Building a New CNN Model to Outperform the Accuracy of Butterfly 50 Images Dataset

Name : MD. ISHTIYAK ALI RIFAT
ID : 18-37927-2
Dept. : CSE

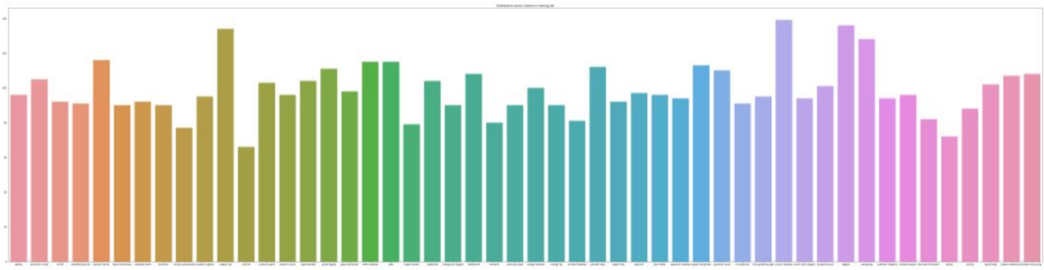
Abstract: This paper is focused on the implementation of different Convolutional Neural Network (CNN) architectures on a specific data-set to analyze the accuracy of every CNN model. Resnet50, VGG16, and a custom model were created for this project. Images of 50 species of butterfly were trained in this project to analyze the accuracy and layer by layer visualization. The main objective is to improve the accuracy with our new CNN model as well as analyze the accuracy of other models also. With the help of multiple convolutions, relu and pooling layers, an efficient model is created to gain the needed accuracy. In VGG16 a good accuracy rate was found. It also found that in Resnet50 the desired accuracy is not achieved due to a lack of proper image visualization. In the part of filter visualization, many different layers were chosen to show the visualization of train images. The need of forming a CNN model to achieve maximum accuracy and visualizing the layers while training is discussed briefly in this paper.

Keywords: cnn; neural network; data training; classification; visualization; computer vision.

1. Introduction

Deep learning, also known as deep neural networks, has shown good prospects in many practical applications. Convolutional neural network (CNN), is the most established algorithm in the field of deep learning. CNN is developed to learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. In this project, a dataset of images is used to train with the help of different CNN models. The name of the dataset is "Butterfly Image Classification 50 species" and is taken from the Kaggle website. All the images are in 224×224×3 in jpg format. The dataset is divided into test, train and valid directory. Also, the dataset is balanced or not is checked at the beginning.

Figure 1. Butterfly image classification data-set with 4 classes



Total Train Examples = 4955

Figure 2. Train data with 50 classes

The convolution layer extracts information from nearby pixels to down sample the picture into features, which are then predicted target values by prediction layers. In this implementation, multiple convolution filters were used to run over the image in order to compute the dot product. It might be said that the learning ability of the network has been a core part of the current success of pattern recognition applications. In this project, the very first model Resnet50 is used to see the prior accuracy. But unfortunately, it couldn't get the desired accuracy as the augmented image is used for this model. If a direct pre-processed image from the dataset is used in the model, then the accuracy rate could be higher. Then VGG16 is implemented to analyze the maximum accuracy and this time the model is trained perfectly with the desired accuracy. After that, a new model is built to outperform the current accuracy. The image is pre-processed with mean-rgb method before train the model.

2. Related Works

There are some related works and research paper available which is evaluated thoroughly to develop the model before starting the project. In a paper, ecological butterfly images were taken to do the classification network that combines the dilated residual network, squeeze-and-excitation (SE) module, and spatial attention (SA) module [1]. The experiment achieved a good result by using class activation mapping (CAM) that can give a good visual explanation for the classification result.

In the paper by Ayad Saad Almryad et al. various deep learning models are used to classify the images of 17,769 butterflies with 10 species [2]. VGG16, VGG19, ResNet50 is used in the model which have been trained on the ImageNet dataset and have achieved high success. In this paper, fine-tuned transfer learning methods are used for the classification of the butterfly images [2].

