Capstone Project
Machine Learning Engineer Nanodegree

Aye Lu Win
September 18, 2020

Definition

Project Overview

Computer vision has been making rapid progress with the advances of deep learning neural networks. Convolutional Neural Networks (CNN) in particular has enabled many machine learning applications such as image classification, and object detection once considered intractable. Yann LeCun et al. introduced a 7-layer convolutional neural networks for classification of images of digits [1] in 1998. Alex Krizhevsky et al. in 2012 introduced AlexNet that outperformed existing model [2]. Karen Simonyan et al. in 2015 proposed a deep CNN known as VGG-16 that had simpler architecture [3]. A very deep CNN known as residual networks or ResNet was introduced by Kaiming He et al. [4] in 2015, and GoogLeNet [5] was introduced by C. Szegedy et al. in 2014.

The dog breed classification problem is well known supervised learning problem and has been studied by Jiongxin Liu et al., using part location [6]. It has also been studied using CNN for example by David Hsu [7] and Jesse Candido [8]. The datasets of images for training, validation, and testing are provided by Udacity. The dog dataset has a total of 8351 images of 133 dog breeds. The dataset is divided into train, validation, and test sets with 6680, 835, and 836 images respectively. They are color images and come in different sizes and file extensions. Udacity has also provided 13233 human images of size 250 x 250 pixels.

Problem Statement

The goal is to develop a CNN model for classification of dog breeds. A dog breed classification algorithm using the CNN model will also be developed. It can accept a user-supplied image and determine whether it is a dog or human. If a dog image is detected the app will predict its breed. If a human face is detected instead, the app will return the dog breed it most resembles.

Convolutional Neural Networks (CNN) will be used to classify dog breeds from the user input images. Since user can input any images, algorithms for detecting human faces and dogs are needed before classifying dog breeds. For face detection, a pre-trained face detector from OpenCV will be used. This face detector will be OpenCV's implementation of Haar feature-based classifiers proposed by Paul Viola and Michael Jones [9]. For dog detection, a pre-trained VGG-16 model will be used. The architecture of the model was proposed by Karen Simonyan and Andrew Zisserman [3] in 2015. The weights of the model have been trained on ImageNet. For predicting dog breed, we will build a convolutional neural networks model that has convolution layers and pooling layers. We will also use a transfer learning model with ResNet50 architecture proposed by He et al. in 2015 [4].

Metrics

Although classes are slightly imbalanced, accuracy is still a valid metrics for evaluating performance of the models for this classification problem and will therefore be used as the evaluation metrics.

accuracy = (*true positives* + *true negatives*) / *dataset size*

Analysis

Data Exploration and Visualization

The dog dataset has a total of 8351 images for 133 dog breeds. The dataset is divided into train, validation, and test sets with 6680, 835, and 836 images respectively corresponding to 80%, 10%, 10% split. They are color images and come in different sizes and file extensions. Udacity has also provided 13233 human images of size 250 x 250 pixels.

Fig. 1. Sample images from the dog dataset.

Fig.2. Sample images from human dataset.

The dog dataset can be obtained at
https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
and the human dataset at
http://vis-www.cs.umass.edu/lfw/lfw.tgz

Algorithms and Techniques

The dog breed classification algorithm first identifies whether the input image is a human, dog or neither. For human face detection, a pre-trained face detector from OpenCV's implementation of Haar feature-based classifiers, is used. For dog detection, a pre-trained convolutional neural network of architecture VGG-16 is used. Convolutional neural networks are today's standard for image classification tasks. They take advantage of parameter sharing and sparsity of connections. Networks with all fully connected layers would have infeasible large parameters to train compared to CNNs. The VGG-16 model has been trained on ImageNet dataset and dog images correspond to dictionary keys 151-268, inclusive.

Benchmark

The random chance of predicting the dog breed correctly is roughly 1 in 133 since the classes are sightly imbalanced. The accuracy for random guessing is thus less than 1%. Liu et al. in 2012 achieved 67% accuracy using part location (e.g., face and eyes) [6].

