CIFAR-100: Object Recognition

Author: Chetna Khanna

Guided by: Professor Jerome J. Braun

Graduate Student, MS in Data Analytics Engineering, Northeastern University, Seattle, 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 has been done using a 9-layer convolutional neural network model. The model uses techniques and processes like max pooling, zero padding, ReLU activation function (for hidden layers), Softmax activation function (for output layer) and Adam as the optimizer. In order to avoid overfitting, techniques and processes like dropout, early stopping have been used. As training a deep learning model on more data can lead to more skillful and robust model, so on-the-fly image data augmentation has been used to expand the dataset. By taking batches of size 64 and training the model for 100 epochs on a GPU, an accuracy of 59 percent has been achieved. 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, Convolutional Neural Network, CIFAR-100, Object Recognition

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 superclasses. Each image has a fine label (class) and a coarse label (superclass). 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 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 is the challenge of achieving a good accuracy score because 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, a deep neural network needs to be built. 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 decided to proceed with this dataset.

In order to build a model with good performance, I built a deep neural network and trained the same on

NVIDIA Tesla K80 GPU with 8 vCPUs having 30 GB memory, all provided by Google Cloud Platform. The total training time was approximately one and a half hour. 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.

The goal of this project has been achieved by building a 9-layer convolution neural network model which has been finalized after a lot of tuning of number of layers and number of filters/units in each layer. The ConvNet architecture of this model has three stacks of CONV-RELU layers followed by a POOL layer and then two fully connected (FC) RELU layers followed by a fully connected output layer. This is one of the good combination of layers for a larger and deeper neural network because multiple stacks of CONV-RELU layers help in extracting more complex features of the input image before undergoing a pooling operation. This 9-layer network helped in getting a good accuracy on not only the training set but also on the testing set.

In order to introduce non-linearity in the model Rectified Linear Unit (ReLU) was used as the activation function for the hidden layers as it is sparse and has reduced likelihood of vanishing gradient problem. ReLU showed good convergence performance and was computationally efficient as well.
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.

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. The CIFAR-100 dataset has images of low quality so a pooling technique was needed which can extract the maximum features from these images. As the max pooling operation calculates the maximum or the largest value in each patch of the feature map, so the same has been used to downsample the image and highlight the most present feature in each and every patch of the feature map.

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 can occur due to the involvement of millions of parameters in the training of a deep neural network. The model built for this project has around 13.8 million parameters so it had a lot of chances of overfitting which had been avoided using dropout.

Using padding, additional pixels can be added to the images while training a convolutional neural network. In order to preserve the same dimension of input in the output, zero-padding has been used in the model by symmetrically adding zeroes to the input matrix.

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 technique has also been used in the model to monitor the validation loss so that 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. 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 terms of the number of epochs on which the training can be monitored 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 the 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 model has been trained for 100 epochs using a batch size of 64. 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.

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 fine tune the same is helping a lot in performance increment and further advancements.

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-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 EffectiveNet 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, pylab, tensorflow, keras, sklearn and skimage.

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. For instance, 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 $\times$ Width $\times$ Height $\times$ 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.

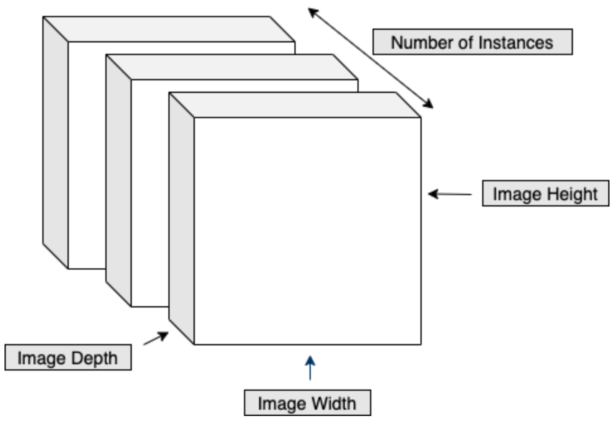

Fig 1. Input shape

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

Fig 2. 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.

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].

The classification task required building a convolutional neural network and since there are 100 classes in the dataset the task can be completed with better performance by building a deep neural network to help the model learn better. I started with 5-layer network with 32, 64 and 128 filters in convolutional layers and 256 units in the first fully connected layer. This network had 643,492 trainable parameters and helped me achieve an accuracy of 38.7% on the validation set and 39.5% on the test set. The graphs of loss versus number of epochs and accuracy versus number of epochs for the same has been shown for comparison in the results section.

