Image Classification on CIFAR-100 using Convolutional Neural Network

Nidhi Rana
MAC
University of WIndsor
Windsor, Ontario
rana71@uwindsor.ca

Azmina Vanzara
MAC
University of WIndsor
Windsor, Ontario
vanzara@uwindsor.ca

***Abstract*—Image Classification is the process of classifying images according to their visual content. In this paper, we propose a model to perform this task with the help of Convolutional Neural Network (CNN). The goal of our project is to train a model using CIFAR-100 dataset increase the accuracy of the classification model. The proposed model optimizes the training loss and the accuracy of classification – accuracy being the main performance metric for training the model – during the training phase to increase its efficiency. The resulting model is able to classify images with an accuracy of 87%.**

Keywords—CNN, CIFAR-100, Maxpooling, Activation, Fine labels

I. Introduction

The problem of image classification is one of the popular research topics in the field of computer vision. It is a task of assigning a label from a fixed set of categories to the image based on its content. This task, despite its simplicity, has many applications in practice.

Recognizing the objects in images, while effortless for a human brain, becomes a difficult and intricate task when it comes to machines. Many approaches have been proposed to accomplish this task, including traditional learning methods and deep learning methods, all with their own advantages and drawbacks. The one method that has become the state-of-art computer vision technique to solve this problem is using CNNs. They are used by Facebook for automatic tagging algorithms, Amazon uses them to generate product recommendations and google uses them to sift through users' photos.

The reason why CNNs are the most popular choice for image classification is because they can automatically extract features from an image. They can effectively down sample the images while maintaining the information obtained spatial interaction between image pixels by using convolution and prediction on the image, thus covering both local and global feature of an image. This facility makes them more versatile than any other classification method.

In this paper, we will be describing the CNN model we designed and trained to classify the images using the standard dataset CIFAR-100 for training and testing. For this project, due to the large amount of time and limited resources, we have focused on training the model based in the fine labels in the dataset. The paper is organized as follows: Section II will discuss the related work, Section III gives an overview materials and methods used to implement the model, and Section IV presents the results and discussion of the output. Finally, the conclusion and references are presented.

II. Related Work

Recent years have seen an increased interest in neural architecture search. It is being used widely in diverse applications and computational environments due to its ability to automatically search for task dependent networks. Recent approaches have been focused around designing deep convolutional neural networks. Some of the best and most used pre-trained classification models available in keras library include ResNet50 which is a 50 layers deep CNN model, VGG-16 and VGG-19 CNN models, Inceptionv3, and EfficientNet. These are considered the state-of-the art image classification models and have been trained over large image datasets.

III. Materials and Methods

A. Neural Networks

The Neural Networks are designed to simulate the working of human brain in a machine. The human brain consists of neurons which transmit and process the information received from our senses. These neurons are arranged together to form a network that passes electric impulses to and from each other. The same principle is utilized to structure a neural network.

Neural networks consist of individual units called neurons which are arranged in groups, i.e., layers (see Fig. 1). Neurons in each layer are connected to those in the next layer and each connection has a separate weight assigned to them. The data is fed to through the input layer and the output is received at the output layer after passing through the hidden layers. Each neuron performs a simple mathematical calculation and then transmits the output to all the nodes connected to it in the next layer. This process continues until the processed data reaches the output layer. A neural network may possess any number of hidden layers, starting with at least one. As the number of layers in a network increase, so does the complexity of the network. And with the increase in complexity, the training time of the network increases as well.

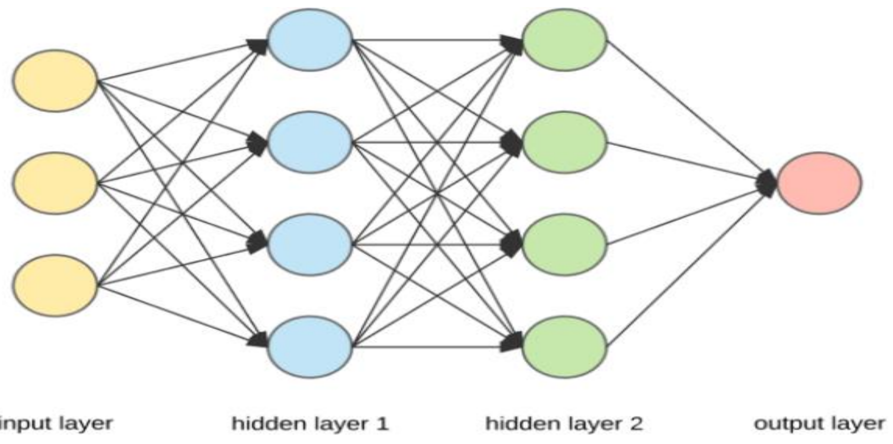


Fig. 1. Basic Structure of a Neural Network

As the computational speed of machines are increasing, so is the complexity of the neural networks. This has led to development of neural networks at such point where a wide range of tasks that were not solvable before having effective solutions available now. One such problem is Image Classification. Among different types of Neural Networks available – such as RNN, LSTM and CNN – CNN works the best to solve this task.

