SISTEME INSTRUIBILE PENTRU ANALIZA IMAGINILOR

HORSES OR HUMANS

CLASIFICATOR: CNN

Teacher: Sl. Dr. Ing Camelia FLOREA
Section: Multimedia Technologies

GAL OSCA

CNN CLASSIFIER EXPLAINED 1

A CNN HAS:

- Convolutional layers
- ReLU layers
- Pooling layers
- a Fully connected layer

CNN ARCHITECTURE WOULD LOOK SOMETHING LIKE THIS:

Input ->Convolution ->ReLU ->Convolution ->ReLU ->Pooling ->
ReLU ->Convolution ->ReLU ->Pooling ->Fully Connected

CNNs have an input layer, and output layer, and hidden layers. The hidden layers usually consist of convolutional layers, ReLU layers, pooling layers, and fully connected layers.

- Convolutional layers apply a convolution operation to the input. This passes the information on to the next layer.
- Pooling combines the outputs of clusters of neurons into a single neuron in the next layer.
- Fully connected layers connect every neuron in one layer to every neuron in the next layer.

GAL OSCA

CNN CLASSIFIER EXPLAINED 2

CNN

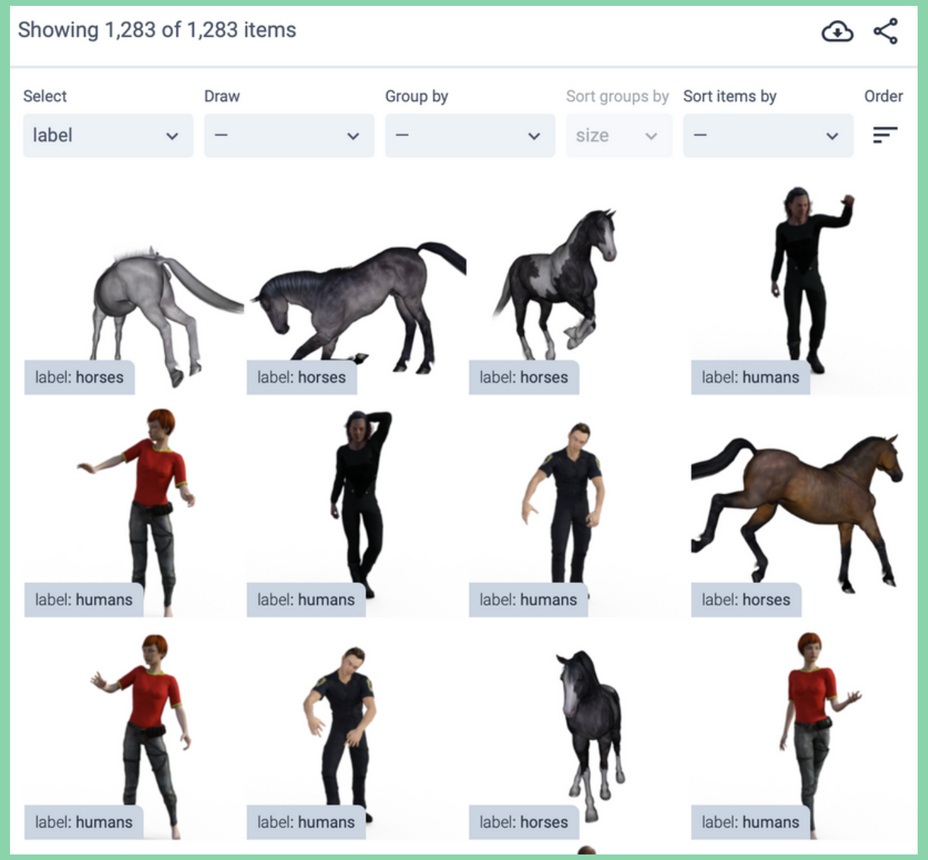
- starts with an input image
- applies many different filters to it to create a feature map
- applies a ReLU function to increase non-linearity
- applies a pooling layer to each feature map
- flattens the pooled images into one long vector.
- inputs the vector into a fully connected artificial neural network.
- processes the features through the network. The final fully connected layer provides the “voting” of the classes that we’re after.
- trains through forward propagation and backpropagation for many, many epochs. This repeats until we have a well-defined neural network with trained weights and feature detectors.