The paper by Ayad Saad Almryad et al. proposed an Android application for detecting butterfly categories. We built and optimized three models and the best one is deployed to an Android mobile device [1]. It is found in the paper that different models are sensitive to parameters. For SVM, the testing accuracy would not improve after data augmentation as the model cannot handle complex scenarios. But data augmentation is significant to 4-Conv CNN, while transfer learning is not sensitive to the size of the dataset because the domain of said dataset is similar to their original [3].

In the dataset "Butterfly Image Classification 50 species" in Kaggle, there are some people who use different types of classification models to get the highest accuracy. The highest accuracy of this image is 99% by using the base model of EffiecientNetB1. The dataset was well balanced with augmented images. 85 batches and 20 train epochs of data are run to complete an epoch in that model. Once the training accuracy reaches 90% the callback function which is created by the author adjusts the learning rate based on the validation loss.

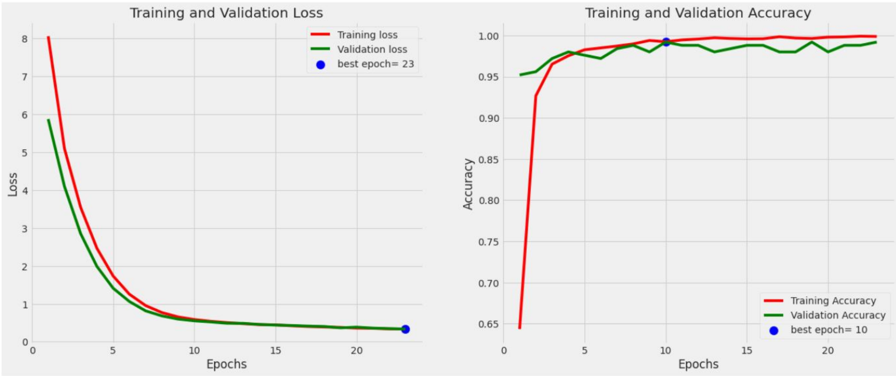


Figure 3. Highest accuracy that was achieved with Butterfly Dataset

3. Proposed Model

A CNN model is implemented in this project to outperform the current accuracy on the "Butterfly Image Classification 50 species" dataset. CNN models always don't work well because of the skewed dataset and these kinds of datasets are tough to handle. Usual classification models and techniques often fail miserably when presented with such a problem. Although this custom model could get a 99% accuracy on such cases.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_53 (Conv2D)           (None, 62, 62, 32)        896
_________________________________________________________________
batch_normalization_53 (Batc (None, 62, 62, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_54 (Conv2D)           (None, 29, 29, 64)        18496
_________________________________________________________________
batch_normalization_54 (Batc (None, 29, 29, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 12, 12, 128)       73856
_________________________________________________________________
batch_normalization_55 (Batc (None, 12, 12, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_56 (Conv2D)           (None, 4, 4, 256)         295168
_________________________________________________________________
batch_normalization_56 (Batc (None, 4, 4, 256)         1024
