CIFAR-100 Classification

1. Abstract

This report will introduce some classification methods to solve image classification problem on the dataset CIFAR-100. The CIFAR-100 dataset consists of 60000 images with 100 classes and 20 superclasses, where each class contains 500 training data and 100 testing data. We are using 3 machine learning algorithms to demonstrate this task: K-Nearest Neighbors, Convolutional Neural Networks and Random Forests. We discuss these algorithms in detail and compare their characteristics, we also provide an analysis of their performance on the dataset. We found out that CNN has the highest accuracy among all models, but it comes with a significant time cost.

2. Introduction

Computer vision and image classification is one of the most important fields of the machine learning and data mining industry. Computer vision is all about letting computers intelligently 'see' like human eyes and analyze, study and classify like humans. With the rise of technology today, machine learning includes deep learning and is playing a huge role in everyday life. For example, self-driving cars rely on deep learning models to identify the objects around the car, image search sites need to use machine learning engines to classify what is inside of an image and display relevant content.

This report is intended to solve a classification problem based on CIFAR-100 dataset which contains 60000 32*32 colour images with 100 classes. The 100 classes are grouped into 20 superclass and the 60000 images are separated with 50000 training images and 10000 testing images, which means, each class contains 500 training images and 100 testing images. The aim of this study is to apply machine learning techniques such as K-Nearest Neighbor, decision trees and CNN algorithms to dataset with zero component analysis and other data preprocessing processes to classify each image into the correct class.

3. Previous Work

It is extremely popular to use deep learning concepts, especially convolutional neural networks for previous models of CIFAR-100 dataset since CNN can obtain better results in image and voice recognition.

The CIFAR-100 dataset is a very popular benchmark and is included as a showcase of performance in many cutting-edge research papers. The best performing model for this task is EfficientNet-B7[1] which produces an accuracy of 91.3% on the CIFAR-100 dataset. An even higher accuracy can be achieved using the techniques described in Sharpness Aware Minimizations(SAM) for Efficiently Improving Generalization[1], which is 96.08%. SAM is an optimizer and is used to boost the performance of already existing CNN architectures. The main idea behind it is that due to the large number of parameters that exist in large models that are utilised for classification of large datasets like CIFAR-100, EfficientNet-B7 being one of these models. The model has a total of 64 million parameters.

We want to also bring attention to the BiT-L[2] model which produces an accuracy of 93.51%. The true power is the aforementioned two model implementations comes from them being trained on a general large dataset that is larger than their benchmark datasets. The highest accuracy model trained on only the CIFAR-100 dataset is ColorNet which is 88.4% and has 19 million parameters.

Pedro Savarese and Michael Maire reduced parameters of CNN by using a method called parameter sharing scheme where the parameters are shared between different layers, to achieve a more modular and compact CNN model compared to normal CNN. In addition, the team finds that after constructing a model, parameter sharing of the model results in different layers being functionally equivalent, allowing them to collapse them into recurrent blocks so that the model can reduce the training time and cost with a better performance. [4]

Irwan Bello and his team uses self-attention as an alternative model compared to normal CNN since convolution can only operate on local instances but missing global instances. The team designed a self-attention mechanism with 2-D that can capture long range interactions and combine with convolutions to prevent the weakness of normal CNN, so that the performance on image classification increases significantly. [5]

We observe that all the high performing algorithms on this classification task are CNNs and thus one of our chosen algorithms is infact CNN. Our approach to it is highly different from these state of the art models though. Due to the lack of high performance hardware, we have chosen a simple neural network architecture with a total of 358,100 parameters. We have also chosen to showcase the performance of KNN and Random Forest classifiers and we expect our simple CNN model to outperform these methods. This expectation is due to the suitability of CNN to image based classification tasks.

4. Methods

4.1 KNN

KNN, A.K.A K-NearestNeighbor algorithm is one of supervised learning algorithms which is commonly used in computer vision and data mining industry. KNN in one sentence is that if most of the K most similar (i.e. the nearest neighbors in the feature space) training data of a testing data is categorized as a certain label, the testing data also belongs to this label. The implement of KNN is as followed:

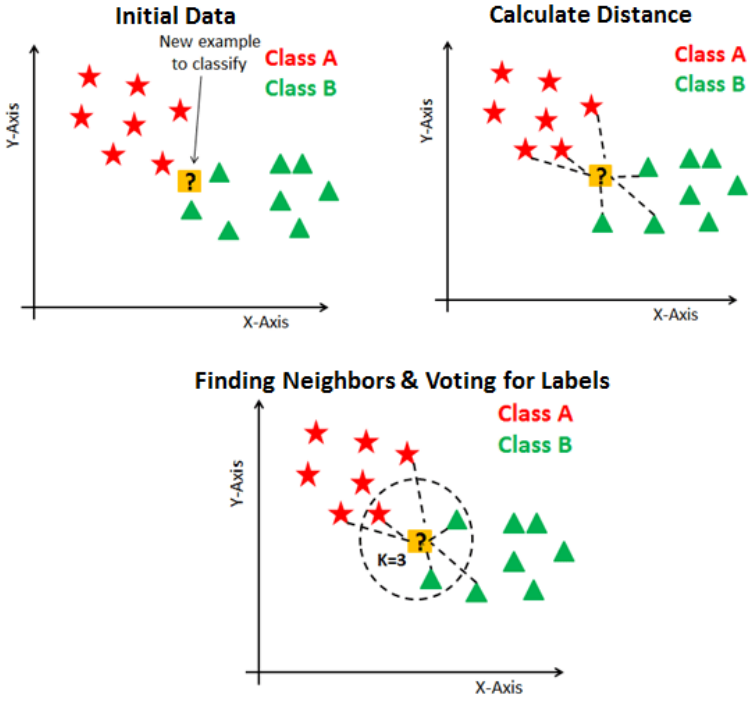
1) Calculate the distance between the test data point and each training data point.

2) Sort the training data points with increasing order of distance.

3) Select the first K points which have the smallest value of distance.

4) Determine the frequency of each category of the selected points.

5) Return the category label with the highest frequency among the select data points as the predicted category of the test data.



(Figure 2: How does knn work) [7]

4.1.1 Hyperparameters

The hyperparameter in the case of KNN is K i.e the number of neighbours to be considered before labelling the test point. The optimum value for K is found using grid search.

4.1.2 Preprocessing methods

4.1.2.1 Normalization

The process can be divided into two steps, centering and scaling:

- Centering data: This step is achieved by subtracting the mean of each feature. The center of the model shifts to 0 but the slope of the regression line is kept but the intercept is equal to zero so that it can avoid overfitting.
- Scaling data: After centering data, we performed scaling data by dividing standard deviation. This can stretch data from different dimensions to the same level of scale.

4.1.2.2 PCA by SVD

We performed a principal component analysis by doing SVD. SVD is the process of decomposing original matrix X into:

$$X = U\Sigma V^T$$

$m \times r$ $\quad r \times r$ $\quad r \times n$

where the middle rxr matrix is a diagonal matrix which contains the singular values of matrix X. By taking the first k numbers of singular value of rxr matrix and times to U and VT we will get a new matrix X' where X' is dimension reduced but keeps most information compare to X

4.1.3 Loss Function

The loss function used to monitor the accuracy of the model is Euclidean distance, given by:

$$D(X,Y) = (\sum_{i=1}^{n} (x_i - y_i)^2)^{1/2}$$

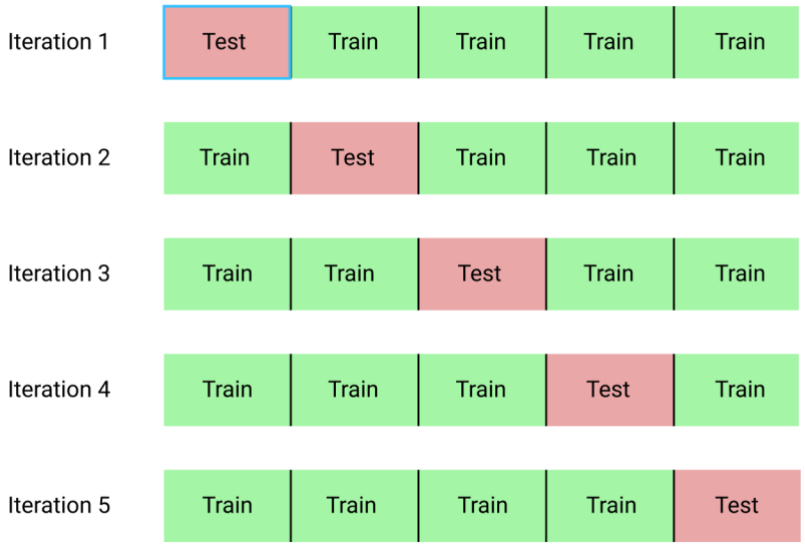
We have chosen to use this loss function because it is the standard loss function to use in the case of 2-dimensional KNN models.