B. *CNN and Image Classification*

CNNs were first proposed in 1980s by Yann LeCun. The earlier versions were able to recognize hand-written digits. The CNN are designed in a way that they can mimic the working of visual cortex of the human brain, which processes the visual stimuli received by the human eye, thus making it an ideal type for the task of image processing and classification.

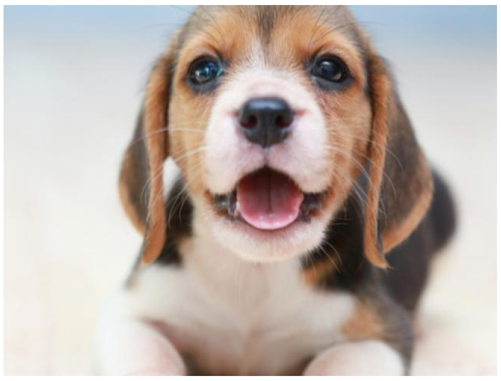


Fig. 2 (a) How human brain perceives an image

Fig. 2 (b) How computer reads an image

Getting a closer look at the role of CNN in image classification, its main task is to take an image as input and assign a label to the image based on its visual content. It is a skill that human brain gets adapted to in the earlier years. But the same information is read by the computer vary vastly. This is illustrated by Fig. 2 (a) and Fig. 2 (b).

To solve this problem, computer searches for basic characteristics to help identify and classify the image. For human brain, it is simple like long ears or nose. For a computer, these features are the boundaries of curvature. While using CNNs, the texture of the image is taken into consideration to extract the said features.

There are three types of layers the images are passed through when being processed by CNN: *Convolutional Layer*, *Activation Layer*, *Pooling Layer*, and *Fully Connected Layer*.

Convolutional Layer: This is the most computation heavy layer in the CNN where the images are convoluted using filters. Filters are applied over a sliding window and the value of each element in the convoluted feature is calculated, thus giving a reduces 2-D matrix as an output.

Activation Layer: This layer applies an activation function to increase the non-linearity of the CNN.

Pooling Layer: This layer is used to reduce the volume of the image by performing down-sampling on the output of Activation Layer.

The complete CNN is constructed using several convolutional layers mixed with activation and pooling layers, where activation layer always follows the convolutional layer, and the pooling layer does the same with activation layer.

Fully Connected Layer: This is added to the network at the very end. This Layer performs the task of flattening the data, i.e., it transforms entire pooled feature matrix obtained into a single N-dimensional vector which is fed to the model for processing, where N is the number of classes from which model selects the desired class.

The schematic diagram of a basic CNN can be observed in Fig. 3. The CNN has five different layers.

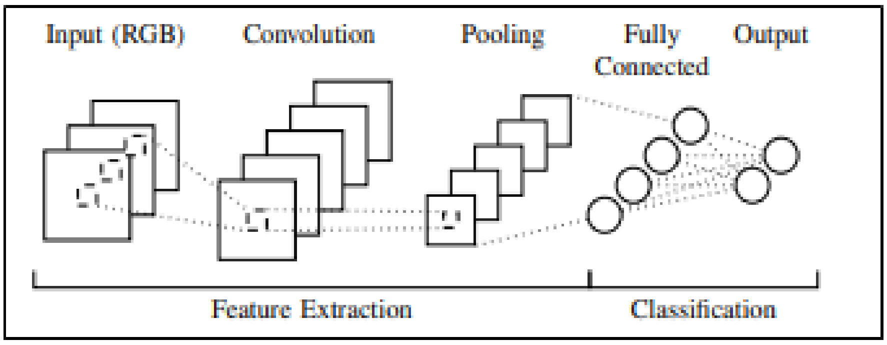


Fig. 3 Schematic diagram of a simple convolutional neural network. The network comprises five different layers. Both feature extraction and classification are learned during training.

For out project, we have utilized Python programing language to develop the CNN. We have made use of *Keras library* and *TensorFlow library* to construct our CNN. The Keras library works as an interface to the TensorFlow library provides python interface for developing ANNs.

