Image Classification with Convolution Neural Networks

Project 2 - Sheyda Zarandi

Abstract—**With the rapid development of high performance computing and parallel computing devices, convolutional neural network (CNN) has drawn increasingly more attention from many researchers for image classification. In this project, we explore the power of CNN in classifying images from CIFAR-10 dataset. We first propose a simple CNN for image classification and exercise many approaches introduced in literature to enhance the accuracy of this simple model. However, we will observe that after some point, we will not be able to further improve the performance of such simple model. At this point, we will start exploring the power of residual networks (Resnet) in enabling us to have deeper networks without loss of performance due to effectively addressing issues that are inherent in deep networks' training process, such as vanishing-gradient. Afterwards, in simulation results we will illustrate the performance of our model which achieves approximately 92.30% accuracy without having to deal with the complexities of implementing a very deep CNN. Furthermore, we extensively analyze the impact of different normalization techniques, colors, and network's models and architectures including number of Resnet block and layers, on the accuracy of the classifier.**

I. Introduction

As a significant application of machine learning and pattern recognition theory, image classification has become an important topic. Image classification is widely employed in different fields and technologies for face recognition, object detection, or signature verification. Regarding these wide range of applications, it is not a surprise that image classification have become a popular field of research among scholars and academia.

This fact motivates us to explore the power of convolution neural network (CNN) in image classification in this project.

Here, we choose CIFAR-10 which is a general image classification dataset, containing 10 different classes of colored-images, such as airplane, truck, frog and etc. The low resolution and complex image content, turn CIFAR-10 classification into a complicated and challenging task. Through the implementation and application of the convolutional neural networks, we aim to figure out how we can deal with this image classification problem.

Moreover, through experiments we will explore different factors that can have an impact on the efficiency and accuracy of CNNs in classification of CIFAR-10 images. Benefiting from the knowledge we gained in this course, we also propose an improved version of CNN, namely residual neural networks (Resnets) which, enables us to take advantage from deeper network architectures without being forced to struggle with challenges that are inherent to the training of these complex networks.

The superiority of deep networks has been spotted in several works in the recent years. In [1], two kinds of deep learning networks, namely AlexNet and Network In Network (NIN) is employed for image retrieval using CIFAR-10 and CIFAR-100 which result in approximately 83% and 57% accuracy on each of these datasets, respectively. In [2], authors combine the idea of deep CNN and K-Nearest Neighbors method and achieves almost 94% accuracy on CIFAR-10. The authors in [3], propose a very deep CNN architecture in which max-pooling layers are all replaced by convolutional layers with increased stride, without any loss in the accuracy of presented neural network which is approximately 95%.

In [4], a new architecture is proposed for Resnets. In this paper, authors propose that instead of using deep Resnets by increasing number of residual blocks, we can widen each block by adding more convolutions to them. This new approach result in approximately 96% accuracy (using pytorch for simulation), by employing more than 128 convolution layers in each residual block.

Deep residual networks were shown to be able to scale up to thousands of layers and still have improving performance. However, each fraction of a percent of improved accuracy costs nearly doubling the number of layers [4]. For instance, as previously mentioned in [4], authors implement more that 128 convolution layers in each of their residual blocks.

Unfortunately, due to hardware limitations which make implementing very deep networks infeasible and extremely time consuming, here we cannot explore such deep architectures. However, we will examine the impact of many of the techniques and approaches introduced in these literature to gain better results, on a much smaller network and try to step by step improve the accuracy of our model by exploiting these methods. Still, the restraints we have regarding the implementation makes us unable to achieve an accuracy as high as the state-of-the art.

The rest of this paper is organized as follows: In Section II, we will propose a simple CNN and explain how different techniques can be used to improve accuracy of this simple model. Then to further improve our work, in Section III, we propose a Resnet architecture for our image classifier. In Section IV, we present simulation results and illustrate how each of the aforementioned methods and architectures effect the accuracy of the proposed model. We further explore our proposed network by comparing our results to the result of the case when no color exists in the images (the images would be converted to gray-scaled images) and will see how colors can help classifiers in better classification of CIFAR-10.

