Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Project Report

Title: Image Classification Using Pretrained Convolutional Networks.

Written by: Jose Burgos **Neptun#:** BX7FN1

Field: Computer Engineering **Specialization:** Data Science and Artificial Intelligence
E-mail: Data Science and Artificial Intelligence

Academic year: 2023/2024 Autumn semester

I. Introduction

This project revolves around the utilization of the CIFAR-10 dataset, comprising ten image classes, for the development and evaluation of two neural network models. The initial phases encompass data acquisition, where raw pixel values are normalized, and dataset division into training and validation sets with one-hot encoded labels.

Two neural network models were then implemented for comparison. The first, a Convolutional Neural Network (CNN) inspired by VGGNet, demonstrated a test accuracy of 85.61% after one hundred epochs. Weight & Biases facilitated hyperparameter optimization and visualization. The second model employed a pre-trained ResNet50 with ImageNet weights, achieving a notable training accuracy of 95.72%. However, its evaluation prompted concerns about potential overfitting, assessed through various metrics and visual inspections.

II. Methods of training

The first step of the project was data acquisition. The selected dataset was CIFAR-10 that is composed of ten classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck) with six thousand images per class. There are fifty thousand training images and ten thousand test images. In the Figure is shown a random group of thirty-two images of the dataset [1].

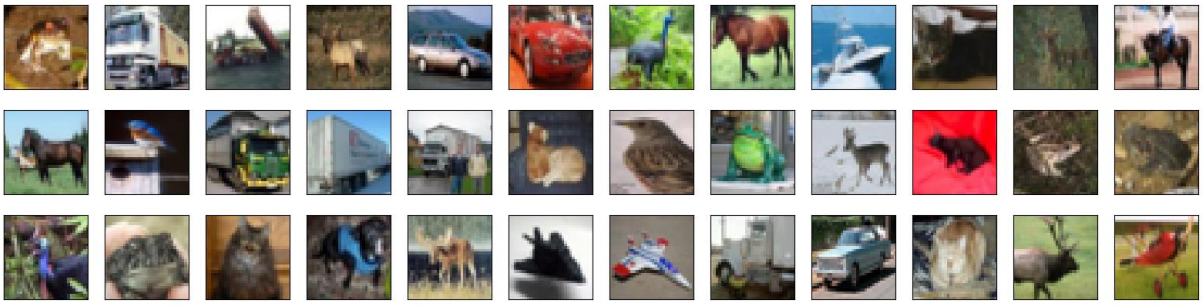

Figure 1. 32 images from CIFAR-10 Dataset randomly selected.

The next step was data preparation where the image that consist of a matrix whit vale from 0 to 255 to normalize the values the preparation step is to divide each value by 255 to make the values part of a range between 0 and 1. The training part of the dataset have to be divided in training a validation and the division was 45000 images for training and 5000 for validation. The labels for testing passed a process of one-hot encoding.

With the data ready to train I started with the models. I decided to use two Neural Networks to make a comparison. First, I make a Convolutional Neural Network based constructed by 6 convolutional layers and 1 fully connected layer with pooling after every 2 layers based in VGGNet, batch normalization after each convolutional layer to normalize the input.

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 32, 32, 92)	2576
activation_12 (Activation)	(None, 32, 32, 92)	0
batch_normalization_12 (BatchNormalization)	(None, 32, 32, 92)	368
conv2d_13 (Conv2D)	(None, 32, 32, 92)	76268
activation_13 (Activation)	(None, 32, 32, 92)	0
batch_normalization_13 (BatchNormalization)	(None, 32, 32, 92)	368
max_pooling2d_6 (MaxPooling2D)	(None, 16, 16, 92)	0
dropout_6 (Dropout)	(None, 16, 16, 92)	0
conv2d_14 (Conv2D)	(None, 16, 16, 184)	152536
activation_14 (Activation)	(None, 16, 16, 184)	0
batch_normalization_14 (BatchNormalization)	(None, 16, 16, 184)	736
conv2d_15 (Conv2D)	(None, 16, 16, 184)	304888
activation_15 (Activation)	(None, 16, 16, 184)	0
batch_normalization_15 (BatchNormalization)	(None, 16, 16, 184)	736
max_pooling2d_7 (MaxPooling2D)	(None, 8, 8, 184)	0
dropout_7 (Dropout)	(None, 8, 8, 184)	0
conv2d_16 (Conv2D)	(None, 8, 8, 368)	609776
activation_16 (Activation)	(None, 8, 8, 368)	0
batch_normalization_16 (BatchNormalization)	(None, 8, 8, 368)	1472
conv2d_17 (Conv2D)	(None, 8, 8, 368)	1219184
activation_17 (Activation)	(None, 8, 8, 368)	0
batch_normalization_17 (BatchNormalization)	(None, 8, 8, 368)	1472
max_pooling2d_8 (MaxPooling2D)	(None, 4, 4, 368)	0
dropout_8 (Dropout)	(None, 4, 4, 368)	0
flatten_2 (Flatten)	(None, 5888)	0
dense_2 (Dense)	(None, 10)	58890

