CIFAR-100: Object Recognition using EfficientNet-B0

Author: Chetna Khanna

Graduate Student, MS in Data Analytics Engineering, Northeastern University, USA

Abstract: Convolutional neural network (CNN) is a class of deep neural network commonly used to analyze images. The objective of this project is to build a convolutional neural network model that can correctly recognize and classify colored images of objects into one of the 100 available classes for CIFAR-100 dataset. The recognition of images in this project has been done using transfer learning approach. The network built in this project uses the state-of-the-art EfficientNet-B0 which was trained on the popular, challenging and large ImageNet dataset. Transfer learning and the idea of intelligently scaling the network (carefully balancing the network's width, depth and resolution) helped in getting a good performance on this dataset. By just training the model for 15 epochs, the model managed to achieve an accuracy of 82 percent. This is definitely a much better performance than the one achieved using a 9-layer convolutional neural network model trained for 100 epochs. The training of the model has been done on a GPU and the model has also been tested on some new random images to visualize the top 5 category predictions along with their probabilities.

Keywords: Deep Learning, Neural Networks, Transfer Learning, Convolutional Neural Network, EfficientNet, EfficientNet-B0, CIFAR-100, Object Recognition, Image Classification

1 INTRODUCTION

CIFAR-100 is a labeled subset of 80 million tiny images dataset where CIFAR stands for Canadian Institute For Advance Research. The images were collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. The dataset consists of 60000 colored images (50000 training and 10000 test) of 32×32 pixel in 100 classes grouped into 20 super classes. Each image has a fine label (class) and a coarse label (super class). The Python version of this dataset has been downloaded from the website of University of Toronto Computer Science and used for the project.

Convolutional neural network is a class of deep neural network commonly used to analyze images as it works well in extracting important features from the images. The initial layers of convolutional neural network helps in extracting simple features from the images and the complexity of feature extraction increases as we move towards the output layer from the input layer.

The objective of this project is to use transfer learning to build a convolutional neural network model that can correctly recognize and classify colored images of objects into one of the 100 available classes. Image recognition is a very simple task for human beings but when it comes to machine it is a complex task because there is a possibility of so many permutations (almost infinite) for an object based on its position and lightning effect in the image.

The motivation behind working on this dataset using transfer learning is the challenge of achieving a good accuracy score (more than 59 % as achieved using a 9-layer convolutional neural network built earlier). The dataset has 100 classes but just 600 images in each class (500 for training and 100 for testing). The most interesting part of this dataset is the image quality. Each of the image in the dataset is of 32×32 pixels which makes recognition a challenging task for machine. So in order to train the machine to correctly recognize and classify the images better than earlier, transfer learning approach has been used. However, the main limitation for building a deep neural network for CIFAR-100 with millions of parameters is memory. But, I felt that dealing with all these challenges would be a great learning and I decided to proceed with this dataset.

In order to build a model that can give good performance, I built a deep neural network using transfer learning and trained the same on NVIDIA Tesla K80 GPU with 8 vCPUs having 30 GB memory, all provided by Google Cloud Platform. The training happened really fast and I think the training can be performed on my local machine as well. The model has been built using the Keras library written in Python along with other data science libraries of Python and the open source web application, Jupyter Notebook.

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. As stated in the Handbook of Research on Machine Learning Applications, Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. EfficientNet is a new family of CNNs built by Google. These CNNs not only provide better accuracy but also improve the efficiency of the models by reducing the number of parameters as compared to the other state-of-the-art models. This project paper uses EfficientNet-B0 model which is a simple mobile-size baseline architecture and trained on ImageNet dataset. EfficientNet works on the idea that providing an effective compound scaling method for increasing the model size can help the model achieve maximum accuracy gains (see figure 1).

According to the paper, original images should be resized to the specified size, which is (224, 224) in case of EfficientNet-B0 (see figure 2). Bicubic interpolation method is used for upscaling the images.

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Fig 2. EfficientNet-B0 Model Architecture - Image from original paper

I used EfficientNet-B0 as my model with ImageNet weights and added my own pooling and dense layers. For pooling, Global average pooling layer has been added in the model to reduce the spatial size of the representation (downsampling) which in turn reduced the number of parameters and computation cost and thus helped in controlling the problem of overfitting.

