



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



25p No 1 TUESDAY 7 OCTOBER 1982

THE INDEPENDENT

'Missile tube blast sank Soviet submarine'

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-I 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

By Mark Urban
Defence Correspondent

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 sea-keeping 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

1982年经贸高级别磋商结束时表示

合作是正确选择

重大原则决不让步 坚决反对加征关税

新华社电 第十一轮中美经贸高级别磋商于5月1日至19日在华盛顿举行。中共中央政治局委员、国务院副总理、中美全面经济对话中方牵头人刘鹤在磋商结束后的媒体吹风会上表示，中美关系十分紧密，经贸关系是中美关系的“压舱石”和“推进器”，不仅涉及两国关系，也涉及世界和平与繁荣。合作是双方唯一正确的选择，但合作是有原则的，在重大原则问题上中方决不让步。

刘鹤强调，双方的协议必须是平等、互利的，在重大原则问题上中方决不马上让步，且双方在很多方面达成了重要共识。他认为三个核心关切问题必须得到解决：一是取消全部加征关税。关税是双方贸易争端的起点，如果要达成协议，加征的关税必须全部取消；二是贸易采购数字要符合实际。双方在阿根廷已对贸易采购数字形成共识，不应随意改变；三是改善文本平衡性。任何国家都有自己的尊严，协议文本必须平衡。目前仍有一些关键问题需要讨论。去年以来，双方谈判出现几次反复，发生了一些曲折，这都是正常的。在双方谈判仍在进行的过程中，随意指责“倒退”是不负责任的。

刘鹤表示，对中国来说，最重要的就是做好自己的事情。中国国内市场需求巨大，供给侧结构性改革将带来产品和服务竞争力的全面提升，财政货币政策仍有充分的空间。中国经济前景非常乐观。在大国博弈的过程中，出现一些曲折是正常的，可以检验我们的能力。在习近平同志为核心的党中央坚强领导下，只要我们坚定信心、共同努力，任何困难都不一定能阻挡中国经济持续健康发展的好态势。

心关切是解决好分歧的前提

CLASSIFIER TASK

DHQW

e i e D

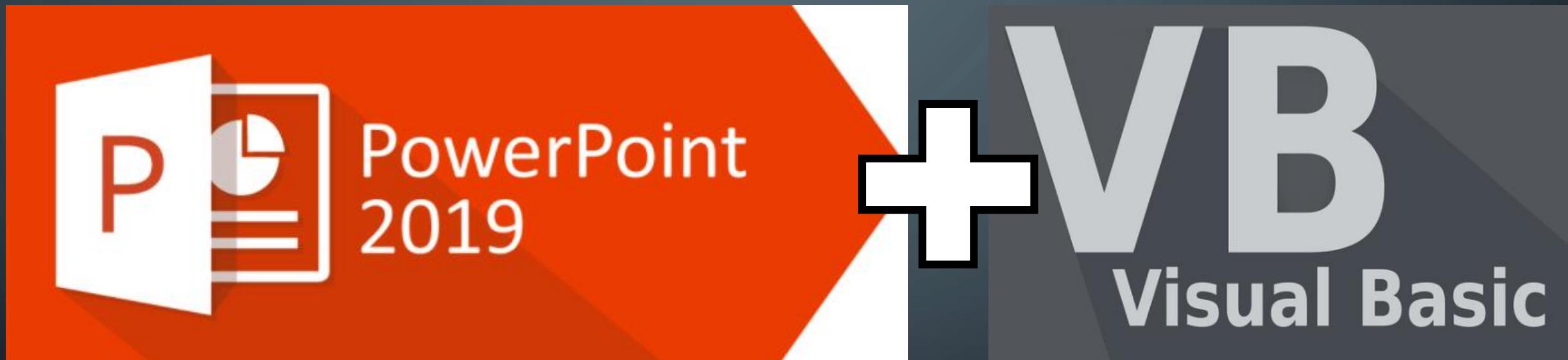
k w Z Q

闖 蹦 女

搔 諸 淪

寧 𠵼 遙





Versuch 2.pptm - PowerPoint

DATEI START EINFÜGEN ENTWURF ÜBERGÄNGE ANIMATIONEN BILDSCHIRMPRÄSENTATION ÜBERPRÜFEN ANSICHT Koch, S...