Total params: 2429270 (9.27 MB)
Trainable params: 2426694 (9.26 MB)
Non-trainable params: 2576 (10.06 KB)

Figure 2. Convolutional Neural Network Summary.

The hyperparameters of this model were randomly selected, the number of epochs for the training were one hundred and at the final of the training the accuracy of the test was 85.61% and is show in the Figure where can be appreciated the learning curves of this model.

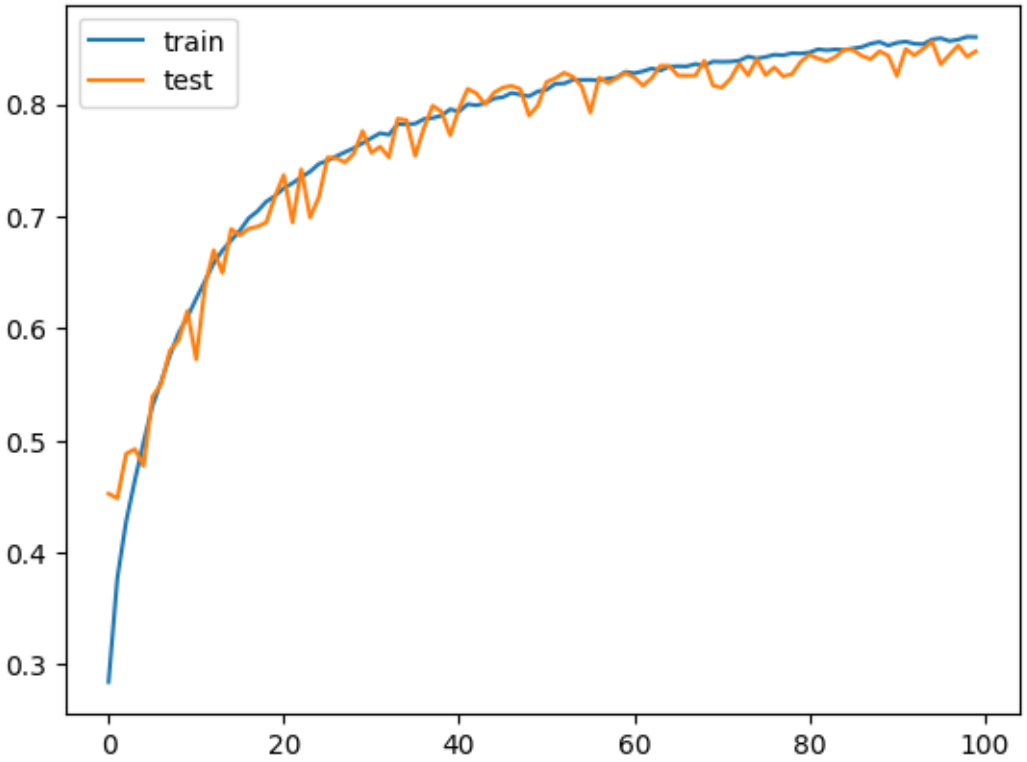

Figure 3. Learning Curves from the Convolutional Neural Network.

To improve the model and the quality of the report I decided to use Weight & Biases where are stored the information about each epoch and a graphic is generated automatically. This helps with the hyperparameters optimization too with a previous configuration in the initialization process. To reduce the number of unnecessary epochs I implemented an early stopping function with a patience of five. The report obtained is in the repository.