In the output layer, Softmax activation function has been used to ensure that the final activations sum up to 1 and thereby fulfill the constraints of probability density. These probabilities later helped in analyzing the model prediction for some new random images.

Dropout technique has also been used to prevent the model from overfitting. Since the outputs of the layers participating in dropout are randomly subsampled thus, it prevented overfitting which could have occured due to the involvement of millions of parameters in the training of a deep neural network. The model built for this project has around 4.2 million parameters so it had a lot of chances of overfitting which had been avoided using dropout.

The Adam optimization algorithm has been used in the model for optimization as it helped the model converge faster and so was more computationally efficient than other optimization algorithms. It also required lesser memory and worked well for my model and also gave good results by very little tuning of its hyperparameter.

The dataset involved a classification task with multiple classes so categorical cross entropy loss has been used. Accuracy as the performance metrics has been used to calculate the accuracy score of the classification model.

Early stopping and reduce learning rate on plateau techniques have been used in the model to monitor the validation loss. Using early stopping, the training can be stopped as soon as the model stops improving on the validation dataset. This has helped to avoid both underfitting by choosing too few epochs and overfitting by choosing too many epochs. Reduce learning rate on plateau helps in adjusting the learning rate as soon as the plateau is detected during the training. As the very first sign of no further improvement may not be the best time to stop so a patience argument has been set to add a delay in monitoring to see any signs of further improvement.

CIFAR-100 dataset has a limited number of images (600) in each class, so in order to help the model learn better and make better predictions, real-time data augmentation has been done to artificially expand the training dataset size by creating modified versions of the existing images in each class. This has helped the model to become robust and even the accuracy improved with the availability of more data. As memory was a topic of concern for my deep neural network, so this real-time data augmentation was a good choice for dataset expansion. The same has been performed using the Python library, Albumentations, as it is fast and supports all common computer vision tasks.

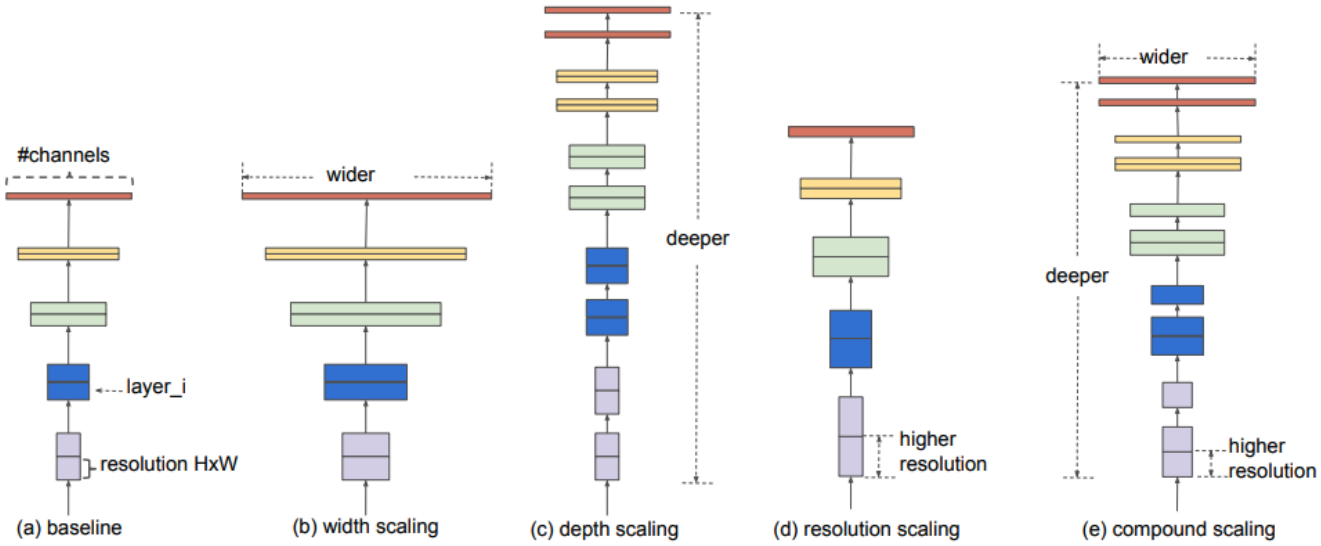

Fig 1. Model Scaling - Image from original paper

The model has been trained for 15 epochs using a batch size of 8. The hyperparameter, batch size, has been chosen carefully so that the number of samples processed before making an update in the model parameters is not a big value. It helped in offering regularization effect to the model and even lowered the generalization error. The overall training process required less memory using this batch size and the network got trained faster as compared to entire training set or a lower batch size.