- The CNN model created from scratch with three convolution layers and pooling layers will serve as the baseline model for which the targeted accuracy is greater than 10%.
- The goal is to achieve the accuracy greater than 80%. A transfer learning model with ResNet50 architecture will be used to achieve this goal.

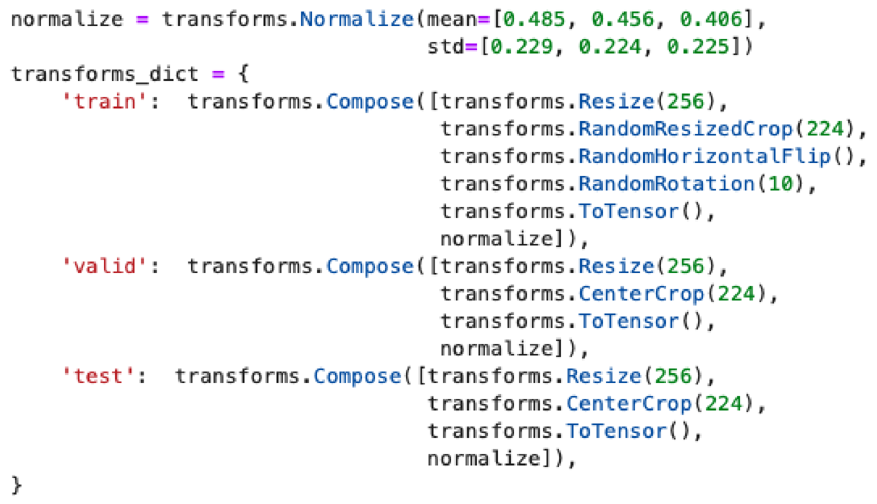
Methodology

Data Preprocessing

- The images are resized to 256 x 256 pixels and then randomly cropped to 224 x 224 pixels for the training dataset. For validation and testing, the images are cropped at center. The size of the input tensor for an image is 3 x 224 x 224. I chose this size because it is the minimum size required for pre-trained models.

- I used the data augmentation for the training dataset by random cropping to size of 224 x 224 pixels, random horizontal flipping, and random rotation of 10 degree.

- After images are converted to tensors, they are normalized using standard parameters.



```
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
transforms_dict = {
    'train':  transforms.Compose([transforms.Resize(256),
                                  transforms.RandomResizedCrop(224),
                                  transforms.RandomHorizontalFlip(),
                                  transforms.RandomRotation(10),
                                  transforms.ToTensor(),
                                  normalize]),
    'valid':  transforms.Compose([transforms.Resize(256),
                                  transforms.CenterCrop(224),
                                  transforms.ToTensor(),
                                  normalize]),
    'test':  transforms.Compose([transforms.Resize(256),
                                 transforms.CenterCrop(224),
                                 transforms.ToTensor(),
                                 normalize]),
}
```

Fig. 3. Data preprocessing.

