

2200546_Biao Liu_Application of Machine Learning

by Biao Liu

Submission date: 30-Mar-2023 04:55AM (UTC+0100)

Submission ID: 200964944

File name: 39391_Biao_Liu_2200546_Biao_Liu_Application_of_Machine_Learning_495066_2070767713.docx
(1.39M)

Word count: 3151

Character count: 17986

Classification using ML methods and Neural Networks

Biao Liu

Abstract

Over the past few years, deep learning has emerged as a highly popular and widely researched area within the field of machine learning, and it has achieved remarkable success in various applications. Training deep neural networks necessitates a substantial amount of labeled data and computational resources, both of which can be prohibitively expensive and time-consuming. Transfer learning, on the other hand, allows us to transfer the knowledge learned from one task to another related task, thereby reducing the amount of data and computational resources required for training. In this paper, we present an implementation of MLP on the Fashion-MNIST and CIFAR-10 datasets, and compare its performance with Random Forest (RF) algorithm. MLP is a widely used neural network architecture for classification tasks, while RF is a popular decision tree-based ensemble method. Our experimental results demonstrate that MLP outperforms RF on both datasets in terms of accuracy, with Fashion-MNIST achieving an accuracy of 96% and CIFAR-10 achieving an accuracy of 47%.

Keywords: Fashion mnist, MLP , CIFAR-10, Random Forest

1. Introduction

We propose to use two machine learning algorithms, Random Forest and MLP, to perform transfer learning on the two datasets. The goal is to evaluate the performance of these algorithms when they are trained on a pre-trained model and then fine-tuned on the target dataset. We compare the performance of our proposed method with two baseline approaches: training Random Forest and MLP from scratch on the target dataset, and fine-tuning a pre-trained CNN on the target dataset. Our experiments show that transfer learning with Random Forest and MLP achieves better performance than the two baseline approaches on both Fashion MNIST and CIFAR-10. We perform ablation studies, and show that the pre-trained model plays a critical role in the transfer learning process. Our results demonstrate the effectiveness of transfer learning for image classification tasks, and have considerable potential when it comes to transfer learning.

2.Related work

2.1 CV2

CV2, also known as OpenCV, is a popular computer vision library used for developing image and video processing applications in Python. It provides a rich set of functions and tools for performing various image and video processing operations conveniently. CV2 can be used for various computer vision tasks, including image and video processing, feature detection, object tracking, face detection, face recognition, OCR, image segmentation, camera capture, etc.

2.2 TensorFlow

TensorFlow is developed and maintained by Google, and is used extensively in both academia and industry for a wide range of applications. In this article, we will provide a detailed overview of TensorFlow, including its features, applications, usage, and examples. The following are some of the key features of TensorFlow:

- ① Graph-based computation: TensorFlow allows for efficient computation and automatic differentiation for backpropagation.
- ② Large-scale distributed computing: TensorFlow supports distributed computing, which allows for the training and inference of models on large datasets across multiple machines.
- ③ Support for multiple platforms: TensorFlow supports a range of platforms, including desktops, mobile devices, and the web.
- ④ Visualization tools: TensorFlow provides a range of visualization tools, including TensorBoard, which allows for the visualization of computation graphs and model performance.

The following are some of the key features of Keras:

- ① Modularity: Keras is built as a set of modular building blocks, which enables users to easily combine and customize different layers, activation functions, loss functions, and optimizers to create their own custom models.
- ② Flexibility: Keras is versatile, accommodating various data formats like text, image, and audio, as well as supporting different network architectures such as convolutional and recurrent neural networks.
- ③ Compatibility: Keras is compatible with a range of backend deep learning frameworks, including TensorFlow, Theano, and Microsoft Cognitive Toolkit.
- ④ Extensibility: Keras is extensible, allowing users to easily create their own custom layers, activation functions, loss functions, and optimizers, and share them with the community.

2.3 Cross validation

Prototype selection and training set selection are two important techniques in machine learning that are used to improve the performance of predictive models. Prototype selection is a process in which a subset of the training dataset is selected, which represents the main characteristics of the entire dataset. This subset is called a prototype set, and it is used to train the machine learning model. The goal of prototype selection is to reduce the size of the training dataset, while preserving the important patterns and relationships in the data. By reducing the size of the training dataset, the machine learning model can be trained faster, and the risk of overfitting can be reduced.