The pretrained Neural Network used was ResNet50 with the weights of "ImageNet". The training of this model was the same as the previous model. The summary of the model is shown in the Figure.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0
 up_sampling2d (UpSampling2  (None, 224, 224, 3)       0
 D)
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712
 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)
 flatten (Flatten)           (None, 2048)              0
 dense (Dense)               (None, 1024)              2098176
 dense_1 (Dense)             (None, 512)               524800
 classification (Dense)      (None, 10)                5130
=================================================================
Total params: 26215818 (100.01 MB)
Trainable params: 26162698 (99.80 MB)
Non-trainable params: 53120 (207.50 KB)
_________________________________________________________________
```

Figure 4. Pretrained Neural Network Summary.

III. Evaluation

The evaluation selected was in three ways, first is the basic evaluation of the Convolutional Neural Network where the function “model.evualate.” is used and the model obtained an accuracy of 83.03%. The second is a visual evaluation where a random group of thirty-two images of the test dataset was evaluated if the predicted label matched the true label with the next results obtained.

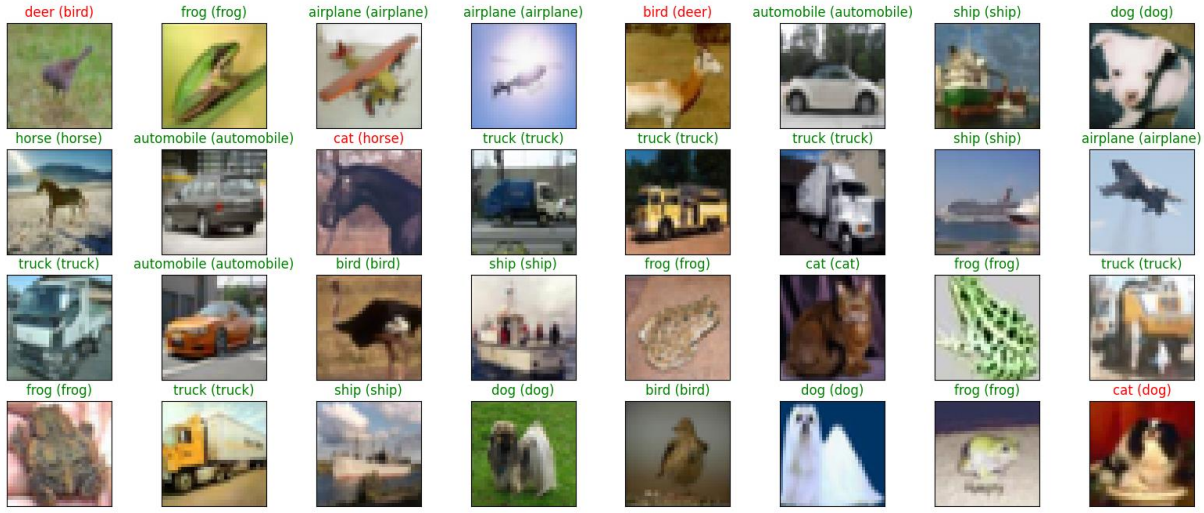

Figure 5. Visual evaluation for the Convolutional Neural Network.

The second way of evaluating is a confusion matrix and the classification report where the most important values are resumed. The model obtained an accuracy of 83%.

