SCRIPT RECOGNITION VIA NEURAL NETWORK(S)

SEBASTIAN, MANPREET, RAHIMA, ADNAN

AGENDA

- Classifier Task
- Creation of Dataset
- Binary Classifier „Latin vs. Chinese“
    * DNN
    * Logistic Regression
- Multiple Classifier „Latin vs. Chinese vs. Kyrillic vs. Georgian“
    * DNN
    * CNN

CLASSIFIER TASK

合作是正确选择

重大原则决不让步 坚决反对加征关税

THE INDEI

25p No 1 TUESDAY 7 OCT

‘Missile tube blast sank Soviet

By Mark Urban
Defence Correspondent

SUBMARINE design experts said last night that an explosion in a missile launch tube caused the damage which sank the stricken Soviet Yankee class submarine in the Atlantic Ocean.

According to one, the force of the blast — caused by missile fuel rather than warheads — was such that it would have blown off access hatches, and may well have,“split the whole tube”.

The Yankee-1 class submarine carries 16 SSN-6 ballistic missiles. The tubes in which they are held are extremely strong, being designed to cope with the forces involved in launching missiles under water. At least one of the liquid-fuelled missiles exploded, following a fire during maintenance.

This allowed water to rush into the rest of the submarine. It is not known whether the explosion happened before or after the submarine surfaced on Saturday night.

After it surfaced the crew were able to assess the damage, which included a gaping hole, where the missile tube hatch had blown off. Soviet merchant ships took most of the 120-strong crew off the boat. Some sailors stayed on board, trying to cut back the fairing over the missile section so that they could patch the hull.

The Soviet merchant vessel Krasnogvardeisk took the submarine in tow on Sunday. The Pentagon believed the boat was out of danger because the Yankee class have great inbuilt buoyancy — all 16 launch tubes can be full of water without affecting seakeeping qualities.

But the blast was so strong that it weakened the whole missile section, which flooded in heavy seas during yesterday morning. The submarine sank in three miles of water — too deep for the boat’s bulkheads to withstand the pressure. They would have “popped” as the boat sank.

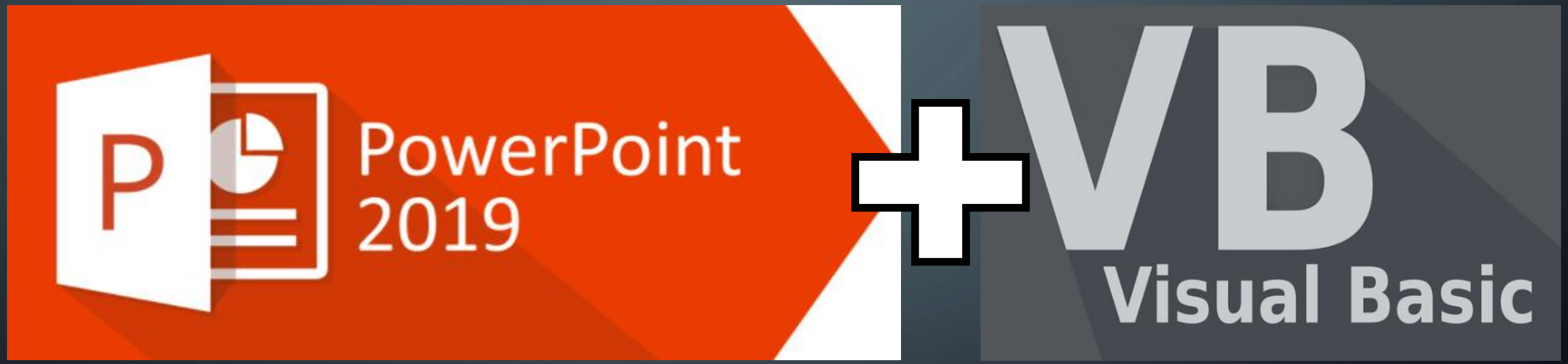
The compartment housing the Yankee’s two nuclear propulsion reactors is the strongest part of the boat and would have been the last section to collapse — the rest of the hull could have imploded first.

The wreck lies so deep that salvage by any current methods would be difficult, if not impossible.