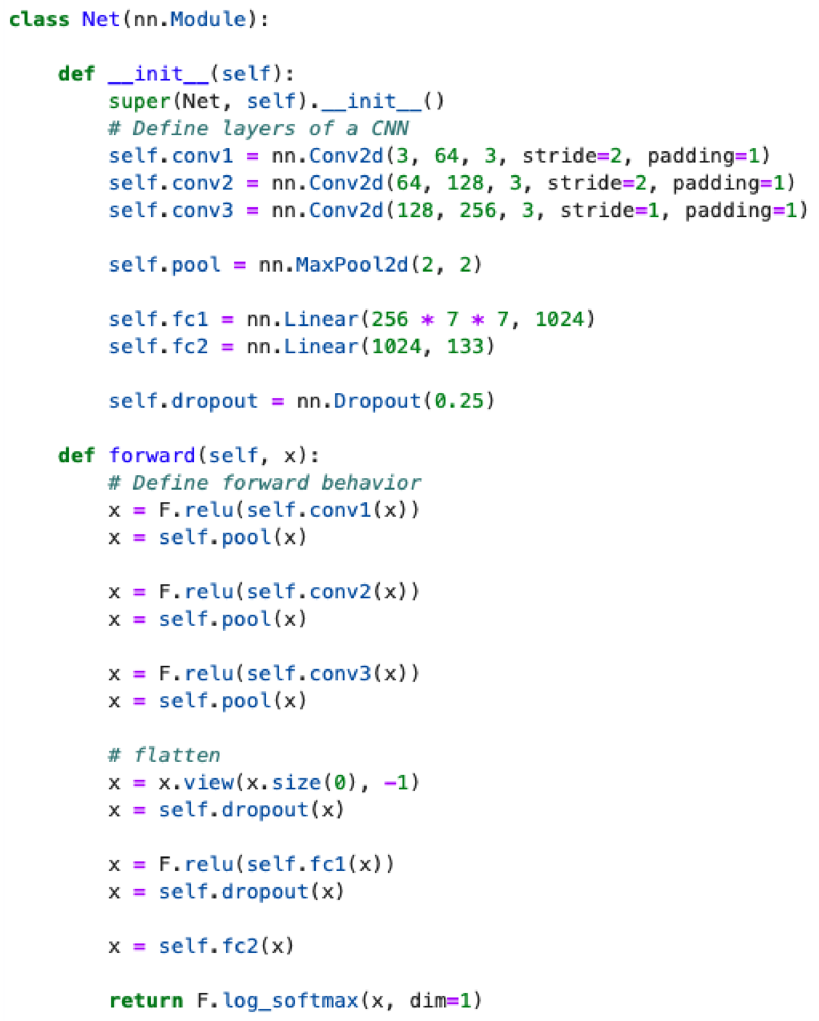
Implementation

The workflow of the project is as follows:

1. Import dog and human image datasets.
2. Detect human faces using OpenCV's implementation of Haar feature-base cascade classifiers and assess the performance of the face detector.
3. Detect dogs using pre-trained VGG-16 models whose weights have already been trained on ImageNet and assess the performance of the detector.
4. Preprocess train, validation and test datasets with necessary transformations such as resizing, random rotation, and converting to tensors.
5. Create a CNN model from scratch with three convolution layers and pooling layers, train the model and evaluate its performance.

6. Create a ResNet50 model using transfer learning, train the model and evaluate its performance.
7. Write a dog breed classification algorithm that accepts a user input image and determines whether it contains a human, dog or neither. If a dog is detected, its predicted breed is returned. If human face is detected, the dog breed it most resembles is returned. If neither is detected, and error message is returned.
8. Test the dog breed classification algorithm with different user input images.

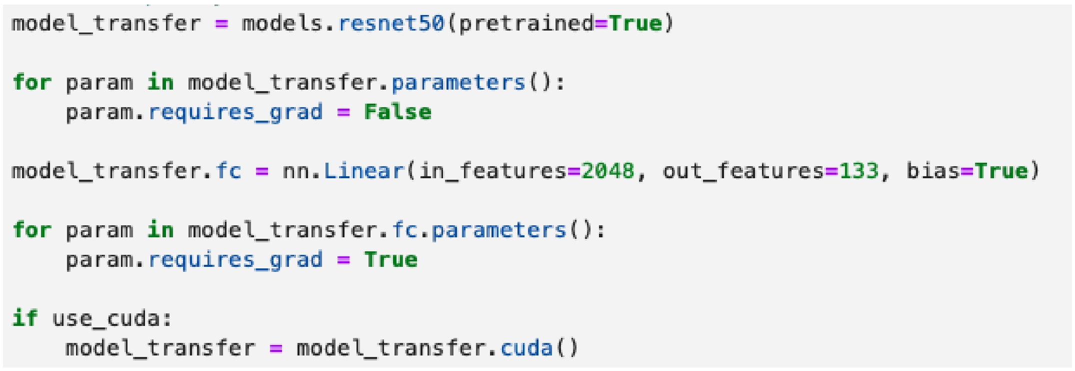
To achieve the target accuracy of greater than 10% with the CNN model created from scratch, I chose a relatively simple 3-layer CNN with rectified linear units (ReLu) as the activation function. I used 3 x 3 kernels and padding of 1 for all convolution layers. The first two layers used stride of 2 and the last layer used stride of 1. The number of kernels for each layer are 64, 128 and 256 respectively. Each convolutional layer is followed by a 2 x 2 pooling layer to further reduce the size of representation. The convolution layers are followed by a fully connected layer, a dropout layer with probability of 0.25 to prevent overfitting, and a fully connected output layer with log_softmax activation function to get the probabilities of each of the 133 dog breeds.



```
class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # Define layers of a CNN
        self.conv1 = nn.Conv2d(3, 64, 3, stride=2, padding=1)
        self.conv2 = nn.Conv2d(64, 128, 3, stride=2, padding=1)
        self.conv3 = nn.Conv2d(128, 256, 3, stride=1, padding=1)

        self.pool = nn.MaxPool2d(2, 2)

        self.fc1 = nn.Linear(256 * 7 * 7, 1024)
        self.fc2 = nn.Linear(1024, 133)

        self.dropout = nn.Dropout(0.25)

    def forward(self, x):
        # Define forward behavior
        x = F.relu(self.conv1(x))
        x = self.pool(x)

        x = F.relu(self.conv2(x))
        x = self.pool(x)

        x = F.relu(self.conv3(x))
        x = self.pool(x)

        # flatten
        x = x.view(x.size(0), -1)
        x = self.dropout(x)

        x = F.relu(self.fc1(x))
        x = self.dropout(x)

        x = self.fc2(x)

        return F.log_softmax(x, dim=1)
```