The best model has been saved using the minimum validation loss as the saving criteria. The training and validation loss versus number of epochs, and training and validation accuracy versus number of epochs have been plotted to visualize the performance.

The model was then evaluated on the test dataset and some new random images. For each new image, the top 5 predictions made by the model was displayed along with their probabilities.

2 BACKGROUND

In the recent past, deep neural networks has helped the researchers achieve good performance on a variety of problems. Images being an important aspect of our life have always been a great area of interest and training a machine to correctly recognize and classify images have always been an important and interesting study area. The enhancement in processing power and ample storage has increased the scope and use of convolutional neural networks extensively. Also, the accessibility of pre-trained deep neural network models and the scope to use transfer learning is helping in achieving better performance on many tasks.

CIFAR-100 is a classic dataset with 100 classes which makes classification task a challenge. In order to achieve performance better than the existing ones a lot of researches have been done. With a lot of experimentation, the accuracy has been improved from 54.23 % to 93.51% from 2012 to 2019 by using techniques like Wide Fractional Max-Pooling, Residual Network, ShakeDrop Regularization, Pipe Parallelism, Efficient Net, Big Transfer (BiT) to name a few.

The current state-of-the-art in CIFAR-100 uses a method called EfficientNet-B7 which has been mentioned in a paper named "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2019)". The research revolves around understanding the reasons that are preventing ConvNet from achieving better accuracy and efficiency. The researchers studied ConvNet scaling and identified that careful balancing of network width, depth and resolution is an important part of achieving better accuracy but is missing in ConvNet. In order to address this issue, a simple and highly effective compound scaling method has been proposed which enables easy scaling up of a baseline ConvNet to any target resource constraints. The effectiveness of this method has been demonstrated in the scaling up of MobileNet and ResNet and then a new baseline network called EfficientNet has been designed which provided accuracy better than ConvNet.

3 APPROACH

The Python version of the dataset used in this project has been downloaded form the website of University of Toronto Computer Science. The available files were

Python pickled object produced using cPickle. Python Pickle module has been used for the purpose of serializing or deserializing these objects in Python. The load() method of pickle has been used to read these files and analyze the structure of the files and the images.

The libraries used for the project include pickle, pandas, numpy, matplotlib, seaborn, tensorflow, keras, sklearn, skimage, cv2 and albumentations.

In order to perform the task of image recognition and classification, a convolutional neural network has to be built which require a 4D array as the input. So, the dataset has been transformed to acquire that shape. The training dataset had 50000 images with the shape (50000, 3072). I have transformed these images to acquire the following shape using the reshape and transpose operation of numpy array:

(Number of instances × Width × Height × Depth)

The width, height and depth are the dimensions of the image where depth is nothing but the number of color channels in the image which is 3 in our case as the images are RGB. The following diagram illustrates the form of 4D input for the convolutional neural network model.

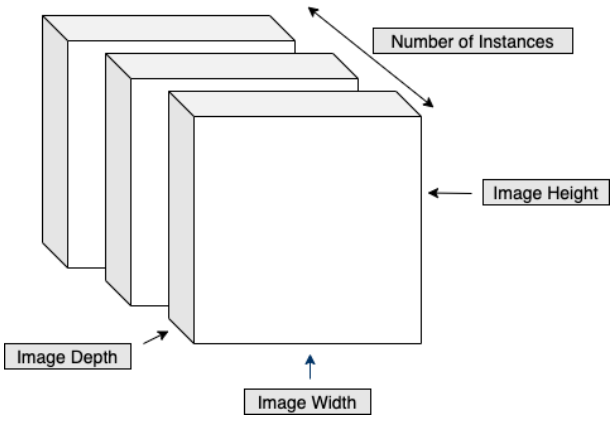

Fig 3. Input shape

Here, is the display of some of the random images from the dataset along with their true labels.