Scientists do not believe that

关切是解决好分歧的前提

CLASSIFIER TASK




Versuch 2.pptm - PowerPoint
DATEI
START
EINFÜGEN
ENTWURF
ÜBERGÄNGE
ANIMATIONEN
BILDSCHIRMPRÄSENTATION
ÜBERPRÜFEN
ANSICHT
Koch, S...
Einfügen
Neue Folie
Schützen
Formen
Anordnen
Schnellformat-vorlagen
Bearbeiten
Zwischenablage
Folien
Schriftart
Absatz
Schutz
Zeichnung
1 闶瑄岸
2 嫸葥賑
3 扒稄餛
4 猳砩爓
5 屁蠑卞
6 垏曦撽
7 誹怊涫
8 鴨齘钔
9 嫌頁頲
10 攜涅敵
11 榤萼濟
闶 瑄 岸
FOLIE 1 VON 2000
ENGLISCH (GROßBRITANNIEN)
NOTIZEN
KOMMENTARE
186 %

```
[106] model = keras.Sequential([
        keras.layers.Flatten(input_shape=(320,100,3)),
        keras.layers.Dense(3, activation=tf.nn.relu),
        keras.layers.Dense(1, activation='sigmoid')
      ])
      model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_3 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 288003
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 4
=================================================================
Total params: 288,007
Trainable params: 288,007
Non-trainable params: 0
_________________________________________________________________
```

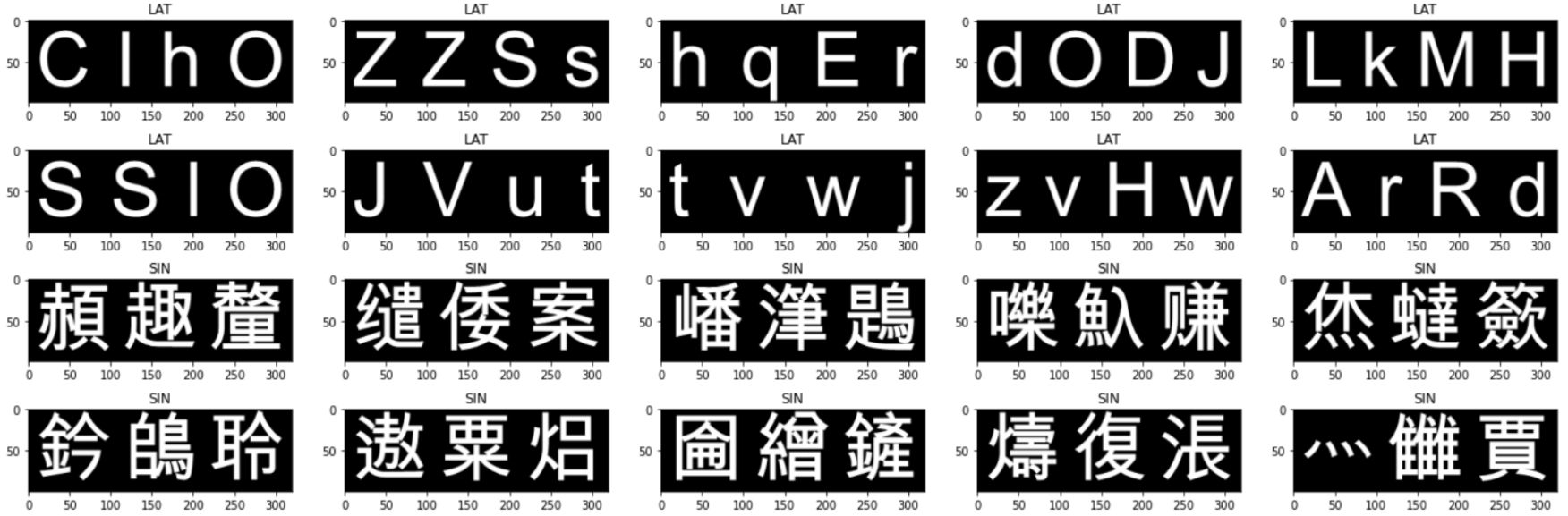
```
Epoch 1/5
1/1 [==============================] - 0s 393ms/step - loss: 0.7054 - accuracy: 0.5500
Epoch 2/5
1/1 [==============================] - 0s 43ms/step - loss: 2.6543 - accuracy: 0.7000
Epoch 3/5
1/1 [==============================] - 0s 44ms/step - loss: 0.3450 - accuracy: 1.0000
Epoch 4/5
1/1 [==============================] - 0s 43ms/step - loss: 0.3446 - accuracy: 1.0000
Epoch 5/5
1/1 [==============================] - 0s 45ms/step - loss: 0.3442 - accuracy: 1.0000
```

```
[110] model.evaluate(test_generator)
```

```
1/1 [==============================] - 0s 126ms/step - loss: 0.3439 - accuracy: 1.0000
[0.34385788440704346, 1.0]
```