GAL OSCA

DATABASE

HORSES_OR_HUMANS

DATASET SIZE: 1283



label

1,283 items

Value	Count
horses	628
humans	655

split

1,283 items

Value	Count
test	256
train	1,027

GAL OSCA

DATA PROCESSING

DATA PRE-PROCESSING

Data normalization

```
Original images range: [ 8 , 255 ]
PreProcessed images range: [ 0.023529411764705882 , 1.0 ]
Reshaped datasets:
     - x_train_p.shape (718, 300, 300, 3)
     - x_test_p.shape (309, 300, 300, 3)
```

LABELS PROCESSING

```
Original shape of labels vector:
    - y_train.shape (718,)
One Hot Encoded, to_categorical, labels:
    - y_cat_train.shape (718, 2)
    - y_cat_train[0].shape (2,)
Before - y_train[0]:  0 , after  - y_cat_train[0]: [1. 0.]
Before - y_train[1]:  1 , after  - y_cat_train[1]: [0. 1.]
Before - y_train[2]:  1 , after  - y_cat_train[2]: [0. 1.]
Before - y_train[3]:  1 , after  - y_cat_train[3]: [0. 1.]
```


GAL OSCA

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 300, 300, 32)	896
activation (Activation)	(None, 300, 300, 32)	0
batch_normalization (BatchNo	(None, 300, 300, 32)	128
conv2d_1 (Conv2D)	(None, 300, 300, 32)	9248
activation_1 (Activation)	(None, 300, 300, 32)	0
batch_normalization_1 (Batch	(None, 300, 300, 32)	128
max_pooling2d (MaxPooling2D)	(None, 150, 150, 32)	0
dropout (Dropout)	(None, 150, 150, 32)	0
conv2d_2 (Conv2D)	(None, 150, 150, 64)	18496
activation_2 (Activation)	(None, 150, 150, 64)	0
batch_normalization_2 (Batch	(None, 150, 150, 64)	256
conv2d_3 (Conv2D)	(None, 150, 150, 64)	36928
activation_3 (Activation)	(None, 150, 150, 64)	0
batch_normalization_3 (Batch	(None, 150, 150, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 75, 75, 64)	0
dropout_1 (Dropout)	(None, 75, 75, 64)	0
flatten (Flatten)	(None, 360000)	0
dense (Dense)	(None, 512)	184320512
activation_4 (Activation)	(None, 512)	0
batch_normalization_4 (Batch	(None, 512)	2048
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026
activation_5 (Activation)	(None, 2)	0