We have made use of Conv2D and Maxpooling2D to add convolutional and pooling layers to the CNN model. We have also used *elu* activation function to increase the non-linearity of the model. We have also added a Dropout layer after flattening the data. The Dropout layer and the Maxpooling layer prevents the model from over-fitting the data.

C. *Dataset*

For the scope of our project, we have used the CIFAR-100 dataset (see Fig. 4) to train and evaluate our model.

CIFAR-100 dataset is one of the standard datasets used in the field of computer vision. It was first created by Geoffrey Hilton, Alex Krizhevsky and Vinod Nair. We have obtained the dataset from link [8] provided in the reference. It contains a collection of 32x32 colour images distributed among a pre-define set of labels.

Fig. 4 CIFAR-100 dataset with 100 different natural images each corresponding to one of the 100 labels

The dataset consists of 100 class labels and 20 superclass labels, called fine and coarse labels, respectively. Fine labels are classified into the coarse labels in such a way that each coarse label contains 5 fine labels as their sub-classes. The distribution of the labels is described in Fig. 5.

The data distribution among the labels is as follows:

Each fine label has 600 corresponding images, 500 for training and 100 for testing per class. That means a total of 60,000 images are present in the dataset of which 10,000 are testing images and 50,000 are training images.

For our project, we have restricted the scope by training the model utilizing the fine labels of the image dataset. The reason for this is the large amount of training time that goes into training any neural network.

D. *Proposed CNN structure and implementation*

In our project, we have constructed a CNN based on the following algorithm:

1. *Create the model object*
2. *Add the different CNN layers as per requirement*
3. *Compile the model*

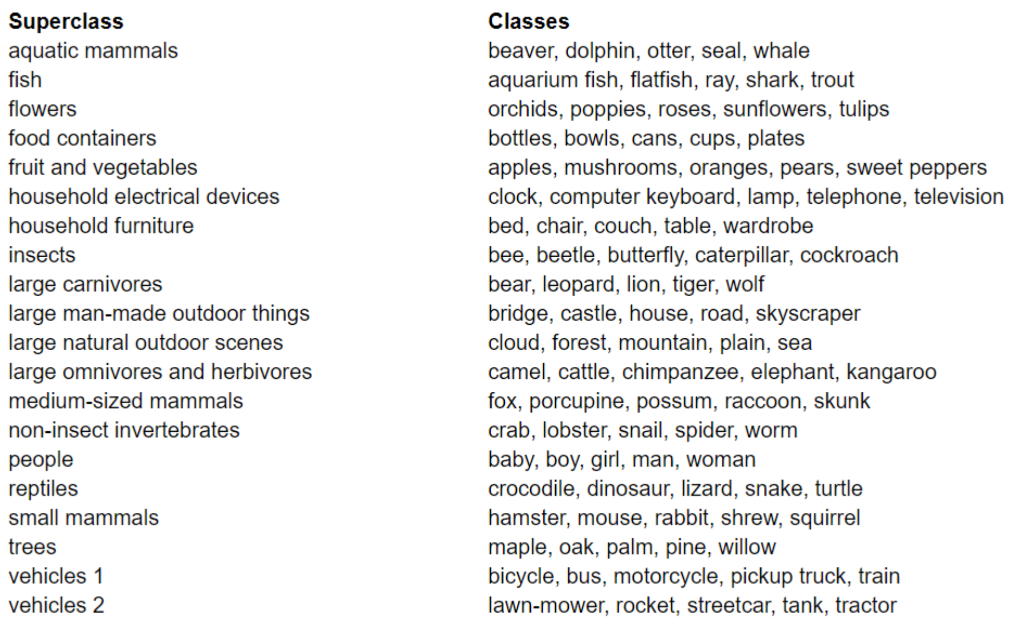
The first step is to simply create a model object for the CNN.

Then, as described in second step, we add convolutional, activation, drop-out and pooling layers as per our requirement.

Now, discussing the details of the actual layers of the CNN, following are the layers added to the model:

1) There are in total 6 convolutional layers each set with output filter 256 and the kernel size (3, 3).
2) There are 6 activation layers, one each after the convolution layer with activation function set to ‘elu’.
3) There are three pooling layers, one each after two consecutive convolutional layers. The pools size for each layer is set to (2, 2).

.



Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Fig. 5 Class labels (fine) and Superclass labels (coarse) of CIFAR-100 Dataset

These layers are set up in 3 groups, each containing 2 convolutional, 2 activation, and 1 pooling layer each. The first convolution layer has an additional component *input_shape* which is the shape of the pixel matrix.