II. A SIMPLE CONVOLUTIONAL NEURAL NETWORK

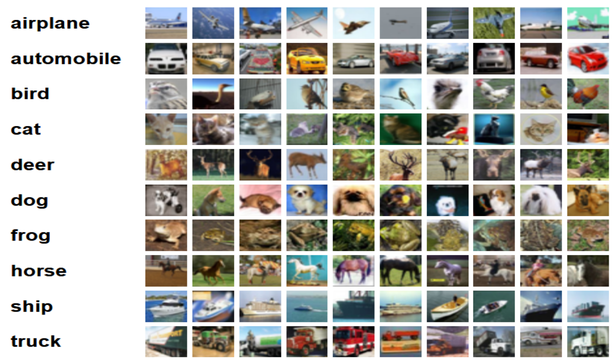


Fig. 1: Pictures from different classes in CIFAR-10 dataset

In this section we propose a simple CNN for classification of CIFAR-10 dataset. Some samples of CIFAR-10 images are given in Fig. 1.

TABLE I: Simple CNN Model

Simple CNN Model
2 convolution layers with 32 filters with kernels of size (3 × 3)
AveragePooling layer with pool size equal to (2×2)
1 convolution layer with 32 filters with kernels of size (3 × 3)
DropOut Layer 1
2 convolution layers with 64 filters with kernels of size (3 × 3)
AveragePooling layer with pool size equal to (2×2)
DropOut Layer 2
2 convolution layers with 128 filters with kernels of size (3 × 3)
MaxPooling layer with pool size equal to (2×2)
DropOut Layer 3
Flatten()
Dense layer with 10 nodes for each of 10 classes

The simple model we implemented for this section is as given in Table 1.

In the above model many techniques that are proposed in literature to improve the accuracy of CNN are employed. These methods are explained in what follows:

- **Dropout:** The term ”dropout” refers to dropping out units (both hidden and visible) in a neural network. Dropout is a regularization technique patented by Google for reducing over-fitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. Here, we implement Dropout layers in between almost all convolution layers. This helps us to considerably alleviate over-fitting problem.

- **Batch Normalization:** Batch normalization will transform inputs so that they are standardized, meaning that they will have a mean of zero and a standard deviation of one (depending on settings these values may vary). During training, the layer will keep track of statistics for each variable and use them to standardize the output which leads to reduction of covariance shift in hidden layers. This would both accelerate the learning process and improves the performance of CNN. Here we have used batch normalization on the output of each convolution layer in our model.

- **Data Augmentation:** One of the simplest way to reduce over-fitting and improve the accuracy of neural networks in general, is to have more training data. Data augmentation is a strategy that significantly increases the diversity of data available for training models, without actually collecting new data and through simply manipulating the data we already have (e.g. cropping, padding, horizontal flipping, rotation, and etc.).

- **Kernel Regularization:** Regularization provides an approach to reduce the over-fitting of a deep learning neural network model on the training data and improve the performance of the model on new data, such as the holdout test set. L1 and L2 are the most common types of regularization. These regularizers update the general value of a parameter (e.g. cost function or weights) by adding another term known as the regularization term. This added value prevents the perfect fitting of parameters' values based on the training data. This would eventually reduce over-fitting.

as we will later show in simulation results, these techniques have a huge impact on the accuracy we achieve. This observation underlines the fact that aside from networks' model and architecture such as number of layers, other factors like having more data through data augmentation, batch normalization, and etc, can have a very conspicuous impact on performance of neural networks.

However, at some level we reach the point where the power of these factors would be exhausted and we cannot gain anymore from manipulating them. This is when we are faced with the fact that we need a stronger model to improve the accuracy of our classifier. This is what motivated us to explore more powerful models such as residual neural networks in the following section.

III. Residual Network

In order to improve the accuracy of the model introduced in the previous section, one way is to deepen the neural network by add more layers to it. However, as previously mentioned, implementing deep neural networks can be quite challenging for many issues that rises during training them such as vanishing gradient. In order to use the power of deep networks without facing some of these sever challenges, one way is to use residual neural networks.

