Image Classification using CNN and CIFAR-10

Pratham Kiran Mehta[1], Rahul Jain[2], Manan Batra[3]

[1,2,3]*B. Tech student, Computer Science and Engineering, Vellore Institute of Technology, Tamil Nadu, India*

---***---

Abstract - *Convolutional Neural Networks (CNN) have been at the forefront of advances in the application of Computer Vision. Their automated and adaptive nature helps in the extraction of the hierarchical structures present in input images, allowing them to capture and interpret complex spatial patterns. This ability has made CNNs a powerful and useful tool in the field of image analysis. The work proposed in this paper reviews the use of CNNs for categorizing images into certain categories and outlines the methods that need to be employed for pre-processing this novel dataset to feed into the CNN model. In this regard, by considering the ten standard object classes of CIFAR-10 dataset, several CNN models were trained to classify these images and compared them against the others to show the effectiveness of each. When training and testing the model, its performance is quantitatively evaluated using measures such as accuracy, precision, validation loss and the loss function. These metrics define the degree of success of the selected architecture. This work contributes to the existing knowledge of CNNs applied to image classification tasks with new datasets and can serve as helpful suggestions in changing the basic structure for object recognition aims.*

***Key Words*: Convolutional Neural Network, CIPHAR-10, Image processing, Image classification, ResNet**

1.INTRODUCTION

Convolutional Neural Networks (CNN) are a subtype of deep learning algorithms that are mostly used to solve identification and detection problems, such as image recognition, detection, and division. These CNNs resemble other neural networks but the use of multiple convolutional layers makes them slightly complex. These convolutional layers use a function called a convolution, which is a form of matrix multiplication. This entails the use of small parts of the input data to extract the appearance characteristics, while at the same time preserving the spatial configurations of pixels.

CIFAR-10 is another popular set of images that is maintained by the Canadian Institute for Advanced Research and is used to train most vision and machine learning models. This is one of the most popular datasets used in studies related to machine learning. CIFAR-10 comprises sixty thousand color images of 32×32 pixels and are classified into ten classes. These classes include airplanes, trucks, cars, frogs, dogs, cats, deer, and birds, and each class has 6000 images.

1.1 Different Models of CNN Used for Image Classification

LeNet which was first introduced in 1998 by Yann LeCun and his co-workers Corinna Cortes and Christopher Burges was targeted for the handwritten digit recognition. LeNet is often described as the 'Hello World 'of deep learning and is one of the first successful convolutional neural network (CNN) architectures. Its network architecture comprises of numerous convolutional layers, pooling layers, and fully connected layers. Particularly exceptional, the presented model has five convolutional layers followed by two fully connected layers. LeNet was the first to introduce CNNs in the field of deep learning for computer vision processes. However, the model above failed at first to learn due to what is known as the vanishing gradients problem. To rectify this, max-pooling layers were included to be added within the convolutional layers to minimize the size of the images. This is not only useful to avoid overfitting but also improves the training speed of CNNs. AlexNet was created by I. Sutskever, G. Hinton and A. Krizhevsky. There are similarities with the LeNet architecture at this level, with a larger number of layers and stacking of convolutional layers. In AlexNet architecture, it has five convolutional layers that are blended with max-pooling and other layers, three fully connected layers, and two dropout layers. In each layer there is an activation function of ReLU kind, and the output layer has an activation function of Softmax kind. In total, the architecture has about 60 million parameters.

ZFNet is a CNN architecture composed of convolutional neural networks and fully connected layers as well. It was created by Rob Fergus and Matthew Zeiler. Like AlexNet, ZFNet also follows the network architecture with several layers of convolutional layers and sets of pooling layers but the size of the middle convolutional layers has been tuned, as has the stride and the filter size of the first layer. The architecture is based on the model developed by Zeiler and Fergus which was used to train the models with the ImageNet data set. ZFNet consists of seven layers: a convolutional layer, a downsampling max-pooling layer, a concatenation layer, another convolutional layer that uses a linear activation function and which has a stride of one. To increase the regularization, dropout is implemented before the output layer which is a fully connected layer. After observing the article, ZFNet is more efficient in terms of computational requirements as compared to AlexNet because deconvolutional layers are placed in between CNNs that provide an approximation inference. GoogLeNet is designed by Jeff Dean, Christian Szegedy, Alexandro Szegegy and their colleagues. This model is superior in the error rate compared with previous ILSVRC competition champions including AlexNet (the 2012 winner) and ZF-Net (the 2013 winner). It also achieves a lower error rate than VGG, the second best algorithm in 2014. There are several unique features used in the architecture to enhance its depth, which include global average pooling and 1x1 convolutional layers. They include short cuts to link the first convolutional layers but add more filters in other CNN layers. Further, it applies several unpooling layers on the top of CNNs during the training stage for removing the spatial redundancy and thus reduces the new parameters to be learned. As for VGG16, it stands for 'Visual Geometry Group 16 'and was created by Andrew Zisserman and Karen Simonyan. The VGGNet has up to 16 layers that are implemented as CNN layers, and this model has up to 95 million parameters that were optimized with the help of training the model on a dataset containing over one billion images belonging to 1000 classes. VGGNet is capable of taking large input images of up to 224 by 224 pixels, with 4096 convolutional features. Nonetheless if the input images are usually sized between 100 x 100 to 350 x 350 pixels which is typical in most image classification problems, architectures such as GoogLeNet perform better than VGGNet. This is so because CNNs with such large filters as those in VGGNet are costly to train and require large amounts of data.

ResNet CNN architecture was designed by Kaiming and his team, and they achieved acclaim by getting the five error rate to 15 percent on the ILSVRC 2015 classification challenge. This network can be regarded as very deep even according to the standards of CNN with 152 layers and over a million parameters. This model was trained on the ILSVRC 2015 dataset for more than forty days on 32 GPUs. As demonstrated by ResNet, there is strong evidence that CNNs can be used in NLP tasks such as machine comprehension and sentence completion.

MobileNets are the types of CNN that have been developed specifically for performance on mobile devices and can be used to serve photo classifications and object identification in real-time. Provided by Andrew G. Trillion, these compact CNN architectures can be used in real-time applications such as smartphones and drones. This flexibility has been shown through a range of CNNs with 100-300 layers in many cases surpassing other network configurations such as VGGNet.

2. LITERATURE REVIEW

This paper by Neha Sharma, Vibhor Jain and Anju Mishra presents an experimental analysis to determine the performance of popular CNN architectures such as AlexNet, GoogLeNet and ResNet50 in real-time object detection and classification scenarios. Through the input of datasets like ImageNet, CIFAR-10, and CIFAR-100, the study proves that GoogLeNet and ResNet50 are more precise compared to AlexNet. It examines the effects of different networks structures and training strategies on CNN's performance in various image classification tasks and discusses the importance of feature extraction and Deep learning for the development of Computer Vision applications.[1]

Krizehvsky et al 2012 succeeded in improving performance through Deep Convolutional Neural Network (CNN) by use of five layers of convolution and three layers of full connection where training was enhanced using Rectified Linear Unit (ReLU). This model utilized local response normalization (LRN) adopted from actual neurons that employ lateral inhibition and was trained through stochastic gradient descent with momentums that enable the search for the optimum in the high dimensional loss of deep learning networks. The well-organized structure of AlexNet placed it at the top of all the previously developed models in the arenas of the ILSVRC.[2]

In the research study by Sultana, A. Sufian and P. Dutta, they investigate the applicability of deep learning methods such as CNNs in the CIFAR-10 image classification task. CNNs are emphasized on their usage in handling large amount of data and the capability in extracting complex features in multi-layered approach. The research encompasses stages like data preparation, model building, and model assessment, focusing on how architectural decisions and proper tuning of hyperparameters contribute to achieving high accuracy and robustness in image classification problems.[3]

Specifically, Ajala's studies focus on Convolutional Neural Networks (CNNs) for image classification using the CIFAR-10 dataset. The paper includes an analysis of the existing methods based on traditional neural networks and points out their shortcomings, such as the lack of efficiency in feature extraction and limited possibilities for generalizing the model. However, CNNs, including learnable filters and pooling layers, can learn features on their own, which further improves the performance of the study. To prevent overfitting and increase the model accuracy some techniques include dropout and regularization are used. It also analyzes different structures of CNN and enhances them for higher performance with the CIFAR-10 data set, which is another focus of the research. [4]

The authors of the paper 'Image classification using SVM and CNN 'seek to demonstrate the superiority of CNN model in image classification over SVM models. CNN models have convolutional, pooling and fully connected layers, which makes it possible for this network to grasp special hierarchy facilitating feature representation at different levels of abstraction. 75% accuracy of CNN model reveals it can process intricate image data and big data.[5]