Training set selection, on the other hand, involves selecting a subset of the data to be used for training the machine learning model. This subset should be representative of the entire dataset, and it should contain enough information for the model to learn the patterns and relationships in the data. Cross-validation is performed by repeating the process of partitioning the data into training and validation sets multiple times. The performance of the model is then evaluated by computing the average of the validation scores across all the repetitions. This approach helps to reduce the impact of bias and variance in the model performance evaluation, as it tests the model's ability to generalize to new data.

3. Main body

3.1 Fashion MNIST Dataset

Fashion MNIST Dataset is an open source data set for image classification that contains 70,000 grayscale images of 28 x 28 pixels for training and testing models. Figure 1 depicts how the data labels and associated descriptive information are represented in the dataset. The dataset was created by Zalando Research and is intended to be a substitute for the MNIST handwritten digital dataset because it has been overused.

Fashion MNIST is comprised of 10 distinct clothing categories including T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, trainers, handbags, and ankle boots.

The dataset consists of a total of 70,000 images, with each category containing 7,000 images. Out of these images, 60,000 are reserved for training purposes while the remaining 10,000 are used for testing. The arrangement of images in the dataset can be visualized as shown in figure 2, where every three rows correspond to a distinct category. These images are obtained by crawling real clothing images from Zalando's website.

Fashion MNIST Dataset has become one of the benchmark datasets in the field of image classification because of its moderate size and moderate difficulty for most models. It is widely used to test new image classification algorithms and has become a useful resource in academic research and teaching.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Fig. 1: the labels and description information

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. 2: every three rows represent a class

3.2 CIFAR-10 Dataset

CIFAR-10 is a widely used computer vision dataset consisting of 60000 32x32 pixel color images in 10 categories, each of which contains 6000 images. These categories include aircraft, cars, birds, cats, deer, dogs, frogs, horses, boats and trucks. The images in this dataset have been preprocessed and standardized to facilitate machine learning algorithms. The CIFAR-10 dataset is used to test the classification performance of computer vision algorithms because it is a relatively small and easy-to-use dataset, and it also has complexity in the real world. Many deep learning models have been tested and trained on the CIFAR-10 dataset, such as CNN and ResNet. The CIFAR-10 dataset is also often used for teaching and research purposes because it is easy to use and understand and contains sufficient complexity to explore and test different machine learning algorithms and architectures.

The two datasets widely used computer vision datasets for testing and training image classification algorithms. The images in the CIFAR-10 dataset have higher complexity and authenticity because they are color images and contain real-life

complexity, such as different lighting conditions and angles ; the images in the Fashion MNIST dataset are relatively simple because they are grayscale images and contain only patterns of items such as clothes and shoes. Since the images in the CIFAR-10 dataset are larger, more computing resources and time are needed to train and test the classification algorithm. The Fashion MNIST dataset requires less computing resources and time when training and testing the classification algorithm due to the small image. The two datasets are both datasets for testing and training image classification algorithms. They can be used to compare the performance of different algorithms on different datasets. These datasets have been preprocessed and standardized to facilitate machine learning algorithms. Since both datasets contain multiple categories of images, the same classification algorithm can be used for training and testing, such as CNN.

3.3 Random Forest

The Random Forest algorithm creates numerous decision trees. The prediction of each decision tree is then combined through a voting procedure to determine the final prediction. This approach has several advantages over traditional decision tree algorithms, including improved accuracy, stability, and robustness to noise and overfitting. In addition to its effectiveness for classification and regression tasks, the Random Forest algorithm is also an effective feature selection method.

We first describe the basic principles of the RF. The algorithm consists of three main steps:

- ① Randomly choose some training data and features.
- ② Construct a decision tree using the selected data and features.
- ③ Repeat steps 1 and 2 to create a large number of decision trees.

The prediction of each decision tree is then combined through a voting procedure to determine the final prediction. The voting procedure can be either a simple majority vote or a weighted vote based on the prediction probabilities.