Fig. 4. CNN model architecture created from scratch.

The accuracy of greater than 80% is desired for this project. To achieve this, a transfer learning model with ResNet50 architecture was chosen. This 50-layer convolutional neural network has been pre-trained on more than a million images from the ImageNet dataset. All of its pre-trained parameters were frozen except for the parameters in the last fully connected layer. The number of units in this output layer was change to 133 to match the number of dog breeds. The ResNet architecture utilizes skipped connections which allow for having deeper network. We also have the advantage of this network having been trained on millions of images.

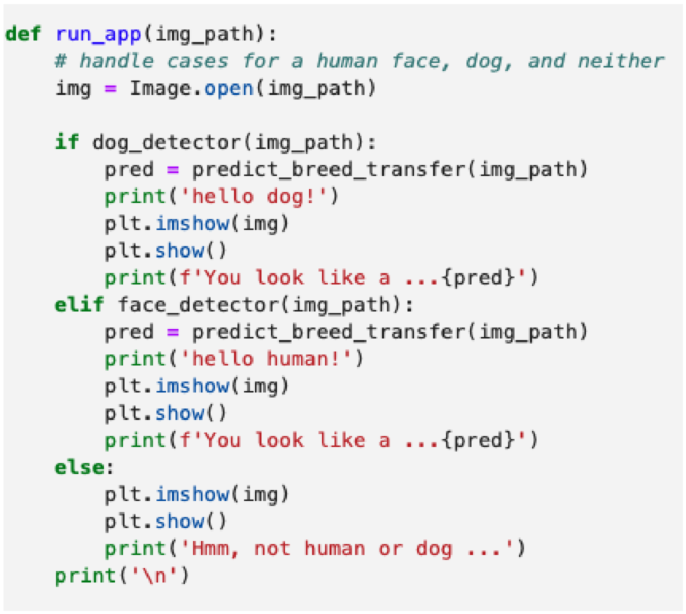


```
model_transfer = models.resnet50(pretrained=True)

for param in model_transfer.parameters():
    param.requires_grad = False

model_transfer.fc = nn.Linear(in_features=2048, out_features=133, bias=True)

for param in model_transfer.fc.parameters():
    param.requires_grad = True

if use_cuda:
    model_transfer = model_transfer.cuda()
```

Fig. 5. Transfer model with ResNet50 architecture.

For both CNN models, Cross Entropy was used as the loss function for this multiclassification problem. Optimizers for performing gradient decent for the CNN model from scratch was SGD and for the transfer model was Adam. The models were trained and evaluated.