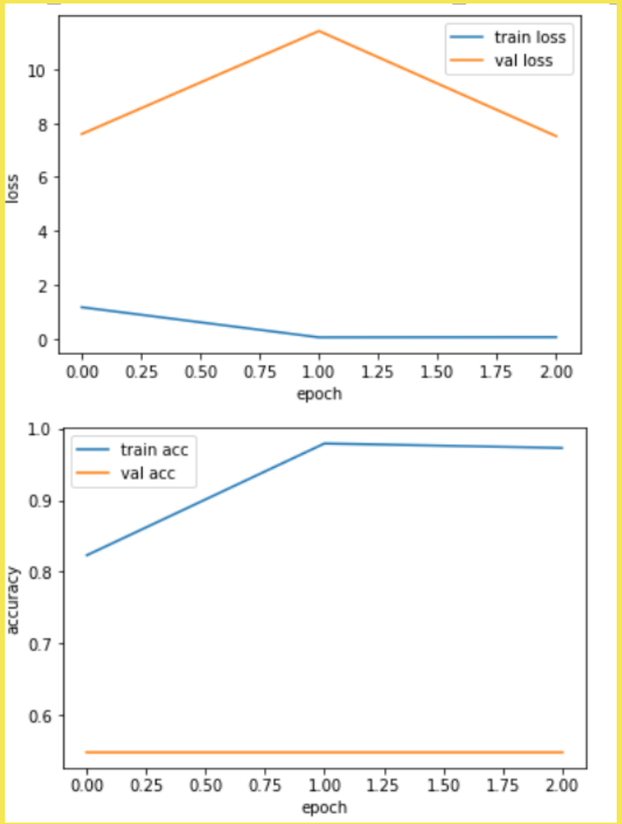
MODEL STRUCTURE

MiniVGGNet Model

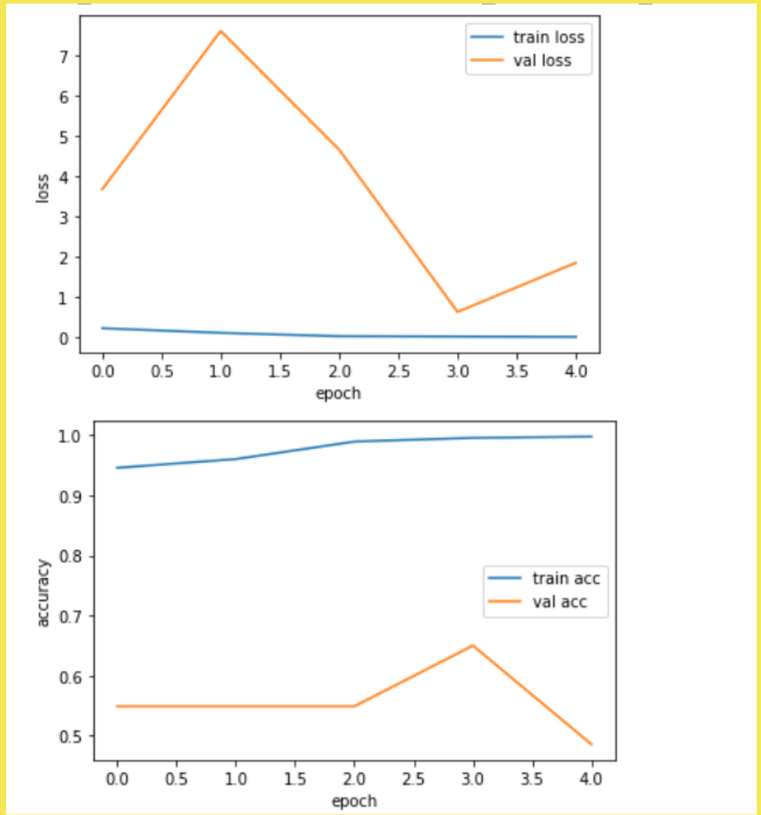
**INPUT ->CONVOLUTION ->RELU ->CONVOLUTION ->RELU ->POOLING ->
RELU ->CONVOLUTION ->RELU ->POOLING ->FULLY CONNECTED**