Fig 4. Images with true labels

In order to make predictions, the labels of the images have been converted to categorical matrix structure from the existing 1D numpy array structure.

The CIFAR-100 dataset had two separate datasets, training and testing. I wanted to keep the testing dataset unknown until the hyperparameter tuning and model optimization, so I made a validation split from the training dataset with a split size of 0.2. Splitting of the dataset has been done using stratified shuffle split as it preserves the percentage of samples in each of the class. The dataset now has 40000 testing images and 10000 validation images along with 10000 testing images. This validation dataset was then used for monitoring the loss and accuracy against the training dataset.

In order to apply transfer learning, images have been resized as per the requirement of EfficientNet-B0 architecture. Since, the images are 32 × 32 pixels, so bicubic interpolation method is used to upscale the images. It considers the closest 4 × 4 neighborhood of known pixels - for a total of 16 pixels. This method produces noticeably sharper images and is the ideal combination of processing time and output quality.

As the performance of deep learning model usually improves with the addition of more data, I planned for image augmentation but memory was a big constraint as the model already had a lot of trainable parameters. So, I opted for the albumentations library of python which helped in real-time data augmentation. I created my own custom data generator using the keras data generator class. The parameters, horizontal flip, vertical flip, grid distortion and elastic transformation was tuned to extend the dataset. As the distribution of the feature values in the images can be very different from each other, the images are normalized by dividing each image by 255 as the range of each individual color is [0,255]. Thus, the rescaled images have all features in the new range [0,1]. All these transformations was done batch-wise, which was set as 8. I only applied augmentation to the training dataset and left the validation and test dataset as it is. The image data generator class helped the model learn better using more images in each class.

The EfficientNet class is available in keras to help in transfer learning with ease. I used the EfficientNet-B0 class with ImageNet weights. Since, I used the EfficientNet-B0 model just for feature extraction, I did not include the fully-connected layer at the top of the network instead specified the input shape and pooling. I also added my own pooling and dense layers. The pooling technique helps in extracting the maximum features from the images. The global average pooling operation has been used in the model which calculates the average output of each feature map in the previous layer. This simple operation with no trainable parameter downsamples the image and highlights the most present feature in each and every patch of the feature map.

Dropout has been used in the hidden layer with value 0.5. This dropout value worked well and was computationally

efficient as well. Upon using the dropout as 0.2 in the hidden layer, the model showed overfitting.

The output dense layer uses Softmax as the activation function to give probabilities as the output by squashing the values in the range [0,1]. This has been done to ensure that the final activations all sum up to 1 and fulfill the constraints of probability distribution.

As the dataset has multiple classes, so categorical cross entropy loss has been used as it measures the performance of a model that outputs a probability value between 0 and 1. An increase in cross entropy loss showed us that the predicted probability is diverging from the actual label and vice-versa. The same has been used to plot the graph and monitor the validation loss.

Accuracy has been used as the performance metrics to monitor the performance of the model. A lot of fluctuations in the same was seen during the training process with different values for hyperparameters.

For training the model, Adam optimizer with learning rate 0.0001 was used in the model training as this value of learning rate helped the model converge faster and provided the best accuracy amongst all the models. The learning rates of 0.01 and 0.001 depicted the inability of the model to learn anything whereas the learning rate of 0.00001 showed that the model was able to learn but at a rate slower than as compared to learning rate 0.0001 and even the performance was lower.
I also used RMSProp optimizer with the same learning rate as in Adam (0.0001) and witnessed a lower performance (a difference of 1%). So, the model was finally trained at learning rate 0.0001 using the Adam optimizer.

In order to stop the training after a certain number of epochs if the validation loss is not reduced further, I used the technique called early stopping. I have monitored validation loss and kept a patience number of epochs to be 10 to help the model add some delay in stopping by giving model a chance to improve further. The number of epochs on which the training stopped was returned and only the best model has been saved for further usage.
I also adjusted learning rate on plateau with patience 5 and minimum learning rate value as 0.000001 and factor 0.5. It helped in adjusting the learning rate on 14th epoch as soon as the plateau is detected during the training.