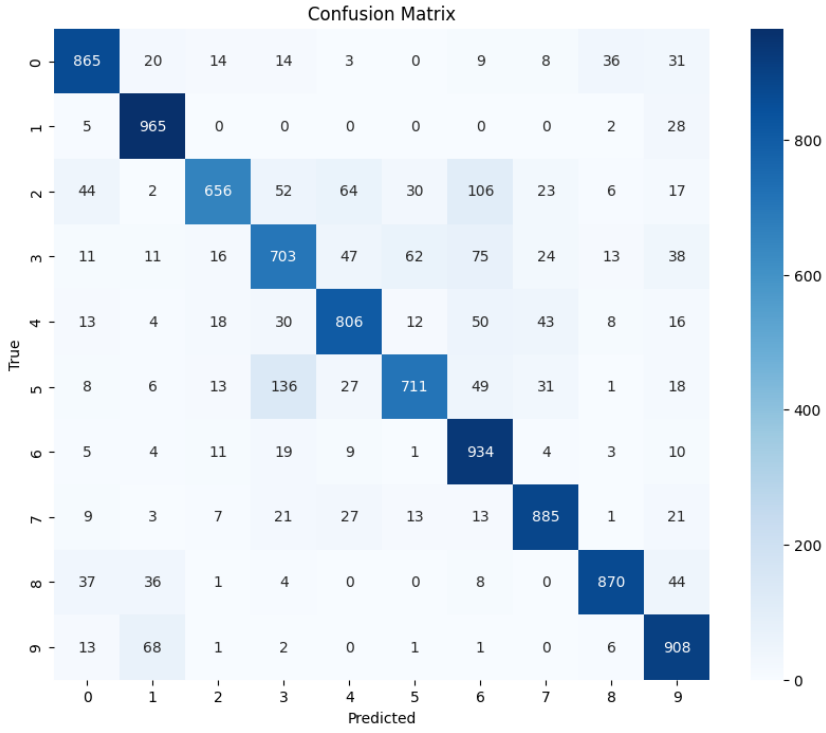

Figure 6. Confusion Matrix from Convolutional Neural Network.

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.86	0.86	1000
1	0.86	0.96	0.91	1000
2	0.89	0.66	0.76	1000
3	0.72	0.70	0.71	1000
4	0.82	0.81	0.81	1000
5	0.86	0.71	0.78	1000
6	0.75	0.93	0.83	1000
7	0.87	0.89	0.88	1000
8	0.92	0.87	0.89	1000
9	0.80	0.91	0.85	1000
accuracy			0.83	10000
macro avg	0.83	0.83	0.83	10000
weighted avg	0.83	0.83	0.83	10000

Figure 7. Classification Report from Convolutional Neural Network.

To improve the model, I used an auto optimization process with Optuna where the hyperparameters optimize where the hidden units, the weight decay, and the learning rate. The number of trails is ten and the results are shown in a table.

Results Table:

	Base Hidden Units	Weight Decay	Learning Rate	Accuracy
0	21	0.002465	0.005163	0.8303
1	27	0.000200	0.002157	0.8303
2	127	0.008437	0.001297	0.8303
3	22	0.000003	0.011092	0.8303
4	48	0.000229	0.000063	0.8303
5	55	0.000054	0.000583	0.8303
6	34	0.001767	0.040526	0.8303
7	57	0.005652	0.000060	0.8303
8	18	0.000739	0.000017	0.8303
9	20	0.000038	0.000059	0.8303

Figure 8. Result table for the auto optimization process.

For the pretrained Neural Network the value obtained in the model evaluation was an accuracy of 95.72%. The visual evaluation is shown in the next figure where just five images do not match the true label.

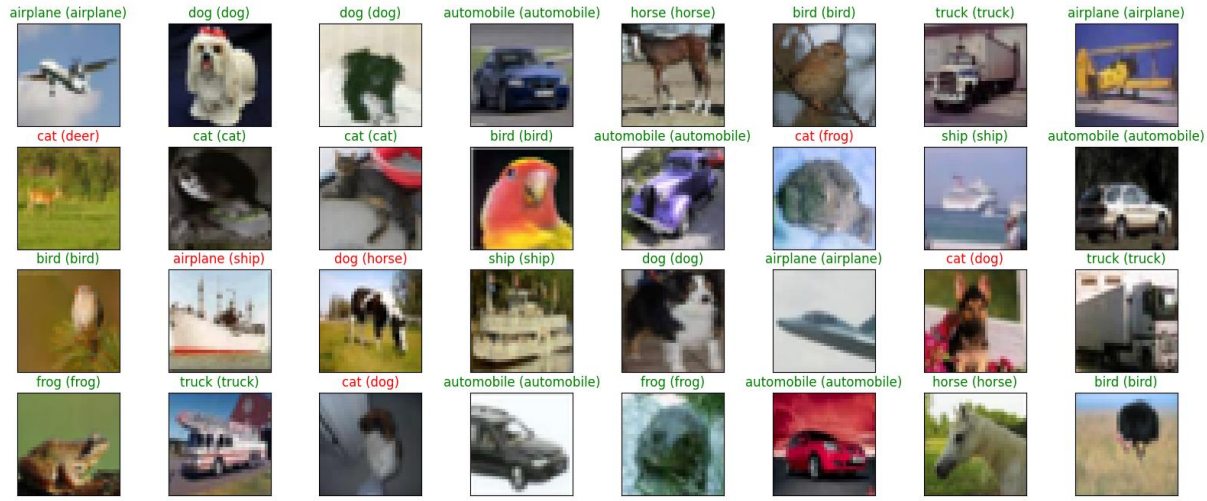

Figure 9. Visual evaluation for the Pretrained Neural Network.