Training Set with Labels

Test Set with Predictions (100% Accuracy)



```
[129] model = keras.Sequential([
        keras.layers.Flatten(input_shape=(320,100,3)),
        keras.layers.Dense(2, activation=tf.nn.relu),
        keras.layers.Dense(1, activation='sigmoid')
      ])
      model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 192002
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 3
=================================================================
Total params: 192,005
Trainable params: 192,005
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/5
1/1 [=====] - 0s 401ms/step - loss: 0.6469 - accuracy: 0.6500
Epoch 2/5
1/1 [=====] - 0s 47ms/step - loss: 0.7744 - accuracy: 0.8500
Epoch 3/5
1/1 [=====] - 0s 43ms/step - loss: 0.1380 - accuracy: 1.0000
Epoch 4/5
1/1 [=====] - 0s 43ms/step - loss: 0.1379 - accuracy: 1.0000
Epoch 5/5
1/1 [=====] - 0s 56ms/step - loss: 0.1377 - accuracy: 1.0000
```

```
135] model.evaluate(test_generator)
```

```
1/1 [=====] - 0s 59ms/step - loss: 0.0729 - accuracy: 1.0000
[0.072921022772789, 1.0]
```

```
[129] model = keras.Sequential([
        keras.layers.Flatten(input_shape=(320,100,3)),
        keras.layers.Dense(2, activation=tf.nn.relu),
        keras.layers.Dense(1, activation='sigmoid')
      ])
      model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 192002
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 3
=================================================================
Total params: 192,005
Trainable params: 192,005
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/5
1/1 [=====] - 0s 401ms/step - loss: 0.6469 - accuracy: 0.6500
Epoch 2/5
1/1 [=====] - 0s 47ms/step - loss: 0.7744 - accuracy: 0.8500
Epoch 3/5
1/1 [=====] - 0s 43ms/step - loss: 0.1380 - accuracy: 1.0000
Epoch 4/5
1/1 [=====] - 0s 43ms/step - loss: 0.1379 - accuracy: 1.0000
Epoch 5/5
1/1 [=====] - 0s 56ms/step - loss: 0.1377 - accuracy: 1.0000
```

```
135] model.evaluate(test_generator)
```

```
1/1 [=====] - 0s 59ms/step - loss: 0.0729 - accuracy: 1.0000
[0.072921022772789, 1.0]
```

```
[150] model = keras.Sequential([
        keras.layers.Flatten(input_shape=(320,100,3)),

        keras.layers.Dense(1, activation='sigmoid')
      ])
      model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_8 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 96001
=================================================================
Total params: 96,001
Trainable params: 96,001
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/5
1/1 [======] - 0s 334ms/step - loss: 0.6652 - accuracy: 0.6500
Epoch 2/5
1/1 [======] - 0s 46ms/step - loss: 2.9788 - accuracy: 0.6000
Epoch 3/5
1/1 [======] - 0s 45ms/step - loss: 4.6914e-11 - accuracy: 1.0000
Epoch 4/5
1/1 [======] - 0s 46ms/step - loss: 4.6914e-11 - accuracy: 1.0000
Epoch 5/5
1/1 [======] - 0s 44ms/step - loss: 4.6914e-11 - accuracy: 1.0000
```

```
[144] model.evaluate(test_generator)
```

```
1/1 [======] - 0s 61ms/step - loss: 3.8721e-10 - accuracy: 1.0000
[3.872054343823095e-10, 1.0]
```