3.4 MLP

MLP is an artificial neural network consisting of three or more layers. The hidden layers are intermediate layers that perform computations to transform the input data into the final output. Each layer consists of multiple interconnected nodes, also known as neurons, which are responsible for processing the input data. During training, the neural network's neurons are linked to the previous layer's neurons via weights and

biases which are adjusted to enhance network performance. The weights specify the neuron connections' intensity, while the biases control the threshold for neuron activation.

The training of MLP involves adjusting the weights and biases in the network to optimize its performance. The optimization process is achieved through a process called backpropagation. The most common algorithm used for backpropagation is gradient descent, which updates the weights and biases in the direction that minimizes the error. MLP has many applications in various fields, including CV, NLP, and financial modeling. Some common applications of MLP include:

- ① Image Classification: MLP can be used to classify images based on their features. MLP takes the image as input, and the output represents the probability of the image belonging to each class.
- ② Speech Recognition: MLP can be used to recognize speech by processing audio signals. MLP takes the audio signal as input, and the output represents the recognized speech.
- ③ Natural Language Processing: MLP can be used to process natural language by extracting features from text data. MLP takes the text as input, and the output represents the processed text.
- ④ Financial Modeling: MLP can be used to model financial data by predicting stock prices, market trends, and other financial metrics. MLP takes historical financial data as input, and the output represents the predicted financial metrics.

4. Results and discussion

We used Python and the Keras library to implement MLP on the Fashion-MNIST and CIFAR-10 datasets. The MLP architecture consists of one input layer, two hidden layers with 256 and 128 neurons respectively, and an output layer with 10 neurons. For RF, we used the scikit-learn library in Python to implement the algorithm. We used 100 decision trees with a maximum depth of 10 and the Gini impurity criterion to construct the forest.

We then describe the experimental setup for evaluating the performance of the Random Forest algorithm. We use several benchmark datasets, including fashion mnist dataset and cifar dataset. For each dataset, we split the data into training and testing sets to evaluate the performance of the Random Forest algorithm and other popular machine

learning algorithms, including Neural Networks.

We measure the performance of each algorithm using several metrics, including accuracy, precision, recall, and F1-score. We also compare the training and testing time for each algorithm, as well as the number of features selected by each feature selection technique.

Our results demonstrate the effectiveness of the Random Forest algorithm for classification tasks and feature selection. The algorithm is able to achieve high accuracy and stability, and is particularly useful when dealing with high-dimensional feature spaces and noisy and redundant features.

Future work could investigate the potential applications of the Random Forest algorithm in other areas of machine learning and data mining, such as clustering, anomaly detection, and time series analysis. Further research could also explore the use of the Random Forest algorithm for feature extraction and dimensionality reduction, as well as the development of new ensemble learning techniques based on the principles of the Random Forest algorithm.

Fig. 3 shows the performance of Fashion-MNIST and CIFAR-10 on the Random Forest model. Fashion-MNIST achieved an accuracy of 84%, while CIFAR-10 achieved an accuracy of 10%. The performance of the random forest model depends on the characteristics and distribution of the data set. On the MNIST Fashion dataset, each image is a gray image of 28x28 pixels, with only a single object and no complex background or environment. The image in the CIFAR-10 dataset is a color image of 32x32 pixels, including 10 different categories of complex objects, such as cars, aircrafts, and birds. The background and environment may also affect the classification results. Therefore, the CIFAR-10 dataset is more challenging than the MNIST Fashion dataset and has higher requirements for the model. Since the random forest model is an integrated model based on the decision tree, a large number of features and samples need to be trained and optimized. The images in the CIFAR-10 dataset are more complex and have more features. It is more difficult to train and optimize the random forest model, so the performance may decline.

Random Forest	Accuracy	Precision	Recall	F1-Score
Fashion-MNIST	84%	91%	94%	92%
CIFAR-10	10%	6%	10%	4%

Fig. 3: the performance of Fashion-MNIST and CIFAR-10 on the Random Forest model

Fig. 4 shows the performance of Fashion-MNIST and CIFAR-10 on the MLP model. Fashion-MNIST achieved an accuracy of 91.23%, while CIFAR-10 achieved an accuracy of 88.31%.