After these three groups of layers, two fully connected layers are added to the model. The first layer is Flatten, which generates a N-dimensional vector from the output of the previous layers and the other is Dense layer – a densely connected layer with the output space of 100 (the total number of classes in the training dataset). Three Dropout layers are also added to the model to prevent overfitting of the data.

The third step is to compile the model. In this step, we also need to define the loss function and an optimizer to model. The performance metric considered during the compilation is *accuracy*.

Loss Function: This function is used to calculate the error rate of the model by considering the difference between the predicted value and the actual value. For our model, since we have used *Softmax* as the final activation function, we use Categorical Cross Entropy as the loss function.

Optimizer: Optimizers set and change the values of weights and learning rates to reduce the losses while training the model. In our case, we have used RMSprop Optimizer with learning rate set to 0.0001 and decay set to 1e-6.

E. *Training the Model*

Fig. 6 (a) and Fig. 6 (b) gives the visualization of the input data for the CNN. As we can observe, the clarity of the image is not very good. So, the very first step we need to perform for training the model is to pre-process and augment the input image.

Data Pre-processing and Augmentation

In order to extract the most information from our input image, we have augmented the data via a number of random transformations, so that our model can make a better feature map from the data. This prevents overfitting of the data and helps the model generalize the input better. We have accomplished this by using the ImageDataGenerator function available in keras library. This function allows us to configure the random transformations and normalization operations that are to be done on the input data. It also generates batches of augmented images which are then fed as input to the model for training.

Training model with training dataset

After the image data has been pre-processed and augmented, the model is finally ready to be trained. CNN training has two important hyper-parameters: *Batch* and *Epoch* [11].

Batch: A hyper parameter that defines the number of samples to work through before updating the internal parameters of the model

Epoch: A hyperparameter that defines how many times a training set is passed through the network

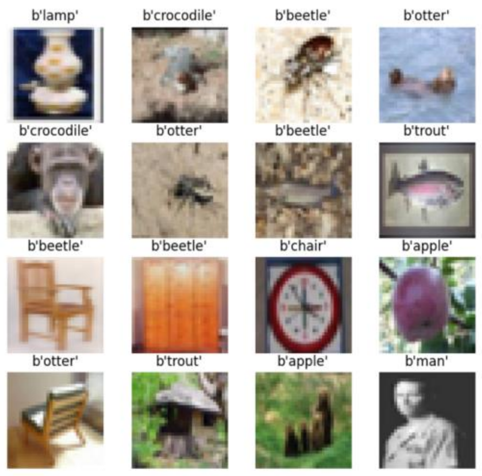


Fig. 6 (a) Sample data images from the CIFAR-100 Dataset

Fig. 6 (b) Visualization of data using a single image from CIFER-100 Dataset

There are different measures through which we can optimize the value of these hyperparameters. For instance, 32, 64 and 128 are the most popular batch sizes for Neural Network models. On the other hand, estimating an optimum number of epochs is not possible until we start training the model. As soon as there is a saturation in the model accuracy or the loss function starts increasing for the model, we need to stop training the model to prevent overfitting. The number of epochs at which such a condition occurs is the optimum number epochs for training the model.

In our case, we have set the batch size as 64, i.e., 64 images are processed by the CNN before its internal parameters are updated. The number of epochs is preset at 200, although we may generate an interrupt if we do not want to train the model further.

A batch can compromise either one or more batches of images based on what gradient algorithm is being followed. We have used the Mini-Batch Gradient Descent where the batch size is greater than 1 but smaller than the size of training set. In this CNN, a single epoch has a total of 781 batches.

The idea behind training the model over epochs is that if we feed data to the model in different ways, it will be able to classify the images during the testing phase more efficiently because, as it is trained over varying input images, it will be able to generalize the images better. Training the model using epochs makes it possible for it to adjust the parameters according t the new information fed to it and reduces the chance of bias while performing classification.

Evaluation during training

Alongside training the model, we are also keeping track of the training accuracy and the loss value of the function. Both training accuracy and testing accuracy are being measured at the end of each epoch.

After all the batches are passed through the network for each epoch, we finally have a fully trained CNN.

Hardware Requirements

The model has been trained on the system with Intel i7 Quad-core processor with 8 GB RAM. It took approximately 68 hours to train the model using the PC with this hardware specifications.

IV. Results and Discussion

A. Evaluation Metrics

We have used two of the classic performance metrics to measure the performance of the model: Classification Accuracy and Loss Value. As mentioned in Section III-E, the accuracy and loss functions for both training and validation are calculated after each epoch.

