Traffic Sign Recognition

Faculty of Computers & Artificial Intelligence

Selected 2

Repository Link: https://github.com/mahmoudhaney/TrafficSignRecognition

Team 1

	ID	NAME	LEVEL	Department
1	202000186	آية عماد طلعت حرب	3	CS
2	202000376	ساره وليد سيد	3	CS
3	202000649	فيروز جمعه منصور محمد	3	CS
4	202000861	محمود هاني سعيد	3	CS
5	202000895	مصطفى جمال عبد الحكيم	3	CS
6	202001010	نورهان طارق عبدالهادي	3	CS

Paper Details

Name

- Autonomous Traffic Sign (ATSR) Detection and Recognition using Deep CNN

Authors

- Danyah A. Alghmghama
- Ghazanfar Latif
- Jaafar Alghazo
- Loay Alzubaidi

Publisher

- Procedia Computer Science

Publication Year

- Jan 2019

Link

- The Paper

Dataset Used

- 2718 images were collected to form the database which we named Saudi Arabian Traffic and Road Signs (SA-TRS-2018)

Result

- 100% Accuracy

Algorithm Used

Abstract

Automatic detection and recognition of traffic signs is very important and could potentially be used for driver assistance to reduce accidents and eventually in driverless automobiles. In this paper, Deep Convolutional Neural Network (CNN) is used to develop an Autonomous Traffic and Road Sign (ATRS) detection and recognition system.

The proposed system works in real time detecting and recognizing traffic sign images. The contribution of this paper is also a newly developed database of 24 different traffic signs collected from random roadsides in Saudi Arabia. The images were taken from different angles and included other parameters and conditions.

A total of 2718 images were collected to form the database which we named Saudi Arabian Traffic and Road Signs (SA TRS-2018). The CNN architecture was used with varying parameters to achieve the best recognition rates. Experimental results show that the proposed CNN architecture achieved an accuracy of 100%, thus higher than those achieved in similar previous studies.

Architecture

The contribution of this paper will be two folds; one is to develop a new database for Arabic Traffic and Road Signs and the other is to develop and design a deep CNN architecture for Arabic Traffic sign recognition. Fig. 1 shows the high-level view of the system. The collected data set is given as an input to the proposed CNN architecture for training, validation, and testing. A detailed explanation of the CNN architecture is provided in the next section.

Once CNN is trained, it is ready to be used for classifying new images which were not part of the collected dataset. The AATS system depends on a group of standard Arabic Traffic signs. In recent years, a number of authors have done work in this field but to the author's knowledge, this is the first time a complete database is developed for Arabic Traffic.

A Deep CNN architecture is also proposed for Arabic traffic sign recognition Generally, CNNs consist of multiple hidden layers between the input and output layers [13]. The design of the proposed CNN is implemented using Python.

Proposed System

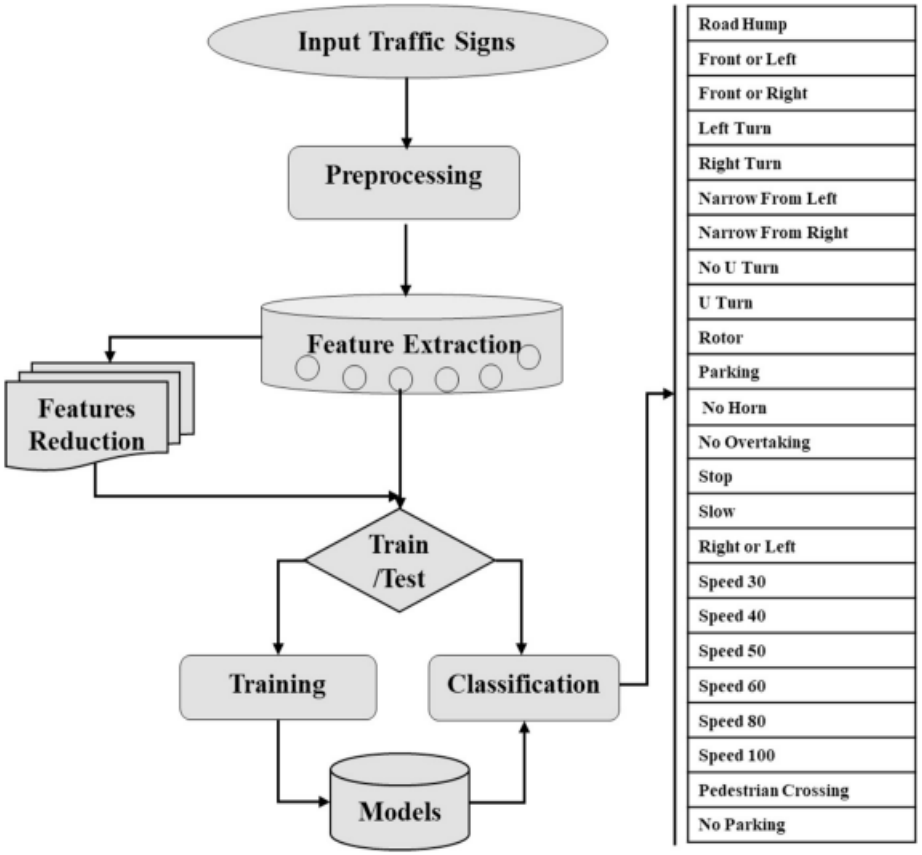


Fig. 1 . Block diagram of supervised learning

Experimental Data

- In this research, a new dataset for Arabic Traffic Signs is developed. As part of this research, 24 most common.

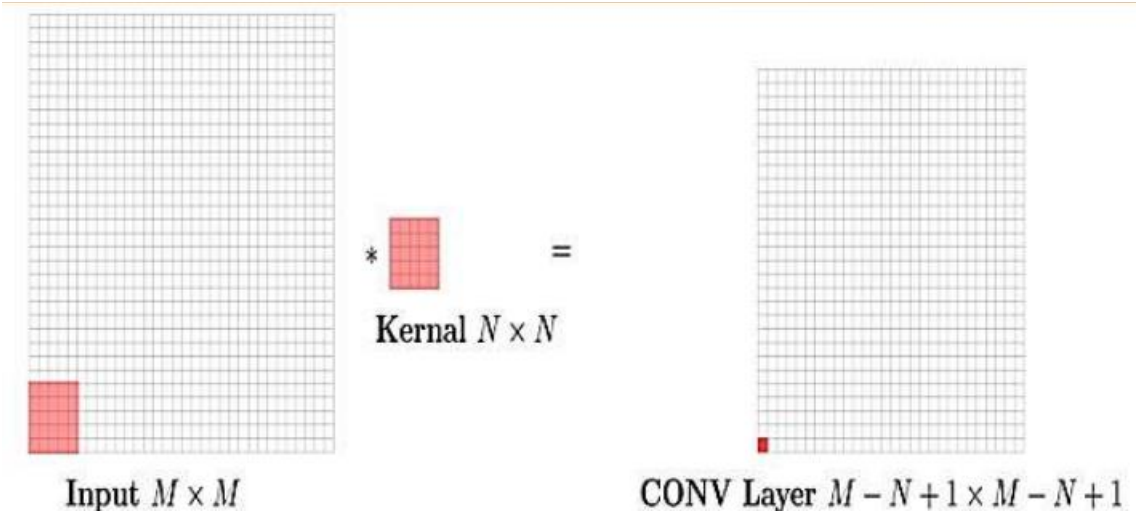
- Arabic traffic signs are selected. The dataset consists of 2,728 images captured for 24 traffic signs.

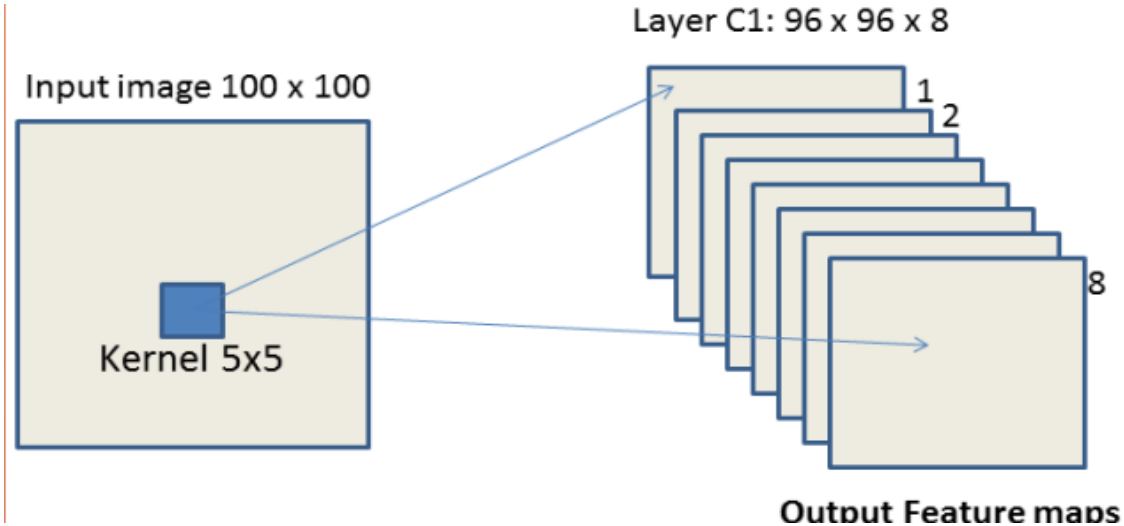
- The images are captured from three connected cities (Khobar, Dammam, and Dhahran) in the Eastern Province of Saudi Arabia. For each traffic sign, sign boards ranging from 20 to 30 location were found and for each location.

- 5 photos were captured with different angles and distances. Table 1 describes the list of all the selected traffic signs visual shape.

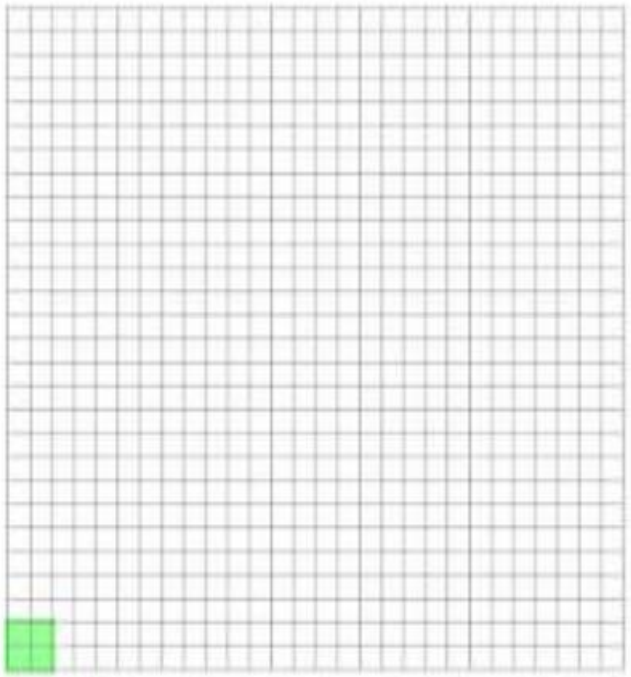
- Their names in English and Arabic and the total number of images captured for each sign as part of the new database.

Convolutional Neural Network



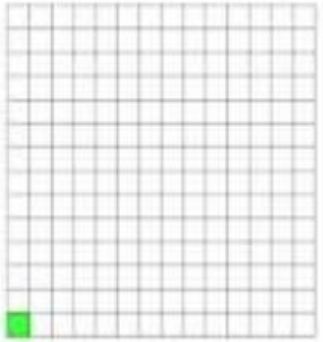


Pooling Layer



Feature map

$M - N + 1 \times M - N + 1$



POOL Layer

$\frac{M-N+1}{2} \times \frac{M-N+1}{2}$

CNN Architecture

The proper choice of different design hyper-parameters, such as non-linearity and pooling variants, directly affects the performance of the network. Since there is no clear guidance on how to choose the CNN hyper parameters, many researchers tend to use trial and error experimentation to discover good settings [15]. We analyzed previous research studies that have been carried out in deep learning generally to choose the hyper-parameters setting that is more likely suitable for this research. So, by analyzing the work done in [16-18], the hyper-parameters setting in this work is as follows. Pooling Layer: It helps in reducing the number of computations needed in the training process by reducing the dimension of the image, and therefore the overfitting problem is reduced. The max-pooling technique was used since the convergence rate is faster as compared to other subsampling techniques, and thus has a better generalization performance. The maximum pixel value in a non-overlapping region, equal to the window of the pooling, is the output of this layer; this is beneficial in creating position invariance. Non-Linearity: Since the relation between the images and their classes is not linear, introducing non-linearity in the CNN is needed. This is achieved by using non-linear activation so that the construction of the non-linear relation between the images and their classes is possible by the CNN. The Rectified Linear Unit (ReLU) is a widely used activation function in CNN.

ReLU has the advantage that CNN trains faster as compared to other functions. One of the ReLU variants was used which is the leaky ReLU since it overcomes the problem of dead neurons that is faced if the original ReLU is used. Basically, leaky ReLU does not output zero when the input values are less than zero, instead it outputs negative value. So, after each convolution layer, and before the pooling layer, we added a leaky ReLU activation function.

Dense Layer Activation Function and Loss Function: As mentioned above, we used the leaky ReLU activation function multiple times in the proposed network. However, the output of the fully connected layer, which is one dimensional vector, is passed to a SoftMax activation function to predict the label of that input. The input vector is transformed to a vector of the same size, but the values range from zero to one only and the summation is always equal to 1. SoftMax function outputs a

probability distribution that is then converted to one-hot encoding vector. So, the element that has the largest portion of probability distribution will have the value one in the one-hot vector and all other elements will have the value zero. To estimate the loss of SoftMax, the categorical cross-entropy function is used. Basically, categorical cross-entropy is used to measure the difference between the output of the SoftMax function and the one-hot encoding of the actual class.

It is used as a part of gradient descent to evaluate CNN performance as an error measure between the true distribution and the predicted one. SoftMax function and categorical cross-entropy are widely used in computer vision tasks when multiple classes are involved. After setting the needed hyper-parameters, the network was compiled using the Adam optimizer. The need of using optimization algorithms is the nature that CNN weights and biases are set automatically, and they are of great importance in the field of machine learning. They work on optimizing a given function; whether maximizing or minimizing it with respect to its parameters. Since the loss function in CNN is differentiable with respect to its parameters, gradient descent-based algorithms are often used; first order partial derivatives are also fast to compute.

Adam is an optimization algorithm that is used to update network weights. It combines the advantages of other classical stochastic gradient descent which are Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). It is computationally efficient, requires less memory and has straightforward implementations. Given different parameters, this algorithm computes each parameter's adaptive learning rate by estimating the gradient's first and second moments [19]. Table 1 shows the summary of the parameters in each layer as well as the total parameters in the proposed network. This CNN network will be referred to as Model 1 in the results section.

Experiments and Results

The improved optimized CNN network layers are shown in Table 3. As observed in the optimized CNN architecture, the number of layers has been reduced without compromising the accuracy of the network. This CNN will be referred to as Model 2. Table 3 shows the experimental results performed on the

proposed CNN architecture. The experiments were performed on epochs size of 10, 50, 100, and 150. For each epoch, the batch sizes of 50,100, 200 and 400 are used.

The table for the experiment shows four columns with Validation Accuracy (Val. Acc), Validation Loss (Val. Loss), Test Accuracy (Test Acc) and finally Test Loss (Test Loss). Different design parameters were changed, and the effect of the changes was tested by evaluating accuracy. A total of 16 different experiments were performed on the CNN architecture to identify an optimized design. Note that all the preprocessing steps remain the same as described above. The proposed CNN models are shown in Table 1 and Table 3. Experimental results show that the number of epochs and number of batch size directly affects the test accuracies achieved for the models. The number of epochs is directly proportional to test accuracy; increasing epoch size increases the test accuracy.

While the opposite is true for batch size as it is observed that increasing batch size decreases test accuracy. Overall, the test accuracy obtained for the improved CNN network-model 2 is better than those achieved for the initial CNN-model 1. In fact, model 2 results for epoch 150 achieved 100% accuracy for all batch sizes. The result is better than those reported in recent literature, however, it is pointed out here that the CNN is applied for Arabic Traffic Sign dataset and no recent literature was found targeting Arabic Traffic sign detection and recognition.

Conclusion

Automatic Arabic Traffic Sign (AATS) recognition system was designed using Convolutional Neural Networks (CNN). The training and testing dataset consists of more than 2,728 sign samples collected for 24 different traffic signs. The dataset went through a preprocessing stage before inputting it to the network. It got partitioned into training, testing and validating datasets.

The initial design was based on similar previous work which was used as a base for the subsequent improved design. The final Deep CNN architecture proposed in this work consists of two convolutional layers, two maxpooling layers one dropout layer and 3 dense layers. 100% accuracy was obtained for epoch 150 for all batch sizes. We demonstrated the usefulness of the designed CNN architecture by implementing a practical system that can take real time Traffic Sign via camera attached to vehicle, classifies them to the corresponding sign, and then give voice notification to driver or take automatic decision for autonomous cars.

Future work will include increasing the size of the dataset and publishing it so that it can be used by other researchers for benchmarking purposes. Work will also continue developing more robust and computationally low-cost recognition systems.

Our Dataset Information

GTSRB - German Traffic Sign Recognition Benchmark

Total Dataset

- **39209**

Training Data

- **31367**

Validation Data

- **7842**

Number of classes

- **43**

Labels

- **From 0 To 42**

Size of images

- **32 * 32**

Implementation

CNN Model

- 4 convolutional layers
- 2 MAX pooling layers
- 1 fully connected layer

Split dataset into

- 31,367 training data
- 7842 testing data
- 0.1(1000) of the testing for validation

Hyperparameters

With Accuracy 92%

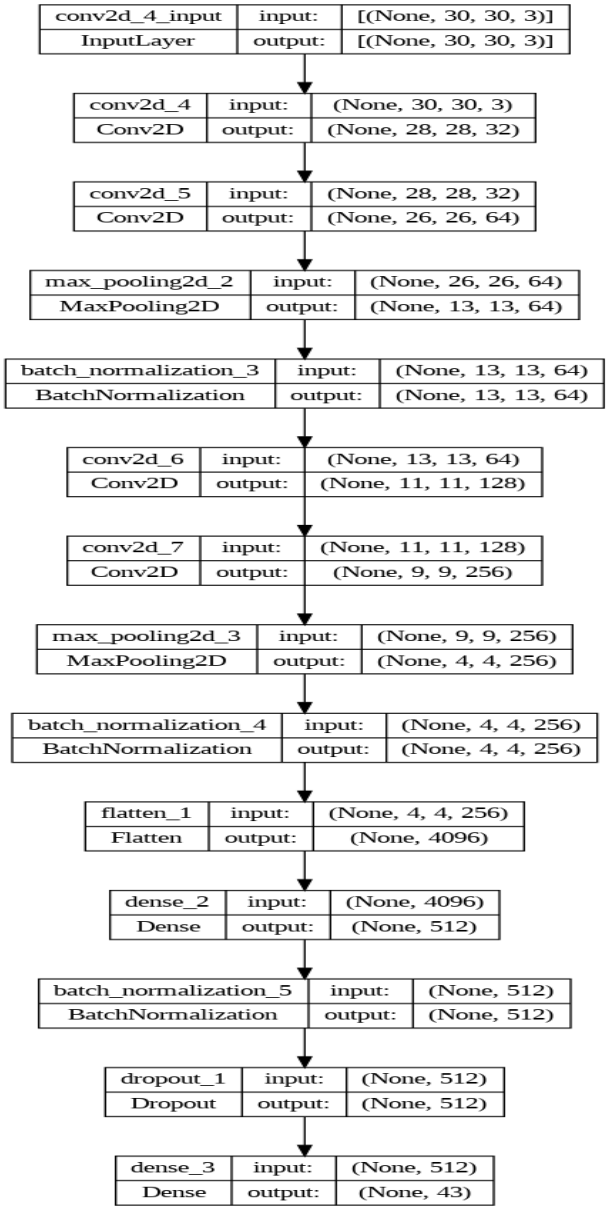
- Test_Size = 0.2
- Batch_size=32
- Kernal_Size = 3 * 3
- Pool_Size = 2 * 2
- Epochs = 30
- Optimizer = Adam
- First convolutional layer filters = 32
- Second convolutional layer filters = 32
- Third convolutional layer filters = 64
- Fourth convolutional layer filters = 64
- Dropout = 0.25 : 0.5
- Nodes in dense = 256
- Learning rate (Lr) = Default

