Identify casting defects from images

Deep Learning

Instructor : Iegor Rudnytskyi

TA : Ilia Azizi

DEFAULT

OK

7348
images

300*300
pixels

KAGGLE
WEBSITE

defective or correctly casted submersible pump impellers.

DATA SPLITTING

TRAINING

DEFAULT [1] = 3007

OK [0] = 2300

VALIDATION

DEFAULT [1] = 751

OK [0] = 575

TEST

DEFAULT [1] = 453

OK [0] = 262

metric_val_loss	metric_val_acc	flag_n_neurons	samples	epochs	learning_rate	script
0.0385	0.997	500	663	100	1.00E-05	vgg16.R
0.1378	0.997	300	663	100	1.00E-04	model1.R
0.0916	0.9947	500	663	100	1.00E-04	model1.R
0.2249	0.9947	200	663	100	1.00E-04	model1.R
0.0617	0.9939	100	663	100	1.00E-05	vgg16.R
0.1429	0.9939	100	663	100	1.00E-04	model1.R
0.0377	0.9939	NA	663	30	1.00E-05	raw vgg16.R
0.0578	0.9932	200	663	100	1.00E-05	vgg16.R
0.0597	0.9924	300	663	100	1.00E-05	vgg16.R
0.0598	0.9917	400	663	100	1.00E-05	vgg16.R
0.2577	0.9909	50	663	100	1.00E-04	model1.R
0.2196	0.9863	400	663	100	1.00E-04	model1.R
						inception_res
0.34	0.9378	NA	663	30	1.00E-05	net_v2.R

Custom CNN

- 500 neurons
- RMSPROP optimizer with learning rate of 0.00001
- loss crossentropy
- 100 epochs
- early stopping with patience = 7

```
#> Model
#> Model: "sequential"
#> _________________________________________________________________
#> Layer (type)                 Output Shape              Param #
#> =================================================================
#> conv2d_3 (Conv2D)            (None, 298, 298, 32)      896
#> _________________________________________________________________
#> max_pooling2d_3 (MaxPooling2D) (None, 149, 149, 32)    0
#> _________________________________________________________________
#> conv2d_2 (Conv2D)            (None, 147, 147, 64)      18496
#> _________________________________________________________________
#> max_pooling2d_2 (MaxPooling2D) (None, 73, 73, 64)      0
#> _________________________________________________________________
#> conv2d_1 (Conv2D)            (None, 71, 71, 128)       73856
#> _________________________________________________________________
#> max_pooling2d_1 (MaxPooling2D) (None, 35, 35, 128)     0
#> _________________________________________________________________
#> conv2d (Conv2D)              (None, 33, 33, 128)       147584
#> _________________________________________________________________
#> max_pooling2d (MaxPooling2D)   (None, 16, 16, 128)     0
#> _________________________________________________________________
#> flatten (Flatten)            (None, 32768)             0
#> _________________________________________________________________
#> dense_1 (Dense)              (None, 300)               9830700
#> _________________________________________________________________
#> dense (Dense)                (None, 2)                 602
#> =================================================================
#> Total params: 10,072,134
#> Trainable params: 10,072,134
#> Non-trainable params: 0
#> _________________________________________________________________
```

Custom CNN Confusion Matrix

- 99.9% ACCURACY
- 99.6% SENSITIVITY
- 100% SPECIFICITY
- 'POSITIVE' CLASS : 0

REFERENCES

PREDICTION	0	1
0	261	0
1	1	453

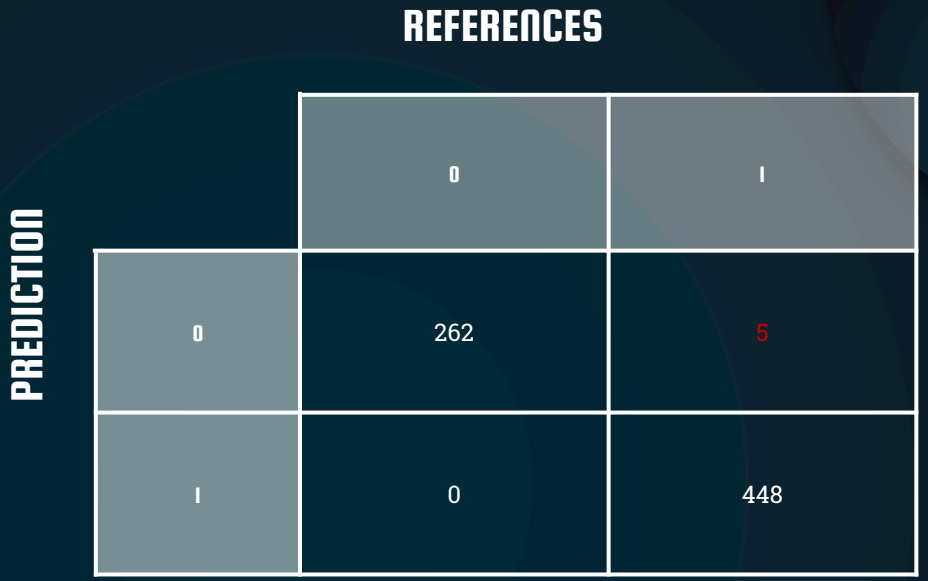
VGG 16 + 2 HIDDEN LAYERS

- 300 neurons
- Relu activation function
- l2 regularizer of 0.01
- RMSPROP optimizer with learning rate of 0.00001
- loss crossentropy
- 100 epochs
- early stopping with patience = 7

```
#> Model
#> Model: "sequential"
#> _________________________________________________________________
#> Layer (type)                 Output Shape              Param #
#> =================================================================
#> vgg16 (Functional)           (None, 9, 9, 512)         14714688
#> _________________________________________________________________
#> flatten (Flatten)            (None, 41472)             0
#> _________________________________________________________________
#> dense_2 (Dense)              (None, 500)               20736500
#> _________________________________________________________________
#> dropout (Dropout)            (None, 500)               0
#> _________________________________________________________________
#> dense_1 (Dense)              (None, 500)               250500
#> _________________________________________________________________
#> dense (Dense)                (None, 2)                 1002
#> =================================================================
#> Total params: 35,702,690
#> Trainable params: 20,988,002
#> Non-trainable params: 14,714,688
#> _________________________________________________________________
```

VGG16
Confusion Matrix

- 99.3% ACCURACY
- 100% SENSITIVITY
- 98.9% SPECIFICITY
- 'POSITIVE' CLASS : 0



CONCLUSION

99.9% ACCURACY

we would recommend our own CNN architecture for this task

both human and AI quality control

Does anyone have any questions?

THANKS!