Accuracy of the model can be found by taking the ratio of the number of correct predictions to that of the total predictions performed by the model. The equation is as follows:

Accuracy = No. of correct predictions / Total no. of predictions

The loss L is defined as the difference between the predicted value by the model and the true value. For our model, the loss function used is the categorical cross entropy, also known as softmax loss. It can be calculated by the following equation:

$$CE = \frac{1}{M}\sum_{p}^{M} -log\left(\frac{e^{s_p}}{\sum_{j}^{C} e^{s_j}}\right)$$

where CE is the categorical cross entropy, M represents the positive class samples, s_p in M is the CNN score for each positive class and C represents all the classes in the dataset.

For evaluation, researchers usually prepare a testing dataset before-hand and use it to measure the performance of the system. In our case, as mentioned in Section II-C, CIFAR-100 already has separate training and testing datasets. We have calculated these metrics for both the training set and the validation set at the end of each epoch. Usually, the training accuracy is higher than the validation accuracy.

B. Results

Table 1 shows the accuracy and loss function value observed at the end an epoch.

TABLE I. Accuracy and Loss Values over Epochs

Epoch	**Accuracy (in %)**		**Loss**	
	Training	***Validation***	***Training***	***Validation***
1	18.15	19.17	3.47	3.47
26	54.78	56.36	1.70	1.63
51	67.93	62.54	1.14	1.45
76	75.03	63.69	0.85	1.45
101	81.12	63.91	0.65	1.49
126	84.10	66.03	0.54	1.63
151	86.07	66.33	0.48	1.67
172	87.18	67.66	0.44	1.60

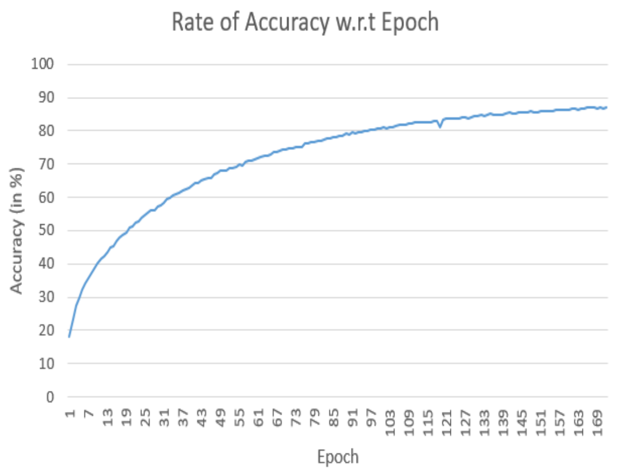


Fig. 7 (a) The rate of increase of Accuracy for the training dataset w.r.t no. of Epochs

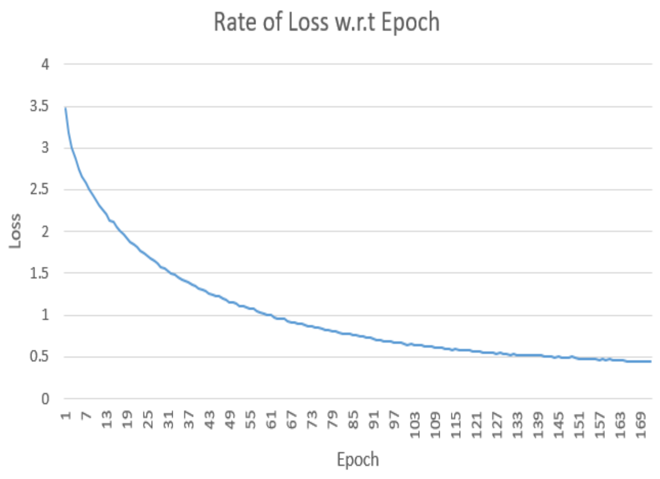


Fig. 7 (b) The rate of decrease of Loss value for training dataset w.r.t no. of Epochs

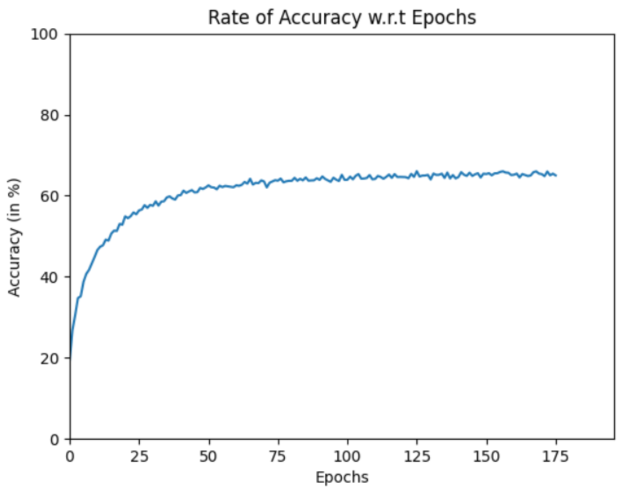


Fig. 8 (a) The rate of increase of Accuracy for testing dataset w.r.t no. of Epochs