4.1.4 Evaluation

We use 10 fold cross-validation and a held out test set to evaluate our model. We would like to discuss about cross-validation in detail here:

The basic idea of cross-validation is to divide the original data set into a training set validation set, then we use the training set to train the classifier, and we use the validation set to valid the performance of the trained model and fine-tuning hyperparameters to make sure the model is in best fit, to avoid overfitting problem.

The cross-validation method we used in this case is 10-fold cross validation. First, shuffle the dataset randomly to avoid bias of the dataset. Then we split the data set into 10 sets with an equal number of datasets, and for each iteration, train the model with 9 sets and apply it on the 10th set, which we call validation set and calculate the accuracy. The iteration takes 10 times so that each set is used as a validation set once. Each test will get the corresponding accuracy. The average accuracy of the results of 10 times is used as an estimate of the accuracy of the algorithm.



(Figure 1: The fundamental of cross validation) [6]

4.2 Random Forests

Random Forests is an ensemble training method that uses a large number of decision trees and takes votes from all the decision trees. The label that gets the most votes is the predicted output of the model.

4.2.1 Decision Trees

Decision Tree is one of the simplest machine learning algorithms where decision tree is a supervised learning algorithm based on if-then-else rules. These rules are trained by training

inside of the tree. Each decision node represents a decision on an attribute, each branch represents the output of a decision, and finally each leaf node represents a classification result.

4.2.2 Hyperparameters

The hyperparameter for Random Forests is the maximum depth that we allow for all the decision trees that are part of the ensemble. We perform grid search to choose the optimum hyperparameter.

4.2.3 Preprocessing

The preprocessing methods used are normalisation followed by Principal Component Analysis using Singular Value Decomposition which have been discussed in the previous section.

4.2.4 Loss Function

The loss function of the model corresponds to the loss function used in each decision tree. The criteria to quantify the quality of a split in a decision tree is the loss function for this model. We use Gini Index as the criteria to evaluate our model.

Gini Index is used when performing feature selection at each node for a decision tree, the tree needs to choose the best suitable feature, so a selection rule is applied. The basic principle for feature selection is to minimize the size of the final decision tree. The rule for Gini Index is to select the parameter with the greatest reduction in impurity. The formula of Gini Index is:

$$gini(D) = 1 - \sum_{j=1}^{n} {p_j}^2$$

4.2.5 Evaluation

We evaluate the model using a held out test set and 10 fold cross-validation process,

4.2.3 CNN

4.1.2 Data Augmentation

- ZCA:

  Zero component analysis is one of the data preprocessing techniques we used in this project. The ZCA whitening ensures that the variance of each dimension of the data remains the same. ZCA whitening is mainly used to reduce the correlation between different features, and try to make the whitened data close to the original input data.

- Horizontal Flip:

We performed a horizontal flip on the dataset so that all images are flipped 180 degree horizontally.

- Rotation:

  We also rotate images after horizontal flip to make sure the input data for the model is in best fit.

Convolutional Neural Networks, CNN is one of the most commonly used feedforward neural networks and also a deep learning model. The concept of CNN consists of the following layer:

- Convolution layer: The parameters of the convolution layer is a set of filters and the activation function. The filters are spatial relationships between pixels and slide the image to generate a 2-D feature map of each filter. The activation function we used is Rectified Linear Unit, ReLU which has a formula of:

  $$max(0, w^T x + b)$$

  The ReLU activation has an advantage over other activation functions as it is computationally efficient as it decreases computation of neurons that produce outputs of less significance by making their output zero. This leads to the creation of some neurons that are never trained and makes certain parts of the network 'dead', but we choose this activation function for it's superior efficiency.

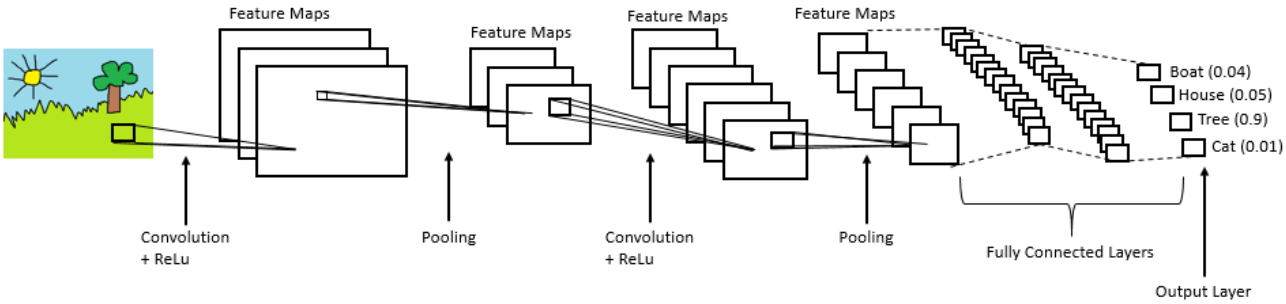
- Pooling layer: Pooling layer has no parameter and plays a role of reducing spatial dimensionality and keeping the most information. In this classification problem, we will use max pooling operation which means we split the input into n x n windows that do not overlap and take the largest value within that window.