Upon optimization of the model further by increasing the number of layers and filters in convolutional layers and number of units in fully connected layer, I later on finalized a model with a 9-layers having 13,870,484 trainable parameters. The ConvNet architecture of this 9-layer deep neural network has 3 stacks of CONV-RELU layers followed by a POOL layer and then 2 fully connected (FC) RELU layers and a final fully connected output layer.

INPUT $\rightarrow$ [[CONV $\rightarrow$ RELU] $\times$ 2 $\rightarrow$ MAX-POOL] $\times$ 3 $\rightarrow$ [FC $\rightarrow$ RELU] $\times$ 2 $\rightarrow$ FC

The specifics of each layer are as follows:

1. Each CONV layer uses zero-padding to ensure than the dimension of output is same as the dimension of input.

2. ReLU has been used as the activation function in all the hidden layers because of its sparsity and advantage to avoid vanishing gradient problem. While training my neural network ReLU showed good convergence performance and was also computationally efficient.

$$\mathrm{f(x) = max(0,\ x)}$$

The reason for this is that ReLU does not activate all the neurons at the same time due to its nature to choose the maximum value between 0 and x. As a few inputs can have negative value too, so ReLU takes the maximum value as 0 and do not activate them. As my model had millions of parameters, this helped to converge faster.

3. A small kernel size of 3 has been used for all the CONV layers as the dataset has 32 $\times$ 32 pixels images and I wanted to extract most of the details form these images. It also lowered down the number of weights in my network and helped in faster training.

4. Max pool of size 2 has been used in the model as max pooling calculates the maximum value in each patch of the feature map. The stride of 2 has been used in the model to move filters 2 pixels at a time and to downsample the image by 2 in both width and height.

5. The number of filters used in the first stack of convolutional layers is 128, the number of filters used in the second stack of convolutional layers is 256 and the number of filters used in the third stack of convolutional layers is 512. Since, CIFAR-100 has 100 classes to classify each image, thus large number of filters helped in achieving better performance. The process of tuning these filters started from 32 and was gradually increased while monitoring the validation loss. The graphs for the same are shown in the results section to show performance difference in models.

6. The output of the convolution and pooling layer is flattened into a single vector of values and fed to the fully connected layers.

7. Three fully connected layers with 1000 units in the first two layers and 100 units (units equal to the number of classes) in the last (output) layer have been used to make the network dense and learn the non-linear combinations of features obtained from the convolutional layers. This combination of units and layers helped the network classify images with more accuracy (5% more than the network with only two fully connected layers).

8. The output 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.

9. Dropout has been used in all the hidden layers with values 0.2 in the input layer and 0.5 in the hidden layers. Upon using the dropout as 0.5 in the input layer as well, the model got low accuracy due to lesser number of parameters and the use of 0.2 in all the layers caused memory issues due to more number of parameters to train. This setting of 0.2 and 0.5 worked well and was computationally efficient as well.

Upon compiling the model, the summary depicted a total of 13,870,484 trainable parameters in the model which is a huge number and required GPU and large storage for fast and efficient computation. So, the model was trained on a GPU along with 8 vCPUs and took a training time of one and a half hour.

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. 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 by performing multiple rounds of hyperparameter tuning.

For training the model, a lot of techniques have been used which required careful tuning of hyperparameters. The 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 so far. The learning rates of 0.01 and 0.001 depicted the inability of the model to learn anything (accuracy around 1%) 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 (36.96% as compared to 59%).
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.

As our dataset had multiple classes to classify any image, 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.

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 20 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. It was noticed that 100 epochs were overfitting my model and thus early stopping helped the training stop at the most appropriate epoch.

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 Keras' ImageDataGenerator() option which helped in real-time data augmentation. The parameters, shear, zoom, horizontal flip, featurewise center, width shift was tuned to extend the dataset and helped the model learn better using more images in each class.

The model was trained for 100 epochs by keeping a batch size of 64 upon considering time and memory constraint in mind. The batch size of 32 took more time in training with a slightly lower accuracy and more processing time.

The model was then fit on this 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.