The study focuses on Convolutional Neural Networks (CNNs) for the classification of images, targeting cow-teat images as a sample dataset. The authors briefly discuss the advantages of using CNNs over conventional approaches and note that CNNs have the ability to learn and extract hierarchical features from images. An efficient guide to designing and training a CNN from scratch in a paper as it accurately demonstrates its effectiveness in classifying images into certain classes. The study focuses on the options for architectural decisions as well as optimization measures to enhance the model's performance and accuracy levels.[6]

The authors discuss the use of the CNN for CBIR, and their work specifically uses the CIFAR-10 dataset. This work relates to the shortcomings of traditional text-based image search techniques and offers an algorithm of enhanced accuracy in image classification. Hence, through training and testing on specific classes (airplane, bird, and car), the study showcases the potential of CNNs in achieving 94% classification accuracy based on the features. The study stresses the need to enhance the performance of algorithms for improving their recall rates.[7]

CIFAR-10 dataset is discussed in this paper as the authors consider Convolutional Neural Networks (CNNs) for image classification. The paper focuses on the fundamental roles of CNNs in computer vision tasks, for example, object recognition acknowledging the ability to learn multiple layers of representation from the data. The paper explores the process of building the CNN model architecture together with the data preparation, training, and testing stages. It also discusses methods like data augmentation and transfer learning that can be used to enhance model performance. Based on the findings of the study, it is argued that through advancement techniques such as regularization and hyperparameter optimization, CNNs perform enhanced accuracy and generalizable performance on image classification tasks. [8]

Emmanuel Adetiba, Oluwaseun T. Ajayi, Jules R. Kala, Joke A. Badejo, Sunday Ajala, Abdullateef Abayomi, the authors of this paper, examined the possibility of deep learning models for the identification of plant species based on the image dataset called Leafsnap. They underscore how accurate identification of plant species is crucial in disciplines like agriculture, climate change and medicine. The classical approaches of plant identification are usually lengthy and may involve expertise that is not easily accessible. To address these challenges, the authors incorporate deep learning methods, where convolutional neural networks (CNNs) have been proven to succeed in many computer vision applications. The study examines five pre-trained CNN models namely VGG-19, AlexNet, MobileNetV2, GoogleNet and ResNet50. The best models on testing were MobileNetV2 where we used the ADAM optimizer with the highest testing accuracy of about 92%. When discussing feature extraction in plant recognition, the authors stress its importance and mention how CNNs eliminate the need for this type of engineering to a certain extent. The paper also provides an experimental comparison of the chosen models on architecture, training, and validation stages. The research then concludes that due to the high performance exhibited by MobileNetV2 the network can be used when designing an automatic species recognition mobile application that would benefit botanists, farmers, and conservationists when identifying plant species in the field. This study adds to the knowledge on the application of deep learning in biological sciences, providing practical solutions for real-world challenges in plant species identification.[9]

3. Method

3.1 Data preparation

The idea here is to pass as much as different transformed images to our model so that the model learns complex patterns and restricts itself from overfitting.

The following steps have been implemented for the same:

- **Image resizing**: As the image size from source is 32X32 we apply resize function just to make sure every image passed to our process is similar in terms of size.
- **Image transformation**: We apply different transformation to the image. Details mentioned below. Here one point to note is that we can randomly apply every type of transformation as this might lead to underfitting. The rationale is that the images already have low resolution, and applying significant transformations such as center cropping, vertical flipping, and others could potentially degrade the model's performance. Apply normalization to image: The mean and standard deviation is taken from different studies researchers have implemented and have them converge faster. One can experiment with different image size and standard deviations as well.
- **Apply Random Erasing**: It helps in randomly adding some noise to the image, so that the model can learn to generalize better.

3.2 Convolution Neural Network Implementation

Steps followed for the modeling process:

1. Define a rough model architecture to follow
2. Starting with 2 Convolution layer and 1 fully connected layer (fc)
3. Keep adding hidden layers to the above model and check of validation accuracy improves.
4. Note if we increase model layers and neurons this might lead to slower training and increase chances of overfitting. Also increased complexity might lead to getting stuck at saddle point or local minima. Hence to cater all these potential problems, experimented with momentum, dropout & the weight initialization

This is how we get the 8x8 image size after cnn1, maxpool1, cnn2 and maxpool2:

1. 'out_2' is a given parameter when the cnn2 was defined, it is the output of cnn2 equal to 32.
2. 8x8 is the result of the 'image size' after. cnn1, maxpool1, cnn2 and maxpool2:

- Input size is 32x32 2.b. after cnn1, size per channel (total of 16 channels) is (32 + 2*2 - 5)/1 + 1 = 32x32
- After maxpool 1, size per channel (still total of 32 channels) is (32 + 2*0 - 2)/2 + 1 (stride size is equal to kernel size of 2) = 16x16
- After cnn2, size per channel is (16+ 2*2 -5)/1 + 1 =16x16
- After maxpool2, size per channel is (16 + 2*0 -2)/2 + 1= 8x8

3.3 CNN MODELS :

CNN Model V1:

1. 2 convolution + max pool layers
2. 1 fully connected layers Default runtime using 0 momentum and 0 dropout value

CNN Model V2:

1. 2 convolution & max pool layers
2. 2 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5

CNN Model V3:

1. 2 convolution & max pool layers
2. 3 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5

CNN Model V3-V2:

1. 3 convolution & max pool layers
2. 3 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5
4. Weights initialized using He weight initialization

CNN Model V3-V3:

1. 3 convolution & max pool layers
2. 4 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5

CNN Model V3-V4:

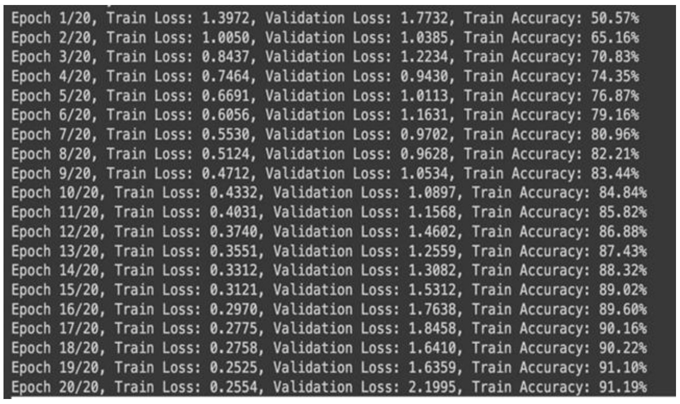
Adding one more hidden layer, dropout value and one more convolution layer.

1. 3 convolution layer + Batch Normalization
2. 3 Hidden layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5

4. Results

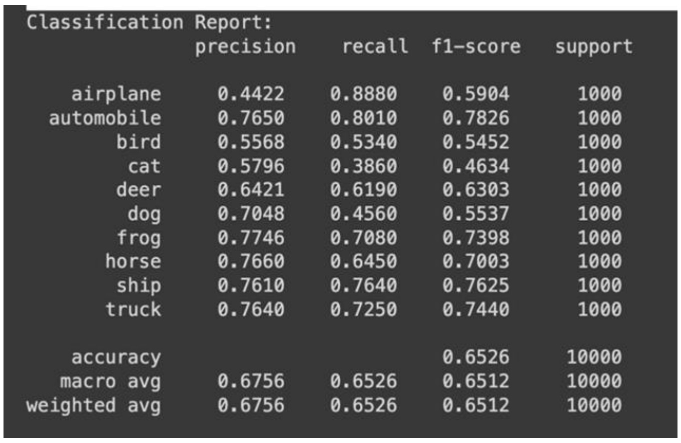
CNN Model V1:

1. 2 convolutions + max pool layers
2. 1 fully connected layer
3. Default runtime using 0 momentum and 0 dropout value



```
Epoch 1/20, Train Loss: 1.3972, Validation Loss: 1.7732, Train Accuracy: 50.57%
Epoch 2/20, Train Loss: 1.0050, Validation Loss: 1.0385, Train Accuracy: 65.16%
Epoch 3/20, Train Loss: 0.8437, Validation Loss: 1.2234, Train Accuracy: 70.83%
Epoch 4/20, Train Loss: 0.7464, Validation Loss: 0.9430, Train Accuracy: 74.35%
Epoch 5/20, Train Loss: 0.6691, Validation Loss: 1.0113, Train Accuracy: 76.87%
Epoch 6/20, Train Loss: 0.6056, Validation Loss: 1.1631, Train Accuracy: 79.16%
Epoch 7/20, Train Loss: 0.5530, Validation Loss: 0.9702, Train Accuracy: 80.96%
Epoch 8/20, Train Loss: 0.5124, Validation Loss: 0.9628, Train Accuracy: 82.21%
Epoch 9/20, Train Loss: 0.4712, Validation Loss: 1.0534, Train Accuracy: 83.44%
Epoch 10/20, Train Loss: 0.4332, Validation Loss: 1.0897, Train Accuracy: 84.84%
Epoch 11/20, Train Loss: 0.4031, Validation Loss: 1.1568, Train Accuracy: 85.82%
Epoch 12/20, Train Loss: 0.3740, Validation Loss: 1.4602, Train Accuracy: 86.88%
Epoch 13/20, Train Loss: 0.3551, Validation Loss: 1.2559, Train Accuracy: 87.43%
Epoch 14/20, Train Loss: 0.3312, Validation Loss: 1.3082, Train Accuracy: 88.32%
Epoch 15/20, Train Loss: 0.3121, Validation Loss: 1.5312, Train Accuracy: 89.02%
Epoch 16/20, Train Loss: 0.2970, Validation Loss: 1.7638, Train Accuracy: 89.60%
Epoch 17/20, Train Loss: 0.2775, Validation Loss: 1.8458, Train Accuracy: 90.16%
Epoch 18/20, Train Loss: 0.2758, Validation Loss: 1.6410, Train Accuracy: 90.22%
Epoch 19/20, Train Loss: 0.2525, Validation Loss: 1.6359, Train Accuracy: 91.10%
Epoch 20/20, Train Loss: 0.2554, Validation Loss: 2.1995, Train Accuracy: 91.19%
```