GAL OSCA

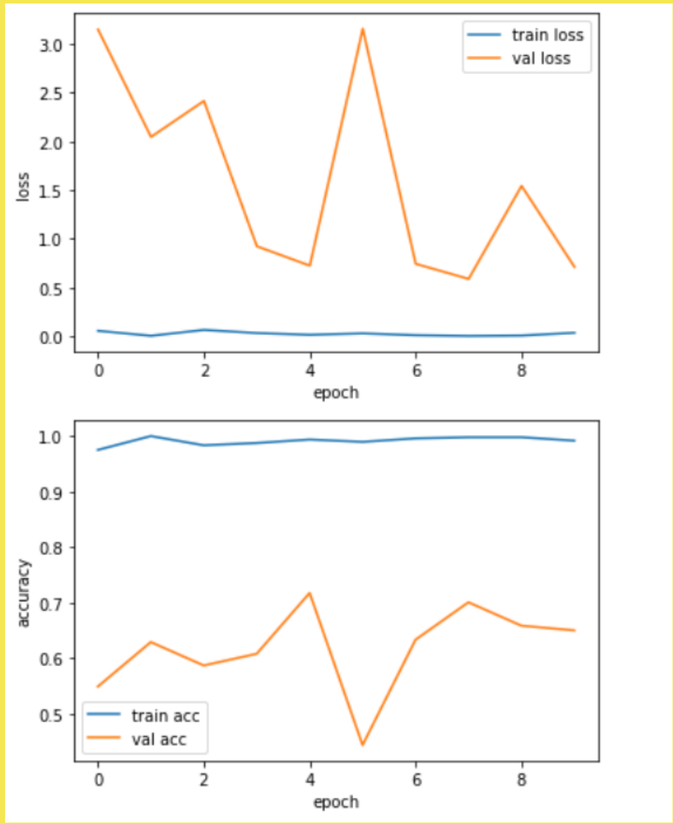
Training Neural Networks



3 epochs

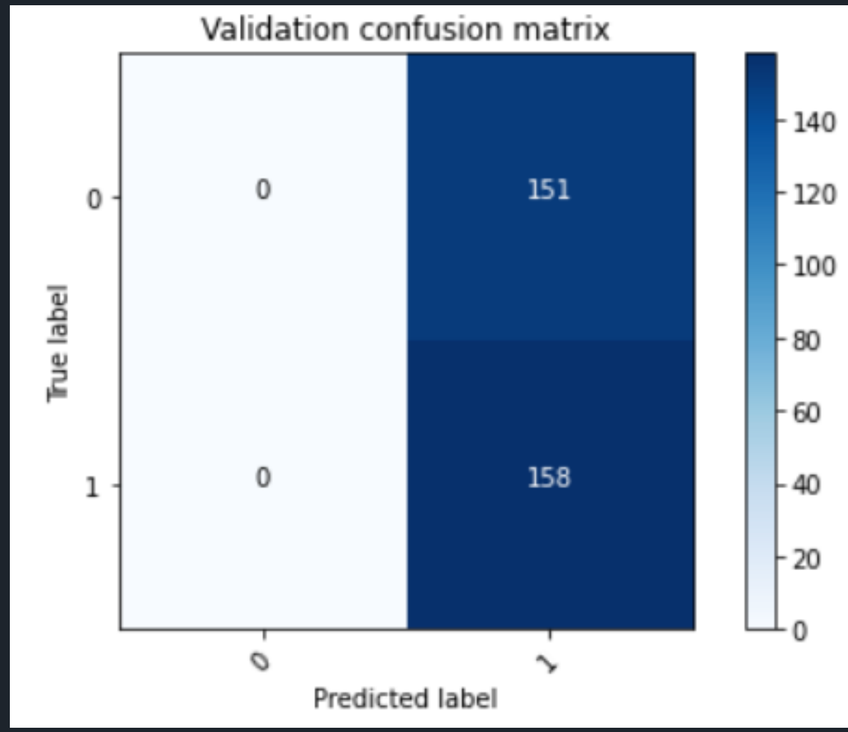


5 epochs



10 epochs

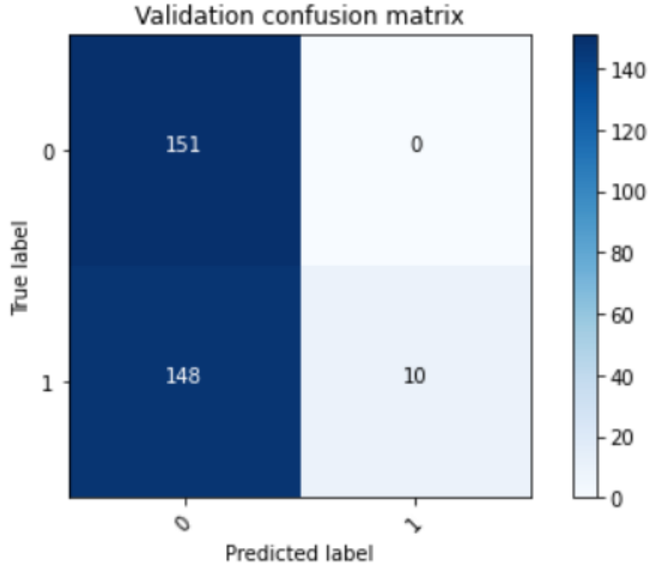
RESULTS 3 EPOCH




```
Classification_report:                precision    recall  f1-score   support

           0       0.00      0.00      0.00       151
           1       0.51      1.00      0.68       158

    accuracy                           0.51       309