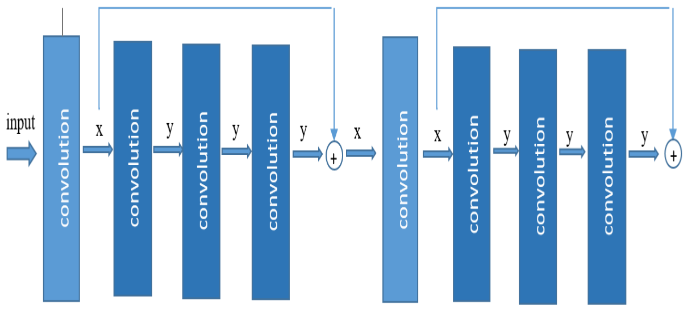


Fig. 2: The overall architecture of Resnet

A Resnet overcomes many of the complexities in training of a deep model by utilizing skip connections or shortcuts to jump over some layers. Typical Resnet models are implemented with double- or triple- layer skips that may contain nonlinearities and batch normalization in between. One motivation for skipping over layers is to avoid the problem of vanishing gradients, by reusing activations from a previous layer until the adjacent layer learns its weights.

Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of training, when all layers are expanded, it stays closer to the manifold and thus learns faster. A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold, and necessitates extra training data to recover.

These benefits and capabilities motivate us to exploit Resnets to improve the accuracy of our classifier.

Overall architecture of the Resnet we implemented for this project is given in Table. 2. As highlighted in this Table, in our model, we have a jump every three convolution layers.

It should be noted that as is proved in [4], having more convolutions in each Resnet block can considerably improve the results' quality. In fact the authors of this paper prove that the impact of having more convolutions in each Resnet block (wider blocks) is much more significant than the performance gain we can obtain by adding more blocks to the network.

However, in absence of powerful hardware, here we only consider four layers of convolution in each Resnet block. Furthermore, there are *elu* activation layer (as given in (1)) followed by batch normalization layer in between each convolution layer in the above proposed model.

$$R(z) = \begin{cases} z & \text{if } z > 0, \\ \alpha(e^z - 1) & \text{otherwise.} \end{cases} \tag{1}$$

It is worth mentioning that due to slower smoothness in *elu* compared to *relu*, employing *elu* activation function, can also be considered as an effective way to deal with vanishing gradient issue that as previously explained is inherent in training process of deep networks.

Moreover, to prevent the issue of overfitting, we also use dropout layers between some of the resnet blocks. It should be noted that since we have a deep network here, excessive use of dropout layers (for example between each convolution layer similar to what we had for the simple model introduced in Section II) can be counter-productive and result in lose of information and thus deteriorate performance.

Based on what was explained above the architecture of our proposed model is as bellow:

TABLE II: Resnet Model

Resnet Model
5 Resnet blocks with 16 (3 × 3) filters
DropOut Layer 1
5 Resnet blocks with 32 (3 × 3) filters
DropOut Layer 2
5 Resnet blocks with 64 (3 × 3) filters
DropOut Layer 3
AveragePooling with filter of size (4 × 4)
Flatten()
Dense layer with 10 nodes for each of 10 classes

where:

- each Resnet block contains 4 convolution layers
- the result of each convolution passes through *elu* activation function
- between each convolution layer we have batch normalization layer

In the following section, we further explore the details of our introduced models and impact of different factors on their performance through extensive simulations.

IV. Simulation Results

In this section we evaluate the performance of our proposed models through simulation using TensorFlow framework. The activation function used throughout simulations is *elu* and all initialization are base on He-normal method.

For better organization, we divide this section into two parts. In the first part we briefly explore the simple CNN model introduced in Section II, and in the second part we focus on evaluation of Resent model presented in Section III.

A. Simple Model

When we proposed the simple CNN model in Table 1, we listed a list of parameters and techniques that can be useful to improve the performance of CNNs, including data augmentation, batch normalization. If we exploit all these techniques, as can be seen in Fig. 3, we will be able to achieve 96% and 90% accuracy on train and test sets, respectively.

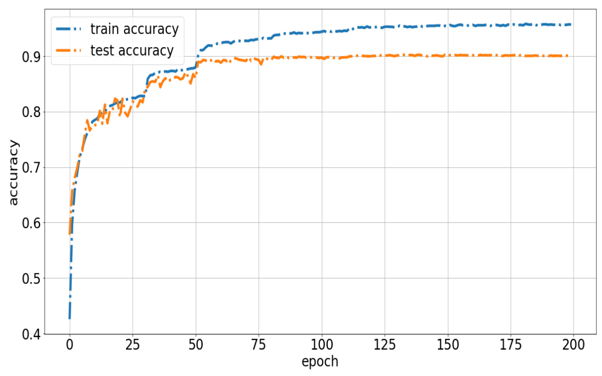


Fig. 3: Accuracy of simple CNN model

To underline the significance of the techniques mentioned in Section II, here we remove all these elements, except for dropouts to avoid extreme overfitting, from the model and as illustrated in Fig. 4, the accuracy achieved would be reduced significantly, from 90% to less than 75%. This observation underlines the fact that aside from networks' model and architecture, other factors such as the techniques explained in Section II can also have a very conspicuous impact on performance of neural networks and thus, should not be underestimated.