Fig. 1. Output of each Epoch using CNN model v1

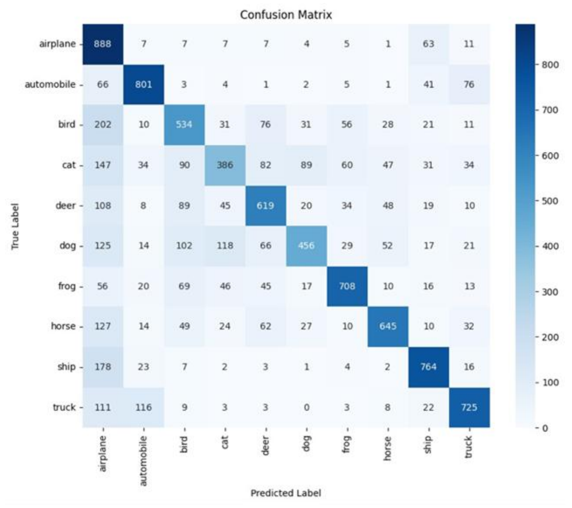


```
Classification Report:
              precision    recall  f1-score   support

    airplane     0.4422    0.8880    0.5904      1000
  automobile     0.7650    0.8010    0.7826      1000
        bird     0.5568    0.5340    0.5452      1000
         cat     0.5796    0.3860    0.4634      1000
        deer     0.6421    0.6190    0.6303      1000
         dog     0.7048    0.4560    0.5537      1000
        frog     0.7746    0.7080    0.7398      1000
       horse     0.7660    0.6450    0.7003      1000
        ship     0.7610    0.7640    0.7625      1000
       truck     0.7640    0.7250    0.7440      1000

    accuracy                         0.6526     10000
   macro avg     0.6756    0.6526    0.6512     10000
weighted avg     0.6756    0.6526    0.6512     10000
```

Fig. 2. Classification report using CNN model v1



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	888	7	7	7	7	4	5	1	63	11
automobile	66	801	3	4	1	2	5	1	41	76
bird	202	10	534	31	76	31	56	28	21	11
cat	147	34	90	386	82	89	60	47	31	34
deer	108	8	89	45	619	20	34	48	19	10
dog	125	14	102	118	66	456	29	52	17	21
frog	56	20	69	46	45	17	708	10	16	13
horse	127	14	49	24	62	27	10	645	10	32
ship	178	23	7	2	3	1	4	2	764	16
truck	111	116	9	3	3	0	3	8	22	725

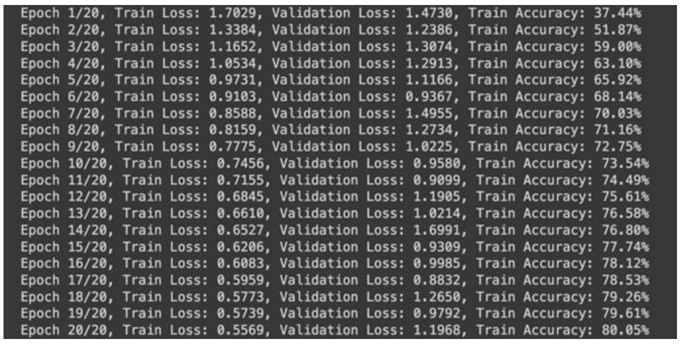
Fig. 3. Confusion matrix of CNN model v1

Fig. 4. Predictions made by CNN model v1

The model was trained over 20 epochs, achieving a final accuracy of over 91% and reducing training loss from 1.3972 to 0.2554. The classification report highlights strong performance for the Automobile class (F1-score: 0.7826), while the Airplane class showed lower precision (0.4422) but high recall (0.8880). The overall weighted average F1-score was 0.6512, indicating solid performance with room for improvement in certain classes.

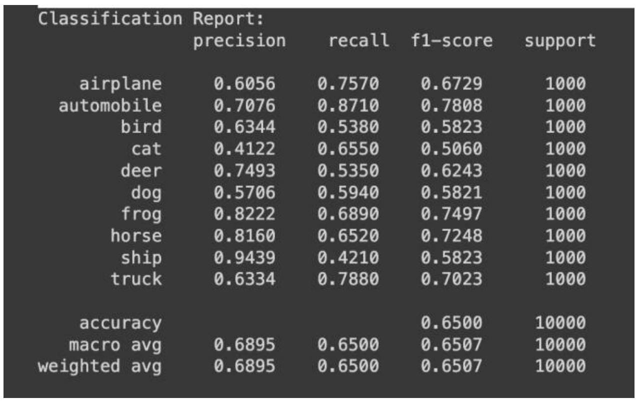
CNN Model V2:

1. 2 convolution & max pool layers
2. 2 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5



```
Epoch 1/20, Train Loss: 1.7029, Validation Loss: 1.4730, Train Accuracy: 37.44%
Epoch 2/20, Train Loss: 1.3384, Validation Loss: 1.2386, Train Accuracy: 51.87%
Epoch 3/20, Train Loss: 1.1652, Validation Loss: 1.3074, Train Accuracy: 59.00%
Epoch 4/20, Train Loss: 1.0534, Validation Loss: 1.2913, Train Accuracy: 63.10%
Epoch 5/20, Train Loss: 0.9731, Validation Loss: 1.1166, Train Accuracy: 65.92%
Epoch 6/20, Train Loss: 0.9103, Validation Loss: 0.9367, Train Accuracy: 68.14%
Epoch 7/20, Train Loss: 0.8588, Validation Loss: 1.4955, Train Accuracy: 70.03%
Epoch 8/20, Train Loss: 0.8159, Validation Loss: 1.2734, Train Accuracy: 71.16%
Epoch 9/20, Train Loss: 0.7775, Validation Loss: 1.0225, Train Accuracy: 72.75%
Epoch 10/20, Train Loss: 0.7456, Validation Loss: 0.9580, Train Accuracy: 73.54%
Epoch 11/20, Train Loss: 0.7155, Validation Loss: 0.9099, Train Accuracy: 74.49%
Epoch 12/20, Train Loss: 0.6845, Validation Loss: 1.1905, Train Accuracy: 75.61%
Epoch 13/20, Train Loss: 0.6610, Validation Loss: 1.0214, Train Accuracy: 76.58%
Epoch 14/20, Train Loss: 0.6527, Validation Loss: 1.6991, Train Accuracy: 76.80%
Epoch 15/20, Train Loss: 0.6206, Validation Loss: 0.9309, Train Accuracy: 77.74%
Epoch 16/20, Train Loss: 0.6083, Validation Loss: 0.9985, Train Accuracy: 78.12%
Epoch 17/20, Train Loss: 0.5959, Validation Loss: 0.8832, Train Accuracy: 78.53%
Epoch 18/20, Train Loss: 0.5773, Validation Loss: 1.2650, Train Accuracy: 79.26%
Epoch 19/20, Train Loss: 0.5739, Validation Loss: 0.9792, Train Accuracy: 79.61%
Epoch 20/20, Train Loss: 0.5569, Validation Loss: 1.1968, Train Accuracy: 80.05%
```