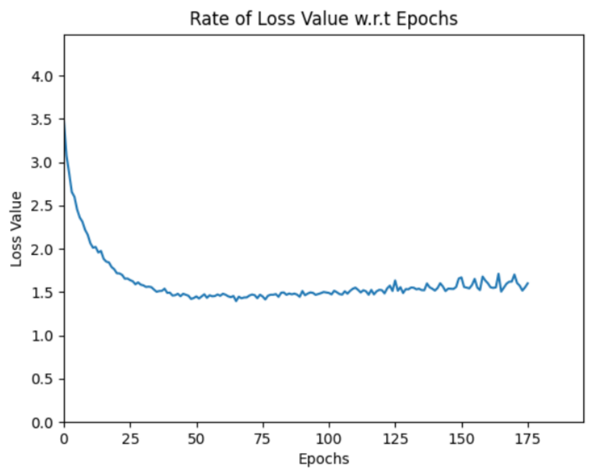


Fig. 8 (b) The rate of decrease of Loss value for testing dataset w.r.t no. of Epochs

C. *Discussion*

The performance of the model during the training phase for both training dataset and testing dataset can be observed in Fig. 7 (a) and Fig. 7 (b), and Fig. 8 (a) and Fig. 8 (b) respectively.

As we can observe from Fig. 7 (a), the accuracy of the model evaluated over training set increases rapidly in the beginning of the training process and after passing through certain number of epochs, it starts getting steady. Finally, it reaches a stage where the model reaches a level of saturation and the accuracy of the model stops increasing. At this point, we should stop the epochs or else the chances of overfitting the data increases.

Similarly, it can be noted from Fig. 7 (b) that the Loss value at the beginning of training stage is very high and it starts decreasing rapidly as epochs run by. After some time, like with accuracy, the rate of decrease in the Loss value reduces until it becomes steady. The number of epoch where the loss value gets saturated is the most optimal number for training the model for that dataset.

In this case, as we can see, the model reaches at the point of saturation at around 165th epoch. So, that is the optimal number of epochs to train the model for CIFAR-100 dataset.

Now, observe Fig. 8 (a) and Fig. 8 (b). These two figures show the performance of model over validation data after each image. It is important to notice that the behavior for both Accuracy and the Loss function for validation data is similar to that of the training data. While the Accuracy increases before saturating after some time, the loss value decreases before it reaches a threshold.

However, it is obvious if we compare the plots that the Accuracy for training and validation set, while at first is almost equal, with each passing epoch, the difference between the performance accuracy for both sets increases. The same is also true for loss value as well. Also, the value of loss function for testing set shows erratic changes between epochs 150 to 175. This is an indication that the model might start overfitting the data if trained any further.

Table 1 consists the observations made at regular interval of epochs during the training phase. We can see that the deductions we made from the plots of Accuracy and Loss value hold true as per the values in the table.

From the performance accuracies mentioned in Section IV-B, we can say that while our designed model is progressing very well in terms of training, it still cannot handle new data efficiently. It means that the quality of our designed model is not up to expected standards. The gap between the training accuracy and the validation accuracy is very huge. There can be various reasons for this, and some of the possibilities are described further.

The most obvious reason for this can be that the model is overfitting the data. We can deduce this from the observations made in Table 1. As we can see, the loss value is increasing over the course of epochs 76 through 151 and onwards.

To remedy this, we can perform cross-validation over the training data, or we can stop the training early to avoid overfitting. We can also use more training data, but since the objective of or project is to classify the CIFAR-100 dataset, this is not a feasible option.

We can also try fine tuning the hyperparameters defined for the CNN, like the learning rate and the batch size, and also the parameters of Drop-out layer since it helps prevent overfitting the data.

There are a few other issues with the model that we observed. If we train the model with mislabeled data, it will lead the model to converge to the wrong best, and as the model is designed to minimize the loss function faster, the model will be overfit to noisy data faster than other models. That is not a problem within the current scope of project as we are working with a standard computer vision dataset.

