University of Sydney – Machine Learning and Data Mining COMP5318

Machine Learning Assignment 2

Classification of the CIFAR100 Dataset

Note: Contribution from each member:
Bernice Data exploration, CNN, SVM, Rewrite report
Smrithika: LR code with zca preprocessing method and create outline for assignment 2 report

Bernice Ziwei Yeow – 510363262;
Smrithika Bura – 490600050
11-7-2021

ContEnts

School of Information Technologies

Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: Machine Learning COMP 5318

Assignment name: Assignment 2

Tutorial time: Wednesday 5:00 PM AEST to 6:00 PM AEST **Tutor name:** Mashud Rana(tut 18)

DECLARATION

We the undersigned declare that we have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*, and except where specifically acknowledged, the work contained in this assignment/project is our own work, and has not been copied from other sources or been previously submitted for award or assessment.

We understand that understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

We realise that we may be asked to identify those portions of the work contributed by each of us and required to demonstrate our individual knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order toarrive at the final assessment mark.

Project team members

Student name	Student ID	Participated	Agree to share	Signature
1.Bernice Ziwei Yeow	510363262	Yes	Yes	[illegible]
2.Smrithika Bura	490600050	Yes	Yes	[illegible]

1 ABSTRACT

The problem that we were trying to solve was of classifying a dataset – CIFAR100, to it's target labels. The dataset consisted of 100 class labels and 50000 training images as well as 10000 testing images. Our assignment was to classify the images to the 100 class labels by using different classification methods.
The significance of this assignment and the classification is that it shows how accurately an image is being classified from the image dataset to the label dataset. It shows how well our classifiers work and how much we understood that the subject of Machine Learning and Data Mining.

The processes we have used varies across the preprocessing and the classifiers. The preprocessing methods that were used are:
1. Flipping the Image upside down
2. Image Data Generator – ZCA Whitening and Brightness
3. Reducing Dimensionality from 4 dimensions to 2 dimensions
4. Feature extraction by taking the mean of pixels values contained in the 3 colour channels
5. Conduct PCA to keep 95% of the variance in the dataset
6.Subsampling 40% of training, validation, test dataset to train a SVM model
7. Data decompression techniques like CP and Tucker's decomposition

The classifier methods that were used to classify the CIFAR100 dataset are:
1. Convolutional Neural Networks (CNN)
2. Support Vector Machines (SVM)
3. Logistic Regression (LR)
The results from these classification methods proved that CNN was the best for classifying the dataset with it giving over 44 % accuracy whereas both the SVM achieved 13% and the LR gave a 11% accuracy for the test datasets.

In conclusion, we have trained on Support Vector Machine (SVM), