Master ST Computer Science
Deep Learning in Computer Vision

University Bordeaux
Faculty S&T
academic year 2019 – 2020

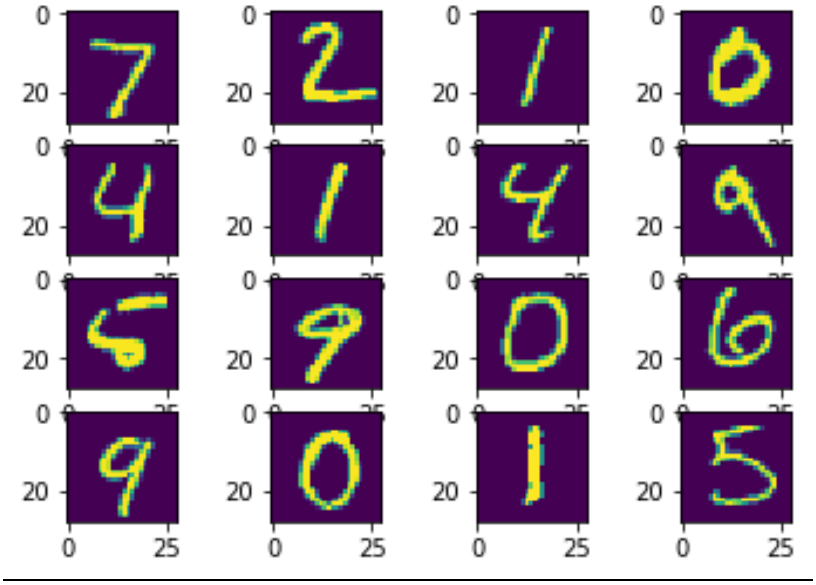
Report on Lab 4
Neural Networks
Implementing a Convolutional Neural Network with Keras

In this lab, we built a convolutional neural network with the Keras framework in Python for the datasets MNIST and CIFAR10.

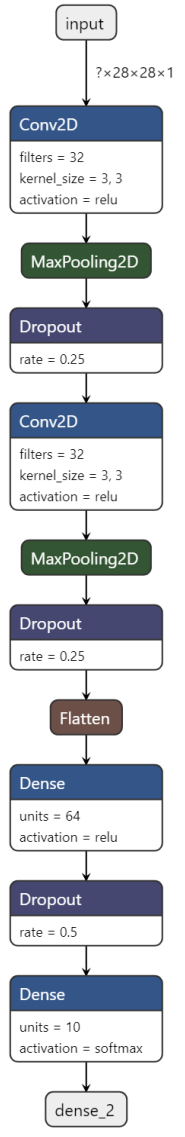
1. Convolutional Neural Network on **MNIST** dataset

Architectural Model

- The best architectural model and its output for MINST dataset which we have obtained is shown in Figure 1.
- Here, the first Conv2D layer takes the low level features like edge, point and the second Conv2D layer considers the higher level feature like structure of the object in an image.
- Generally, pooling region is doing the down sampling of the convolutional layer output and MaxPooling2D is taking the maximum pooling in the pooling regions.
- Dropout is a regularization method which is dropping the units of the layer in a rate of 0.25.
- Flatter layer is converting the feature matrix to feature vector.
- Then the first Dense layer is putting the relu activation function in the hidden layer whereas second Dense layer is putting the softmax activation function in the output layer to recognize the 10 digits.



(a) First 16 input training images

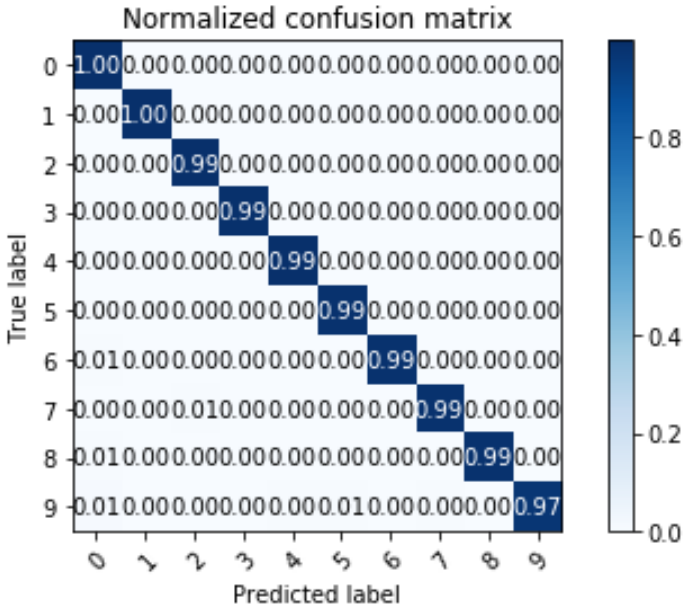


(b) Architectural Model

Fig. 1. Training images and Architectural Model of Convolutional Neural Network for MINST dataset

Analysis of Confusion Matrix

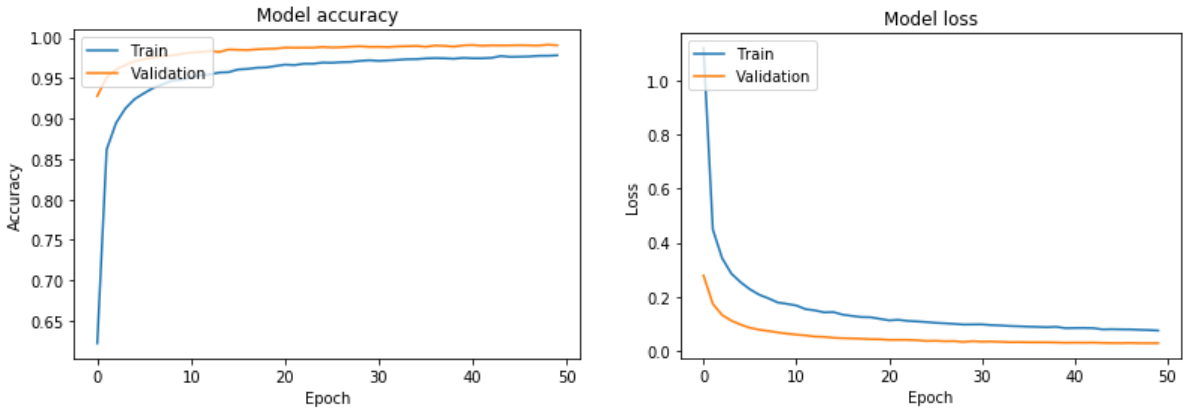
The confusion matrix for the convolutional neural network of MINST image dataset is shown below. Here it can be seen that the digits (0-8) can be recognized by the network efficiently providing on average 99% accuracy. Only for digit 9, the network provides 97% accuracy.



Analysis of Model accuracy and Model loss curve

As MNIST is a small dataset we considered the whole training dataset for training the network and the test dataset for validation purpose. It can be noticed from both in the accuracy and loss curve that the marginal difference between the training data and validation data is very small for which this convolutional neural network cannot be considered neither overfitting nor under fitting. Rather it is a well fitted model in our observation.

Here, the final accuracy on test set is 99.05%.



(a) (b)

Fig.2. (a) Plot of the accuracy for Convolutional Neural Network of MNIST dataset

(b) Plot of the loss function for Convolutional Neural Network of MNIST dataset

Analysis of 10 worst classified images:

Fig 3 shows the True label and the Predicted Label of the digits which are wrongly classified by the convolutional neural network. From the confusion matrix this wrong classification can be explained. For example:

The confusion matrix shows that true label of 6 is misclassified as 0 for which result 0.01 is coming.

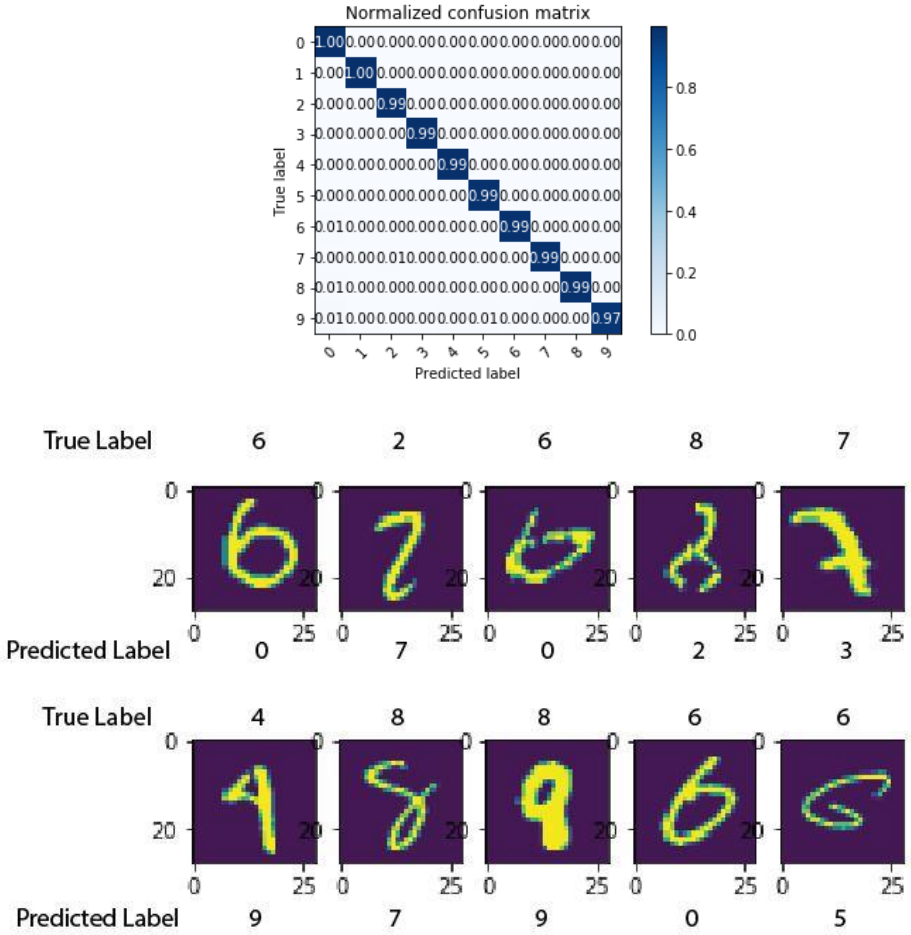


Fig 3. 10 worst classified images by the convolutional network indicating the True Label and the Predicted Label

2. Convolutional Neural Network on **CIFAR10** dataset

CIFAR10 dataset consists of color images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Program a convolutional neural network to recognize the 10 classes. It has total 50000 instances and 10000 instances in training and test set respectively. Each image is 32x32 pixel color image in which the class object exists in a colorful background which makes the classification a challenging task even for human.

To classify the classes with convolutional Neural Networks, at first, we tried the network that we developed for MNIST dataset. At first, we plotted first 16 images of different classes. The images are given in the figure 4. From the figure, we can see that the instances of the CIFAR10 dataset is quite difficult to classify due to very low resolution and background.

As we don't have a big training dataset in CIFAR10, we have used the whole training set for training the network and test set as our validation set.

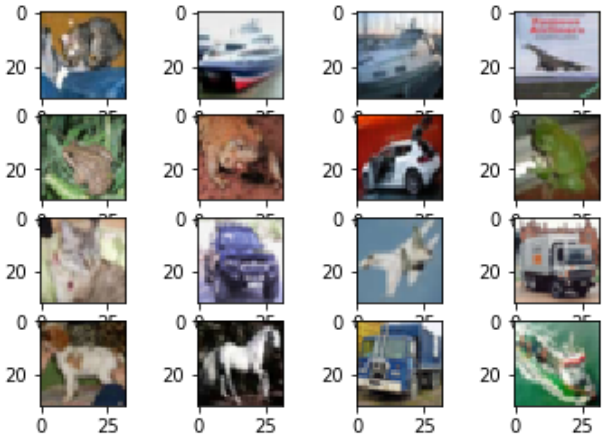
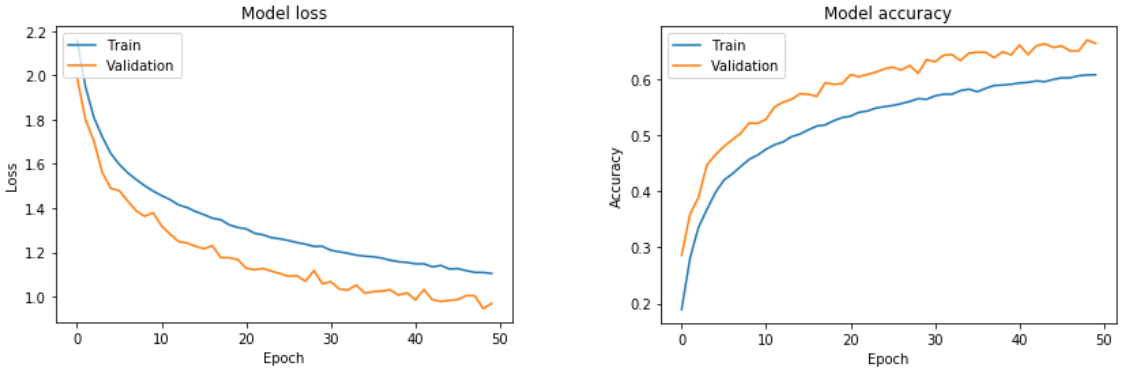


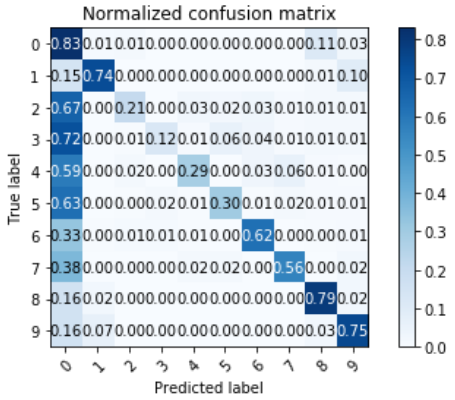
Figure 4: First 16 instances of Training Dataset.



a. Model Loss vs Epoch b. Model Accuracy vs Epoch

Figure 5: Model loss and accuracy of training and validation dataset for 50 epochs.

We have trained the network for 50 epochs. From the figure 5, we can see that our network does not achieve very good accuracy. We increased epochs to 100 but it does not achieve more than 75% accuracy for both Training dataset and validation dataset. Even for 150 epochs it does not provide significant change as the loss for training dataset does not decrease than 0.9. On the other hand, the model loss and model accuracy difference between training and validation dataset is not significant as a result we can't comment whether the model is over-fit or under-fit. Since, it does not achieve good classification accuracy the network is suitable for this task. The confusion matrix for this network is given below. Here, the classes airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck represents 0,1, … 8,9 respectively.



Form the confusion matrix, we can see that the network can identify these airplane, automobile, ship, truck classes better than the other classes. It is possible that, the network has been able to learn more structural classes compared to action changing classes. In figure 6, we presented first 10 wrong classified classes.

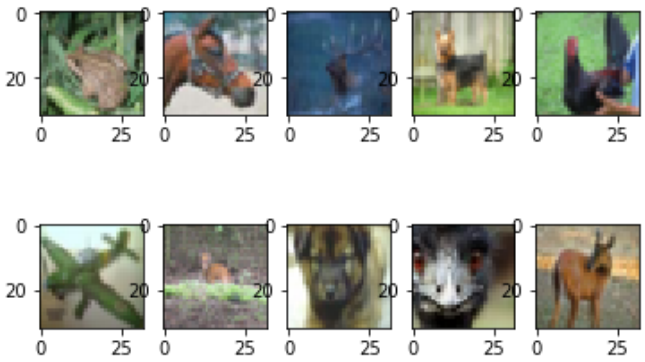
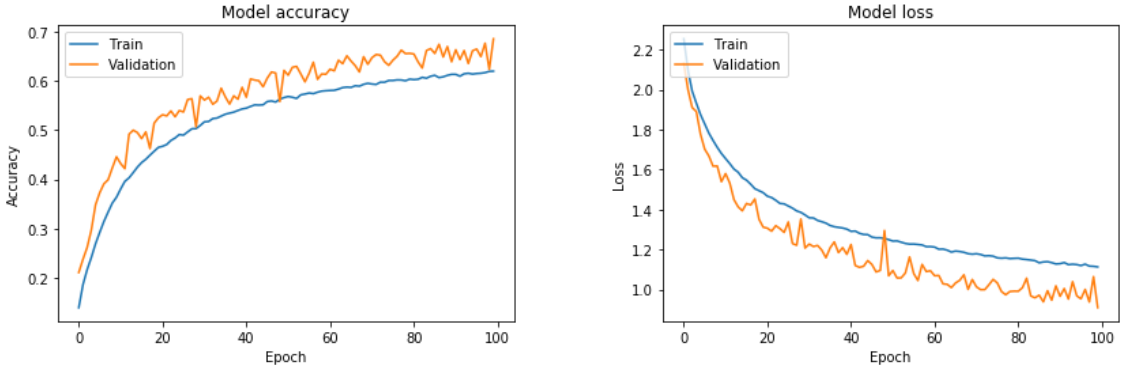


Fig 6. 10 worst classified images by the convolutional network.

