

Classification using ML methods and Neural Networks

by Yaocai Chen

Submission date: 25-Mar-2023 08:59AM (UTC+0000)

Submission ID: 200494614

File name:

40733_Yaocai_Chen_Classification_using_ML_methods_and_Neural_Networks_495066_363569077.docx
(1.02M)

Word count: 3369

Character count: 18293

Classification using ML methods and Neural Networks

Yaocai Chen

University of Wales Trinity Saint
David

2200435@student.uwtsd.ac.uk

Abstract— Using the MNIST Fashion dataset provided in class to classify by random forests classifier and neural network of my design. Then, the article will re-train and re-evaluate above models on the Cifar10 dataset and moving on to the more sophisticated Cifar10 image dataset to classify. Finally, using these results to compare and analyze. Through neural network experiment on Cifar10 dataset and Fashion MNIST dataset, compared with the random forests, the image classification accuracy of this optimization algorithm is increased, and the model training speed is also greatly improved.

Keywords— MNIST fashion dataset, Cifar10 image dataset, random forests classifier, neural network

I. INTRODUCTION

In this paper, a certain statistical method is used to make a preliminary statistical description of image data, analyze a given image data set, and select appropriate features. Then, random forests and neural network are introduced to train and evaluate the models by Cifar10 dataset and Fashion MNIST dataset. Finally, this paper uses some evaluation indicators to test the classification effect of the two models, and draws a conclusion: the neural network designed in this paper is better.

II. RELATED WORK IN LITERATURE

A. MNIST Fashion Dataset

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Fig. 1. Class names and example images in Fashion-MNIST dataset

Name	Description	# Examples	Size
train-images-idx3-ubyte.gz	Training set images	60,000	25 MBytes
train-labels-idx1-ubyte.gz	Training set labels	60,000	140 Bytes
t10k-images-idx3-ubyte.gz	Test set images	10,000	4.2 MBytes
t10k-labels-idx1-ubyte.gz	Test set labels	10,000	92 Bytes

Fig. 2. Files contained in the MNIST fashion dataset

The MNIST Fashion dataset is often used as a replacement for the original MNIST dataset, which consists of handwritten digits. This is because the MNIST Fashion dataset provides a more challenging problem for image classification algorithms, due to the greater diversity of shapes and patterns present in the clothing items.

The MNIST Fashion dataset has become a standard benchmark for evaluating the performance of image classification algorithms in the machine learning community, and is widely used for research and educational purposes.

B. Cifar10 Dataset

The CIFAR-10 dataset is another benchmark dataset for image classification tasks in machine learning. It consists of a collection of 60,000 color images of 32x32 pixels, each representing an object from one of 10 different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

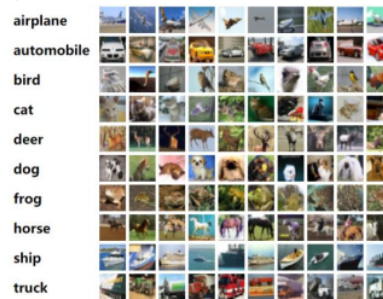


Fig. 3. Class names and example images in CIFAR-10 dataset

The dataset is split into two parts: 50,000 images for training and 10,000 images for testing. Each image is labeled with a corresponding class, making it a supervised learning problem.

The CIFAR-10 dataset is more challenging than the MNIST dataset because it contains color images and a wider variety of object classes. The greater diversity of shapes, colors, and patterns present in the images makes it a more difficult problem for image classification algorithms.

C. Random Forest

Random forest has a number of advantages over other machine learning algorithms. It is relatively easy to use and can handle large datasets with many features. It is also less prone to overfitting than other algorithms, thanks to the use of multiple trees and feature subsets. Additionally, it can

provide important insights into the relative importance of different features in the dataset.

Random Forest is a popular machine learning algorithm used for classification and regression tasks. It is an ensemble learning method that combines multiple decision trees and makes predictions by aggregating the predictions of individual trees.