- Fully connected (dense) layer: This is the layer that all neurons connect to each other with weight, usually the fully connected layer is at the end of the convolutional neural network. There are two fully connected layers at the end of the network and they use different activation functions to produce their output. The first dense layer utilises the ReLU activation function and the second one uses the Softmax activation function. The Softmax activation function is given by:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \quad true$$

where i is the i-th class, C is the total number of classes and $e^z$is the exponential of the

vector z



(Figure 3: Process of CNN) [8]

4.3 Other Methods

4.3.1 Adam Optimizer

One of the optimizing methods we used is Adam Optimizer. It is very common now to use Adam Optimizer in deep learning models, especially tasks such as computer vision and natural language processing. Adam optimization is an extension of the stochastic gradient descent algorithm.

Compared to stochastic gradient descent which keeps one unchanging learning rate for training, Adam designs independent adaptive learning rates for different parameters by calculating the first moment estimation and the second moment estimation of gradient. So by using Adam Optimizer, we can reduce training cost significantly and solve the CNN CIFAR-100 problem which is a deep learning problem more efficiently. [3]

4.3.2 Loss function:

We choose categorical cross-entropy as our loss function of choice. The goal of the training process is to reduce this loss and hence we need to ensure it is consistent with what we want our deep learning model to do.

The categorical cross-entropy loss function is a loss function that calculates the cross-entropy between the probability distributions of the predicted labels and the given labels.

$$Loss = -\sum {y_i}^{true}.log y_i$$

Where $y_i$is the probability of the input image to belong to the i-th class as predicted by our model and ${y_i}^{true}$is the label that is given to the image as part of the dataset. ${y_i}^{true}$is 1 for the correct class and 0 for all other classes. The negative sign ensures that the loss is getting minimised as ${y_i}^{true}$and $y_i$get closer to each other.

5. Experiment Methodology and Observations

All experiments are performed on a machine with the following specifications:

Processor: Intel Core i7 @ 2.2GHz (12 CPUs)

Ram: 8 GB

Graphics Card: 8 GB NVIDIA GeForce GTX 1060 Max-Q

5.1 KNN

5.1.1 Preprocessing

The KNN model is implemented by first preprocessing the input data using normalisation and Singular Value Decomposition.

5.1.2 Observation and Discussion

The knn model with normalization and PCA data preprocessing methods on dataset has two hyperparameters: pca components and k. When choosing the right number of pca components, we want to keep most of variance so we decide to keep at least 80% of variance therefore we found that having a pca component number as 27 can give us at least 80% of variance kept and reduce most unimportant dimension. We find that even there are 3072 dimensions, but first 27 dimensions contains 80% of information. So, performing PCA becomes a critical to reduce the resource cost especially time cost when training large scale of data.

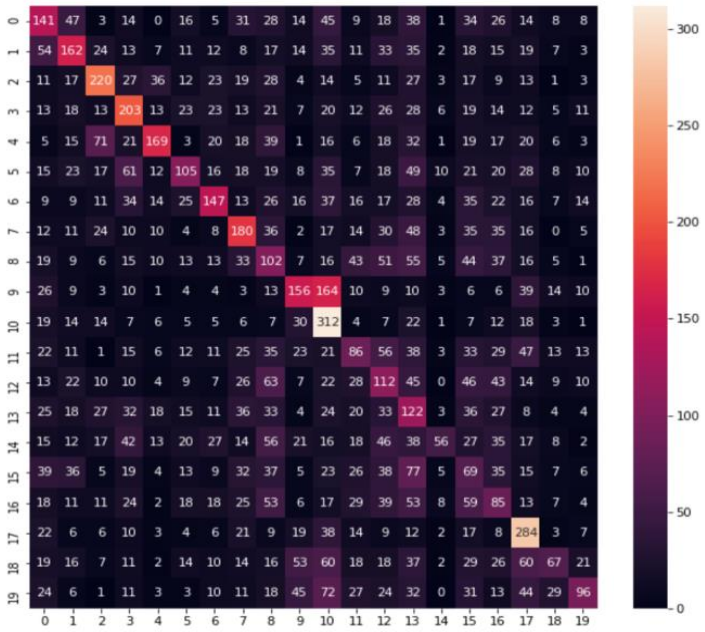
Compare to other models we constructed, knn model is the easiest model to construct and the time cost is the lowest.