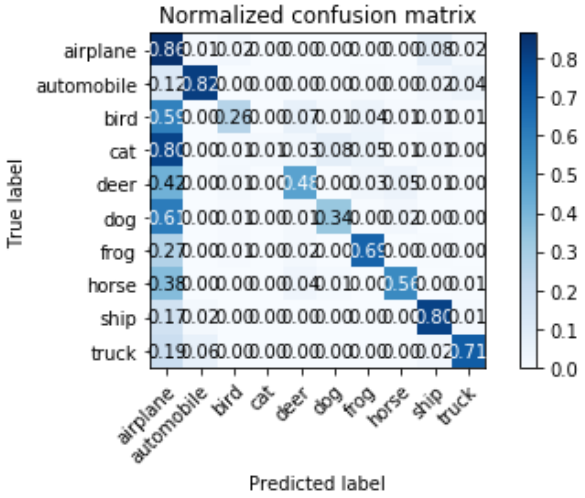
To increase the accuracy, our first intuition was to learn more features from the images so we added two more convolutional layers with pooing layers. In the Dense layer, we added another hidden layer with 128 neurons after the convolutions layers.



c. Model Loss vs Epoch

d. Model Accuracy vs Epoch

Figure 7: Model loss and accuracy of training and validation dataset for 100 epochs.

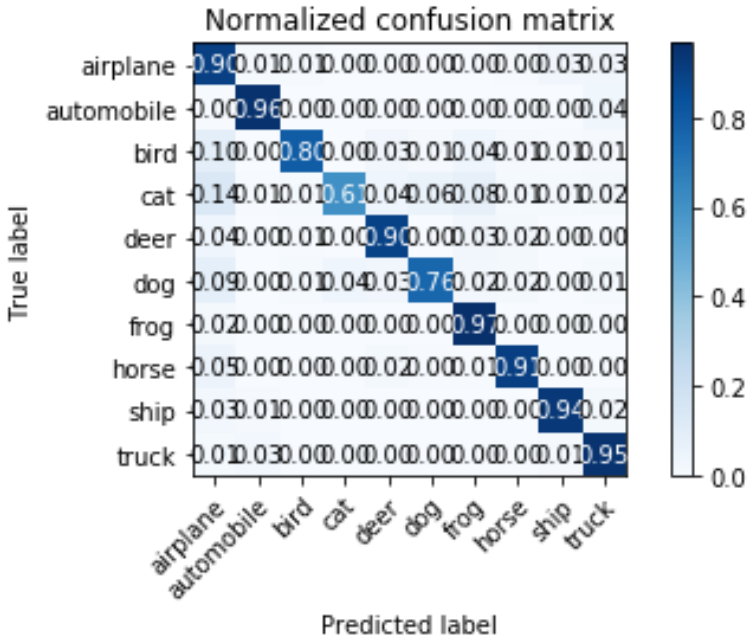


After 150 epochs, test accuracy remains 66.82% only.

Fig 8. 10 worst classified images by the convolutional network.

After trying out with different convolutional layers and dense layers, we understood that our networks are failing to learn the features of the training dataset. So we applied data augmentation to increase the amount of training dataset instances. In this implementation, we have used Exponential Linear Unit (ELU) in convolutional neural network as an activation. Total 5 convolutional layers along with Batch normalization has been used in this network. Only one Dense layer with softmax activation is used for classification task. In convolution layer, max-pooling, Batch normalization and Dropout has been introduced. Batch Normalization normalizes the activation of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1. It addresses the problem of internal covariate shift. It also acts as a regularizer, in some cases eliminating the need for Dropout. Batch Normalization achieves the same accuracy with fewer training steps thus speeding up the training process. Dropout helps to reduce the overfitting of the network by dropping random units. It is used after the max pooling layers. The figure is given in Figure 9. Model loss and accuracy is given in the next page.

Here we have presented the confusion matrix where we can see that, our classification is significantly improved in the new network. Only the class “Cat” and “Dog” has poor classification result. As these two class looks similar, our classifier is misclassifying with each other.



