test = tfds.load(
    "imdb_reviews",
    split=["train", "test"],
    as_supervised=True,
    batch_size=16)

# Mostra alguns exemplos
for text, label in imdb_train.take(1):
    for i in range(len(text)):
      print('Revisão:', format(i), '\n', text[i].numpy())
      print('Classificação:', label[i].numpy())
```

```
Revisão: 0
 b"This was an absolutely terrible movie. Don't be lured in by
Christopher Walken or Michael Ironside. Both are great actors, but
this must simply be their worst role in history. Even their great
acting could not redeem this movie's ridiculous storyline. This movie
is an early nineties US propaganda piece. The most pathetic scenes
were those when the Columbian rebels were making their cases for
revolutions. Maria Conchita Alonso appeared phony, and her pseudo-love
affair with Walken was nothing but a pathetic emotional plug in a
movie that was devoid of any real meaning. I am disappointed that
there are movies like this, ruining actor's like Christopher Walken's
good name. I could barely sit through it."
Classificação: 0
Revisão: 1
 b'I have been known to fall asleep during films, but this is usually
due to a combination of things including, really tired, being warm and
comfortable on the sette and having just eaten a lot. However on this
occasion I fell asleep because the film was rubbish. The plot
development was constant. Constantly slow and boring. Things seemed to
happen, but with no explanation of what was causing them or why. I
admit, I may have missed part of the film, but i watched the majority
```

of it and everything just seemed to happen of its own accord without any real concern for anything else. I cant recommend this film at all.'
Classificação: 0
Revisão: 2
 b'Mann photographs the Alberta Rocky Mountains in a superb fashion, and Jimmy Stewart and Walter Brennan give enjoyable performances as they always seem to do.

But come on Hollywood - a Mountie telling the people of Dawson City, Yukon to elect themselves a marshal (yes a marshal!) and to enforce the law themselves, then gunfighters battling it out on the streets for control of the town?

Nothing even remotely resembling that happened on the Canadian side of the border during the Klondike gold rush. Mr. Mann and company appear to have mistaken Dawson City for Deadwood, the Canadian North for the American Wild West.

Canadian viewers be prepared for a Reefer Madness type of enjoyable howl with this ludicrous plot, or, to shake your head in disgust.'
Classificação: 0
Revisão: 3
 b'This is the kind of film for a snowy Sunday afternoon when the rest of the world can go ahead with its own business as you descend into a big arm-chair and mellow for a couple of hours. Wonderful performances from Cher and Nicolas Cage (as always) gently row the plot along. There are no rapids to cross, no dangerous waters, just a warm and witty paddle through New York life at its best. A family film in every sense and one that deserves the praise it received.'
Classificação: 1
Revisão: 4
 b'As others have mentioned, all the women that go nude in this film are mostly absolutely gorgeous. The plot very ably shows the hypocrisy of the female libido. When men are around they want to be pursued, but when no "men" are around, they become the pursuers of a 14 year old boy. And the boy becomes a man really fast (we should all be so lucky at this age!). He then gets up the courage to pursue his true love.'
Classificação: 1
Revisão: 5
 b"This is a film which should be seen by anybody interested in, effected by, or suffering from an eating disorder. It is an amazingly accurate and sensitive portrayal of bulimia in a teenage girl, its causes and its symptoms. The girl is played by one of the most brilliant young actresses working in cinema today, Alison Lohman, who was later so spectacular in 'Where the Truth Lies'. I would recommend that this film be shown in all schools, as you will never see a better on this subject. Alison Lohman is absolutely outstanding, and one marvels at her ability to convey the anguish of a girl suffering from this compulsive disorder. If barometers tell us the air pressure, Alison Lohman tells us the emotional pressure with the same degree of accuracy. Her emotional range is so precise, each scene could be measured microscopically for its gradations of trauma, on a scale of

rising hysteria and desperation which reaches unbearable intensity. Mare Winningham is the perfect choice to play her mother, and does so with immense sympathy and a range of emotions just as finely tuned as Lohman's. Together, they make a pair of sensitive emotional oscillators vibrating in resonance with one another. This film is really an astonishing achievement, and director Katt Shea should be proud of it. The only reason for not seeing it is if you are not interested in people. But even if you like nature films best, this is after all animal behaviour at the sharp edge. Bulimia is an extreme version of how a tormented soul can destroy her own body in a frenzy of despair. And if we don't sympathise with people suffering from the depths of despair, then we are dead inside."