Fig. 5. Output of each Epoch using CNN model v2

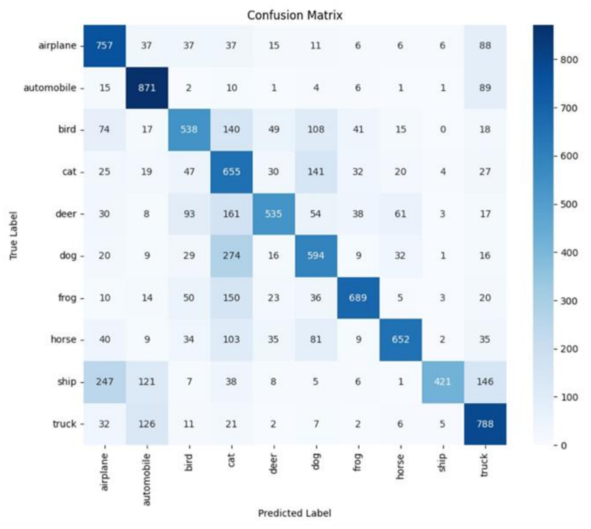


```
Classification Report:
              precision    recall  f1-score   support

    airplane     0.6056    0.7570    0.6729      1000
  automobile     0.7076    0.8710    0.7808      1000
        bird     0.6344    0.5380    0.5823      1000
         cat     0.4122    0.6550    0.5060      1000
        deer     0.7493    0.5350    0.6243      1000
         dog     0.5706    0.5940    0.5821      1000
        frog     0.8222    0.6890    0.7497      1000
       horse     0.8160    0.6520    0.7248      1000
        ship     0.9439    0.4210    0.5823      1000
       truck     0.6334    0.7880    0.7023      1000

    accuracy                         0.6500     10000
   macro avg     0.6895    0.6500    0.6507     10000
weighted avg     0.6895    0.6500    0.6507     10000
```

Fig. 6. Classification report using CNN model v2



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	757	37	37	37	15	11	6	6	6	88
automobile	15	871	2	10	1	4	6	1	1	89
bird	74	17	538	140	49	108	41	15	0	18
cat	25	19	47	655	30	141	32	20	4	27
deer	30	8	93	161	535	54	38	61	3	17
dog	20	9	29	274	16	594	9	32	1	16
frog	10	14	50	150	23	36	689	5	3	20
horse	40	9	34	103	35	81	9	652	2	35
ship	247	121	7	38	8	5	6	1	421	146
truck	32	126	11	21	2	7	2	6	5	788

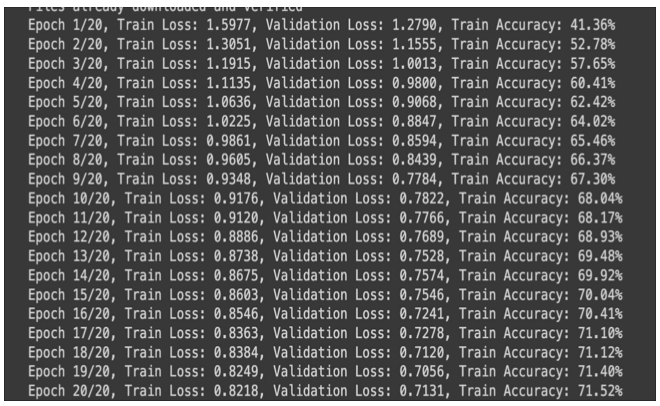
Fig. 7. Confusion matrix of CNN model v2

Fig. 8. Predictions made by CNN model v2

The model achieved 80% training accuracy after running it for 20 Epochs with validation loss of 0.5569. The model classification report shows average weighted accuracy of 68% with only ship class having accuracy over 90%. The average F1-score came 0.6507 showing room for improvement in the model.

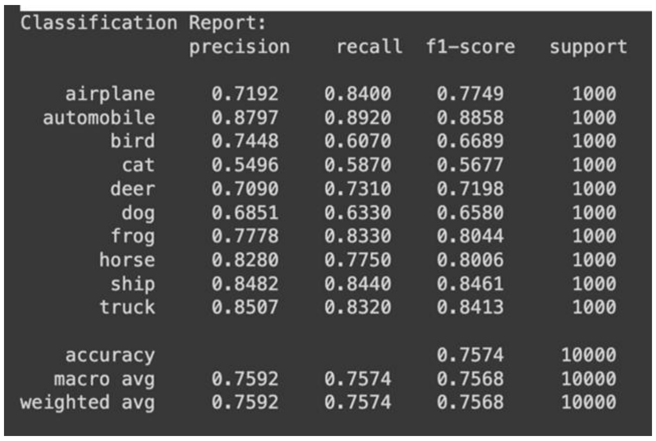
CNN Model V3:

1. 2 convolution & max pool layers
2. 3 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5



```
Files already downloaded and verified
Epoch 1/20, Train Loss: 1.5977, Validation Loss: 1.2790, Train Accuracy: 41.36%
Epoch 2/20, Train Loss: 1.3051, Validation Loss: 1.1555, Train Accuracy: 52.78%
Epoch 3/20, Train Loss: 1.1915, Validation Loss: 1.0013, Train Accuracy: 57.65%
Epoch 4/20, Train Loss: 1.1135, Validation Loss: 0.9800, Train Accuracy: 60.41%
Epoch 5/20, Train Loss: 1.0636, Validation Loss: 0.9068, Train Accuracy: 62.42%
Epoch 6/20, Train Loss: 1.0225, Validation Loss: 0.8847, Train Accuracy: 64.02%
Epoch 7/20, Train Loss: 0.9861, Validation Loss: 0.8594, Train Accuracy: 65.46%
Epoch 8/20, Train Loss: 0.9605, Validation Loss: 0.8439, Train Accuracy: 66.37%
Epoch 9/20, Train Loss: 0.9348, Validation Loss: 0.7784, Train Accuracy: 67.30%
Epoch 10/20, Train Loss: 0.9176, Validation Loss: 0.7822, Train Accuracy: 68.04%
Epoch 11/20, Train Loss: 0.9120, Validation Loss: 0.7766, Train Accuracy: 68.17%
Epoch 12/20, Train Loss: 0.8886, Validation Loss: 0.7689, Train Accuracy: 68.93%
Epoch 13/20, Train Loss: 0.8738, Validation Loss: 0.7528, Train Accuracy: 69.48%
Epoch 14/20, Train Loss: 0.8675, Validation Loss: 0.7574, Train Accuracy: 69.92%
Epoch 15/20, Train Loss: 0.8603, Validation Loss: 0.7546, Train Accuracy: 70.04%
Epoch 16/20, Train Loss: 0.8546, Validation Loss: 0.7241, Train Accuracy: 70.41%
Epoch 17/20, Train Loss: 0.8363, Validation Loss: 0.7278, Train Accuracy: 71.10%
Epoch 18/20, Train Loss: 0.8384, Validation Loss: 0.7120, Train Accuracy: 71.12%
Epoch 19/20, Train Loss: 0.8249, Validation Loss: 0.7056, Train Accuracy: 71.40%
Epoch 20/20, Train Loss: 0.8218, Validation Loss: 0.7131, Train Accuracy: 71.52%
```

Fig. 9. Output of each Epochs using CNN model v3



```
Classification Report:
              precision    recall  f1-score   support

    airplane     0.7192    0.8400    0.7749      1000
  automobile     0.8797    0.8920    0.8858      1000
        bird     0.7448    0.6070    0.6689      1000
         cat     0.5496    0.5870    0.5677      1000
        deer     0.7090    0.7310    0.7198      1000
         dog     0.6851    0.6330    0.6580      1000
        frog     0.7778    0.8330    0.8044      1000
       horse     0.8280    0.7750    0.8006      1000
        ship     0.8482    0.8440    0.8461      1000
       truck     0.8507    0.8320    0.8413      1000

    accuracy                         0.7574     10000
   macro avg     0.7592    0.7574    0.7568     10000
weighted avg     0.7592    0.7574    0.7568     10000
```

Fig 10 Classification report using CNN model v3

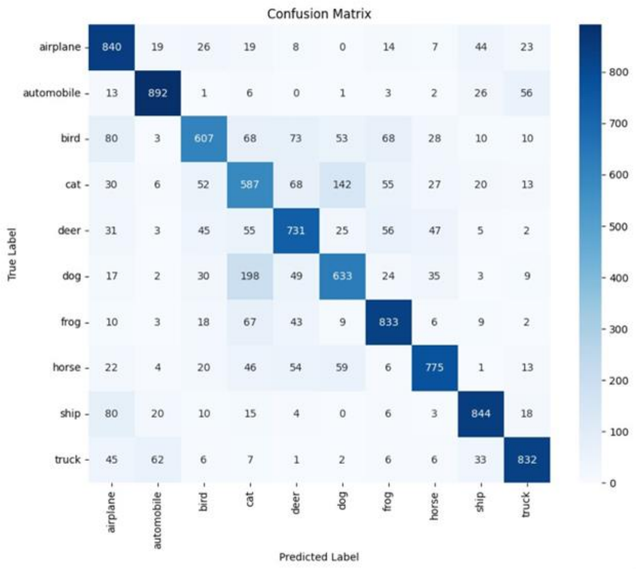


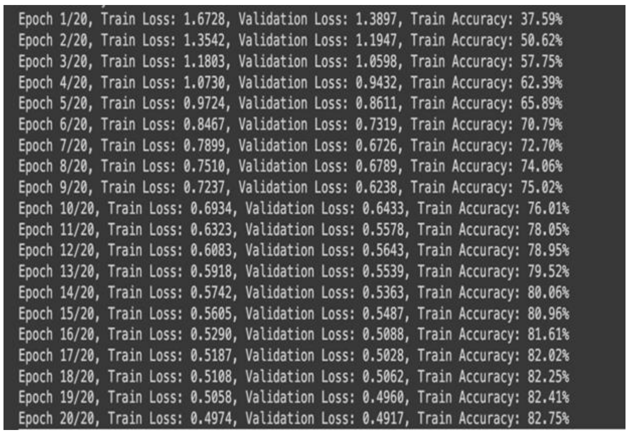
Fig. 11. Confusion matrix of CNN model v3

Fig. 12. Predictions made by CNN model v3

The model's training accuracy increased from 40% to over 71% after 20 epochs. Simultaneously, training loss decreased from 1.5977 to 0.8218. Overall, the model achieved a weighted average precision of 0.7592, recall of 0.7574, and F1-score of 0.7568, indicating balanced performance across most classes, yet also shows much needed improvement.