The model was trained for just 15 epochs by keeping time and memory constraint in mind.

Upon compiling the model, the summary depicted a total of 4,177,664 trainable parameters which was very less than 13,870,484 trainable parameters in my earlier model that did not use transfer learning.

The model was then fit on the extended dataset and its history has been stored so that the graphs of loss versus number of epochs and accuracy versus number of epochs can be plotted to monitor the model performance. Confusion matrix for the model has been plotted but since there are 100 classes in the dataset so interpretation from its visualization was a challenge. Finally, the classification report was generated which showed which category has low precision and/or recall.

The predictions from the model have been visualized along with the true label to see how the model is performing on the unknown testing dataset. Some new random images were also tested to know the top 5 predictions by the model.

4 RESULTS

This project is based on CIFAR-100 dataset which has 60000 colored images of 32 × 32 pixel categorized in 100 classes and 20 superclasses collected by researchers at University of Toronto Computer Science. The task involved in the project is to build a convolutional neural network model using transfer learning so that we can correctly recognize and classify these images.

A lot of experiments have been performed in optimizing the model and selecting the best one. The following graph shows the best accuracy achieved by building a 9-layer convolutional neural network model by fine tuning the hyperparameters. The graph has been plotted between training and validation accuracy and number of epochs.

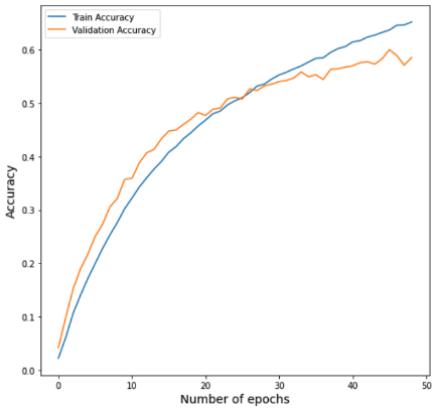

Fig 5. Accuracy of 9-layer CNN model

The following graph shows the training and validation loss of the 9-layer convolutional neural network model.

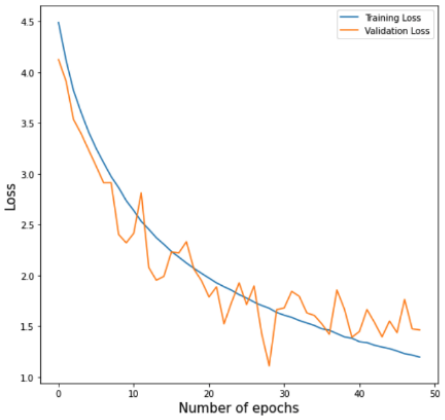

Fig 6. Loss of 9-layer CNN model

The following graph shows the training and validation loss and accuracy of the model built using transfer learning from EfficientNet-B0 against the number of epochs.