Classificação: 1
Revisão: 6
 b'Okay, you have:

Penelope Keith as Miss Herringbone-Tweed, B.B.E. (Backbone of England.) She\'s killed off in the first scene - that\'s right, folks; this show has no backbone!

Peter O\'Toole as Ol\' Colonel Cricket from The First War and now the emblazered Lord of the Manor.

Joanna Lumley as the ensweatered Lady of the Manor, 20 years younger than the colonel and 20 years past her own prime but still glamourous (Brit spelling, not mine) enough to have a toy-boy on the side. It\'s alright, they have Col. Cricket\'s full knowledge and consent (they guy even comes \'round for Christmas!) Still, she\'s considerate of the colonel enough to have said toy-boy her own age (what a gal!)

David McCallum as said toy-boy, equally as pointlessly glamourous as his squeeze. Pilcher couldn\'t come up with any cover for him within the story, so she gave him a hush-hush job at the Circus.

and finally:

Susan Hampshire as Miss Polonia Teacups, Venerable Headmistress of the Venerable Girls\' Boarding-School, serving tea in her office with a dash of deep, poignant advice for life in the outside world just before graduation. Her best bit of advice: "I\'ve only been to Nancherrow (the local Stately Home of England) once. I thought it was very beautiful but, somehow, not part of the real world." Well, we can\'t say they didn\'t warn us.

Ah, Susan - time was, your character would have been running the whole show. They don\'t write \'em like that any more. Our loss, not yours.

So - with a cast and setting like this, you have the re-makings of "Brideshead Revisited," right?

Wrong! They took these 1-dimensional supporting roles because they paid so well. After all, acting is one of the oldest temp-jobs there is (YOU name another!)

First warning sign: lots and lots of backlighting. They get around it by shooting outdoors - "hey, it\'s just the sunlight!"

Second warning sign: Leading Lady cries a lot. When not crying, her eyes are moist. That\'s the law of romance novels: Leading Lady is "dewy-eyed."

Henceforth, Leading Lady shall be known as L.L.

Third warning sign: L.L. actually has stars in her eyes when she\'s in love. Still, I\'ll give Emily Mortimer an award just for having to act with that spotlight in

her eyes (I wonder . did they use contacts?)

And lastly, fourth warning sign: no on-screen female character is "Mrs." She\'s either "Miss" or "Lady."

When all was said and done, I still couldn\'t tell you who was pursuing whom and why. I couldn\'t even tell you what was said and done.

To sum up: they all live through World War II without anything happening to them at all.

OK, at the end, L.L. finds she\'s lost her parents to the Japanese prison camps and baby sis comes home catatonic. Meanwhile (there\'s always a "meanwhile,") some young guy L.L. had a crush on (when, I don\'t know) comes home from some wartime tough spot and is found living on the street by Lady of the Manor (must be some street if SHE\'s going to find him there.) Both war casualties are whisked away to recover at Nancherrow (SOMEBODY has to be "whisked away" SOMEWHERE in these romance stories!)

Great drama.'
Classificação: 0
Revisão: 7
 b'The film is based on a genuine 1950s novel.

Journalist Colin McInnes wrote a set of three "London novels": "Absolute Beginners", "City of Spades" and "Mr Love and Justice". I have read all three. The first two are excellent. The last, perhaps an experiment that did not come off. But McInnes\'s work is highly acclaimed; and rightly so. This musical is the novelist\'s ultimate nightmare - to see the fruits of one\'s mind being turned into a glitzy, badly-acted, soporific one-dimensional apology of a film that says it captures the spirit of 1950s London, and does nothing of the sort.

Thank goodness Colin McInnes wasn\'t alive to witness it.'
Classificação: 0
Revisão: 8
 b'I really love the sexy action and sci-fi films of the sixties and its because of the actress\'s that appeared in them. They found the sexiest women to be in these films and it didn\'t matter if they could act (Remember "Candy"?). The reason I was disappointed by this film was because it wasn\'t nostalgic enough. The story here has a European sci-fi film called "Dragonfly" being made and the director is fired. So the producers decide to let a young aspiring filmmaker (Jeremy Davies) to complete the picture. They\'re is one real beautiful woman in the film who plays Dragonfly but she\'s barely in it. Film is written and directed by Roman Coppola who uses some of his fathers exploits from his early days and puts it into the script. I wish the film could have been an homage to those early films. They could have lots of cameos by actors who appeared in them. There is one actor in this film who was popular from the sixties and its John Phillip Law (Barbarella). Gerard Depardieu, Giancarlo Giannini and Dean Stockwell appear as well. I guess I\'m going to have to continue waiting for a director to make a good homage to the films of the sixties. If any are reading this, "Make it as sexy as you can"! I\'ll be waiting!'