V. Conclusion

To conclude, in this paper, we have presented you with a very simple CNN network that can be used for image classification trained over CIFAR-100 dataset. We have intentionally limited ourselves to a fewer number of layers and designed a very basic model that allows us to concentrate on critical aspects of the model rather than focusing on unnecessary details. Due to lack of good hardware, we have tried to create an architecture that is easy to train on a simple hardware system and limited the project scope and had to contend ourselves with very few configurations. We are going to continue testing and optimizing the model until it gives better accuracy.

For future works, as was mentioned in Section IV-C, the model we proposed needs fine tuning and optimizing to reduce the effect of overfitting. Furthermore, as of now, we have run our experiments on CIFAR-100 with two levels of semantic hierarchy and a fixed number of fine-grained classes per coarse-grained class. However, this structure only captures a tiny fraction of real-world semantic hierarchies. Semantic hierarchy trees even in a domain like "animals" are usually asymmetric trees, with vastly different number of subtrees per tree node and different depths for the leaf nodes. Hence, we would like to explore how well our model can generalize to more degenerate semantic hierarchies. We hypothesize that the CIFAR-100 dataset with only two hierarchy levels might not be able to showcase the full potential of the CNN model, and thus hope to experiment with the model on more complex semantic hierarchies, e.g. using the ImageNet dataset [9] or the CompCars dataset [10].

The CompCars dataset contains 163 car makes with 1,716 car models, perfect for establishing a nice semantic hierarchy. For example, we could establish a hierarchy as follows: 1) car type, 2) car make, 3) car model, and 4) model year. We are especially interested in observing how the attention mechanism ties in with this dataset; we would be interested to observe which features of the input image the model attends to as it iterates through the above-described hierarchy.

Acknowledgment

It is with great pleasure that we present this report on our topic "Image Classification using CNN". We would like to thank our instructor Dr. Luis Rueda for his sharing his knowledge and offering us guidance throughout the course. His help is the reason for the successful completion of our project.

References

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[2] Ertam, Fatih & Aydin, Galip. (2017). Data classification with deep learning using Tensorflow. 755-758. 10.1109/UBMK.2017.8093521.

[3] I. Kandel and M. Castelli, "A Novel Architecture to Classify Histopathology Images Using Convolutional Neural Networks," *Applied Sciences*, vol. 10, no. 8, p. 2929, Apr. 2020.

[4] Asim Suhail, Manoj Jayabalan and Vinesh Thiruchelvam, "CONVOLUTIONAL NEURAL NETWORK BASED OBJECT DETECTION: A REVIEW ," *Journal of Critical Reviews* 7 (2020), 786-792.

[5] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[6] Khan, A., Sohail, A., Zahoora, U. *et al.* A survey of the recent architectures of deep convolutional neural networks. *Artif Intell Rev* 53, 5455–5516 (2020). DOI: https://doi.org/10.1007/s10462-020-09825-6

[7] Yapıcı, Mutlu & Tekerek, Adem & Topaloglu, Nurettin. (2019). Literature Review of Deep Learning Research Areas. 5. 188-215. 10.30855/gmbd.2019.03.01.

[8] Krizhevsky, A., 2020. CIFAR-10 and CIFAR-100 datasets. [online] University of Toronto. Available at: https://www.cs.toronto.edu/~kriz/cifar.html [Accessed 22 December 2020]

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009

[10] X. Liu, W. Liu, H. Ma, and H. Fu. Large-scale vehicle reidentification in urban surveillance videos. In Multimedia and Expo (ICME), 2016 IEEE International Conference on, pages 1–6. IEEE, 2016.

[11] Brownlee, J., Difference between a Batch and an Epoch in a Neural Network. Machine Learning Mastery [online] Available at: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ [Accessed 22 December, 2020]

[12] Nobel, B., The Ethics of AI Image Recognition. [online] Cloudera blog. Available at: https://blog.cloudera.com/the-ethics-of-ai-image-recognition/ [Accessed 22 December 2020]

[13] I. Kanellopoulos & G. G. Wilkinson (1997) Strategies and best practice for neural network image classification, International Journal of Remote Sensing, 18:4, 711-725, DOI: 10.1080/014311697218719

APPENDIX

APPENDIX I

ACRONYMS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long-Short Term Memory
ANN	Artificial Neural Networks
ELU	Exponential Linear Units
CE	Cross Entropy
ResNet	Residual Neural Network

APPENDIX II

AUTHORS AND WORK DISTRIBUTION

The paper is presented by: Azmina Vanzara and Nidhi Rana

The task distribution among the team members was as follows:

Task	Azmina	Nidhi
Dataset Analysis	🟩	
Literature Review	🟩	🟩
Data Visualization	🟩	
Model development	🟩	🟩
Result Visualization		🟩
Result Analysis		🟩
Project Report	🟩	🟩

Since only one of the team member's was capable of handling the training task, we divided the tasks accordingly. The construction of the model was done after discussing the strategy as a team and the coding it and running it on a single machine.