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. A lot of research has been done in order to achieve better performance since this dataset has been made public. The task involved in project is to build a convolutional neural network which can correctly recognize and classify these images.

A lot of experiments have been performed in optimizing the model and selecting the best model. The following graphs show the fine tuning of learning rate for Adam optimizer and how the model improved which changing rates (keeping the same number of trainable parameters, epochs and batch size). The graphs have been plotted between training and validation accuracy and number of epochs.

It is clear from the graphs that the model is not learning at learning rates 0.01 and 0.001 whereas a lot of learning has been happening at the learning rate of 0.00001. However, the final optimized model uses Adam optimizer and learning rate 0.0001 as it provided better accuracy than model with learning rate 0.00001.

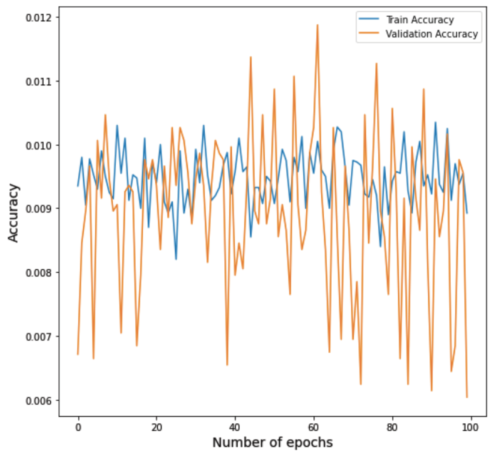

Fig 3. Adam optimizer with learning rate 0.01

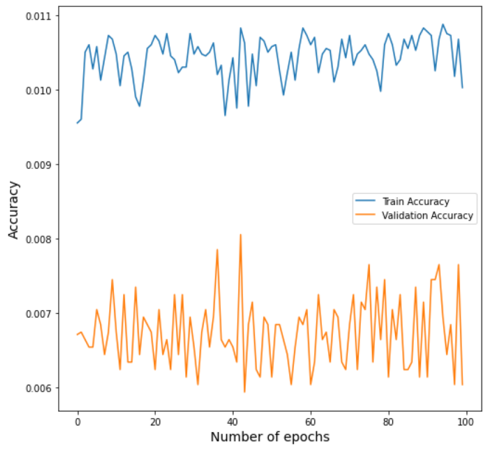

Fig 4. Adam optimizer with learning rate 0.001

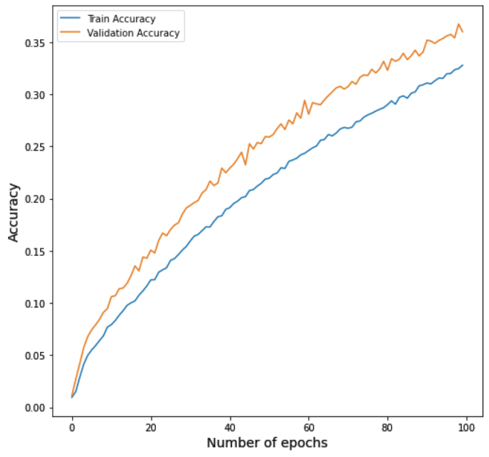

Fig 5. Adam optimizer with learning rate 0.00001

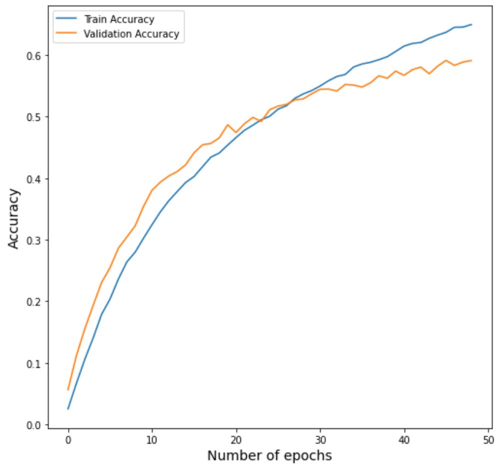

Fig 6. Accuracy of final model using Adam optimizer (learning rate 0.0001)

The following graph depicts how Adam optimizer outperforms RMSProp at the same learning rate of 0.0001 by around 1% in accuracy. The comparison can be done in between Fig 6 and Fig 7.