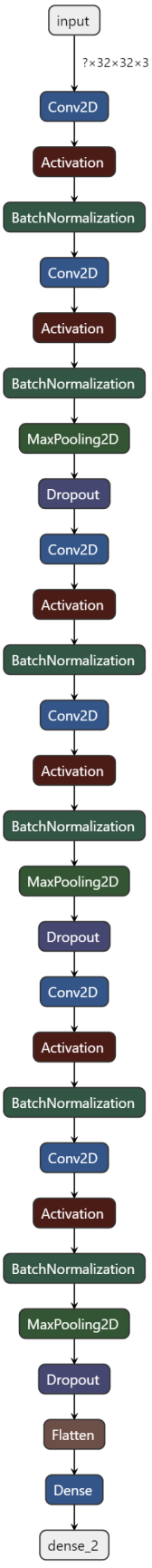
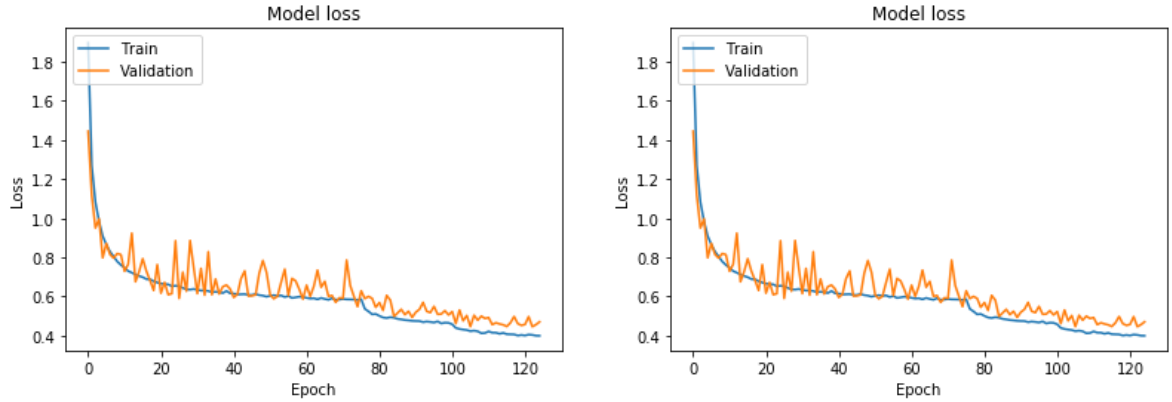


Fig 9: Model Architecture

From figure 10 it is evident that, the model loss and model accuracy difference between training and validation dataset is quite similar as a result we can conclude that this model is well-fit. Also from this curve, it can be seen that, if we increase the number of epochs, the accuracy can be increased a little bit.



a. Model Loss vs Epoch

b. Model Loss vs Epoch

Figure 10: Model loss and accuracy of training and validation dataset for 100 epochs.

In figure 11, we have presented 10 wrong classifications.

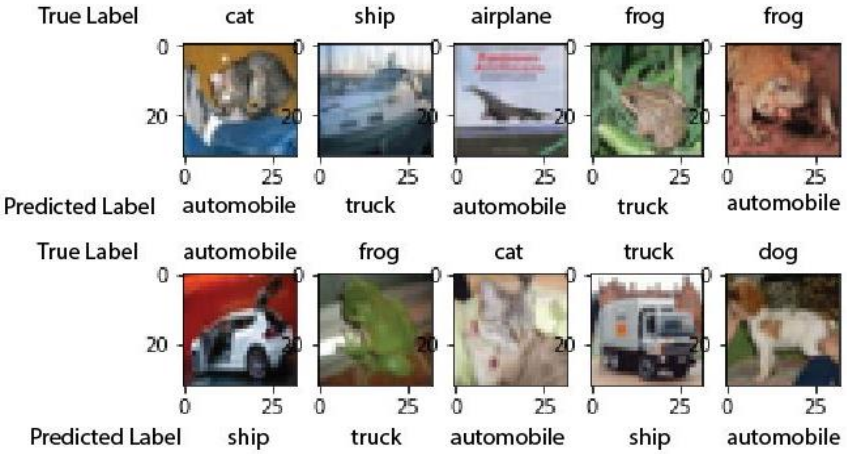


Fig 11. 10 worst classified images by the convolutional network.