DIRECT COMPARISON: LATIN - CHINESE

赏鲔gxFh觜戛

赏樀aRDg椹会

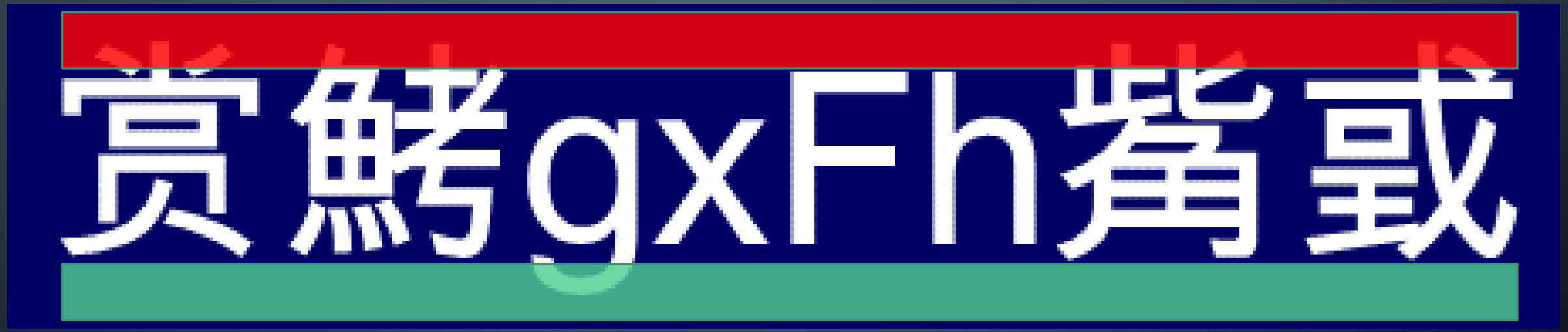
夋颅UIEu扇禄

榴牯zygb婿沛

DIRECT COMPARISON: LATIN - CHINESE

DIRECT COMPARISON: LATIN - CHINESE

DIRECT COMPARISON: LATIN - CHINESE




LAT
G J C I
LAT
H a y c
LAT
d L Y m
LAT
z p k V
LAT
O S W x
LAT
h S F J
SIN
跳岫笞
SIN
嫌頁頸
SIN
閃瑄岸
SIN
猳砭燸
SIN
屁蠑卞
SIN
垏曦撽


LAT
G J C I
LAT
H a y c
LAT
d L Y m
LAT
z p k V
LAT
O S W x
LAT
h S F J
SIN
跳 岫 笭
SIN
嫌 頁 頸
SIN
闶 瑄 岸
SIN
猳 砭 爓
SIN
屁 蠑 卞
SIN
垏 曦 撽


LAT
G J C I
LAT
H a y c
LAT
d L Y m
LAT
z p k V
LAT
O S W x
LAT
h S F J
SIN
跳 岫 笭
SIN
嫌 頁 頸
SIN
閃 瑄 岸
SIN
猳 砭 爓
SIN
屁 蠑 卞
SIN
垏 曦 撽


LAT
G J C I
LAT
H a y c
LAT
d L Y m
LAT
z p k V
LAT
O S W x
LAT
h S F J
SIN
跳 岫 笞
SIN
嫌 頁 頸
SIN
闅 瑄 岸
SIN
猳 砭 爓
SIN
屁 蠑 卞
SIN
垏 曦 撽

cation:

sin (213).png

sin (214).png

sin (215).png

sin (216).png

sin (217).png

sin (218).png

sin (219).png

sin (220).png

sin (221).png

sin (222).png

sin (223).png

sin (224).png

lat (215).png

lat (216).png

lat (217).png

lat (218).png

lat (219).png

lat (220).png

lat (221).png

lat (222).png

lat (223).png

lat (224).png

lat (225).png

lat (226).png

```
1/1 [==============================] - 0s 439ms/step - loss: 0.0090 - accuracy: 1.0000
Epoch 28/30
1/1 [==============================] - 0s 439ms/step - loss: 0.0063 - accuracy: 1.0000
Epoch 29/30
1/1 [==============================] - 0s 440ms/step - loss: 0.0059 - accuracy: 1.0000
Epoch 30/30
1/1 [==============================] - 0s 453ms/step - loss: 0.0063 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
2/2 [==============================] - 1s 197ms/step - loss: 0.0860 - accuracy: 0.9700
```

```
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(320,100,3)),
  #keras.layers.Dense(50, activation='relu'),
  #keras.layers.Dense(10, activation='relu'),
  keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 96001
=================================================================
Total params: 96,001
Trainable params: 96,001
Non-trainable params: 0
```

```
1/1 [==============================] - 0s 439ms/step - loss: 0.0090 - accuracy: 1.0000
Epoch 28/30
1/1 [==============================] - 0s 439ms/step - loss: 0.0063 - accuracy: 1.0000
Epoch 29/30
1/1 [==============================] - 0s 440ms/step - loss: 0.0059 - accuracy: 1.0000
Epoch 30/30
1/1 [==============================] - 0s 453ms/step - loss: 0.0063 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
2/2 [==============================] - 1s 197ms/step - loss: 0.0860 - accuracy: 0.9700
```

```
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(320,100,3)),
  #keras.layers.Dense(50, activation='relu'),
  #keras.layers.Dense(10, activation='relu'),
  keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 96001
=================================================================
Total params: 96,001
Trainable params: 96,001
Non-trainable params: 0
```

```
Epoch 29/30
1/1 [==============================] - 1s 688ms/step - loss: 0.0493 - accuracy: 0.9850
Epoch 30/30
1/1 [==============================] - 1s 651ms/step - loss: 0.0289 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
2/2 [==============================] - 1s 204ms/step - loss: 0.0852 - accuracy: 0.9700
```

```
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(320,100,3)),
  keras.layers.Dense(50, activation='relu'),
  keras.layers.Dense(10, activation='relu'),
  keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 96000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                4800050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 4,800,571
Trainable params: 4,800,571
Non-trainable params: 0
```

4 – LANGUAGES CLASSIFIER

cation: Public

GEO (223).png

GEO (224).png

GEO (225).png

GEO (226).png

GEO (227).png

GEO (228).png

GEO (229).png

GEO (230).png

GEO (231).png

GEO (232).png

GEO (233).png

GEO (234).png

cation:

GEO (223).png

GEO (224).png

GEO (225).png

GEO (226).png

GEO (227).png

GEO (228).png

GEO (229).png

GEO (230).png

GEO (231).png

GEO (232).png

GEO (233).png

GEO (234).png

KYR (229).png

KYR (230).png

KYR (231).png

KYR (232).png

KYR (233).png

KYR (234).png

KYR (235).png

KYR (236).png

KYR (237).png

KYR (238).png

KYR (239).png

KYR (240).png


GEO (229).png


GEO (230).png

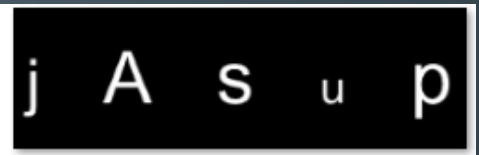

GEO (231).png


GEO (232).png


GEO (233).png


GEO (234).png


lat (215).png


lat (216).png


lat (217).png


lat (218).png


lat (219).png


lat (220).png


sin (221).png


sin (222).png


sin (223).png


sin (224).png


KYR (237).png


KYR (238).png


KYR (239).png


KYR (240).png

```
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(320,100,1)),
  keras.layers.Dense(80, activation='relu'),
  keras.layers.Dense(30, activation='relu'),
  keras.layers.Dense(4, activation='softmax')
])
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_9 (Flatten)          (None, 32000)             0
_________________________________________________________________
dense_27 (Dense)             (None, 80)                2560080
_________________________________________________________________
dense_28 (Dense)             (None, 30)                2430
_________________________________________________________________
dense_29 (Dense)             (None, 4)                 124
=================================================================
Total params: 2,562,634
Trainable params: 2,562,634
Non-trainable params: 0
```

```
Epoch 28/30
1/1 [==============================] - 0s 405ms/step - loss: 0.2512 - accuracy: 0.9250
Epoch 29/30
1/1 [==============================] - 0s 409ms/step - loss: 0.1807 - accuracy: 0.9700
Epoch 30/30
1/1 [==============================] - 0s 393ms/step - loss: 0.1475 - accuracy: 0.9800
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 81s 27s/step - loss: 0.8307 - accuracy: 0.6500
[0.8306537866592407, 0.6499999761581421]
```

200 train; 100 test

```
Epoch 28/30
1/1 [==============================] - 0s 405ms/step - loss: 0.2512 - accuracy: 0.9250
Epoch 29/30
1/1 [==============================] - 0s 409ms/step - loss: 0.1807 - accuracy: 0.9700
Epoch 30/30
1/1 [==============================] - 0s 393ms/step - loss: 0.1475 - accuracy: 0.9800
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 81s 27s/step - loss: 0.8307 - accuracy: 0.6500
[0.8306537866592407, 0.6499999761581421]
```

```
Epoch 58/60
1/1 [==============================] - 0s 413ms/step - loss: 0.0036 - accuracy: 1.0000
Epoch 59/60
1/1 [==============================] - 0s 410ms/step - loss: 0.0036 - accuracy: 1.0000
Epoch 60/60
1/1 [==============================] - 0s 405ms/step - loss: 0.0033 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 1s 171ms/step - loss: 1.4010 - accuracy: 0.6275
[1.4010444487953186, 0.6274999976158142]
```