The reference papers [1]-[7] were divided between the two members and perused to come up with an idea of how to proceed with the project. Research paper distribution is as follows:
Azmina Vanzara: [1], [3], [4], [6]
Nidhi Rana: [1], [2], [5], [7]

APPENDIX III

ETHICAL, LEGAL AND SOCIETAL ISSUES WITH MACHINE LEARNING

1. Ethical aspects:

The use of Deep Learning for image recognition offers great potential for business transformation and problem-solving, but numerous responsibilities are interwoven with that potential. Predominant among them is the need to understand how the underlying technologies work, and the safety and ethical considerations required to guide their use [12]. Many organizations use recognition capabilities in helpful and transformative ways. But just as most technologies can be used for good, there are always those who seek to use them intentionally for ignoble or even criminal reasons. In-between intentional beneficial use and intentional harmful use, there are grey areas and unintended consequences. Governments and corporate governance bodies likely will create guidelines and laws that apply to these types of tools. Physical safety is a prime concern. Customers demand accountability from companies that use these technologies. Transparency helps create trust and that trust will be necessary for any business to succeed in the field of image recognition.

As per as our project we have used the CIFAR 100 dataset that gives a very good approach to classify images as terms of coarse labels and fine labels which can be carried along with other great Image classification problems according to their semantic hierarchy. While previous works usually only classify objects into the leaf categories, we argue that generating hierarchical labels can describe how the leaf categories evolved from higher level coarse-grained categories, thus can provide a better understanding of the objects. In this paper, we propose to utilize the CNN framework to address the hierarchical image classification task. It pans across the image performing a series of these evaluations (convolution) where each calculation is sensitive to a different kind of arrangement of pixels. These values are passed up (or not) to another layer such that each successive layer responds to more complicated patterns or shapes. By analogy with the brain, each layer is imagined as a series of connected neurons. Each neuron responds to a different aspect of an image and all of these layers activate in different weightings of connections. The combination of those weightings allows for the identification of what's going on in the image. In automatic image captioning, the final layer of neurons is used to assign the labels. This framework can not only generate hierarchical labels for images, but also improve the traditional leaf-level classification performance due to incorporating the hierarchical information.

2. Legal Aspects:

The owner of the data - CIFAR 100 is Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton which gathered data tiny images dataset on which we based all our experiments that was collected by colleagues at MIT and NYU over the span of six months. This data is available freely at [8] as people can do more analysis in this domain and contribute to the society of Image classification. They assembled it by searching the web for images of every non-abstract English noun in the lexical database WordNet. The search term used to find an image provides it with a rough label, although it is extremely unreliable due to the nature of online image search technology. In total, the dataset contains 80 million colour images downscaled to 32 × 32 and spread out across 79000 search terms. We have been using convolutional neural/neuron networks, a technology that mimics the way the eye and the brain makes sense of an image. CNN (convolutional neural networks) is currently the state-of-the-art for image recognition. Most of our experiments with various learnings were performed on a subset of about 2 million images. Using these labels, we show that object recognition is Significantly improved by pre-training a layer of features on a large set of unlabelled tiny images

3. Societal Aspects:

Choice of reference materials embodies their own priorities, privileges, and perspectives on what constitutes a suitable candidate, with all of the racial and other systemic biases that exist with this societal 'grey area'. In that case, we need to ask, what, and how much, harm can our research accomplish? This product is designed for all people and does not have a bias. This data is majorly used to do various research on how Image classification works. The data and images that are obtained are public and does not harm any individual as it is a raw data and freely available to anyone who can bring improvement in the area of Image Classification/recognition. It is, in principle, an excellent dataset for training of deep generative models, but previous researchers who have tried this have found it difficult to learn a good set of filters from the images. Therefore, with a better initiative authors of this data have come better dataset that can be used that is easy to understand and apply various combination on the pixels that have been formed for each image.

Appendix IV

Source Codes and Files

Training and Evaluation of the model	cifar100.py
Data Visualization	DataVisualization.py
Evaluation Plot	Plot.py
Accuracy and Loss Evaluation for Testing data (pickle file)	loss_validation.p