With Accuracy 98%

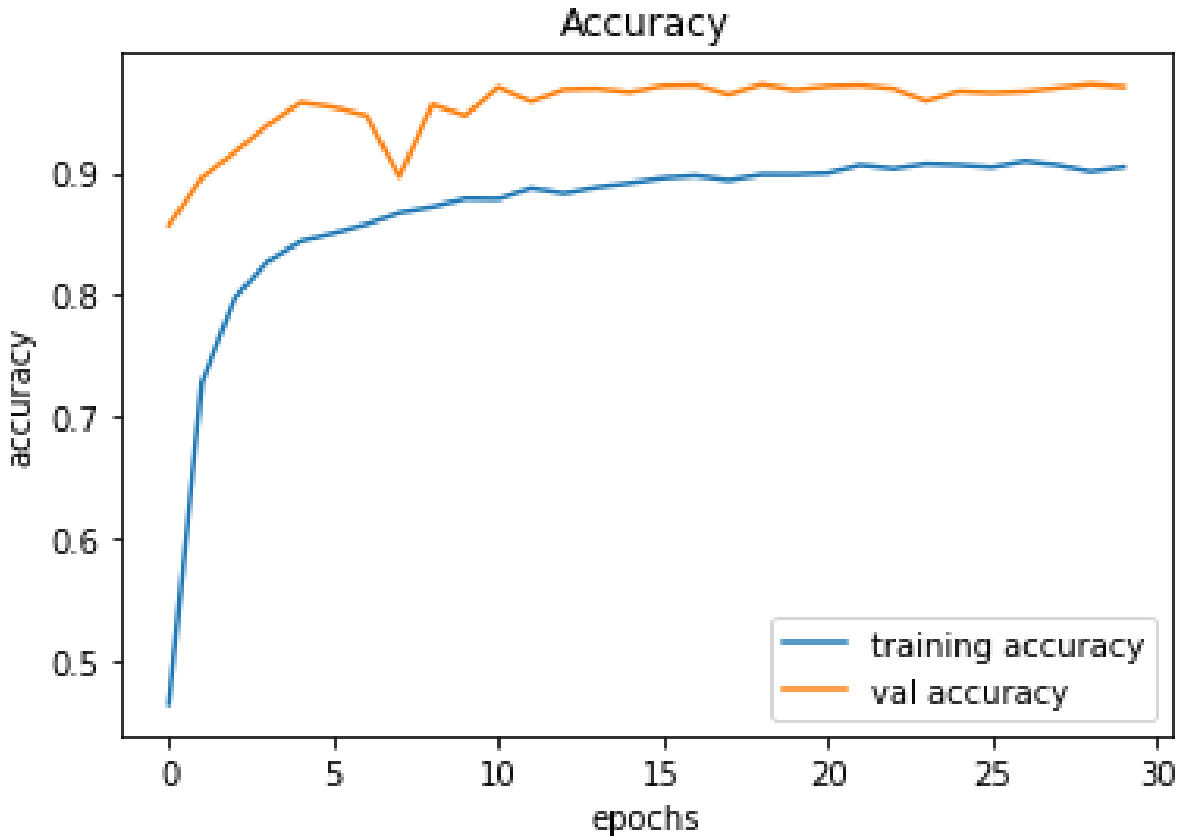
- Test_Size = 0.2
- Batch_size=32
- Kernal_Size = 3*3
- Pool_Size = 2 * 2
- Epochs = 30
- Optimizer = Adam
- First convolutional layer filters = 32
- Second convolutional layer filters = 64
- Third convolutional layer filters = 128
- Fourth convolutional layer filters = 256
- Dropout = 0.5
- Nodes in dense = 512
- Learning rate (Lr) = 0.001
- Image Augmentation