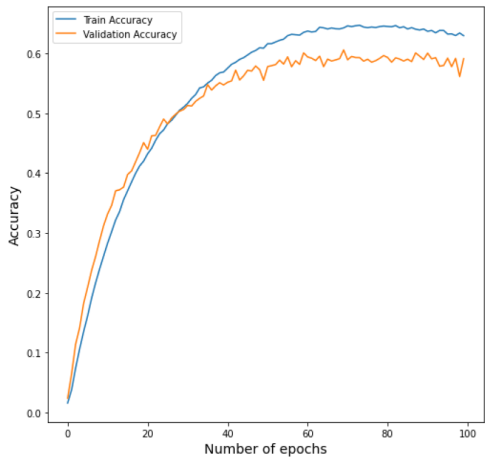

Fig 7. RMSProp with learning rate 0.0001

The number of layers and filters and/or units in each layer are tuned several times to achieve better and better accuracy. Graph of accuracy for a 5-layer neural network model with just 643,492 parameters is shown below.

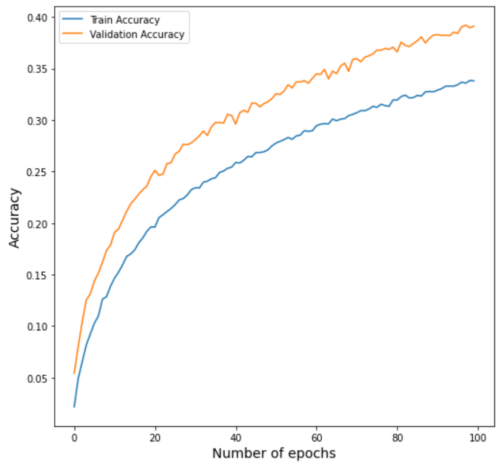

Fig 8. A 5-layer neural network with validation accuracy 38.74% and loss 2.3989

The graph of loss versus number of epochs for the finalized model is shown below. In order to prevent overfitting, early stopping was implemented in the model and the model stopped training at 49th epoch after a patience of 20 epochs to provide the best model for the selected hyperparameters. The reported accuracy is 59% on the testing dataset and a loss of 1.47.

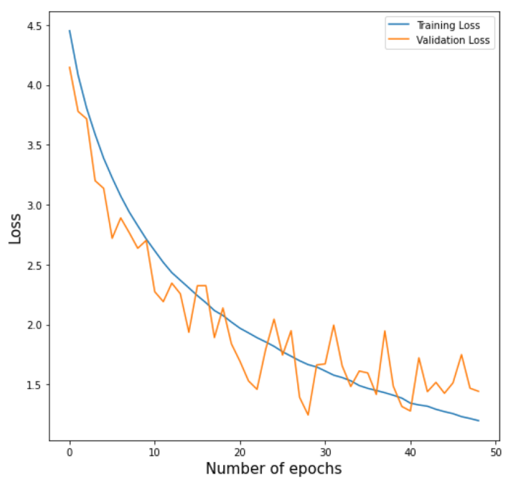

Fig 9. Final model depicting the training and validation loss

The confusion matrix and classification report for the model has been generated which depicts that in most of the categories the precision and recall both are low. This suggests that the model needs further optimization.

```
[[74  0  0 ...  0  0  0]
 [ 0 72  0 ...  1  0  0]
 [ 0  1 39 ...  0  8  1]
 ...
 [ 0  1  0 ... 63  0  1]
 [ 0  0  5 ...  0 35  4]
 [ 1  0  0 ...  0  0 68]]
```

Fig 10. Confusion matrix for the final model

```
              precision    recall  f1-score   support

Category 0         0.81      0.74      0.77       100
Category 1         0.74      0.72      0.73       100
Category 2         0.45      0.39      0.42       100
Category 3         0.53      0.17      0.26       100
Category 4         0.53      0.09      0.15       100
Category 5         0.30      0.56      0.39       100
Category 6         0.56      0.57      0.56       100
Category 7         0.72      0.29      0.41       100
Category 8         0.48      0.70      0.57       100
Category 9         0.71      0.64      0.67       100
```

Fig 11. 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.