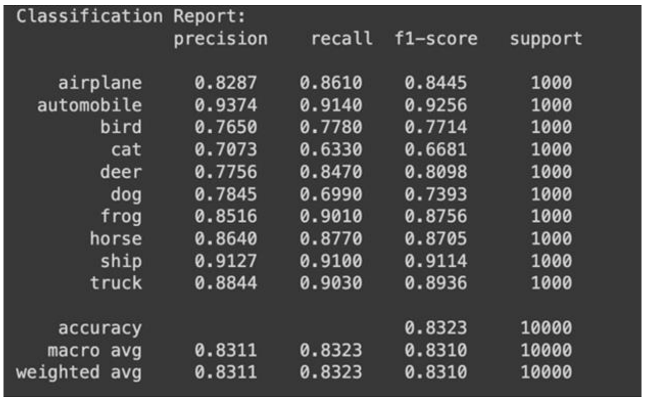
CNN Model V3-V2:

1. 3 convolution & max pool layers
2. 3 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5



```
Epoch 1/20, Train Loss: 1.6728, Validation Loss: 1.3897, Train Accuracy: 37.59%
Epoch 2/20, Train Loss: 1.3542, Validation Loss: 1.1947, Train Accuracy: 50.62%
Epoch 3/20, Train Loss: 1.1803, Validation Loss: 1.0598, Train Accuracy: 57.75%
Epoch 4/20, Train Loss: 1.0730, Validation Loss: 0.9432, Train Accuracy: 62.39%
Epoch 5/20, Train Loss: 0.9724, Validation Loss: 0.8611, Train Accuracy: 65.89%
Epoch 6/20, Train Loss: 0.8467, Validation Loss: 0.7319, Train Accuracy: 70.79%
Epoch 7/20, Train Loss: 0.7899, Validation Loss: 0.6726, Train Accuracy: 72.70%
Epoch 8/20, Train Loss: 0.7510, Validation Loss: 0.6789, Train Accuracy: 74.06%
Epoch 9/20, Train Loss: 0.7237, Validation Loss: 0.6238, Train Accuracy: 75.02%
Epoch 10/20, Train Loss: 0.6934, Validation Loss: 0.6433, Train Accuracy: 76.01%
Epoch 11/20, Train Loss: 0.6323, Validation Loss: 0.5578, Train Accuracy: 78.05%
Epoch 12/20, Train Loss: 0.6083, Validation Loss: 0.5643, Train Accuracy: 78.95%
Epoch 13/20, Train Loss: 0.5918, Validation Loss: 0.5539, Train Accuracy: 79.52%
Epoch 14/20, Train Loss: 0.5742, Validation Loss: 0.5363, Train Accuracy: 80.06%
Epoch 15/20, Train Loss: 0.5605, Validation Loss: 0.5487, Train Accuracy: 80.96%
Epoch 16/20, Train Loss: 0.5290, Validation Loss: 0.5088, Train Accuracy: 81.61%
Epoch 17/20, Train Loss: 0.5187, Validation Loss: 0.5028, Train Accuracy: 82.02%
Epoch 18/20, Train Loss: 0.5108, Validation Loss: 0.5062, Train Accuracy: 82.25%
Epoch 19/20, Train Loss: 0.5058, Validation Loss: 0.4960, Train Accuracy: 82.41%
Epoch 20/20, Train Loss: 0.4974, Validation Loss: 0.4917, Train Accuracy: 82.75%
```

Fig. 13. Output of Epoch using CNN model v3-v2



```
Classification Report:
              precision    recall  f1-score   support

    airplane     0.8287    0.8610    0.8445      1000
  automobile     0.9374    0.9140    0.9256      1000
        bird     0.7650    0.7780    0.7714      1000
         cat     0.7073    0.6330    0.6681      1000
        deer     0.7756    0.8470    0.8098      1000
         dog     0.7845    0.6990    0.7393      1000
        frog     0.8516    0.9010    0.8756      1000
       horse     0.8640    0.8770    0.8705      1000
        ship     0.9127    0.9100    0.9114      1000
       truck     0.8844    0.9030    0.8936      1000

    accuracy                         0.8323     10000
   macro avg     0.8311    0.8323    0.8310     10000
weighted avg     0.8311    0.8323    0.8310     10000
```

Fig 14 Classification report using CNN model v3-v2

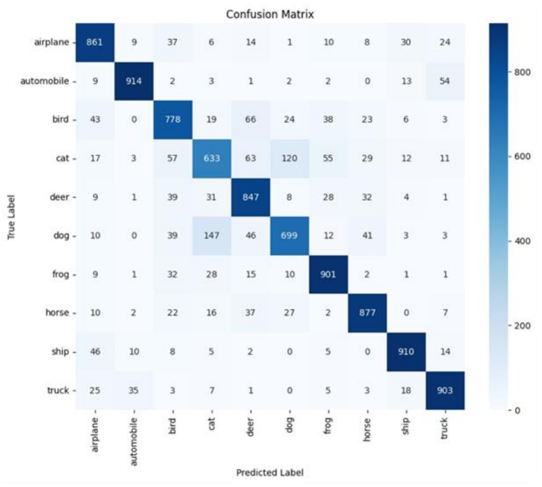


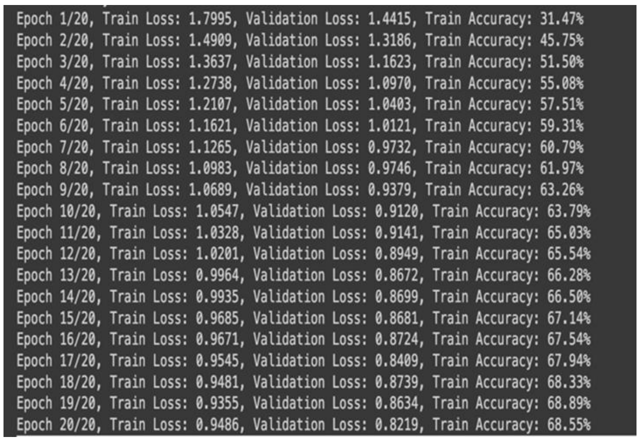
Fig. 15. Confusion matrix of CNN model v3-v2

Fig. 16. Predictions made by CNN model v3-v2

After 20 Epochs training, the model achieved over 83% accuracy. From the report, we see that the model optimally classifies certain classes like Automobiles, ships, and trucks. However, it struggled to classify bird, cat, and dog images showing further need for improvement and adjustments. The average F1-score of the model was 0.8310.

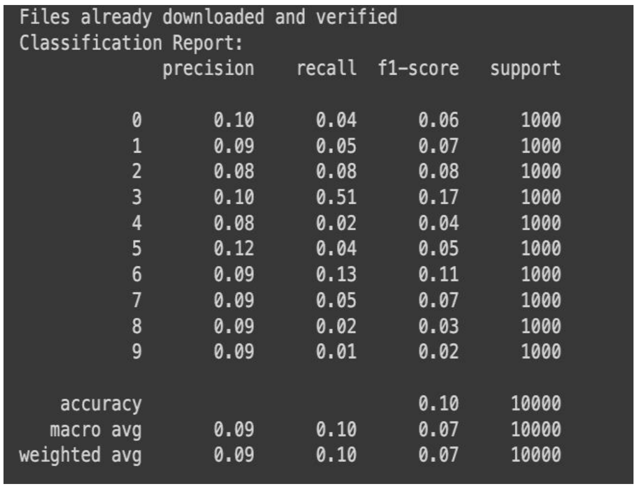
CNN Model V3-V3:

1. 3 convolution & max pool layers
2. 4 fully connected layers
3. Default runtime using 0.2 momentum and dropout value p = 0.5
4. Weights initialized using He weight initialization



```
Epoch 1/20, Train Loss: 1.7995, Validation Loss: 1.4415, Train Accuracy: 31.47%
Epoch 2/20, Train Loss: 1.4909, Validation Loss: 1.3186, Train Accuracy: 45.75%
Epoch 3/20, Train Loss: 1.3637, Validation Loss: 1.1623, Train Accuracy: 51.50%
Epoch 4/20, Train Loss: 1.2738, Validation Loss: 1.0970, Train Accuracy: 55.08%
Epoch 5/20, Train Loss: 1.2107, Validation Loss: 1.0403, Train Accuracy: 57.51%
Epoch 6/20, Train Loss: 1.1621, Validation Loss: 1.0121, Train Accuracy: 59.31%
Epoch 7/20, Train Loss: 1.1265, Validation Loss: 0.9732, Train Accuracy: 60.79%
Epoch 8/20, Train Loss: 1.0983, Validation Loss: 0.9746, Train Accuracy: 61.97%
Epoch 9/20, Train Loss: 1.0689, Validation Loss: 0.9379, Train Accuracy: 63.26%
Epoch 10/20, Train Loss: 1.0547, Validation Loss: 0.9120, Train Accuracy: 63.79%
Epoch 11/20, Train Loss: 1.0328, Validation Loss: 0.9141, Train Accuracy: 65.03%
Epoch 12/20, Train Loss: 1.0201, Validation Loss: 0.8949, Train Accuracy: 65.54%
Epoch 13/20, Train Loss: 0.9964, Validation Loss: 0.8672, Train Accuracy: 66.28%
Epoch 14/20, Train Loss: 0.9935, Validation Loss: 0.8699, Train Accuracy: 66.50%
Epoch 15/20, Train Loss: 0.9685, Validation Loss: 0.8681, Train Accuracy: 67.14%
Epoch 16/20, Train Loss: 0.9671, Validation Loss: 0.8724, Train Accuracy: 67.54%
Epoch 17/20, Train Loss: 0.9545, Validation Loss: 0.8409, Train Accuracy: 67.94%
Epoch 18/20, Train Loss: 0.9481, Validation Loss: 0.8739, Train Accuracy: 68.33%
Epoch 19/20, Train Loss: 0.9355, Validation Loss: 0.8634, Train Accuracy: 68.89%
Epoch 20/20, Train Loss: 0.9486, Validation Loss: 0.8219, Train Accuracy: 68.55%
```

Fig. 17. Output of each Epoch using CNN model v3-v3



```
Files already downloaded and verified
Classification Report:
              precision    recall  f1-score   support

           0       0.10      0.04      0.06      1000
           1       0.09      0.05      0.07      1000
           2       0.08      0.08      0.08      1000
           3       0.10      0.51      0.17      1000
           4       0.08      0.02      0.04      1000
           5       0.12      0.04      0.05      1000
           6       0.09      0.13      0.11      1000
           7       0.09      0.05      0.07      1000
           8       0.09      0.02      0.03      1000
           9       0.09      0.01      0.02      1000

    accuracy                           0.10     10000
   macro avg       0.09      0.10      0.07     10000
weighted avg       0.09      0.10      0.07     10000
```

Fig 18 Classification report using CNN model v3-v3

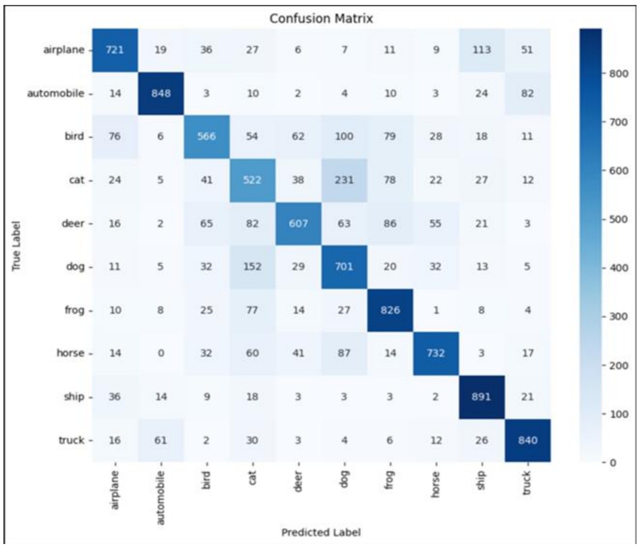


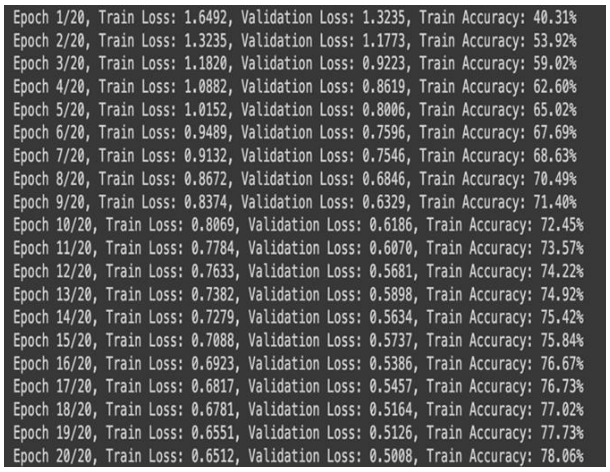
Fig. 19. Confusion matrix of CNN model v3-v3

Fig. 20. Predictions made by CNN model v3-v3

The CNN model V3-V3 demonstrates a decline in both training and validation losses, with training accuracy of less than 70%. However, it struggles to generalize, achieving low validation accuracy. The classification report highlights poor performance across most classes, with class cat having the highest recall and F1-score. The confusion matrix shows a strong tendency to misclassify, frequently predicting cat for various true labels.

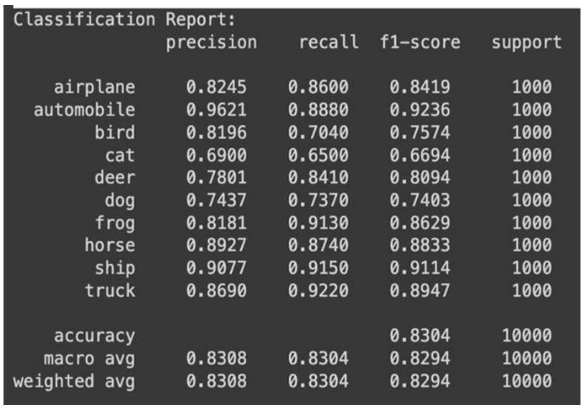
CNN Model V3-V4:

1. Adding one more hidden layer & dropout value & one more convolution layer
2. Total 3 hidden layers, 3 convolution layer & Batch Normalization



```
Epoch 1/20, Train Loss: 1.6492, Validation Loss: 1.3235, Train Accuracy: 40.31%
Epoch 2/20, Train Loss: 1.3235, Validation Loss: 1.1773, Train Accuracy: 53.92%
Epoch 3/20, Train Loss: 1.1820, Validation Loss: 0.9223, Train Accuracy: 59.02%
Epoch 4/20, Train Loss: 1.0882, Validation Loss: 0.8619, Train Accuracy: 62.60%
Epoch 5/20, Train Loss: 1.0152, Validation Loss: 0.8006, Train Accuracy: 65.02%
Epoch 6/20, Train Loss: 0.9489, Validation Loss: 0.7596, Train Accuracy: 67.69%
Epoch 7/20, Train Loss: 0.9132, Validation Loss: 0.7546, Train Accuracy: 68.63%
Epoch 8/20, Train Loss: 0.8672, Validation Loss: 0.6846, Train Accuracy: 70.49%
Epoch 9/20, Train Loss: 0.8374, Validation Loss: 0.6329, Train Accuracy: 71.40%
Epoch 10/20, Train Loss: 0.8069, Validation Loss: 0.6186, Train Accuracy: 72.45%
Epoch 11/20, Train Loss: 0.7784, Validation Loss: 0.6070, Train Accuracy: 73.57%
Epoch 12/20, Train Loss: 0.7633, Validation Loss: 0.5681, Train Accuracy: 74.22%
Epoch 13/20, Train Loss: 0.7382, Validation Loss: 0.5898, Train Accuracy: 74.92%
Epoch 14/20, Train Loss: 0.7279, Validation Loss: 0.5634, Train Accuracy: 75.42%
Epoch 15/20, Train Loss: 0.7088, Validation Loss: 0.5737, Train Accuracy: 75.84%
Epoch 16/20, Train Loss: 0.6923, Validation Loss: 0.5386, Train Accuracy: 76.67%
Epoch 17/20, Train Loss: 0.6817, Validation Loss: 0.5457, Train Accuracy: 76.73%
Epoch 18/20, Train Loss: 0.6781, Validation Loss: 0.5164, Train Accuracy: 77.02%
Epoch 19/20, Train Loss: 0.6551, Validation Loss: 0.5126, Train Accuracy: 77.73%
Epoch 20/20, Train Loss: 0.6512, Validation Loss: 0.5008, Train Accuracy: 78.06%
```