Classificação: 0
Revisão: 9

b'Sure, this one isn\'t really a blockbuster, nor does it target such a position. "Dieter" is the first name of a quite popular German musician, who is either loved or hated for his kind of acting and thats exactly what this movie is about. It is based on the autobiography "Dieter Bohlen" wrote a few years ago but isn\'t meant to be accurate on that. The movie is filled with some sexual offensive content (at least for American standard) which is either amusing (not for the other "actors" of course) or dumb - it depends on your individual kind of humor or on you being a "Bohlen"-Fan or not. Technically speaking there isn\'t much to criticize. Speaking of me I find this movie to be an OK-movie.'
Classificação: 0
Revisão: 10
b'During a sleepless night, I was switching through the channels & found this embarrassment of a movie. What were they thinking?

If this is life after "Remote Control" for Kari (Wuhrer) Salin, no wonder she\'s gone nowhere.

And why did David Keith take this role? It\'s pathetic!

Anyway, I turned on the movie near the end, so I didn\'t get much of the plot. But this must\'ve been the best part. This nerdy college kid brings home this dominatrix-ish girl...this scene is straight out of the comic books -- or the cheap porn movies. She calls the mother anal retentive and kisses the father "Oh, I didn\'t expect tongue!" Great lines!

After this, I had to see how it ended..

Well, of course, this bitch from hell has a helluva past, so the SWAT team is upstairs. And yes...they surround her! And YES YES! The kid blows her brains out!!!! AHAHHAHAHAHA!!

This is must-see TV.

'
Classificação: 0
Revisão: 11
b'Cute film about three lively sisters from Switzerland (often seen running about in matching outfits) who want to get their parents back together (seems mom is still carrying the torch for dad) - so they sail off to New York to stop the dad from marrying a blonde gold-digger he calls "Precious". Dad hasn\'t seen his daughters in ten years, they (oddly enough) don\'t seem to mind and think he\'s wonderful, and meanwhile Precious seems to lead a life mainly run by her overbearing mother (Alice Brady), a woman who just wants to see to it her daughter marries a rich man. The sisters get the idea of pushing Precious into the path of a drunken Hungarian count, tricking the two gold-digging women into thinking he is one of the richest men in Europe. But a case of mistaken identity makes the girls think the count is good-looking Ray Milland, who goes along with the scheme \'cause he has a crush on sister Kay.

This film is enjoyable, light fare. Barbara Read as Kay comes across as sweet and pretty, Ray Milland looks oh so young and handsome here (though, unfortunately, is given little to do), Alice Brady is quite good as the scheming mother - but it is Deanna Durbin, a real charmer and cute as a button playing youngest sister Penny, who pretty much steals the show. With absolutely beautiful vocals, she sings several songs

throughout the film, though I actually would have liked to have seen them feature her even more in this. The plot in this film is a bit silly, but nevertheless, I found the film to be entertaining and fun.'
Classificação: 1
Revisão: 12
 b"This 1984 version of the Dickens' classic `A Christmas Carol,' directed by Clive Donner, stars George C. Scott as Ebenezer Scrooge. By this time around, the challenge for the filmmaker was to take such familiar material and make it seem fresh and new again; and, happily to say, with this film Donner not only met the challenge but surpassed any expectations anyone might have had for it. He tells the story with precision and an eye to detail, and extracts performances from his actors that are nothing less than superlative, especially Scott. One could argue that the definitive portrayal of Scrooge-- one of the best known characters in literary fiction, ever-- was created by Alastair Sim in the 1951 film; but I think with his performance here, Scott has now achieved that distinction. There is such a purity and honesty in his Scrooge that it becomes difficult to even consider anyone else in the role once you've seen Scott do it; simply put, he IS Scrooge. And what a tribute it is to such a gifted actor; to be able to take such a well known figure and make it so uniquely his own is quite miraculous. It is truly a joy to see an actor ply his trade so well, to be able to make a character so real, from every word he utters down to the finest expression of his face, and to make it all ring so true. It's a study in perfection.

The other members of the cast are splendid as well, but then again they have to be in order to maintain the integrity of Scott's performance; and they do. Frank Finlay is the Ghost of Jacob Marley; a notable turn, though not as memorable, perhaps, as the one by Alec Guinness (as Marley) in the film, `Scrooge.' Angela Pleasence is a welcome visage as the Spirit of Christmas Past; Edward Woodward, grand and boisterous, and altogether convincing as the Spirit of Christmas Present; and Michael Carter, grim and menacing as the Spirit of Christmas Yet To Come.