K	Validation Accuracy	Time in second
5	0.27664	101
10	0.28506	112
15	0.28856	118
20	0.28758	121
25	0.28572	124

30	0.28458	125

(Table 1: accuracy and time cost on different K)

From table 1 we notice that as k increases, the time cost increases but not increases significantly. The accuracy first increases but then decreases. So picking too large k or too small k is always not the most suitable k value for a model. A too small k will make the model sensitive to noise and a too large k will make k neighbors include points from other class. Since k=15 produce best validation accuracy, we choose k=15 and observe the accuracy and other metrics on the test set. The model produces an accuracy of 0.2905 on the test, the confusion matrix and the table with recall, accuracy and f1 score of each label can be found below:



(Figure 4 Confusion matrix of knn)

	precision	recall	f1-score	support
0	0.27	0.28	0.28	500
1	0.34	0.32	0.33	500
2	0.45	0.44	0.44	500
3	0.34	0.41	0.37	500
4	0.51	0.34	0.41	500
5	0.32	0.21	0.25	500
6	0.38	0.29	0.33	500
7	0.33	0.36	0.34	500
8	0.16	0.20	0.18	500
9	0.35	0.31	0.33	500
10	0.31	0.62	0.41	500
11	0.21	0.17	0.19	500
12	0.18	0.22	0.20	500
13	0.15	0.24	0.18	500
14	0.47	0.11	0.18	500
15	0.11	0.14	0.13	500
16	0.17	0.17	0.17	500
17	0.40	0.57	0.47	500
18	0.32	0.13	0.19	500
19	0.41	0.19	0.26	500
accuracy			0.29	10000
macro avg	0.31	0.29	0.28	10000
weighted avg	0.31	0.29	0.28	10000

(Figure 5 precision, recall and f1-score table)

From figure 5 we notice that the recall value for class 10 is significantly higher than other classes then we look at confusion table we find out that 312 testing data is predicted as class 10. When we look at recall function Recall = TP / (TP+FN) we find that the proportion of the number of positive samples classified to the number of positive samples is high for class 10. On the other hand, f1 score for class 15 and 16 are lower than other classes indicating that it is difficult for knn models to classify these data.

5.2 Random Forests

5.2.1 Data preprocessing:

The decision tree model is implemented by first preprocessing the input data using normalisation and Singular Value Decomposition.

5.2.2 Observation and Discussion:

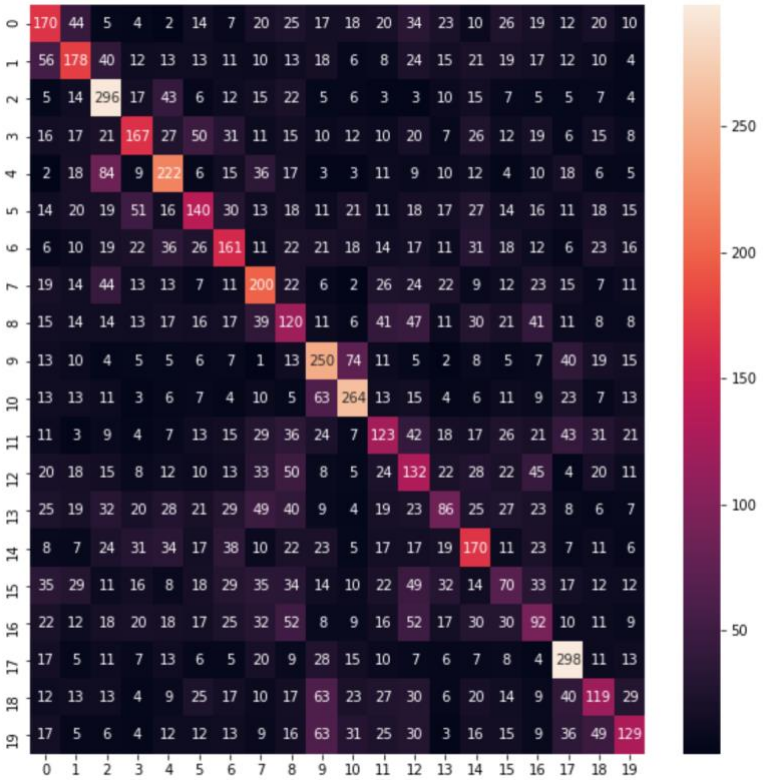
The decision tree model with normalization and PCA data preprocessing methods on the dataset has two hyperparameters: pca components and depth. Since we've discussed pca components for knn model, the only hyperparameter we need to find is the depth of the tree. The depth of the decision tree is defined as the longest path traversing the tree from top to bottom. From table 2 we notice that as the depth increases the accuracy increases significantly but when depth reaches to a point, the accuracy will not increase but decrease a bit.