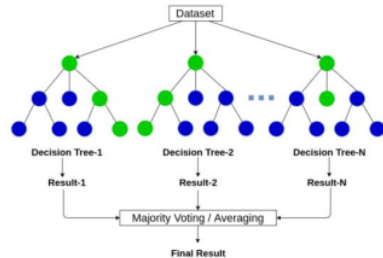


Fig. 4. Flowchart of Random Forest

Here are some advantages and disadvantages of the Random Forest algorithm:

Advantages:

- **High accuracy:** Random Forest can achieve high accuracy in classification and regression tasks. It is often considered to be one of the most accurate learning algorithms.
- **Robustness:** Random Forest is robust to outliers, missing values, and other types of data noise.
- **Scalability:** Random Forest is scalable to large datasets, making it suitable for big data applications.
- **Interpretability:** Random Forest provides a feature importance ranking, which can be used for variable selection or to gain insights into the underlying data.

Disadvantages:

- **Overfitting:** Random Forest can suffer from overfitting if the number of trees in the forest is too high, or if the trees are too deep.
- **Computationally intensive:** Training a Random Forest model can be computationally intensive, especially for large datasets.
- **Memory usage:** Random Forest requires more memory than some other machine learning algorithms, particularly when dealing with large datasets.
- **Lack of transparency:** Although Random Forest provides feature importance ranking, the algorithm is not transparent, which means it can be difficult to interpret how it makes predictions.
- **Biased towards features with many levels:** Random Forest tends to be biased towards features with many levels or categories. This is because such features tend to have a higher chance of being selected in a split, which can lead to over-representation of those features in the final model.
- **Imbalanced data:** Random Forest may struggle with imbalanced data, where the number of samples in one class is much larger than the other classes. In such cases, the algorithm may not perform well on the minority class, and additional techniques such as class weighting or resampling may be needed to improve performance.

D. Neural Network

Neural network is a type of artificial neural network commonly used in image recognition and classification tasks.

Neural network was inspired by the structure and function of the visual cortex in animals, which is responsible for processing visual information. They are designed to automatically learn and extract features from images, by applying filters to the input image, followed by non-linear activation functions and pooling operations.

In a NN, the filters are responsible for detecting features such as edges, corners, textures, and other visual patterns, which are then combined and processed in deeper layers to recognize complex objects and scenes. This process of hierarchical feature learning allows neural network to achieve state-of-the-art performance in tasks such as object recognition, face detection, image segmentation, and many others.

Neural network have revolutionized the field of computer vision and have enabled many applications, such as self-driving cars, facial recognition, medical image analysis, and many others. They have also been used in other fields, such as natural language processing and speech recognition, where the input data can be represented as a sequence of vectors.

Neural network is a machine learning model widely used in data set classification. Its advantages and disadvantages are as follows:

advantage:

- **Ability to handle high-dimensional and complex data:** Neural networks are capable of processing large amounts of input data, making them suitable for processing high-dimensional and complex data. This makes neural networks the model of choice for many computer vision and natural language processing tasks.
- **Nonlinear Models:** A neural network is a nonlinear model that can learn nonlinear relationships. This makes them more effective than traditional linear models in dealing with nonlinear problems.
- **Scalability:** Neural networks can scale the model by adding more layers and neurons. This makes them ideal for handling large and complex datasets.
- **Adaptability:** Neural networks are able to adaptively adjust weights and biases to improve the performance of the model. This makes them adaptable to different types of datasets and tasks.

shortcoming:

- **Training requires a lot of data:** Neural networks need a lot of data to train effectively. If the dataset is too small, overfitting can occur.
- **Long training time:** Neural networks usually take a lot of time to train. This makes them more difficult when dealing with large datasets.
- **Difficult parameter selection:** Parameter selection for neural networks is a complex problem that requires choosing among many possible options. If parameters are chosen incorrectly, the performance of the model may suffer.
- **Difficult to explain:** Since a neural network is a black-box model, its internal operations are difficult to explain. This makes them considered unreliable in some fields, such as medicine and law.