The results obtained in the confusion matrix and classification repost show an accuracy of just 80%, that is less than the accuracy obtained in the training.

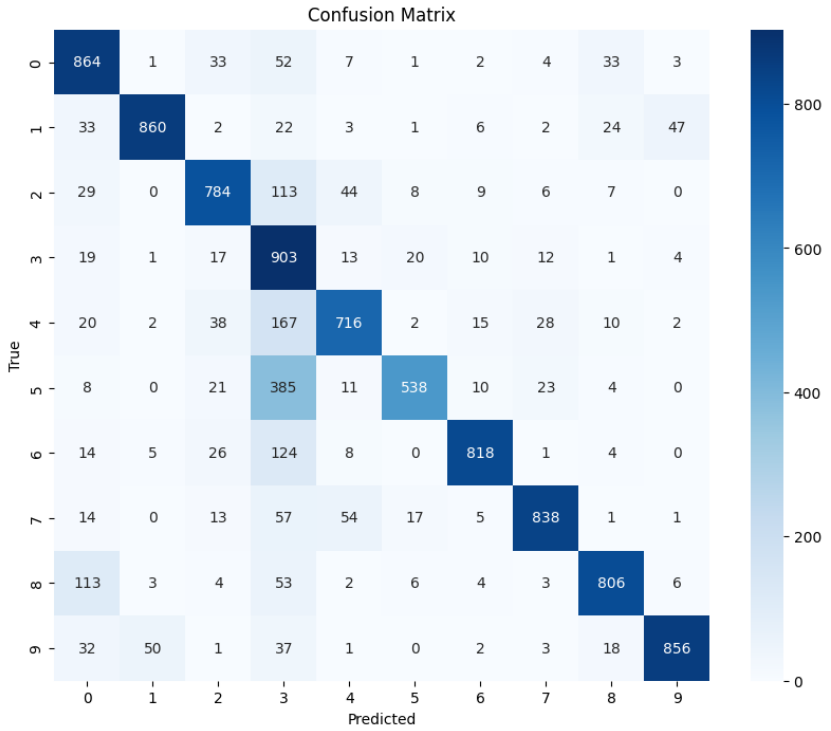

Figure 10. Confusion Matrix from Pretrained Neural Network.

Classification Report:

	precision	recall	f1-score	support
0	0.75	0.86	0.81	1000
1	0.93	0.86	0.89	1000
2	0.83	0.78	0.81	1000
3	0.47	0.90	0.62	1000
4	0.83	0.72	0.77	1000
5	0.91	0.54	0.68	1000
6	0.93	0.82	0.87	1000
7	0.91	0.84	0.87	1000
8	0.89	0.81	0.84	1000
9	0.93	0.86	0.89	1000
accuracy			0.80	10000
macro avg	0.84	0.80	0.81	10000
weighted avg	0.84	0.80	0.81	10000

Figure 11. Classification Report from Pretrained Neural Network.

IV. Conclusion

The pretrained model shows the best results in the training part but in the evaluation shows an accuracy approximated to the Convolutional Neural Network. I could suppose that the pretrained model is overfitting the dataset that producing wrong predictions. The recommendation is to use less numbers of epochs for the pretrained model and change the monitoring value in the early stopping and reduce the patience.

References:

[1] Alex Krizhevsky, *The CIFAR-10 dataset,* University of Toronto, 2009. [Online] Available: https://www.cs.toronto.edu/~kriz/cifar.html

[2] Mohamed Elgendy, *Deep Learning for Computer Vision,* Manning Publications, 2019

[3] Kaiming He & Xiangyu Zhang & Shaoqing Ren & Jian Sun, *Deep Residual Learning for Image Recognition,* arXiv 1512.03385, 2015. [Online] Available: https://arxiv.org/pdf/1512.03385.pdf