Einfügen Neue Folie Folien Schriftart Absatz Schutz Zeichnung Bearbeiten

Zwischenablage

1 阅瑄岸

2 媚薦賑

3 扱稷餳

4 猥砣爌

5 屁蠟卞

6 垣曦檄

7 誹忿洎

8 鵠齡钔

9 嫌貢頤

10 攜涅敵

11 櫟莺濟

阅瑄岸

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 #
<hr/>		
flatten_4 (Flatten)	(None, 96000)	0
dense_8 (Dense)	(None, 2)	192002
dense_9 (Dense)	(None, 1)	3
<hr/>		

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

赏鰐gxFh醤或

赏楠aRDg棍会

叟頽UIEu扇祿

榴桔zygb媂沛

DIRECT COMPARISON: LATIN - CHINESE

貢魚考gxFh肴或

DIRECT COMPARISON: LATIN - CHINESE

貢魚考gxFh鰐角或

DIRECT COMPARISON: LATIN - CHINESE

貢魚考gxFh嘴角或

LAT

G J C I

LAT

z p k V

SIN

跳 岘 穗

SIN

狎 破 煙

LAT

H a y c

LAT

O S W x

SIN

塘 頁 頭

SIN

屁 蠍 卍

LAT

d L Y m

LAT

h S F J