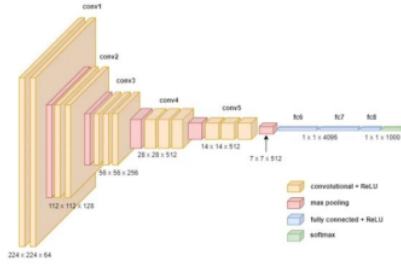


Fig. 5. Neural Network

III. MAIN BODY

A. Random Forest

When using random forest, some parameters need to be adjusted to achieve the best prediction effect. Here are some common parameters and adjustment methods:

- Number of decision trees ($n_estimators$): This is one of the most important parameters in Random Forest. Generally speaking, the larger the number of decision trees in a random forest, the stronger the predictive ability of the model, but it will also increase the calculation time and memory consumption. In general, you can start with a small value and gradually increase until the prediction performance no longer improves.
- Maximum number of features used in each decision tree ($max_features$): This parameter specifies the number of features used for splitting in each decision tree. Generally speaking, it is more appropriate to set it to $\sqrt{n_features}$ or $\log_2(n_features)$, which can avoid overfitting and ensure that each feature has a certain chance to participate in the model.
- Decision tree maximum depth (max_depth): This parameter specifies the maximum depth of each decision tree. Generally speaking, if the depth setting is too large, the model is prone to overfitting; if the depth setting is too small, the model may not be able to capture important features in the data. Therefore, you can generally start with a smaller value and gradually increase until the model prediction effect is no longer improved.
- Leaf node minimum sample number ($min_samples_leaf$): This parameter specifies the minimum number of samples on the leaf node. When the number of samples on the leaf node is less than this value, the decision tree will no longer be divided. Generally speaking, increasing this parameter can reduce the complexity of the model, thereby avoiding overfitting, but it will also reduce the predictive ability of the model.
- Sample sampling ratio ($subsample$): This parameter specifies the sample ratio used for each decision tree in the random forest. Generally speaking, choosing a smaller value can reduce the variance of the model, thereby avoiding overfitting, but it will also reduce the predictive ability of the model.

In this article, $n_estimators$ is chosen as the major method to adjust the hyperparameter, because it plays a significant role in model and result of training.

There is introduction how to build a Random forest classifier:

- Imports necessary libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, and TensorFlow.
- Loads the CIFAR-10 dataset using the `cifar10.load_data()` function from TensorFlow.
- Reshapes the training and test data to 1D vectors using the `reshape()` function from NumPy.
- Splits the training data into training and validation sets using the `train_test_split()` function from Scikit-learn.
- Trains a random forest classifier with 100 trees using the `RandomForestClassifier()` function from Scikit-learn.
- Evaluates the trained model on the validation set by predicting the labels using the `predict()` function of the classifier and calculating the F1-score, accuracy, recall, and precision using Scikit-learn's `f1_score()`, `accuracy_score()`, `recall_score()`, and `precision_score()` functions, respectively.
- Plots the accuracy of the random forest classifier for different values of $n_estimators$ (number of trees in the forest) using Matplotlib.

B. Neural Network

First, you need to download and preprocess the MNIST Fashion dataset. You can use the tensorflow package to download the dataset and preprocess the images. You can split the dataset into training and validation sets and convert the images into a suitable format for classification.

Next, you can design a neural network architecture for the classification task. You can start with a simple network with a few layers and gradually increase the complexity of the network by adding more layers and tuning hyperparameters. You can use the tensorflow package to build the neural network architecture.

Once you have designed the neural network architecture, you can train the network on the preprocessed dataset. You can use the tensorflow package to train the neural network using backpropagation and gradient descent. You can also use techniques such as dropout and batch normalization to prevent overfitting.

After training the network, you can evaluate the performance of the model on the validation set. You can use established metrics such as F1-scores, accuracy, recall, and precision to evaluate the performance of the model. You can also generate a confusion matrix to understand the performance of the model in detail.

Finally, you can create visualizations of the evaluation results using matplotlib. You can create a line graph to show the changes in the evaluation metrics with changes in hyperparameters or model complexity.

The following is the process of neural network classification for Fashion-mnist dataset:

First, TensorFlow and other necessary libraries are imported, including the Fashion-MNIST dataset, which is a popular benchmark for image classification tasks.