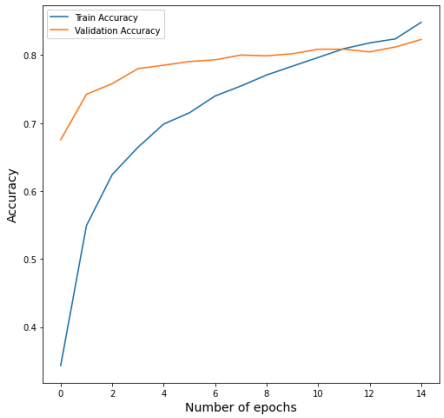

Fig 7. Accuracy of final model

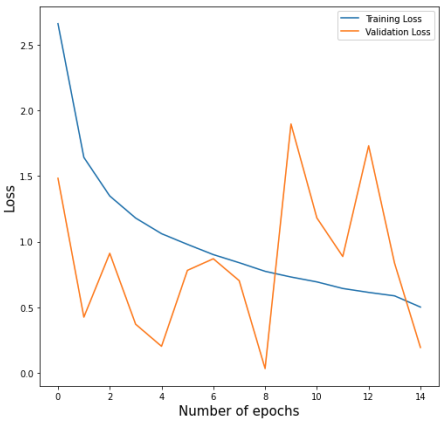

Fig 8. Loss of final model

In order to prevent overfitting, early stopping and reduce learning rate on plateau was implemented in the model and the model adjusted the learning rate at 14th epoch after a patience of 5 epochs to provide the best model for the selected hyperparameters. The reported accuracy is 82% on the testing dataset and a loss of 0.19. The confusion matrix and classification report for the model has been generated which depicts that in most of the categories the precision and recall are high but a few categories still have a low value.

```
[[93  0  0 ...  0  0  0]
 [ 0 92  0 ...  0  0  0]
 [ 0  0 81 ...  0  0  0]
 ...
 [ 0  0  0 ... 88  0  0]
 [ 0  0  3 ...  0 66  0]
 [ 0  0  0 ...  0  1 84]]
```

Fig 9. Confusion matrix for the final model

	precision	recall	f1-score	support
Category 0	0.89	0.93	0.91	100
Category 1	0.88	0.92	0.90	100
Category 2	0.60	0.81	0.69	100
Category 3	0.75	0.82	0.78	100
Category 4	0.72	0.71	0.71	100
Category 5	0.80	0.78	0.79	100
Category 6	0.87	0.90	0.88	100
Category 7	0.92	0.78	0.84	100
Category 8	0.91	0.96	0.94	100
Category 9	0.89	0.92	0.91	100
Category 10	0.71	0.55	0.62	100
Category 11	0.52	0.39	0.45	100
Category 12	0.85	0.80	0.82	100
Category 13	0.83	0.80	0.82	100
Category 14	0.95	0.82	0.88	100
Category 15	0.83	0.92	0.87	100

Fig 10. Classification report for a few categories

The below image depicts the predictions made by the model and the true labels. It can be seen that the model misclassified when the images had multiple objects. For instance, the third image had multiple trees so it classified it as forest instead of willow tree.

Fig 11. True and predicted labels

The model has also been tested on some new random images and the top 5 predictions along with their probabilities have been depicted using a graph to get visual clarity.

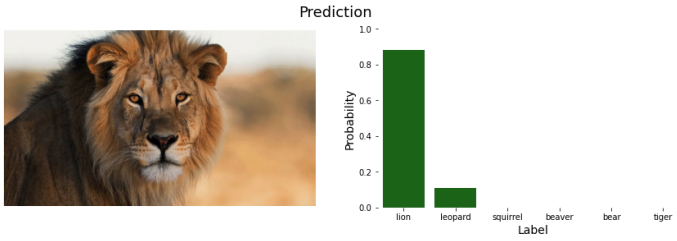

Fig 12. First correct prediction

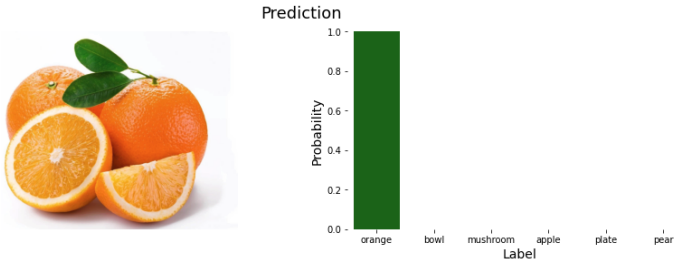

Fig 13. Second correct prediction

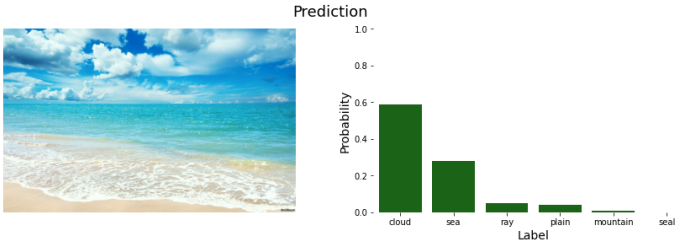

Fig 14. First incorrect prediction

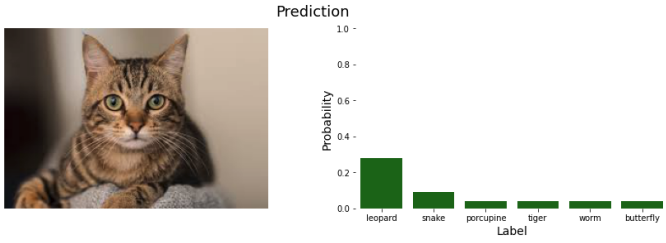

Fig 15. Second incorrect prediction

It can be seen that the model correctly predicted a lion and an orange and that too with high probability but was unable to predict the sea and cat correctly. The cat was predicted as a leopard and the surprising part was that the option of cat was not even among the top 5 options. Similarly, a compound image of sea with clouds and land was predicted as cloud but atleast sea was the second closest option.