   macro avg       0.26      0.50      0.34       309
weighted avg       0.26      0.51      0.35       309

Set date/imagini eronate:  (151, 300, 300, 3)
```

RESULTS 5 EPOCH

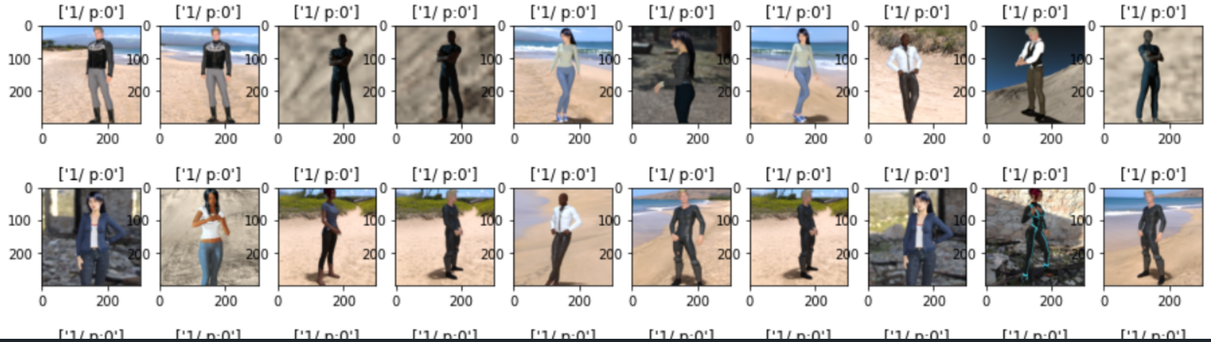
Confusion matrix, without normalization



Classification_report:

	precision	recall	f1-score	support
0	0.51	1.00	0.67	151
1	1.00	0.06	0.12	158
accuracy			0.52	309
macro avg	0.75	0.53	0.40	309
weighted avg	0.76	0.52	0.39	309

Set date/imagini eronate: (148, 300, 300, 3)