We believe that as the depth increases, the data (number of instances) contained in the deepest node will decrease. This will cause this segmentation to end before reaching a certain depth. This will result that the prediction will affected by some certain noise of the data and the model may have overfitting problem.

Depth	Validation Accuracy
5	0.2293
10	0.2884
15	0.3233
20	0.3338
25	0.3366
30	0.3377

(Table 2: Accuracy on different depth)

When depth is 30, the validation accuracy reaches to 0.3377 which is the highest for all depth values. The accuracy on the test set is evaluated to 0.3387. Therefore we will use depth = 30 to construct the confusion matrix.



(Figure 6: confusion matrix on decision tree model)

k = 30 CV_Accuracy = 0.33774 Test Accuracy = 0.3387

	precision	recall	f1-score	support
0	0.34	0.34	0.34	500
1	0.38	0.36	0.37	500
2	0.43	0.59	0.49	500
3	0.39	0.33	0.36	500
4	0.41	0.44	0.43	500
5	0.33	0.28	0.30	500
6	0.33	0.32	0.33	500
7	0.34	0.40	0.37	500
8	0.21	0.24	0.22	500
9	0.38	0.50	0.43	500
10	0.49	0.53	0.51	500
11	0.27	0.25	0.26	500
12	0.22	0.26	0.24	500
13	0.25	0.17	0.20	500
14	0.33	0.34	0.33	500
15	0.19	0.14	0.16	500
16	0.21	0.18	0.20	500
17	0.48	0.60	0.53	500
18	0.29	0.24	0.26	500
19	0.37	0.26	0.30	500
accuracy			0.34	10000
macro avg	0.33	0.34	0.33	10000
weighted avg	0.33	0.34	0.33	10000

(Figure 7: precision, recall and f1-score on decision model)

The f1-score for class 2, 10 and 17 are significantly higher compare to other classes, this shows the decision tree model can classify images of those categories very well. And the accuracy for decision tree is slightly higher than knn model indicates that for image classification problems like CIFAR-100 dataset, the decision tree will have a better performance compare to knn, but still have a huge difference compare to deep learning model CNN since CNN can extract inherent property of an image.

5.3 Convolutional Neural Networks

5.3.1 Preprocessing

We use data augmentation techniques and ZCA whitening to preprocess images before being input to the neural network.