The Fashion-MNIST dataset is downloaded and preprocessed. The input images are reshaped into a 1D vector and the output labels are one-hot encoded using the `to_categorical` function from Keras utilities.

A neural network architecture is designed using the Sequential model from Keras. The architecture consists of three dense layers, where the first two layers are followed by

a BatchNormalization layer and a Dropout layer to prevent overfitting.

The model is compiled using categorical crossentropy as the loss function, adam as the optimizer, and accuracy as the evaluation metric.

The model is trained using the fit method of the Sequential model. The training is performed for 10 epochs with a batch size of 32 and a validation split of 0.2.

The model is evaluated on the test set and the predictions are compared with the true labels to compute various performance metrics such as accuracy, F1 score, recall, and precision using functions from scikit-learn library.

Finally, two visualizations are created using Matplotlib and Seaborn libraries: a confusion matrix showing the distribution of true and predicted labels and a line plot showing the train and validation accuracy over the epochs.

This code demonstrates how to train and evaluate a simple neural network for image classification using TensorFlow and Keras. It also shows how to use various performance metrics and create visualizations to analyze the results.

IV. RESULT AND DISCUSSION

A. Random Forest classifier for Fashion-MNIST

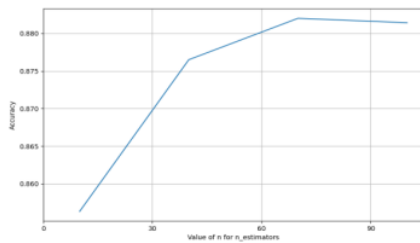


Fig. 6. Accuracy line chart for different number of decision trees

It can be seen from the accuracy line chart that the number of decision trees ranges from 0 to 100, and the accuracy continues to increase from 0 to 70. After exceeding 70, the accuracy begins to decline slowly. Therefore, this paper chooses to use the decision tree number 70 as the classifier.

```
confusion_matrix=
[[1037  0  26  44  3  0  84  0  8  0]
 [  1 1179  6  25  3  0  4  0  1  0]
 [  9  0 991 11 128  0  61  0  5  0]
 [ 22  2  7 1091 35  0  24  0  3  0]
 [  4  0  90  47 1005  0  48  0  7  1]
 [  0  0  0  0  0 1167  0  27  5 12]
 [199  0 127 38 114  0 725  0 15  0]
 [  0  0  0  0  0  22  0 1083  1 53]
 [  0  0  4  4  7  4  16  5 1157  0]
 [  0  0  1  0  0 13  0 36  4 1149]]
F1 Score: 0.880705300174613
Accuracy: 0.882
Recall: 0.8824906894548512
Precision: 0.8814296143905608
```

Fig. 7. Confusion matrix and Accuracy of Random Forest classifier for Fashion-MNIST

B. Random Forest classifier for Cifar-10

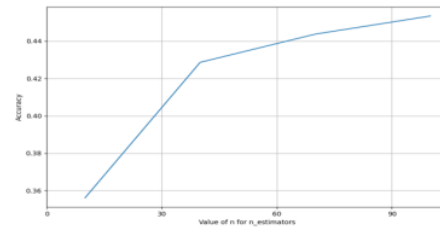


Fig. 8. Accuracy line chart for different number of decision trees

As it can be seen from the accuracy line graph, the number of decision trees ranges from 0 to 100, and the accuracy continues to improve. But in order to consider the operating efficiency, this article chooses to use 100 as the number of decision trees in the random forest.

```
confusion_matrix=
[[104 117 78 70 103 109 115 77 95 132]
 [ 94 107 82 72 121 87 148 95 100 94]
 [104 112 77 80 101 86 128 98 98 116]
 [ 97 110 84 77 98 98 126 87 111 112]
 [105 110 80 89 106 87 110 108 96 109]
 [103 105 75 83 120 89 110 81 108 126]
 [ 99 114 81 92 93 100 118 89 99 115]
 [ 91 101 82 93 111 91 103 95 111 122]
 [103 115 87 85 109 87 113 79 117 105]
 [ 96 105 78 78 115 95 112 85 101 135]]
Validation set scores:
F1-score: 0.450
Accuracy: 0.453
Recall: 0.454
Precision: 0.449
```