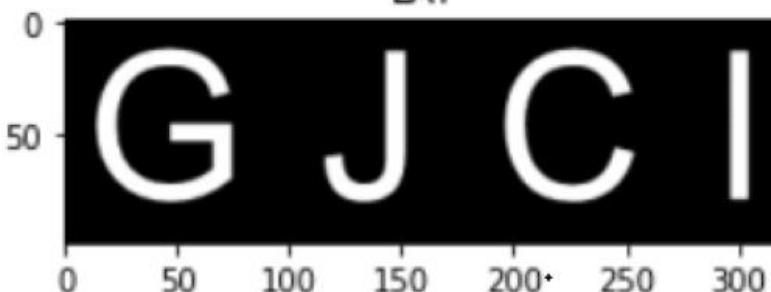
SIN

閔 瑣 岸

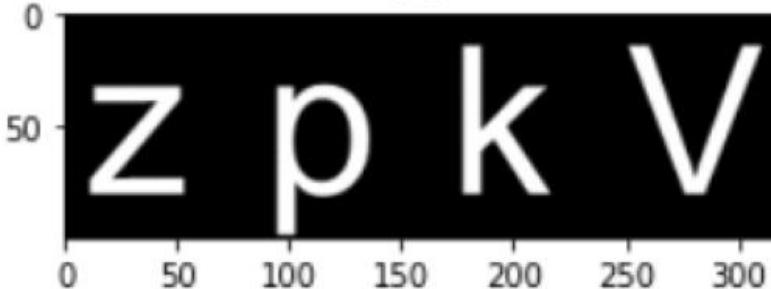
SIN

坪 曜 撒

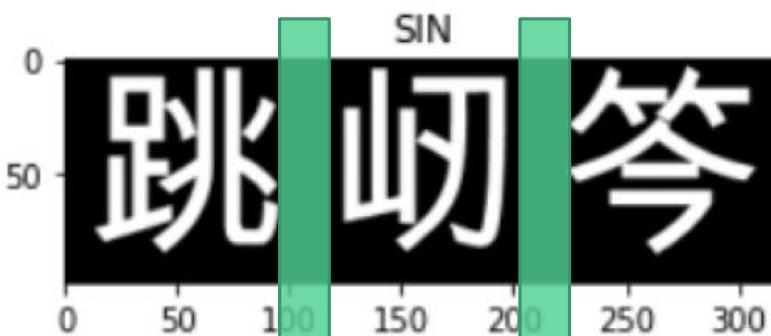
LAT



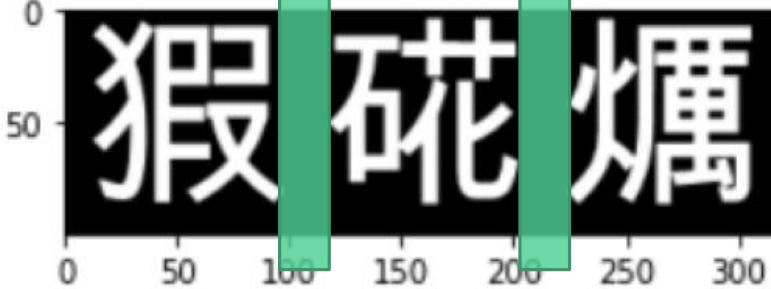
LAT



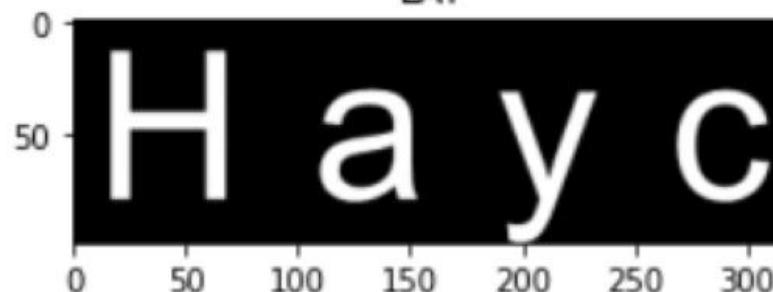
SIN



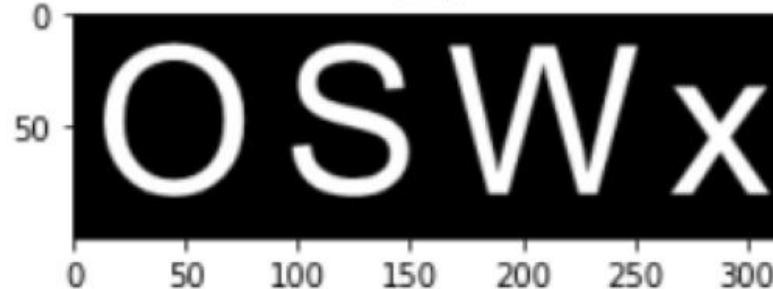
SIN



LAT



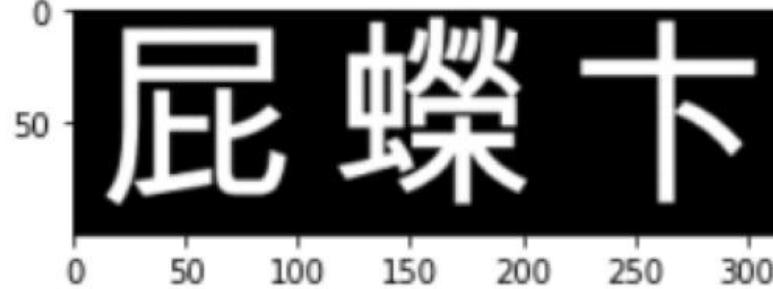
LAT



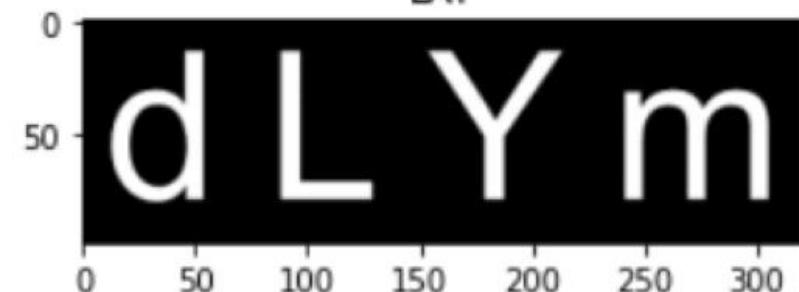
SIN



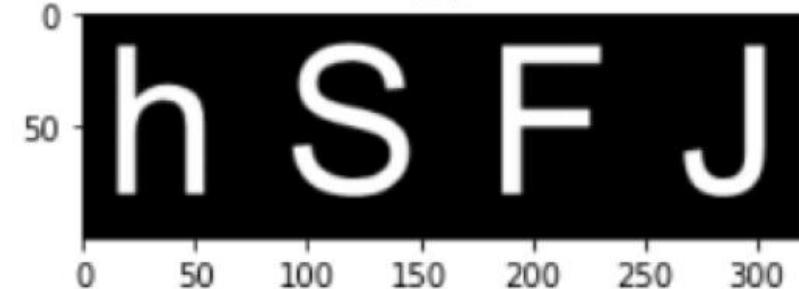
SIN



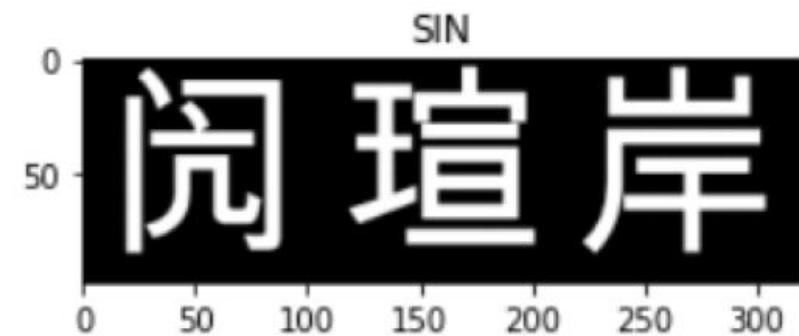
LAT



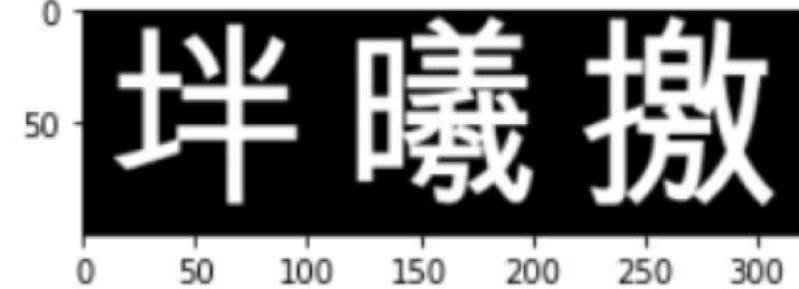
LAT



SIN



SIN







筭 決 猪 挖

sin (213).png

艸 延 初
咗

sin (214).png

伙 划 慢
峩

sin (215).png

呻 蔭 鮫
嬾

sin (216).png

蹙 烹 探 喜

sin (217).png

啾 翱 力 軒

sin (218).png

鵬 捏 謬
葉

sin (219).png

櫟 渗 狼 駘

sin (220).png

鞚 畦 悔 嘻

sin (221).png

执 粢 圜 株

sin (222).png

聯 賜 榴 煙

sin (223).png

蠚 舢 經 驁

sin (224).png

𩷶 眇 韓 这

sin (225).png

葩 蠶 越 台

sin (226).png

cation:

g P I C d

lat (215).png

j A s u p

lat (216).png

z k l e F

lat (217).png

n D S o B

lat (218).png

d U O f b

lat (219).png

y M M g F

lat (220).png

K f Z y Q

lat (221).png

R R n T o

lat (222).png

R s s E C

lat (223).png

R s d T r

lat (224).png

R B D z a

lat (225).png

v R M j K

lat (226).png

h O b v H

lat (227).png

q u q q h

lat (228).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 #
<hr/>		
flatten_2 (Flatten)	(None, 96000)	0
<hr/>		
dense_4 (Dense)	(None, 1)	96001
<hr/>		
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