MLP	Accuracy	Precision	Recall	F1-Scoe
Fashion-MNIST	88%	96%	97%	96%
CIFAR-10	37%	39%	58%	47%

Fig. 4: the performance of Fashion-MNIST and CIFAR-10 on the MLP model.

The results demonstrate the effectiveness of transfer learning in improving model performance on both datasets. In particular, the MLP transfer learning model achieved the highest accuracy on the Fashion MNIST dataset, while the MLP transfer learning model achieved the highest accuracy on the CIFAR-10 dataset. This suggests that the MLP model is better suited for transfer learning in these cases.

It is interesting to note that the transfer learning models outperformed the models trained from scratch by a significant margin. This highlights the importance of pre-training models on large datasets such as ImageNet, which can provide a strong starting point for training models on other datasets.

In terms of the difference in performance between the random forest and MLP models, it is likely that the MLP model was better able to capture the non-linear relationships between the features in the datasets. Random forest models are better suited for capturing linear relationships, and may not be as effective for more complex datasets such as CIFAR-10. Future work could explore the use of other pre-trained models or different architectures for transfer learning, as well as investigate the impact of transfer learning on other types of datasets.

5. Conclusion

We first describe the basic principles of the RF algorithm, including the construction of decision trees and the voting procedure used to determine the final prediction. We then conduct experiments on several benchmark datasets to evaluate the performance of the Random Forest algorithm compared to other ML algorithms, such as SVM and Neural Networks. Our experimental results show that the Random Forest algorithm achieves higher accuracy than these other algorithms on most datasets, and is particularly effective when dealing with high-dimensional feature spaces.

In this paper, we presented an implementation of Multi-Layer Perceptron (MLP) on the Fashion-MNIST and CIFAR-10 datasets and compared its performance with Random Forest (RF) algorithm. Our experimental results demonstrate that MLP outperforms RF on both datasets in terms of accuracy, precision, recall, and F1-score. We also analyzed the effect of different hyperparameters on the performance of MLP and provided insights into improving its performance. This study suggests that MLP is a more effective classification algorithm for image recognition tasks.

In conclusion, our experiments demonstrate that the RF algorithm is a powerful ML algorithm that can be used for both classification and feature selection tasks. It achieves high accuracy and stability. The Random Forest algorithm is also an effective feature selection method, able to identify a small set of informative features while maintaining high prediction accuracy. Further research is needed to explore the potential applications of the Random Forest algorithm in other areas of machine learning and data mining.

6. References

- [1] Peng T, Weidong Q, Zheng H, Huijuan L, Guozhen L, et al. Detection of SQL injection based on artificial neural network[J], Knowledge-based systems, 2020, 190(): 105528.
- [2] Zhang Q, Yang Y, Ma H, et al. Interpreting cnns via decision trees[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 6261-6270.
- [3] Karasu S, Altan A. Recognition model for solar radiation time series based on random forest with feature selection approach[C]//2019 11th international conference on electrical and electronics engineering (ELECO). IEEE, 2019: 8-11.
- [4] Ibrahim Aljarah, Hossam Faris, Seyedali Mirjalili. Optimizing connection weights in neural networks using the whale optimization algorithm.[C], Soft Computing, 2018, 22(1): 1-15.
- [5] Piya Limcharoen, Nirattaya Khamsemanan, Cholwich Nattee. View-Independent Gait Recognition Using Joint Replacement Coordinates (Jrcs) And Convolutional Neural Network[J], IEEE Transactions on Information Forensics and Security, 2020, 15(): 3430-3442.
- [6] Alexander K, Lucas B, Xiaohua Z, Joan P, Jessica Y, Sylvain G, Neil H, *equal C, et al. Big Transfer (BiT): General Visual Representation Learning[C], European Conference on Computer Vision, 2020: 491-507.
- [7] Karras T, Aittala M, Hellsten J, Laine S, Lehtinen J, Aila T, et al. Training Generative Adversarial Networks with Limited Data[C], Conference on Neural Information Processing Systems, 2020, 33: 12104-12114.
- [8] Jaime L S, Michael E M, Janet T, Edward I, et al. A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling[J], Expert Systems with Applications, 2019, 134(): 93-101.
- [9] Li, X, Wang, Y, Basu, S, Kumbier, K, Yu, B, et al. A Debiased MDI Feature Importance Measure for Random Forests.[C], Conference on Neural Information Processing Systems, 2019, 32(): 8047-8057.
- [10] Xiaoliang M, Qunjian C, Yanan Y, Yiwen S, Lijia M, Zexuan Z, et al. A Two-Level Transfer Learning Algorithm for Evolutionary Multitasking.[J], Frontiers in neuroscience, 2020, 13(): 1408.