David Warner hits just the right mark with his Bob Cratchit, bringing a sincerity to the role that measures up well to the standard of quality set by Scott's Scrooge, and Susannah York fares just as well as Mrs. Cratchit. The real gem to be found here, though, is the performance of young Anthony Walters as Tiny Tim; it's heartfelt without ever becoming maudlin, and simply one of the best interpretations-- and the most real-- ever presented on film.

The excellent supporting cast includes Roger Rees (Fred Holywell, and also the narrator of the film), Caroline Langrishe (Janet Holywell), Lucy Gutteridge (Belle), Michael Gough (Mr. Poole) and Joanne Whalley (Fan). A flawless presentation, this version of `A Christmas Carol' sets the standard against which all others must be gauged; no matter how many versions you may have seen, watching this one is like seeing it for the first time ever. And forever after, whenever you think of Scrooge, the image your mind will conjure up will be that of George C. Scott. A thoroughly entertaining and

satisfying experience, this film demands a place in the annual schedule of the holiday festivities of every home. I rate this one 10/10."
Classificação: 1
Revisão: 13
 b'Put the blame on executive producer Wes Craven and financiers the Weinsteins for this big-budget debacle: a thrash-metal updating of "Dracula", with a condescending verbal jab at Bram Stoker (who probably wouldn\'t want his name on this thing anyway) and nothing much for the rest of us except slasher-styled jolts and gore. Christopher Plummer looks winded as Van Helsing in the modern-day--not just a descendant of Van Helsing but the real thing; he keeps himself going with leeches obtained from Count Dracula\'s corpse, which is exhumed from its coffin after being stolen from Van Helsing\'s vault and flown to New Orleans. This is just what New Orleans needs in the 21st Century! The film, well-produced but without a single original idea (except for multi-racial victims), is both repulsive and lazy, and after about an hour starts repeating itself. * from ****'
Classificação: 0
Revisão: 14
 b'Hilarious, evocative, confusing, brilliant film. Reminds me of Bunuel\'s L\'Age D\'Or or Jodorowsky\'s Holy Mountain-- lots of strange characters mucking about and looking for..... what is it? I laughed almost the whole way through, all the while keeping a peripheral eye on the bewildered and occasionally horrified reactions of the audience that surrounded me in the theatre. Entertaining through and through, from the beginning to the guts and poisoned entrails all the way to the end, if it was an end. I only wish i could remember every detail. It haunts me sometimes.

Honestly, though, i have only the most positive recollections of this film. As it doesn\'t seem to be available to take home and watch, i suppose i\'ll have to wait a few more years until Crispin Glover comes my way again with his Big Slide Show (and subsequent "What is it?" screening)... I saw this film in Atlanta almost directly after being involved in a rather devastating car crash, so i was slightly dazed at the time, which was perhaps a very good state of mind to watch the prophetic talking arthropods and the retards in the superhero costumes and godlike Glover in his appropriate burly-Q setting, scantily clad girlies rising out of the floor like a magnificent DADAist wet dream.

Is it a statement on Life As We Know It? Of course everyone EXPECTS art to be just that. I rather think that the truth is more evident in the absences and in the negative space. What you don\'t tell us is what we must deduce, but is far more valid than the lies that other people feed us day in and day out. Rather one "WHAT IS IT?" than 5000 movies like "Titanic" or "Sleepless in Seattle" (shudder, gag, groan).

Thank you, Mr. Glover (additionally a fun man to watch on screen or at his Big Slide Show-- smart, funny, quirky, and outrageously hot). Make more films, write more books, keep the nightmare alive.'

```
Classificação: 1
Revisão: 15
 b'It was disgusting and painful. What a waste of a cast! I swear, the
audience (1/2 full) laughed TWICE in 90 minutes. This is not a lie. Do
not even rent it.<br /><br />Zeta Jones was just too mean to be
believable.<br /><br />Cusack was OK. Just OK. I felt sorry for him
(the actor) in case people remember this mess.<br /><br />Roberts was
the same as she always is. Charming and sweet, but with no purpose.
The "romance" with John was completely unbelievable.'
Classificação: 0

# Compila modelo
classificador.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"])

# Retreina modelo
classificador.fit(imdb_train, validation_data=imdb_test,
epochs=EPOCHS)

Epoch 1/10
 136/1563 ━━━━━━━━━━━━━━━━━━━━ 36:58 2s/step - accuracy: 0.4998 -
loss: 0.8037

# Classifica dois exemplos novos
preds = classificador.predict(["What an amazing movie!", "A total
waste of my time."])
print(preds)
```