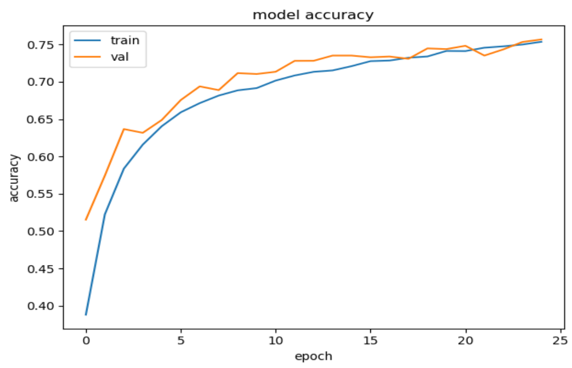


Fig. 4: Accuracy of simple CNN model

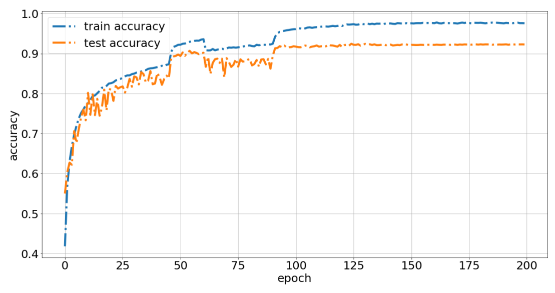
However, we would eventually reach the point where the power of these factors would be exhausted and we cannot gain anymore from manipulating them. This is when we would need stronger models, such as Resnets, to further improve the accuracy of the classifier.

B. Resnet Model

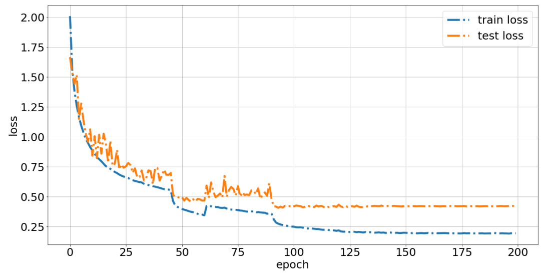
The model used for Resnet was previously given in Table 2. This model with its 15 Resnet blocks results in almost 98% accuracy on train set and 92.30% accuracy on test set. In Fig. 5, the accuracy and loss of our model is illustrated.

As can be seen, at the beginning due to the fact that network's parameters are not yet trained, accuracy is fairly low and loss is high. However, as the number of epochs increases, the performance of network improves until it almost converges to the values provided in Table 3 around epoch number 150.

Even though a part of the improvement we achieved here, compared to the simple model in the previous section, might be due to having deeper network, the effect of skipping-connection approach in residual networks should not be underestimated.



(a) Accuracy



(b) Loss

Fig. 5: Performance of Resnet model introduced in Table 2.

TABLE III: Resnet Model

	Accuracy (%)	Loss
Train set	97.57	0.1938
Test set	92.30	0.4209

To evaluate how such simple act can impact the accuracy of our model, we have removed the skip-connection part from the Resnet and compared the results obtained with the results from the default model in Fig. 6.

As can be seen in this figure, also from the statistics provided in Table 4, where we have given the accuracy and loss report of the model with removed skip connection, the simple act of adding the result of previous layers directly with the result of layers ahead, can significantly increase the accuracy of neural network even if we overlook the fact that this skipping connection act also enables us to train up to hundreds or even thousands of layers and still achieve compelling performance.