Fig. 21. Output of each Epoch using CNN model v3-v4



```
Classification Report:
              precision    recall  f1-score   support

    airplane     0.8245    0.8600    0.8419      1000
  automobile     0.9621    0.8880    0.9236      1000
        bird     0.8196    0.7040    0.7574      1000
         cat     0.6900    0.6500    0.6694      1000
        deer     0.7801    0.8410    0.8094      1000
         dog     0.7437    0.7370    0.7403      1000
        frog     0.8181    0.9130    0.8629      1000
       horse     0.8927    0.8740    0.8833      1000
        ship     0.9077    0.9150    0.9114      1000
       truck     0.8690    0.9220    0.8947      1000

    accuracy                         0.8304     10000
   macro avg     0.8308    0.8304    0.8294     10000
weighted avg     0.8308    0.8304    0.8294     10000
```

Fig 22 Classification report using CNN model v3-v4

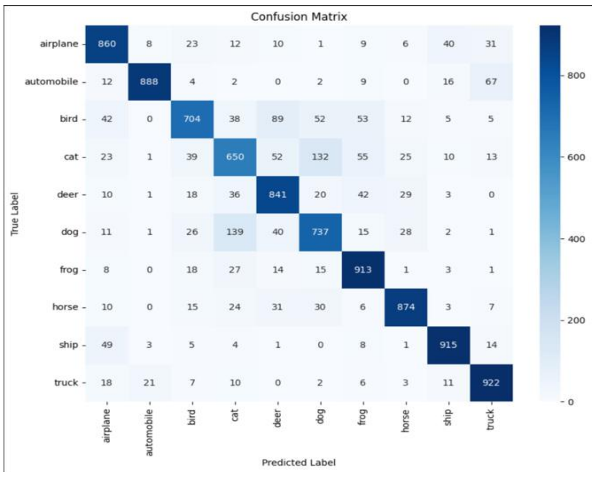


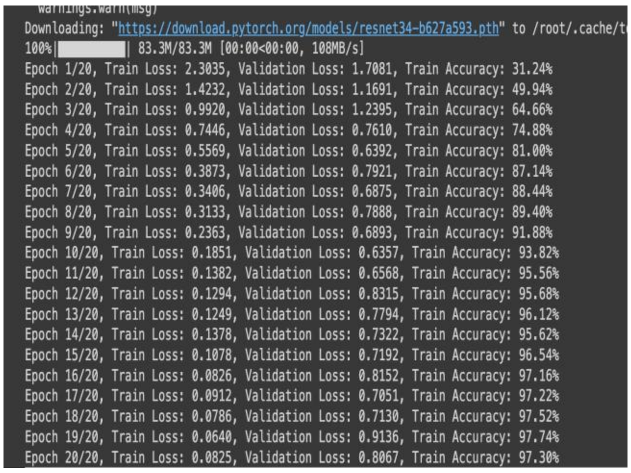
Fig. 23. Confusion matrix of CNN model v3-v4

Fig. 24. Predictions made by CNN model v3-v4

The model achieved over 83% accuracy in classifying the images with an average F1-score of 0.8294 and 0.8304 recall score. It showed a strong performance in classifying classes like Automobile, Ship and Truck with Automobile class having 96.21% precision and 88.80% recall. Classes Bird, Cat and Dog show low precision and recall hinting need for improvement and adjustments to the model.

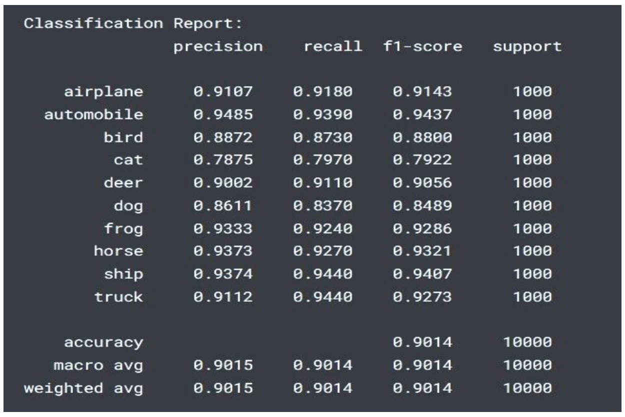
CNN Model Resnet:

The CNN model ResNet enhances deep learning accuracy through residual connections, enabling deeper networks. This architecture addresses the vanishing gradient problem by allowing alternate pathways for gradients to flow, aiding the effective training of deep networks.



```
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet34-b627a593.pth" to /root/.cache/t
100%|██████████| 83.3M/83.3M [00:00<00:00, 108MB/s]
Epoch 1/20, Train Loss: 2.3035, Validation Loss: 1.7081, Train Accuracy: 31.24%
Epoch 2/20, Train Loss: 1.4232, Validation Loss: 1.1691, Train Accuracy: 49.94%
Epoch 3/20, Train Loss: 0.9920, Validation Loss: 1.2395, Train Accuracy: 64.66%
Epoch 4/20, Train Loss: 0.7446, Validation Loss: 0.7610, Train Accuracy: 74.88%
Epoch 5/20, Train Loss: 0.5569, Validation Loss: 0.6392, Train Accuracy: 81.00%
Epoch 6/20, Train Loss: 0.3873, Validation Loss: 0.7921, Train Accuracy: 87.14%
Epoch 7/20, Train Loss: 0.3406, Validation Loss: 0.6875, Train Accuracy: 88.44%
Epoch 8/20, Train Loss: 0.3133, Validation Loss: 0.7888, Train Accuracy: 89.40%
Epoch 9/20, Train Loss: 0.2363, Validation Loss: 0.6893, Train Accuracy: 91.88%
Epoch 10/20, Train Loss: 0.1851, Validation Loss: 0.6357, Train Accuracy: 93.82%
Epoch 11/20, Train Loss: 0.1382, Validation Loss: 0.6568, Train Accuracy: 95.56%
Epoch 12/20, Train Loss: 0.1294, Validation Loss: 0.8315, Train Accuracy: 95.68%
Epoch 13/20, Train Loss: 0.1249, Validation Loss: 0.7794, Train Accuracy: 96.12%
Epoch 14/20, Train Loss: 0.1378, Validation Loss: 0.7322, Train Accuracy: 95.62%
Epoch 15/20, Train Loss: 0.1078, Validation Loss: 0.7192, Train Accuracy: 96.54%
Epoch 16/20, Train Loss: 0.0826, Validation Loss: 0.8152, Train Accuracy: 97.16%
Epoch 17/20, Train Loss: 0.0912, Validation Loss: 0.7051, Train Accuracy: 97.22%
Epoch 18/20, Train Loss: 0.0786, Validation Loss: 0.7130, Train Accuracy: 97.52%
Epoch 19/20, Train Loss: 0.0640, Validation Loss: 0.9136, Train Accuracy: 97.74%
Epoch 20/20, Train Loss: 0.0825, Validation Loss: 0.8067, Train Accuracy: 97.30%
```

Fig. 25. Output of each Epoch using CNN model Resnet



```
Classification Report:
              precision    recall  f1-score   support

    airplane     0.9107    0.9180    0.9143      1000
  automobile     0.9485    0.9390    0.9437      1000
        bird     0.8872    0.8730    0.8800      1000
         cat     0.7875    0.7970    0.7922      1000
        deer     0.9002    0.9110    0.9056      1000
         dog     0.8611    0.8370    0.8489      1000
        frog     0.9333    0.9240    0.9286      1000
       horse     0.9373    0.9270    0.9321      1000
        ship     0.9374    0.9440    0.9407      1000
       truck     0.9112    0.9440    0.9273      1000

    accuracy                         0.9014     10000
   macro avg     0.9015    0.9014    0.9014     10000
weighted avg     0.9015    0.9014    0.9014     10000
```

Fig. 26. Classification report using CNN model Resnet

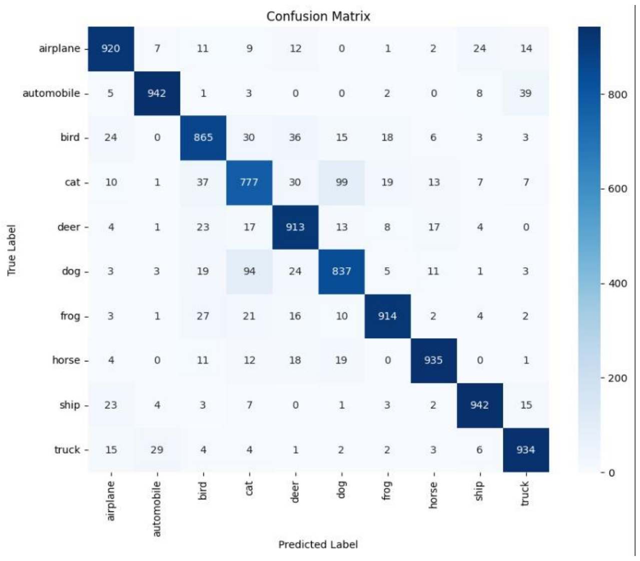
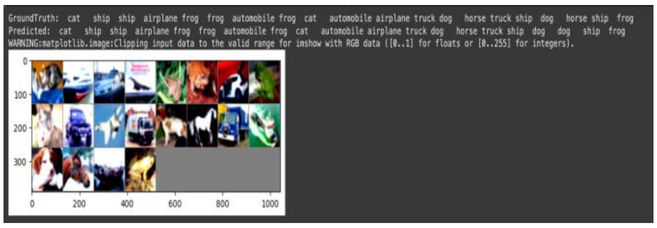