፩፻፭፻፪፲

GEO (223).png

፩፻፭፻፪፳

GEO (224).png

፩፻፭፻፪፴

GEO (225).png

፩፻፭፻፪፵

GEO (226).png

፩፻፭፻፪፶

GEO (227).png

፩፻፭፻፪፷

GEO (228).png

፩፻፭፻፪፸

GEO (229).png

፩፻፭፻፪፹

GEO (230).png

፩፻፭፻፪፺

GEO (231).png

፩፻፭፻፪፻

GEO (232).png

፩፻፭፻፪፼

GEO (233).png

፩፻፭፻፪፽

GEO (234).png

፩፻፭፻፪፾

GEO (235).png

፩፻፭፻፪፷

GEO (236).png

cation: Public

Օ Ե Կ Բ Ժ Ւ

GEO (223).png

Ժ Ա Ս Փ Ը Ծ Յ

GEO (224).png

cation:

Կ Վ Ի Ե Յ Յ Վ

KYR (229).png

Մ Կ Ի Թ Ե Վ Ի Ւ

KYR (230).png

Լ Ե Յ Յ Կ Կ Վ

GEO (225).png

Ո Ե Ց Տ Ե Ճ Ճ

GEO (226).png

Կ

Կ Վ Կ Շ Յ Յ Ե

KYR (231).png

Ռ Յ Ե Կ Ե Շ Յ Ե

KYR (232).png

Յ Ժ Ե Ր Ա Յ Յ

GEO (227).png

Ի Բ Ղ Ե Ֆ Ե Ե

GEO (228).png

Կ

Խ Ա Խ Ե Թ Ի Ո

KYR (233).png

Յ Ջ Հ Ո Ս Ե Շ Ե

KYR (234).png

Թ Ա Ե Յ Կ Յ Յ

GEO (229).png

Ե Ա Ֆ Գ Ժ Ժ

GEO (230).png

Կ

Կ Ի Վ Վ Շ Յ Յ Ե

KYR (235).png

Շ Կ Ի Ր Վ Փ Վ Ք

KYR (236).png

Դ Ե Մ Ա Խ Ա

GEO (231).png

Դ Ե Մ Ե Ց Ա Ֆ

GEO (232).png

Կ

Յ Ի Ւ Լ Ի Յ Խ

KYR (237).png

Շ Յ Հ լ Վ Վ Կ

KYR (238).png

Յ Խ Ո Ե Ե Յ

GEO (233).png

Դ Վ Ե Մ Ա Բ Ա

GEO (234).png

Կ

Ե յ Ա դ ֆ տ

KYR (239).png

Ա Ր պ յ կ օ

KYR (240).png

Ձ Ե յ ա ֆ Գ

GEO (235).png

Վ Յ Ն Ո Յ Յ

ա

Թ Ա Ե մ Ե Մ Ֆ

KYR (236).png

Վ Ե Կ Ի Վ Ր Գ

KYR (237).png

თ ბ ხ ლ

GEO (229).png

ბ ყ ზ გ ძ ძ

GEO (230).png

თ ხ ს ი ა

GEO (231).png

თ ხ ძ ს მ ჭ

GEO (232).png

ვ ჯ ე ნ ვ

GEO (233).png

თ ყ ძ ვ ჩ ს

GEO (234).png

გ პ ი ც დ

lat (215).png

ჯ ა ს უ რ

lat (216).png

ჟ კ ი ე ჭ

lat (217).png

ნ დ ს ი ვ

lat (218).png

დ უ ი ფ ბ

lat (219).png

უ მ მ გ ჭ

lat (220).png

靶 畜 懈 嘻

sin (221).png

执 粪 圃 株

sin (222).png

聯 賜 榴 煙

sin (223).png

蠶 駢 繼 驁

sin (224).png

鮋 盱 韜 迤

sin (225).png

葩 蠻 越 台

sin (226).png

ჟ ქ თ ი ჟ ხ

KYR (237).png

ў ჟ ლ ყ უ ქ

KYR (238).png

ъ ი დ ფ თ

KYR (239).png

ä ე რ უ კ ი

KYR (240).png

ä ე მ წ შ ჭ

KYR (241).png

ც ე ნ ი უ გ

KYR (242).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  
[1.401044487953186, 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 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 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  
[0.6566987633705139, 0.737500011920929]
```

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 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 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 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 !