Finally, the dog breed classification algorithm was put together using face detector, dog detector and the ResNet50 transfer model.



```
def run_app(img_path):
    # handle cases for a human face, dog, and neither
    img = Image.open(img_path)

    if dog_detector(img_path):
        pred = predict_breed_transfer(img_path)
        print('hello dog!')
        plt.imshow(img)
        plt.show()
        print(f'You look like a ...{pred}')
    elif face_detector(img_path):
        pred = predict_breed_transfer(img_path)
        print('hello human!')
        plt.imshow(img)
        plt.show()
        print(f'You look like a ...{pred}')
    else:
        plt.imshow(img)
        plt.show()
        print('Hmm, not human or dog ...')
    print('\n')
```

Fig. 6. Dog classification algorithm.

Refinement

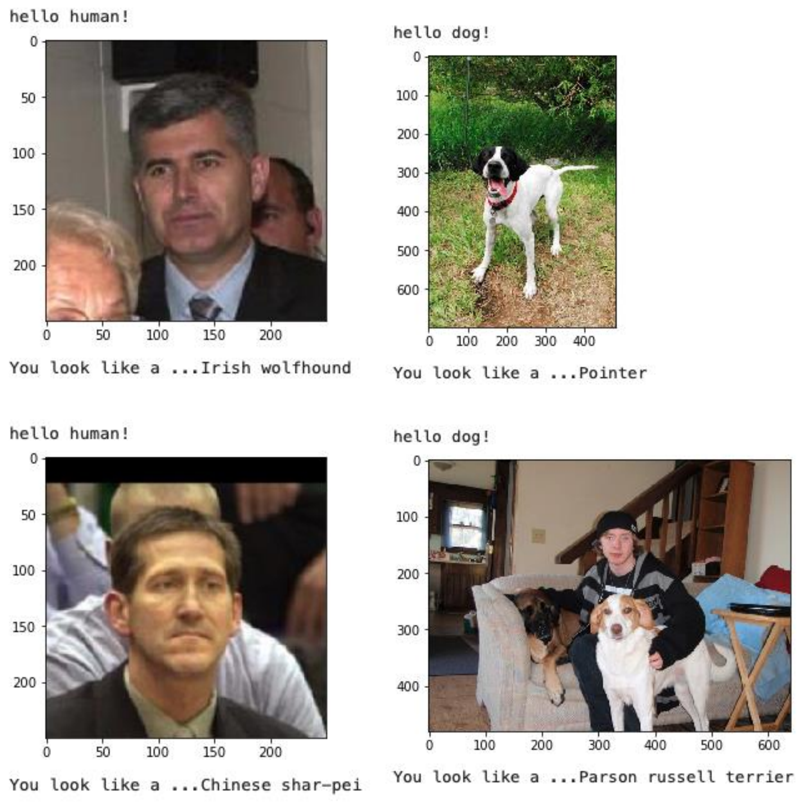
For the CNN model created from scratch, initially 32 filters were used in the first convolution layer and the first fully connected layer has 500 units. The accuracy with the initial model was 16%. To further improve the accuracy, the number of filters was increased to 64 in the first convolution layer and 1024 units in the first fully connected layer. The accuracy of 22% was achieved with the final CNN model created from scratch. The transfer model with ResNet50 architecture accuracy of 84% was achieved using Adam optimizer with the learning rate of 0.001.

Results

Model Evaluation and Validation

The CNN model created from scratch was trained for 30 epochs. The best cross entropy loss for the validation dataset is 3.46. The model achieved the accuracy of 22% (187/836) on the test dataset.

The transfer learning model with ResNet50 architecture was trained for 20 epochs. The best cross entropy loss for the validation dataset is 0.52. The model achieved the accuracy of 84% (703/836).



```
hello human!
You look like a ...Irish wolfhound

hello dog!
You look like a ...Pointer

hello human!
You look like a ...Chinese shar-pei

hello dog!
You look like a ...Parson russell terrier
```