4.1 Discussion

Conclusion: Transfer learning was successfully applied on the CIFAR-100 dataset using EfficientNet-B0. After training the model for just 15 epochs an accuracy of 82% was achieved on the test dataset. The reported validation loss and accuracy is 0.19.
The results depict that it is highly likely that the scope of the project can be extended further to gain better accuracy. Complex systems can be built which can facilitate image recognition in complex tasks like medical analysis, face detection for security purposes.

Future Work: The CIFAR-100 dataset is a large dataset and has a lot of categories to classify the images. Transfer learning has helped in improving the performance of the model. However, the performance can be improved further using the other state-of-the-art versions of EfficientNet. As stated in the original paper, the highest accuracy achieved on CIFAR-100 is 92% using the EfficientNet-B0 model.

Optimizers like SGD, Nesterov momentum can be used to monitor the performance of the model as it is said that these optimizers speed up the training process and improve convergence significantly. I tried using RMSProp in my model but did not see much difference in performance. However, I feel that this might be due to less hyperparameter tuning for this optimizer. All the hyperparameters used in building this model can be tuned further with different values and combinations with each other. There is a high chance of increment in the model performance by their further tuning.

Since the dataset has very few images in each class so the addition of more images in this dataset could possibly and positively affect the model performance.

The data augmentation technique needs to be chosen carefully by knowing the dataset context and thus a lot of experimentation is required to come up with the best augmentation techniques for a dataset. This is one more area which can lead to improvement in the performance.

The number of epochs can be increased further to allow the model to train for a longer duration.

5 CONCLUSION

Recognition of different images is a simple task for we humans as it is easy for us to distinguish between different features. Somehow our brains are trained unconsciously with a similar type of images that has helped us distinguish between features (images) without putting much effort into the task. For instance, after seeing a few cats, we can recognize almost every different type of cat we encounter in our life. However, machines need a lot of training for feature extraction which becomes a challenge due to high computation cost, memory requirement and processing power.

The transfer learning model built in this project for CIFAR-100 dataset recognizes and classifies colored images of objects in one of the 100 available categories with 82% accuracy. The reported loss is 0.19. The model used techniques like early stopping, reduce learning rate on plateau and dropout to avoid overfitting.

Even after training the model with millions of parameters, the model predicted the class for a few images

completely wrong. It is considered that the performance of a deep learning model increases with the amount of data used in its training. It is believed that the accuracy of this dataset can be further improved by using the other versions of EfficientNet.

6 ACKNOWLEDGEMENT

I would like to express my great appreciation to Professor Jerome J. Braun for his valuable and constructive suggestions during the planning and execution of this project. It is due to his motivation, guidance and support that I was able to work on such a big project and learn to apply different techniques required in building a deep convolutional neural network. His willingness to give his time generously to each and every student of the class is and will always be appreciated.
I would like to offer my special thanks to my husband and daughter for being my constant support during my journey of learning. Their encouragement and support is my biggest strength.
A big thanks to Google Cloud Platform for providing free credits and GPU which made training hassle-free.

References

1. CIFAR-100 dataset:
https://www.cs.toronto.edu/ kriz/cifar.html

2. Stanford Convolutional Neural Networks for Visual Recognition:
http://cs231n.stanford.edu/

3. Original research paper:
https://arxiv.org/pdf/1905.11946.pdf

4. Keras API Docs:
https://keras.io/api/
https://keras.io/api/applications/efficientnet/efficientnetb0-function

5. Udemy Deep Learning A-Z: Hands-On Artificial Neural Networks

Github Repo:
https://tinyurl.com/ydf62tl2