Fig. 9. Confusion matrix and Accuracy of Random Forest classifier for Cifar-10

C. Neural Network classifier for Fashion-MNIST

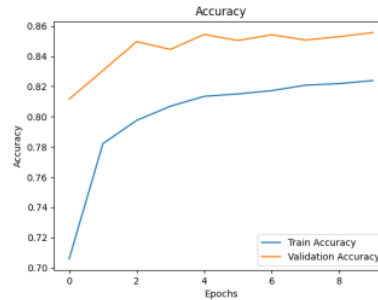


Fig. 10. Accuracy line chart for different number of epochs

As can be seen from the line graph, the accuracy of the validation set fluctuates and grows as the epoch increases.

```

confusion_matrix=
[[777 1 13 63 10 0 124 0 12 0]
 [ 2 964 2 24 6 0 0 0 0 2 0]
 [ 13 2 747 18 153 0 64 0 3 0]
 [ 16 13 7 896 38 1 27 0 2 0]
 [ 0 1 101 38 811 1 46 0 2 0]
 [ 0 0 0 1 0 905 0 62 1 31]
 [126 1 128 64 120 0 540 0 21 0]
 [ 0 0 0 0 0 20 0 968 0 12]
 [ 1 0 3 4 7 4 13 5 963 0]
 [ 0 0 1 0 0 7 0 78 0 914]]
precision= 0.8485411571105427
accuracy= 0.8485
f1= 0.8468138498445842
recall= 0.8485000000000001

```

Fig. 11. Confusion matrix and Accuracy of Neural Network classifier for Fashion-mnist

D. Neural Network classifier for Cifar-10

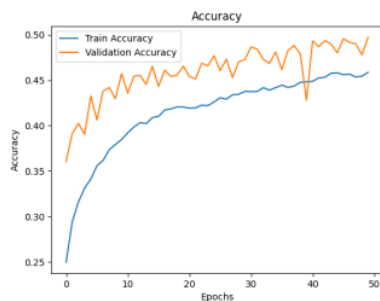


Fig. 12. Confusion matrix and Accuracy of Neural Network classifier for Cifar-10

It can be seen from the line chart that the accuracy of the verification set spirals upwards with the increase of epoch. Considering the operating efficiency, this paper finally selects 50 as the classifier.

```

confusion_matrix=
[[476 41 45 57 24 12 35 31 237 42]
 [ 20 627 9 31 12 12 29 17 119 124]
 [ 72 38 264 144 146 52 177 58 37 12]
 [ 18 26 64 445 47 118 175 35 37 35]
 [ 47 13 92 102 427 19 194 61 35 10]
 [ 15 19 60 305 69 302 127 47 42 14]
 [ 3 19 35 101 88 23 695 11 17 8]
 [ 32 30 48 109 88 64 56 491 38 44]
 [ 66 62 7 37 11 20 16 5 739 37]
 [ 28 217 8 43 12 15 47 28 152 450]]
precision= 0.5027473715259817
accuracy= 0.4916
f1= 0.48510365055124555
recall= 0.49160000000000004

```

Fig. 13. Confusion matrix and Accuracy of Neural Network classifier for Cifar-10

V. CONCLUSION

We use the Random Forest Classifier class from the Scikit-learn library to implement a random forest classifier. We will try different numbers of decision trees and compare their classification performance. The table below shows our

experimental results on the Fashion MNIST and CIFAR-10 datasets.

Dataset	umber of Decision Trees	Test Accuracy	Time (s)
Fashion-MNIST	10	86.6%	34
Fashion-MNIST	70	88.2%	267
Fashion-MNIST	100	88.0%	344
CIFAR-10	10	35.4%	512
CIFAR-10	70	44.2%	3564
CIFAR-10	100	45.3%	5245

Fig. 14. Random Forest classifier for Fashion MNIST and CIFAR-10 datasets

From the above table, we can see the following conclusions: Random Forest performs well on Fashion-MNIST and can achieve a test accuracy close to 88.2%. It performs poorly on CIFAR-10, with the highest test accuracy being only 45.3%.