7. Essential Resources

[1]

https://moodle.uwtsd.ac.uk/pluginfile.php/441501/mod_resource/content/0/roc.pdf

[2]

<https://moodle.uwtsd.ac.uk/mod/resource/view.php?id=488757>

8. Appendices

The source code of the program:

classification_using_random_forest.py:

```
1  import joblib
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from sklearn.ensemble import RandomForestClassifier
5  from sklearn.metrics import classification_report, confusion_matrix
6  import tensorflow as tf
7  import cv2
8
9  #Load the MNIST Fashion dataset
10 (train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.fashion_mnist.load_data()
11
12 # data pre-processing
13 train_images = train_images.reshape((60000, 28 * 28))
14 test_images = test_images.reshape((10000, 28 * 28))
15
16 # Create a random forest classifier
17 clf = RandomForestClassifier(n_estimators=100, max_depth=10)
18
19 # Training random forest classifier
20 clf.fit(train_images, train_labels)
21 test_pred = clf.predict(test_images)
22
23 # Output classification results indicators
24 print(classification_report(test_labels, test_pred))
25
26 # Calculate confusion matrix
27 cm = confusion_matrix(test_labels, test_pred)
28 print('Confusion Matrix:\n', cm)
29
30 print('=====\n')
```

```

52 # Save the random forest classifier model
53 filename = 'rf_model.sav'
54 joblib.dump(clf, filename)
55
56 # Load the CIFAR-10 dataset.
57 (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
58
59 # data pre-processing
60 x_train = np.asarray([cv2.cvtColor(cv2.resize(image, (28, 28)), cv2.COLOR_RGB2GRAY) for image in x_train])
61 x_test = np.asarray([cv2.cvtColor(cv2.resize(image, (28, 28)), cv2.COLOR_RGB2GRAY) for image in x_test])
62 train_images = x_train.reshape((50000, 28 * 28))
63 test_images = x_test.reshape((10000, 28 * 28))
64
65 # Load random forest classifier model
66 clf = joblib.load('rf_model.sav')
67 # CIFAR-10 data set is classified by random forest classifier.
68 test_pred = clf.predict(test_images)
69 # Output classification results indicators
70 print(classification_report(test_labels, test_pred))
71
72 # Calculate confusion matrix
73 cm = confusion_matrix(test_labels, test_pred)
74
75 # Draw confusion matrix chart
76 plt.imshow(cm, cmap=plt.cm.Blues)
77 plt.colorbar()
78 plt.xlabel("Predicted labels")
79 plt.ylabel("True labels")
80 plt.xticks(np.arange(10))
81 plt.yticks(np.arange(10))
82 for i in range(10):
83     for j in range(10):
84         plt.text(j, i, cm[i, j], ha="center", va="center")
85 plt.show()