200 train; 100 test

```
Epoch 28/30
1/1 [==============================] - 0s 405ms/step - loss: 0.2512 - accuracy: 0.9250
Epoch 29/30
1/1 [==============================] - 0s 409ms/step - loss: 0.1807 - accuracy: 0.9700
Epoch 30/30
1/1 [==============================] - 0s 393ms/step - loss: 0.1475 - accuracy: 0.9800
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 81s 27s/step - loss: 0.8307 - accuracy: 0.6500
[0.8306537866592407, 0.6499999761581421]
```

```
Epoch 58/60
1/1 [==============================] - 0s 202ms/step - loss: 0.3843 - accuracy: 0.8500
Epoch 59/60
1/1 [==============================] - 0s 208ms/step - loss: 0.2442 - accuracy: 0.9400
Epoch 60/60
1/1 [==============================] - 1s 762ms/step - loss: 0.2546 - accuracy: 0.9000
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 81s 27s/step - loss: 0.6567 - accuracy: 0.7375
[0.6566987633705139, 0.737500011920929]
```

```
Epoch 58/60
1/1 [==============================] - 0s 413ms/step - loss: 0.0036 - accuracy: 1.0000
Epoch 59/60
1/1 [==============================] - 0s 410ms/step - loss: 0.0036 - accuracy: 1.0000
Epoch 60/60
1/1 [==============================] - 0s 405ms/step - loss: 0.0033 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 1s 171ms/step - loss: 1.4010 - accuracy: 0.6275
[1.401044487953186, 0.6274999976158142]
```

200 train; 100 test

500 train; 100 test

```
Epoch 28/30
1/1 [==============================] - 0s 405ms/step - loss: 0.2512 - accuracy: 0.9250
Epoch 29/30
1/1 [==============================] - 0s 409ms/step - loss: 0.1807 - accuracy: 0.9700
Epoch 30/30
1/1 [==============================] - 0s 393ms/step - loss: 0.1475 - accuracy: 0.9800
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 81s 27s/step - loss: 0.8307 - accuracy: 0.6500
[0.8306537866592407, 0.6499999761581421]
```

```
Epoch 58/60
1/1 [==============================] - 0s 202ms/step - loss: 0.3843 - accuracy: 0.8500
Epoch 59/60
1/1 [==============================] - 0s 208ms/step - loss: 0.2442 - accuracy: 0.9400
Epoch 60/60
1/1 [==============================] - 1s 762ms/step - loss: 0.2546 - accuracy: 0.9000
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 81s 27s/step - loss: 0.6567 - accuracy: 0.7375
[0.6566987633705139, 0.737500011920929]
```

```
Epoch 58/60
1/1 [==============================] - 0s 413ms/step - loss: 0.0036 - accuracy: 1.0000
Epoch 59/60
1/1 [==============================] - 0s 410ms/step - loss: 0.0036 - accuracy: 1.0000
Epoch 60/60
1/1 [==============================] - 0s 405ms/step - loss: 0.0033 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 1s 171ms/step - loss: 1.4010 - accuracy: 0.6275
[1.401044487953186, 0.6274999976158142]
```

200 train; 100 test

```
Epoch 998/1000
1/1 [==============================] - 0s 202ms/step - loss: 5.9799e-04 - accuracy: 1.0000
Epoch 999/1000
1/1 [==============================] - 0s 204ms/step - loss: 3.8059e-04 - accuracy: 1.0000
Epoch 1000/1000
1/1 [==============================] - 0s 190ms/step - loss: 2.9140e-04 - accuracy: 1.0000
```

```
model.evaluate(test_generator)
```

```
4/4 [==============================] - 1s 156ms/step - loss: 1.0517 - accuracy: 0.7600
[1.0517429113388062, 0.7599999904632568]
```

500 train; 100 test

4 – LANGUAGES CLASSIFIER

Time for CNN !