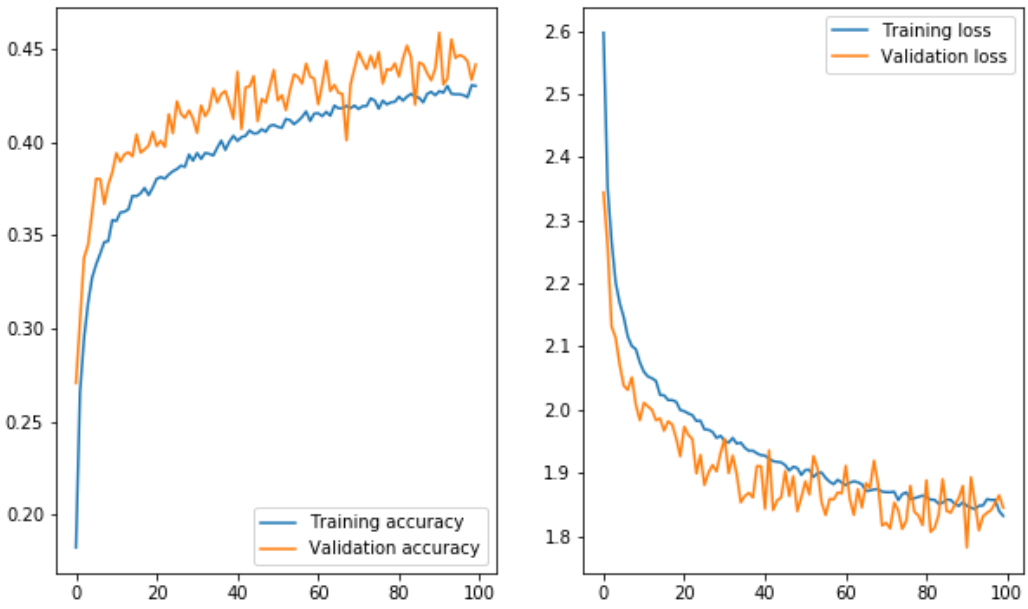
i) Data Augmentation

ii) ZCA Whitening with Data Augmentation

5.3.2 Observations

We train the model for 100 epochs, monitoring the training accuracy as well as the validation accuracy. It should be noted that this validation set is part of the training set but is static so the training set is a total of 45000 images and the validation set is a total of 5000 images.

i) Data Augmentation



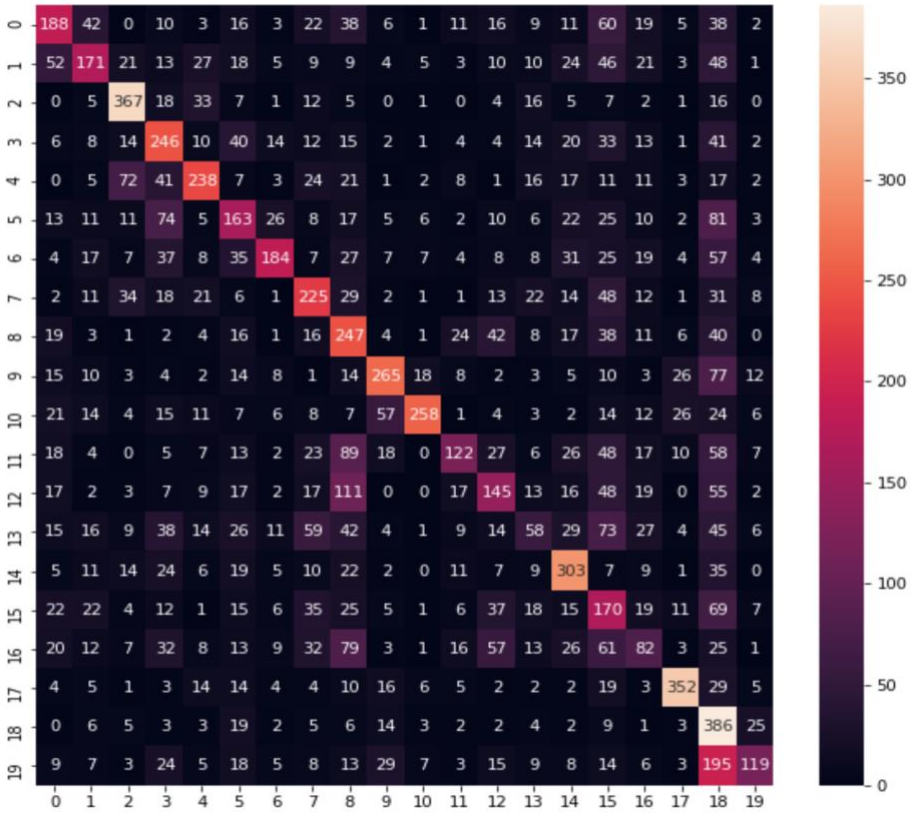
(Figure 8)

We notice that there is a steep increase in performance initially after which the performance on the validation set is erratic, suggesting the occurrence of overfitting. We also notice that the validation accuracy is higher than the training accuracy in all cases, this is because of data augmentation. Using data augmentation we have essentially made the training problem harder than the kind of problem the model will be tested on and hence we see better performance on the simpler non-augmented validation set.

We find the best performing model using the above graph and construct the following confusion matrix and precision-recall table:

	precision	recall	f1-score	support
0	0.44	0.38	0.40	500
1	0.45	0.34	0.39	500
2	0.63	0.73	0.68	500
3	0.39	0.49	0.44	500
4	0.55	0.48	0.51	500
5	0.34	0.33	0.33	500
6	0.62	0.37	0.46	500
7	0.42	0.45	0.43	500
8	0.30	0.49	0.37	500
9	0.60	0.53	0.56	500
10	0.81	0.52	0.63	500
11	0.47	0.24	0.32	500
12	0.35	0.29	0.32	500
13	0.23	0.12	0.16	500
14	0.51	0.61	0.55	500
15	0.22	0.34	0.27	500
16	0.26	0.16	0.20	500
17	0.76	0.70	0.73	500
18	0.28	0.77	0.41	500
19	0.56	0.24	0.33	500
accuracy			0.43	10000
macro avg	0.46	0.43	0.43	10000
weighted avg	0.46	0.43	0.43	10000