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 50)                25650
=================================================================
Total params: 613,362
Trainable params: 612,402
Non-trainable params: 960
```

Figure 4. Proposed CNN model architecture

A custom new Sequential CNN model which is inspired from the AlexNet Architecture with a fixed and different kernel size and filters is used in the proposed model. The kernel size 3 is defined based on the colored image and the number of channels. In the architecture 4 convolutional layer including the MaxPooling layer, 1 Average-Pooling Layer, Global Pooling layer instead of flatten layer is used that will lead to a better feature map, reshape the environment in which the model's layers are visualized. The dropout method was used from TensorFlow to remove the overfitting. The activation function was chosen carefully while building this model. The activation function is the non-linear transformation that is used over the input images. The transformed result is then sent to the next neuron layer as input. The softmax activation function is used to the last dense layer in this proposed model as the number of image classes is much higher in this dataset. The softmax activation function calculates the relative probabilities to determine the final probability value. ReLU is also used in convolutional layer in this architecture. Batch normalization is also done in each convolutional layer.

Two fully connected dense layer with 256 neurons in the first layer 512 neurons in the second layer is used at the last of the model which received the from the neurons of the previous layer. The dense layer basically calculates matrix-vector multiplication which later on changes the dimension of the vector. The last dense layer visualizes the output with the help of the softmax activation function.

4. Results and Discussion

To check the needed accuracy, two CNN models are used. At first, Resnet50 is implemented with proper image augmentation with a total of 50 epochs and 128 batch sizes. The overall accuracy is not good enough while running in a personal machine but the accuracy is much higher in the Kaggle notebook which is 84%. In a personal machine, the accuracy is 70%. Then again VGG16 is implemented to verify the highest accuracy. But this time 99% accuracy is achieved by this model with 50 epochs. The main problem in Resnet50 was the lower image size which is then outperformed by VGG16. In this VGGnet16, 13 Convolutional Layers and 3 fully connected Dense layers are used to train the dataset.