This increase in achieved accuracy that comes with skipping connections in Resnet, is due to the fact that doing so can be quite effective in fighting with issues like gradient-vanishing. Deep networks are hard to train because of this notorious problem which is due to the fact that as the gradient is back-

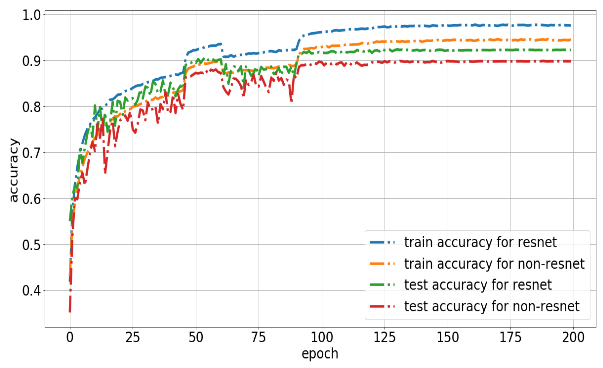


Fig. 6: Resnet model accuracy compared to removed-skipped-connection case

propagated to earlier layers, repeated multiplications may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly. Skip connections is a mechanism that strengthens the result of previous layers and in this way reduces the aforementioned issue.

TABLE IV: Results from removed skipping-connection

	Accuracy (%)	Loss
Train set	94.55	0.2862
Test set	89.77	0.4727

An other important parameter in Resnets' performance is the number of Resnet blocks. To evaluate the effect of this parameter here we consider two cases: first when we have 15 Resnet blocks as in our default model and second when we have 9 resnet blocks. The results of these two cases are given in Fig. 7. Although in simulation results the improvement is nominal and cannot be clearly perceived, we can see this improvement in Table 5 where the results of these two cases as well as when we have 12 blocks, on the test set are provided.

As proved in [4], the number of convolution in each Resnet blocks (width of Resnet), can have an even deeper effect on the performance of Resnets than the number of layers.

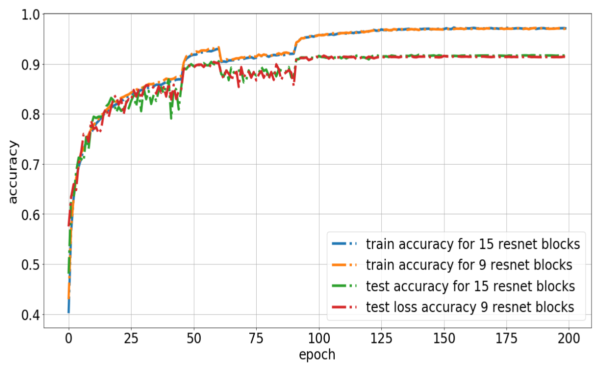


Fig. 7: The overall architecture of Resnet

TABLE V: Results from different number of Resnet blocks

	Accuracy (%)	Loss
15 Resnet blocks	92.30	0.4209
12 Resnet blocks	91.58	0.4210
9 Resnet blocks	91.45	0.4211

To prove this point, in [4], authors consider a sixteen block Resnet, where in each block we have $16 \times k$ convolutions, where $k \in \{1, 2, 3\}$. Unfortunately we are not able to experiment with such deep and complex Resnet, however here we consider a simpler case and compare the results from when we have three convolutions in each block with the default model with 4 convolutions in each block. The results from these two scenario are given in Table 6.

As is clear from the data provided in this table, even adding one convolution layer to each block (in general we have fifteen extra convolutions in the whole resnet as there are 15 blocks in our model) can empower the model and considerably increase the accuracy.

TABLE VI: Results from different number of convolutions in each blocks

	Accuracy (%)	Loss
4 convolutions in each blocks	92.30	0.4209
3 convolutions in each blocks	91.64	0.4205

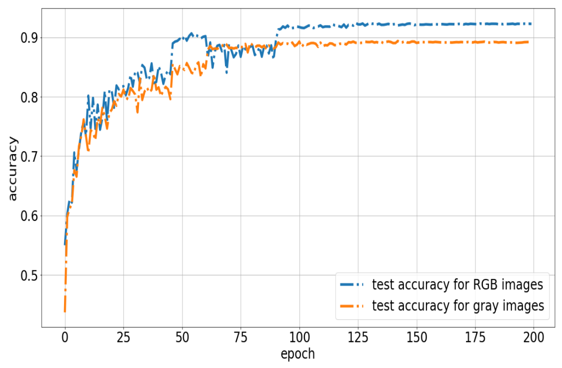
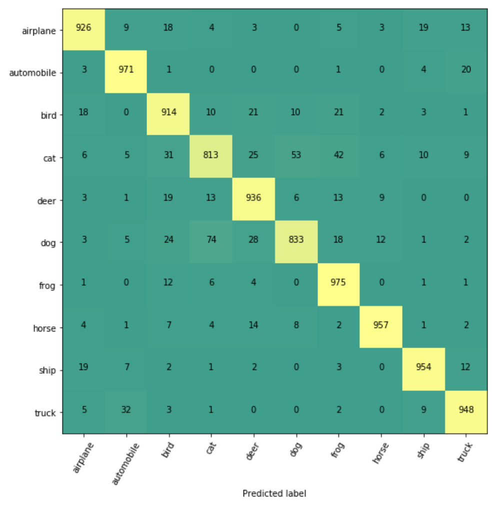