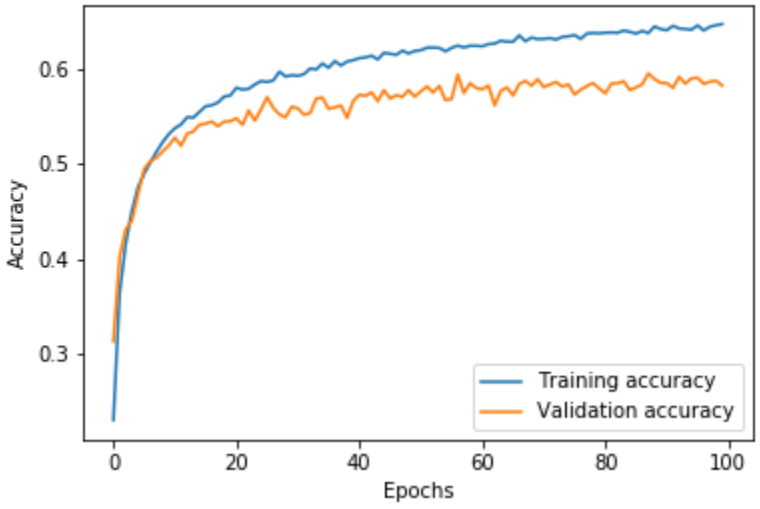
(Figure 9)



(Figure 10)

The accuracy of this model on the test set is 47%.

ii) ZCA Whitening and Data Augmentation



(Figure 11)

We notice that as the training accuracy keeps on increasing the validation accuracy is erratic and we can notice clear overfitting as we get past the 60 epoch mark. The best validation accuracy is obtained at epoch 59 and we use the model weights obtained at epoch 59 to predict the test data.

The accuracy obtained through this preprocessing method is 54% and is much higher than the simpler preprocessing methods but the trade-off is time. It takes a significantly longer time to train using the ZCA Whitening process.

6. Conclusion and Future Work

6.1 Result Conclusion

In this report, we have introduced KNN, a machine learning algorithm which finds k nearest neighbor data points of a test data and return the label with most frequence; we have introduced random forest, a machine learning algorithm based on if-or-else rule; we have introduced CNN, a deep learning model and also a feedforward neural network work. We performed data preprocessing methods for each model and recorded confusion matrix and accuracy. The accuracy for both KNN and random forest is about 30% but the execution time is relatively fast. CNN model can achieve highest accuracy with 54% in all three models but is the most time-cost training model.

6.2 Future Work

For KNN model, we didn't add any weight function inside KNN model such that, close neighbor data points can have higher weight and as the distance increases, the weight of the neighbor data point decreases. We want to add different weight functions to find the best suitable weight function for CIFAR-100 dataset.

Also there are some deep learning models we want to try. We want to try to design more CNN models such as CNN VGG model and CNN ResNet model since after doing some research, we find that VGG and ResNet models can achieve significant high accuracy for image classification. Therefore we want to apply those models to the CIFAR-100 dataset and see how good they can be, and also compare with our CNN model to make improvements.

References

1. Pierre Foret, Ariel Kleiner, Hossein Mobahi, Behnam Neyshabur. Sharpness-Aware minimization for efficiently improving generalization. Oct 3rd, 2020; available from: https://arxiv.org/pdf/2010.01412v1.pdf
2. Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, Neil Houlsby. Big Transfer (BiT): general visual representation learning. May 5th, 2020; available from: https://arxiv.org/pdf/1912.11370v3.pdf
3. Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning. July 3, 2017; available from: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
4. Pedro Savarese, Michael Maire. Learning implicitly recurrent CNNs through parameter sharing. 2019; available from: https://arxiv.org/pdf/1902.09701v2.pdf
5. Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, Quoc V. Le. Attention augmented convolutional networks. 2019; available from: https://paperswithcode.com/paper/190409925
6. Raheel Shaikh. Cross validation explained: evaluating estimator performance. Nov 26, 2018; available from: https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85
7. Avinash Navlani. KNN classification using Scikit-learn. August 3rd, 2018; available from: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

8. Prabhu. Understanding of convolutional neural network (CNN) — deep learning. Mar 4, 2018; available from: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148