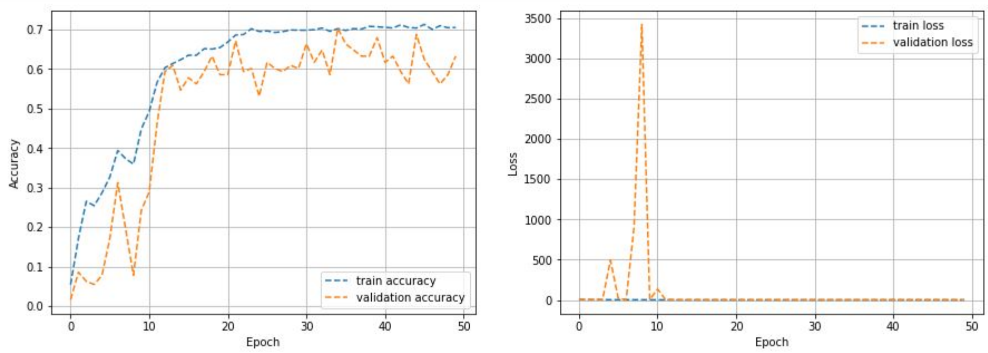


Figure 5. **Train and validation graph of Resnet50**

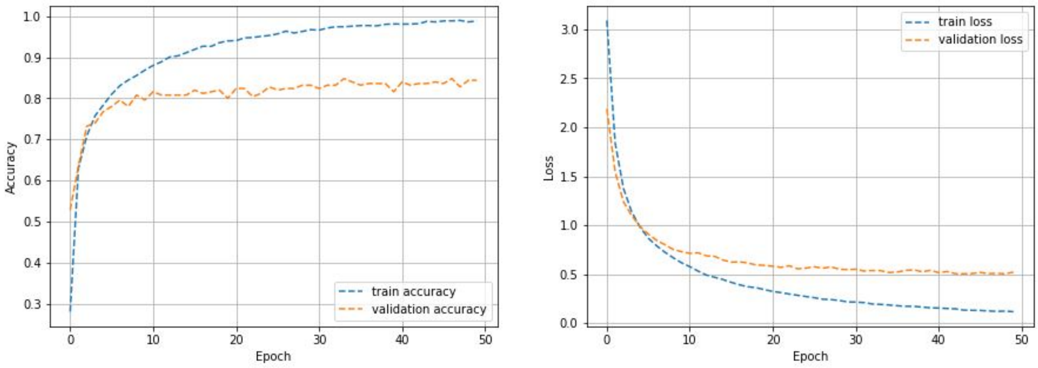


Figure 6. Train and validation graph of VGG16 model

In the new custom model, the highest accuracy of 99.74% is achieved with 50 epochs. But the validation accuracy is not that good. For this model, the mean-rgb method is used to pre-process the image dataset, and also proper reshape method is used before training the dataset. The image shuffle method is also used to shuffle the images of each classes.

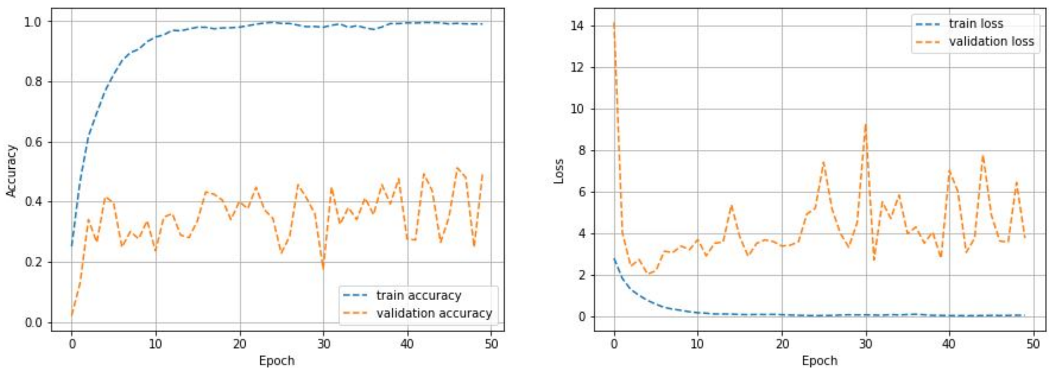


Figure 7. Train and validation graph of Proposed model

Table 1. All the achieved accuracy of CNN Models

Model Name	50th epoch accuracy (%)	Highest Accuracy (%)	Highest Validation Accuracy (%)
Resnet50	70.54	71.33	70.31
VGG16	99	99.23	84
Proposed New Model	99.07	99.56	51.04

According to the project instruction, single-layer visualization is also implemented for the VGG16 and the new custom CNN model. The trained filters are just weights in neural network , but by training the specific two-dimensional structure of the filters, the weight values have a spatial relationship to each other, therefore visualizing each filter as a two-dimensional picture is important.The first step is to review the filters in the model, to see what we have to work with.Small or inhibitory weights are represented by dark squares, whereas big or excitatory weights are represented by bright squares.

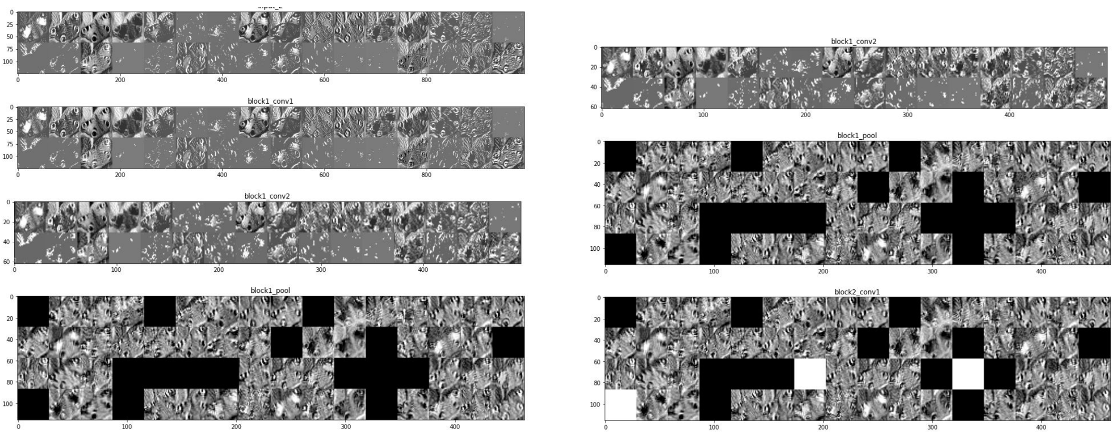


Figure 8. Layer by layer Visualization of proposed model

5. Conclusion

At the end of the project, it can be said the new CNN model outperformed the current accuracy but the validation accuracy result is not that good enough. Some future work could be done to increase the validation accuracy of the new CNN model. The VGG16 model also achieved the needed accuracy. After finshed all the training model it can be said that, if we take the higher image size then the accuracy will be little bit more higher.

6. Dataset Link

Link of the dataset : https://www.kaggle.com/gpiosenka/butterfly-images40-species

References

1. Dongjun Xin, Yen-Wei Chen and Jianjun Li, “Fine-Grained Butterfly Classification in Ecological Images Using Squeeze-And-Excitation and Spatial Attention Modules”.
2. Ayad Saad Almryad, Hakan Kutucu, “Automatic identification for field butterflies by convolutional neural networks”.
3. Lili Zhu and Petros Spachos, “Butterfly Classification with Machine Learning Methodologies for an Android Application”, 10.1109/GlobalSIP45357.2019.8969441