Fig. 8: RBG images vs. Gray images classification

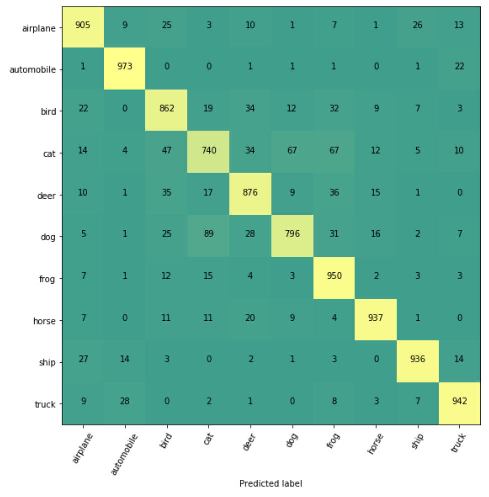
One of the questions that usually arises while working with different datasets is how we can simplify the structure of data through methods such as dimension reduction and how these simplifications would effect the performance of the neural network since in all these approaches we lose some of the information at hand. To this end, here we have removed the color from the dataset's images (gray-scaled all the images) and compared the result to when we use the original colored images. As can be perceived from Fig. 8, colors can be quite beneficial in image classification of CIFAR-10 dataset. In fact, removing colors from images decreases the accuracy of neural network from 92.30% to 89.17%, which is a notable reduction in accuracy.

To clarify the reason behind this observation, we have provided the confusion matrix of these two scenarios in Fig. 9.

In Fig. 9 we can see that colors are widely exploited by the Resnet to correctly classify images. For example , we can see that colors have a quite notable effective in differentiating between cats and frogs. In fact when we remove the colors from images, cat/frog miss-classification rises from 42 cases for when we have color to 67 in gray-scaled images. This is something understandable for us, since we are also less likely to classify a vague image as cat if its color is green or dark red. However, there are some cases such as dog/cat in which the classification error rises from 74 to 89 that implies although unclear to us, colors can be used for differentiating between dogs and cats as well.



(a) RGB images



(b) Gray images

Fig. 9: Confusion matrix

V. Conclusion

In this project we explored two models proposed for image classification using CIFAR-10 dataset. We first focused on a simple CNN model and saw that through exploiting techniques such as batch normalization and data augmentation we can reach 90% accuracy even with this small neural network. Then to empower the classifier, we proposed a residual network with 15 blocks, each containing 4 convolutions. Through this model we were able to improve the accuracy from 90% to 92.30%. Although in [4], the authors were able to reach almost 96% accuracy while using pytorch for their simulations, we should note that any of such deep and multi-layer architecture is out of the hardware capabilities we have at hand. Due to this here we sufficed to explore the techniques proposed in these papers in a much smaller scale. Base on this, in simulations we extensively investigated the effect of normalization techniques, data augmentation, use of residual networks, number of residual blocks, and number of convolutions in each block on the accuracy of classifier. Also, through simulations we draw this conclusion that use of colors can be extremely effective in classification of CIFAR-10 images and gray-scaling images can considerably decrease the accuracy of proposed model.

References

[1] H. Huang, C. Chiu, C. Kuo, Y. Wu, N. N. Y. Chu, and P. Chang, "Mixture of deep cnn-based ensemble model for image retrieval," in *2016 IEEE 5th Global Conference on Consumer Electronics*, Oct 2016, pp. 1–2.

[2] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, "Cifar-10: Knn-based ensemble of classifiers," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec 2016, pp. 1192–1195.

[3] T. B. M. R. Jost Tobias Springenberg, Alexey Dosovitskiy, "Striving for simplicity: The all convolutional net," in *Arxiv*, Apr. 2015.

[4] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Arxiv*, Apr. 2016.