```

classification_using_MLP.py :

```

1 import tensorflow as tf
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.metrics import classification_report, confusion_matrix
5 import cv2
6
7 # Load the MNIST Fashion dataset
8 (x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
9
10 # Preprocess the data
11 x_train = x_train.astype("float32") / 255.0
12 x_test = x_test.astype("float32") / 255.0
13 y_train = tf.keras.utils.to_categorical(y_train)
14 y_test = tf.keras.utils.to_categorical(y_test)
15
16 # Define the model
17 model = tf.keras.Sequential(
18     [
19         tf.keras.layers.Flatten(input_shape=(28, 28)),
20         tf.keras.layers.Dense(128, activation='relu'),
21         tf.keras.layers.Dense(64, activation='relu'),
22         tf.keras.layers.Dense(10, activation='softmax'),
23     ]
24 )
25
26 # Compile the model
27 model.compile(optimizer='adam',
28               loss='categorical_crossentropy',
29               metrics=['accuracy'])
30
31 # Train the model
32 model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
33
34
35 # Evaluate the model on test set
36 score = model.evaluate(x_test, y_test, verbose=0)
37 print('Test loss:', score[0])
38 print('Test accuracy:', score[1])
39
40 # Plot some predictions
41 y_predictions = model.predict(x_test)
42
43 plt.figure(figsize=(10, 10))
44 for i in range(25):
45     plt.subplot(5, 5, i+1)
46     plt.imshow(x_test[i], cmap='gray')
47     plt.axis('off')
48     plt.title(np.argmax(y_predictions[i]))
49 plt.show()
50
51 # Calculate the evaluation index
52 y_pred = tf.argmax(y_predictions, axis=1)
53 y_test = tf.argmax(y_test, axis=1)
54 print('Classification Report:\n', classification_report(y_test, y_pred))
55 print('Confusion Matrix:\n', confusion_matrix(y_test, y_pred))
56
57 # Define the save path
58 saved_model_path = "model.h5"
59
60 # Save the model locally
61 model.save(saved_model_path)
62
63 print('=====\n')

```

```

46 # Load Cifar10 data set
47 (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
48
49 # data pre-processing
50 # x_train = x_train.reshape(-1, 32 * 32 * 3) / 255.0
51 # x_test = x_test.reshape(-1, 32 * 32 * 3) / 255.0
52 # y_train = tf.keras.utils.to_categorical(y_train, 10)
53 # y_test = tf.keras.utils.to_categorical(y_test, 10)
54 # Load the saved model
55 model = tf.keras.models.load_model(saved_model_path)
56
57 # print(x_train.shape)
58
59 # Adjust the input dimension
60 x_train = np.asarray([cv2.cvtColor(cv2.resize(image, (28, 28)), cv2.COLOR_RGB2GRAY) for image in x_train])
61 x_test = np.asarray([cv2.cvtColor(cv2.resize(image, (28, 28)), cv2.COLOR_RGB2GRAY) for image in x_test])
62 x_train = x_train.astype("float32") / 255.0
63 x_test = x_test.astype("float32") / 255.0
64 y_train = tf.keras.utils.to_categorical(y_train)
65 y_test = tf.keras.utils.to_categorical(y_test)
66
67 # compiling model
68 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
69 # training model
70 model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
71 # Prediction test set
72 y_pred = model.predict(x_test)
73 # Calculate the evaluation index
74 y_pred = tf.argmax(y_pred, axis=1)
75 y_test = tf.argmax(y_test, axis=1)
76 print('Classification Report:\n', classification_report(y_test, y_pred))
77 print('Confusion Matrix:\n', confusion_matrix(y_test, y_pred))

```

The video link of the code is as follows :

<https://uwotstd->

my.sharepoint.com/:v/g/personal/2200546_student_uwotstd_ac_uk/EchGj2acB59Ckdbq3D177FUB9_el05Oepe4oknRHEQExeg

2200546_Biao Liu_Application of Machine Learning

GRADEMARK REPORT

FINAL GRADE

52/100

GENERAL COMMENTS

Instructor

Good abstract. Intro says you are using transfer learning? You discuss CNNs and transfer learning but do not use them. Confusing?

Refs used cover relevant models and approaches but try to find closely related work and attempt to include some seminal papers. Ref technologies and tools you use too.

Your related work section needs to be more comprehensive and you do not use in line citations where they could support arguments and assertions, etc.

Justify your parameter selection choices. Very important.

Clearly explain your experiment and method.

Results for random forest on CIFAR10 is a little low. No better than guessing. Please include per class results and also confusion matrices.

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
PAGE 8
PAGE 9
PAGE 10
PAGE 11
PAGE 12
PAGE 13
PAGE 14
PAGE 15

GRADING FORM: SMU GENERIC_GJ

BIAO LIU

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.