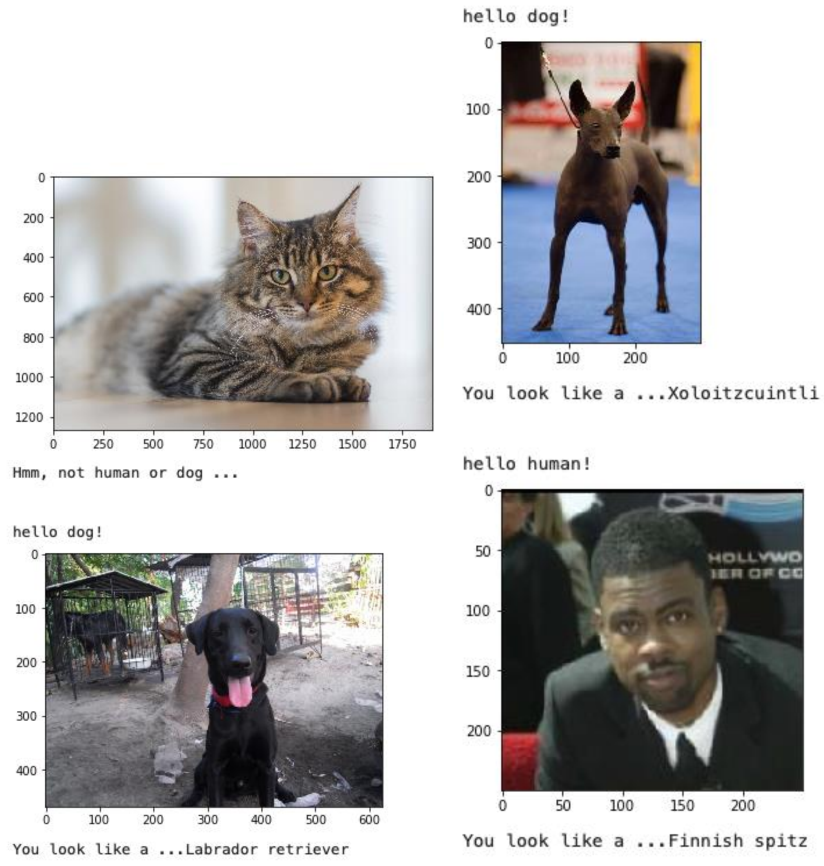
Fig. 7. Sample outputs of dog breed classifier.

Justification

The CNN model from scratch had the accuracy of 22%. While it performed a lot better than random guessing, the accuracy was not suitable for the application. The transfer learning model with ResNet50 architecture, on the other hand, had the accuracy of 84% which is sufficient high for using it in light applications.

Conclusion

Free-Form Visualization



```
hello dog!
You look like a ...Xoloitzcuintli

Hmm, not human or dog ...

hello human!
You look like a ...Finnish spitz

hello dog!
You look like a ...Labrador retriever
```

Fig. 8. Sample outputs of the dog breed classifier.

Reflection

In this project, a dog breed classification algorithm was developed using pre-trained VGG-16 dog classifier, Haar feature-based face detector, and convolutional neural networks (CNN) models. The 3-convolution layer CNN model created from scratch achieved the accuracy of 22% while the transfer learning model with ResNet50 architecture achieved far better accuracy of 84%. This advantage over the model from scratch, however, was expected since the transfer model utilized parameters trained on more than a million images.

Improvement

- Include a functionality to handle the case when both human and dog are seen in the image.
- Tune the model for example by including additional dropout layer and fully connected layers.
- Expand the dataset to include more dog breeds to train the network so that it will be able to classify more than 133 breeds.
- Experiment with different architectures such as GoogLeNet, ResNet101, etc.

References

[1]. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, Vol. 86 (1998).

[2]. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Network", NIPS (2012).

[3]. Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition", ICLR (2015).

[4]. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385, (2015).

[5]. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions", arXiv:1409.4842, (2014).

[6]. Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur, "Dog Breed Classification Using Part Location", European Conference on Computer Vision, Springer, Berlin, Heidelberg, 2012.

[7] David Hsu, "Using Convolutional Neural Networks to Classify Dog Breed", http://cs231n.stanford.edu/reports/2015/pdfs/fcdh_FinalReport.pdf (2015).

[8] Jesse Candido, "Dog Breed Classification and Visualization", http://cs230.stanford.edu/projects_spring_2018/reports/8291007.pdf (2018).

[9]. Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", Conference on Computer Vision and Pattern Recognition (2001).