Fig. 27. Confusion matrix of CNN model Resnet



```
GroundTruth:  cat   ship  ship  airplane frog  frog  automobile frog  cat   automobile airplane truck dog   horse truck ship  dog   horse ship  frog
Predicted:  cat   ship  ship  airplane frog  frog  automobile frog  cat   automobile airplane truck dog   horse truck ship  dog   dog   ship  frog
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Fig. 28. Prediction output of CNN model using Resnet

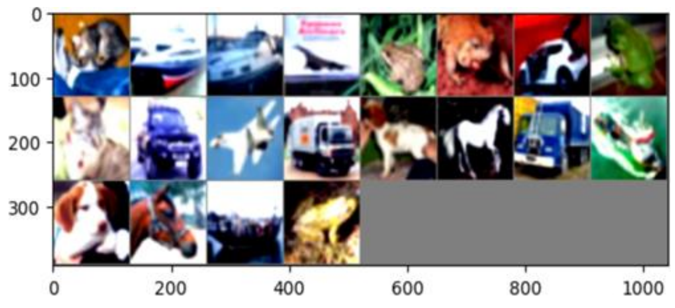


Fig. 29. Prediction results

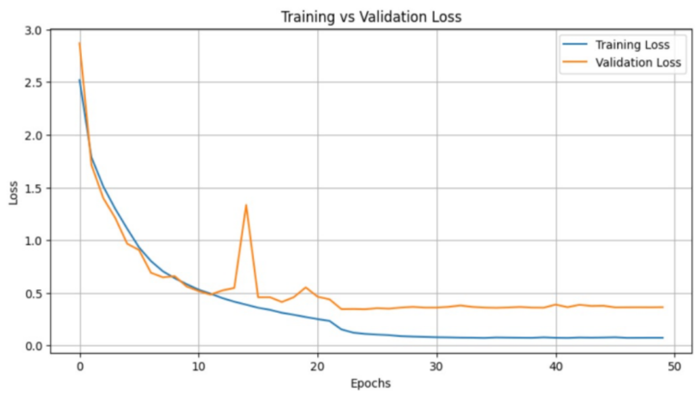
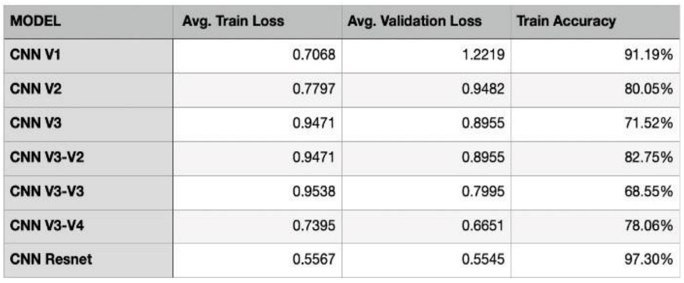


Fig. 30. Training and validation loss graph for Resnet

The graph shows the training and validation loss of a ResNet-34 model over 50 epochs. Initially, both losses decrease rapidly, indicating that the model is learning effectively. Around epoch 10, the training loss continues to steadily decrease, suggesting the model is fitting well to the training data. However, the validation loss plateaus and shows fluctuations, with a noticeable spike around epoch 15, indicating potential overfitting. Beyond epoch 20, the validation loss remains consistently higher than the training loss, which signifies that the model is learning to fit the training data but struggles to generalize to the validation set.



MODEL	Avg. Train Loss	Avg. Validation Loss	Train Accuracy
CNN V1	0.7068	1.2219	91.19%
CNN V2	0.7797	0.9482	80.05%
CNN V3	0.9471	0.8955	71.52%
CNN V3-V2	0.9471	0.8955	82.75%
CNN V3-V3	0.9538	0.7995	68.55%
CNN V3-V4	0.7395	0.6651	78.06%
CNN Resnet	0.5567	0.5545	97.30%

Fig. 31. Comparison data of all CNN models

The table summarizes the performance of various CNN models that were trained and used for this paper. Amongst these models, ResNet emerged on top achieving the lowest average training and validation losses and the highest training accuracy, recall and F1-score. This suggests that the ResNet architecture was particularly effective in learning the underlying patterns in the Cifar10 data. While CNN V1 exhibited high training accuracy, it suffered from overfitting, as evidenced by the significant gap between training and validation loss. Other models, such as CNN V2 and CNN V3, demonstrated moderate performance, with variations in training accuracy. Different variants of CNN V3 showed mixed results, indicating that architectural modifications did not consistently improve performance. Overall, ResNet architecture's superior performance highlights its effectiveness in classifying Cifar10 images.

5. CONCLUSIONS

CNN models, including simple CNNs and advanced ResNet-34 architectures, effectively classify images. Simple CNNs are quick and efficient for smaller datasets but limited in performance. ResNet-34, with deeper layers and residual connections, offers significantly better accuracy. For high accuracy and large datasets, ResNet-34 with SWA and mixed precision is ideal. For quicker, less resource-intensive tasks, a simple CNN suffices. The choice depends on the application's complexity and available resources.

REFERENCES

[1] D. Kornack and P. Rakic, “Cell Proliferation without Neurogenesis in Adult Primate Neocortex,” Science, vol.

294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

[2] Neha Sharma, Vibhor Jain, Anju Mishra, An Analysis Of Convolutional Neural Networks For Image Classification, Procedia Computer Science, Volume 132, 2018, Pages 377-384, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.198.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In NIPS. 1106–1114.

[4] F. Sultana, A. Sufian and P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 2018, pp. 122-129, doi: 10.1109/ICRCICN.2018.8718718.

[5] Ajala, Sunday. (2021). Convolutional Neural Network Implementation for Image Classification using CIFAR-10 Dataset. 10.13140/RG.2.2.27690.54724.

[6] S. Y. Chaganti, I. Nanda, K. R. Pandi, T. G. N. R. S. N. Prudhvith and N. Kumar, "Image Classification using SVM and CNN," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2020, pp. 1-5, doi: 10.1109/ICCSEA49413.2020.9132851. keywords: {Support vector machines;Deep learning;Computer science;Neural networks;Education;Support vector machine;Hardware;Python;Support Vector Machines (SVM) and Convolutional Neural Networks (CNN)}.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, & Neil Houlsby. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

[8] Sharma, Atul. (2021). INTERNATIONAL CONFERENCE ON INNOVATIVE COMPUTING AND COMMUNICATION (ICICC 2021) Image Classification Using CNN.

[9] Image Classification using CNN with CIFAR-10 Dataset", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.10, Issue 7, page no. 702-707, July, 2023.

[10] Adetiba, E., Ajayi, O. T., Kala, J. R., Badejo, J. A., Ajala, S. & Abayomi, A. (2021). LeafsnapNet: An Experimentally Evolved Deep Learning Model for Recognition of Plant Species based on Leafsnap Image Dataset. Journal of Computer Science, 17(3), 349-363. https://doi.org/10.3844/jcssp.2021.349.36.

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... &Rabinovich, A. (2015) "Going deeper with convolutions." in Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[12] Chen, L.; Li, S.; Bai, Q.; Yang, J.; Jiang, S.; Miao, Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. Remote Sens. 2021, 13, 4712. https://doi.org/10.3390/rs13224712.

[13] P. Gavali and J. Saira Banu, "Deep Convolutional Neural Network for Image Classification on CUDA Platform" in Deep Learning and Parallel Computing Environment for Bioengineering Systems, A. K. Sangaiah, Ed., Academic Press, 2019, pp. 99-122.

[14] R. Doon, T. Kumar Rawat and S. Gautam, "Cifar-10 Classification using Deep Convolutional Neural Network," 2018 IEEE Punecon, Pune, India, 2018, pp. 1-5, doi: 10.1109/PUNECON.2018.8745428. keywords: {Convolution;Max-pooling;CIFAR-10}.

[15] Pang, Y., Shah, G., Ojha, R., Thapa, R., & Bhatta, B. (2023). Comparison of CNN Architecture on Image Classification Using CIFAR10 Datasets. International Journal on Engineering Technology, 1(1), 37–52. https://doi.org/10.3126/injet.v1i1.60898.