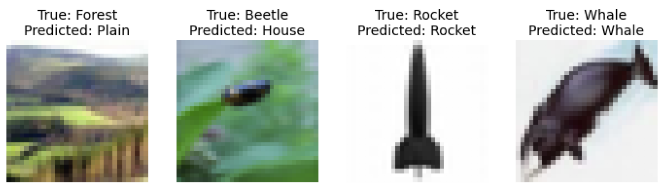

Fig 12. True and predicted labels

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.

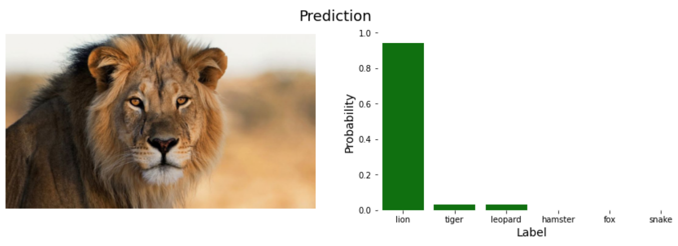

Fig 13. Correct prediction

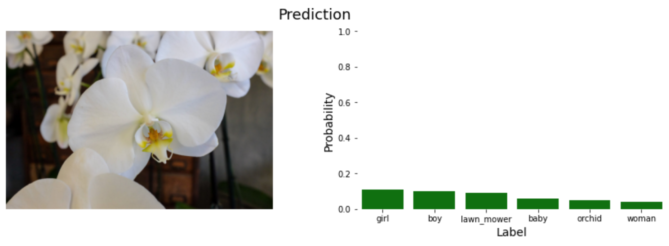

Fig 14. Incorrect prediction

It can be seen that the model correctly predicted a lion and that too with high probability but was unable to predict orchid. It can be seen that it was unable to predict the image as orchid however orchid is among the top 5 predictions given by the model.

4.1 Discussion

Conclusion: The final model for this project is a 9-layer deep neural network with 13,870,484 trainable parameters. After training the model for just 49 epochs an accuracy of 59.28% has been achieved on the validation dataset and 59.09% on the testing dataset. The accuracy on training dataset is 64.96% if we train the model for just 49 epochs to avoid overfitting. The reported validation loss is 1.42 and the test loss is 1.47.
The results depict that it is highly likely that the scope of the project can be extended further to gain much better accuracy. Complex systems can be built which can facilitate image recognition in complex tasks like medical analysis, face detection for security purposes with a 100% accuracy to avoid any misrecognition.

Future Work: The CIFAR-100 dataset is a large dataset and has a lot of categories to classify the images. Creating a deep neural network for this dataset has clearly helped in increasing the performance of the model. However, the network could have been made even more deeper and/or wider provided there is enough storage memory and computation power. Adding more GPUs and storage memory can definitely help the model perform better on not only the training dataset but testing dataset as well.

There is a possibility that ELU activation function can perform better than ReLU activation function if applied on the hidden layers as ReLU does not support negative values which results in negative mean activation creating a bias shift. ELU on the other hand allows negative values too and thus brings the mean activation closer to zero. However, I was unable to find much documentation supporting and using the same. After applying the same in my case, I hardly saw difference in performance of my model. It is quite possible that it can be beneficial for deeper and/or wider networks and thus can be an area of experimentation.

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 model can be optimized for larger kernel sizes as well like 5 or 7 to see if the same can help improve the model performance. Even an experiment can be done with stride as well.

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 along with an increase in patience for early stopping to monitor the model to improve further.

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 9-layer deep neural network model built in this project for CIFAR-100 dataset recognizes and classifies colored images of objects in one of the 100 available categories with 59% accuracy. The ConvNet architecture of the model has three stacks of CONV-RELU layers followed by a POOL layer and then two fully connected (FC) RELU layers followed by a fully connected output layer. The model uses 13,870,484 trainable parameters which has been trained for an hour an half on a GPU with 8vCPUs. The Adam optimizer with learning rate 0.0001 and categorical cross entropy loss has been used to used to support the training process which involved 100 epochs and 64 as the batch size. The reported loss is 1.47. The model used techniques like early stopping 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. As it is considered that the performance of a deep learning model increases with the amount of data used in its training, it would be highly possible that such a mediocre accuracy was due to the limited size of the dataset for each class. It is believed that the accuracy of this dataset can be further improved by working on different factors related to model building and hyperparameter tuning.

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 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. Keras API Docs: https://keras.io/api/
4. Udemy Deep Learning A-Z: Hands-On Artificial Neural Networks