Results & Visualizations

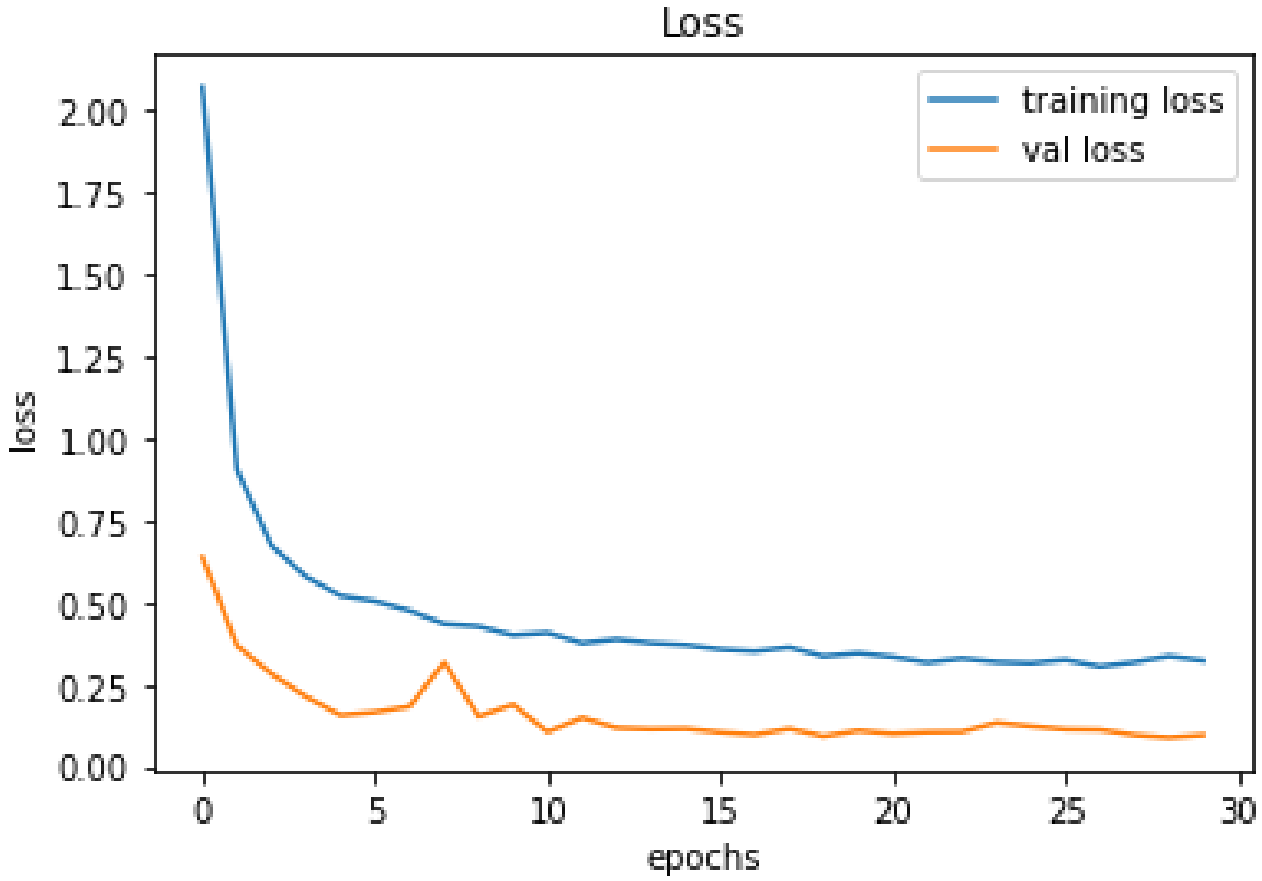
Model Description



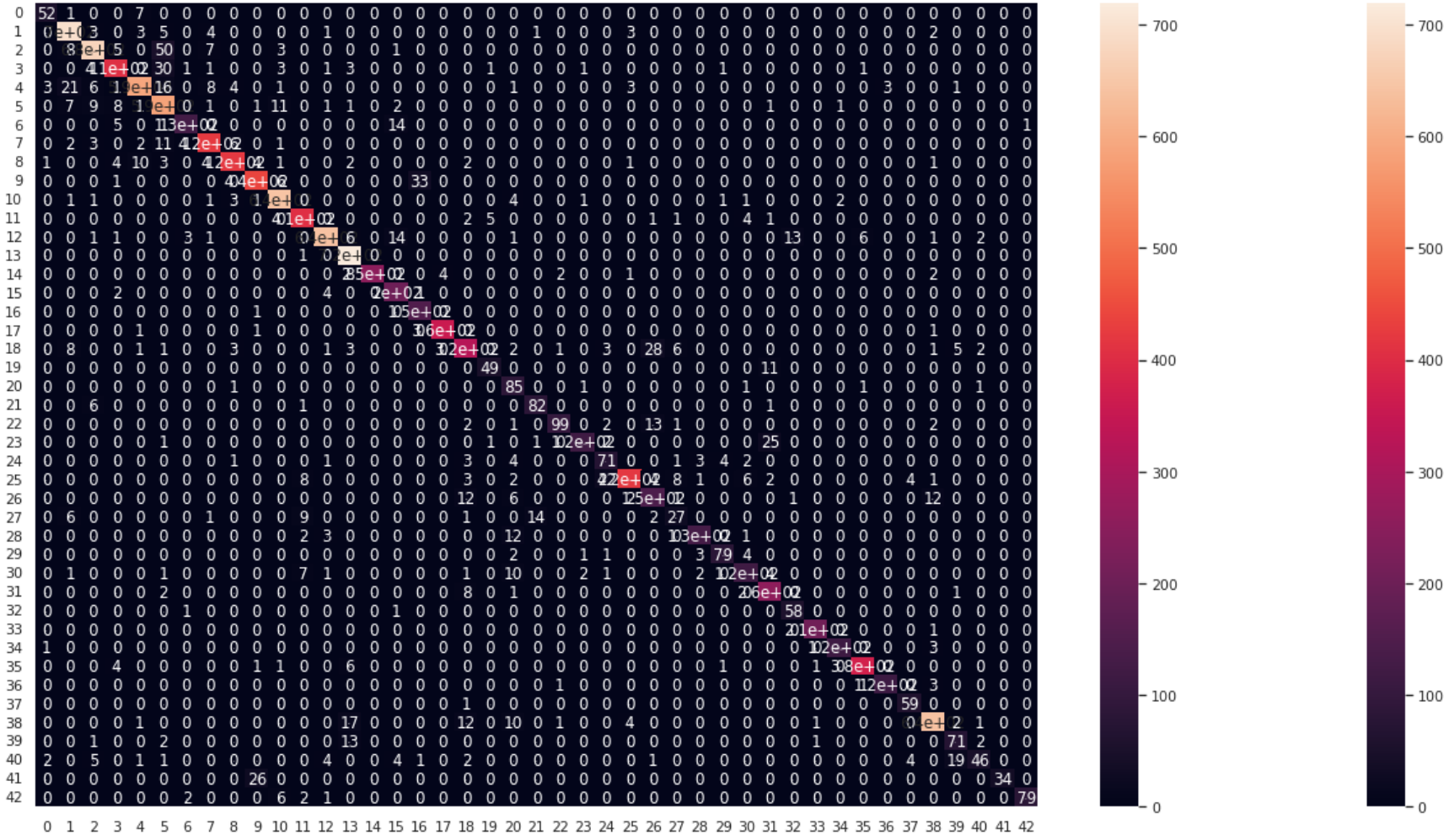
Accuracy Curve



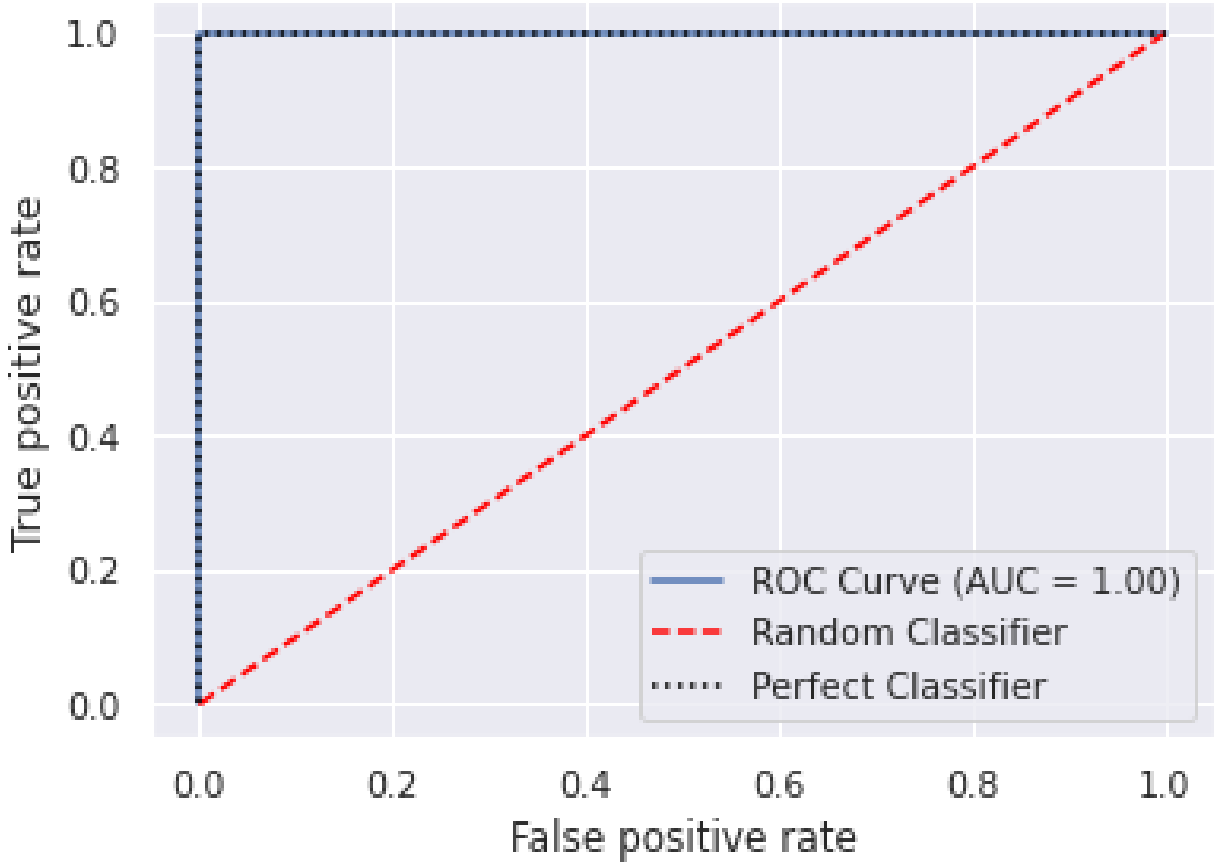
Loss Curve


Loss
training loss
val loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
loss
0
5
10
15
20
25
30
epochs

Confusion Matrix



ROC Curve


1.0
0.8
0.6
0.4
0.2
0.0
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
ROC Curve (AUC = 1.00)
Random Classifier
Perfect Classifier

Layered View