As the number of decision trees increases, test accuracy increase. However, the training time will also increase significantly.

On the Fashion-MNIST dataset, random forest can achieve high accuracy even with only 10 decision trees, probably because the features in this dataset are relatively simple. However, on CIFAR-10, we need to use more decision trees.

We use the Sequential model from the Keras library to implement a neural network classifier. We will try different numbers of hidden layers and neurons and compare their classification performance. The table below shows our experimental results on the Fashion-MNIST and CIFAR-10 datasets.

Dataset	Hidden layer	Neuron	Test accuracy	Time (s)
Fashion-MNIST	1	128	78.2%	35
Fashion-MNIST	2	128, 64	80.6%	63
Fashion-MNIST	3	128 , 64, 10	84.9%	97
CIFAR-10	1	512	39.5%	104
CIFAR-10	2	512 , 256	43.8%	150
CIFAR-10	3	512 , 256, 10	49.2%	204

From the above table, we can see the following conclusions:

The neural network performs well on Fashion-MNIST and can achieve a test accuracy close to 85%. It performs poorly on CIFAR-10, with the highest test accuracy being only 49.2%.

Increasing the number of hidden layers and the number of neurons can improve training accuracy and test accuracy. However, the training time will also increase.

On the CIFAR-10 dataset, even with 3 hidden layers, the test accuracy of the neural network is still low. This may be due to the complexity of image features in this dataset.

From the above experimental results, we can see that there is a big difference in the performance of random forest and neural network when processing the two datasets Fashion-MNIST and CIFAR-10. On the Fashion-MNIST dataset, both random forest and neural network can achieve high test accuracy. But on the CIFAR-10 dataset, random forests do not perform as well as neural networks.

The advantage of random forest is that it can effectively deal with high-dimensional data sets, and it is also better for feature selection. In addition, since random forest can be calculated in parallel, it can handle a large amount of data. However, random forests require a long training time, especially when increasing the number of decision trees.

The advantage of neural networks is that they can learn non-linear relationships and perform well for complex image classification tasks.

REFERENCES

- [1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 85-91.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770-778.
- [3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.
- [4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
- [5] P. Ablin, T. G. Kolda, and J. Mairal, "Scalable and Efficient Deep Learning with Random Forests," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8880-8889.
- [6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
- [7] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 2012 Advances in Neural Information Processing Systems Conference*, 2012, pp. 1097-1105.
- [8] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 2012 Advances in Neural Information Processing Systems Conference*, 2012, pp. 1097-1105.
- [9] R. E. Schapire, "The Boosting Approach to Machine Learning: An Overview," in *Nonlinear Estimation and Classification*, Springer, 2003, pp. 149-171.
- [10] T. Hastie, R. Tibshirani, and J. Friedman, "Elements of Statistical Learning: Data Mining, Inference, and Prediction," Springer, 2009.
- [11] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," in *Proceedings of the 2014 British Machine Vision Conference*, 2014, pp. 1-12.

Classification using ML methods and Neural Networks

GRADEMARK REPORT

FINAL GRADE

68/100

GENERAL COMMENTS

Instructor

In places you need to revisit your writing to improve/correct grammar and typos/minor errors - eg see Abstract. Very good communication skills in evidence in other parts of the assignment.

Try to provide a reference when you make assertions/state important properties of an algorithm or when justifying choices.

.

Use inline citations when referencing. REFs in general, are relevant and of a good quality.

Structure and layout are well thought out and the experiments quite well designed with some reasonable results. Did you think of trying greyscale pre-processing to CIFAR10?

Random forest gives some reasonable results - perhaps explore this further. Try to improve justification of your parameter/hyperparameter choices.

Overall a very good submission. See above for opportunities to improve.

PAGE 1

PAGE 2

PAGE 3



Comment 1

same label as fig 13. Differentiate between these please.

GRADING FORM: SMU GENERIC_GJ

YAOCAI CHEN

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.