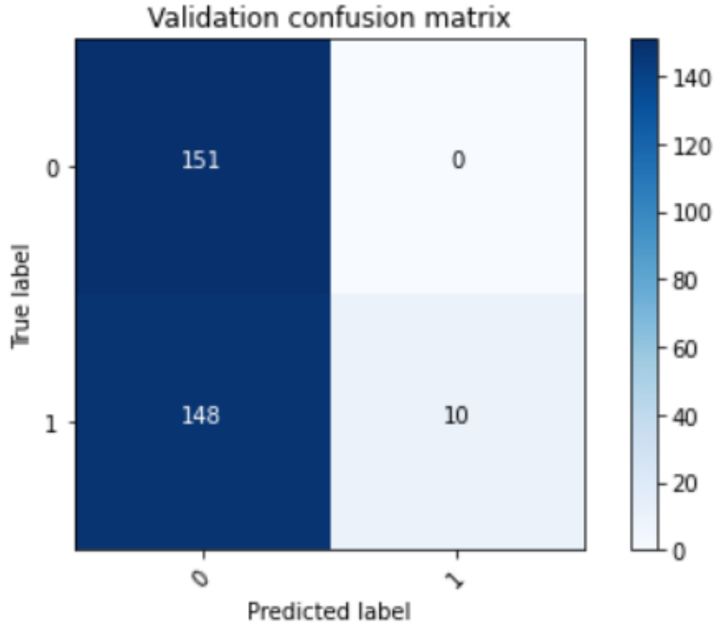


SISTEME INSTRUIBILE PEMTRU ANALIZA IMAGINILOR

GAL OSCA

RESULTS 10 EPOCH

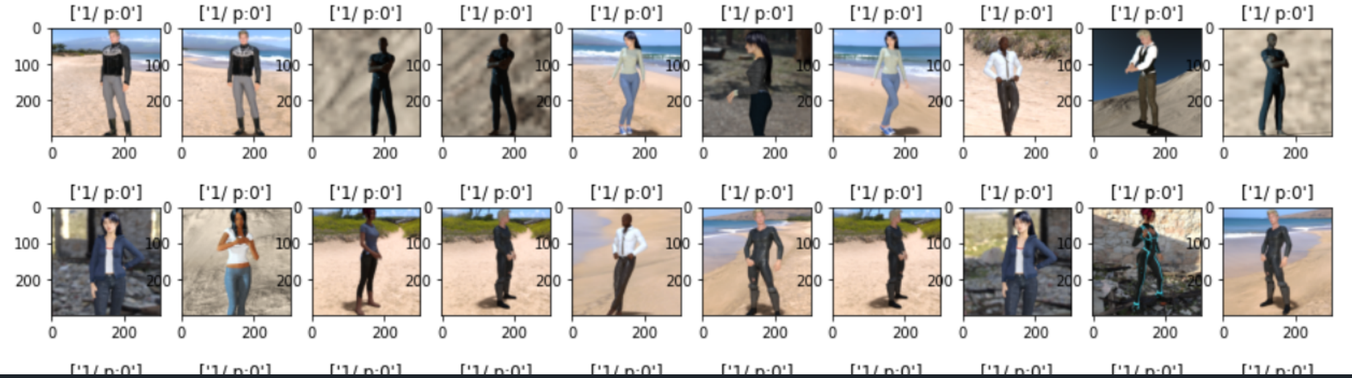
Confusion matrix, without normalization

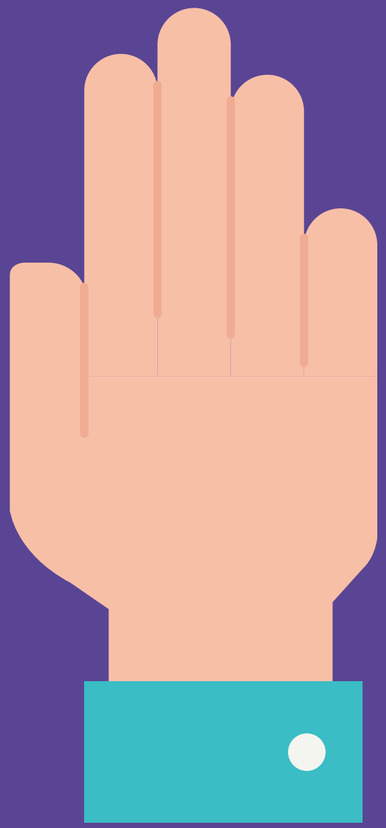


Classification_report:

	precision	recall	f1-score	support
0	0.51	1.00	0.67	151
1	1.00	0.06	0.12	158
accuracy			0.52	309
macro avg	0.75	0.53	0.40	309
weighted avg	0.76	0.52	0.39	309

Set date/imagini eronate: (148, 300, 300, 3)





CONCLUSION

The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.

CNNs are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs. Also, CNNs were developed keeping images into consideration but have achieved benchmarks in text processing too. CNNs are trained to identify the edges of objects in any image.

THANK YOU!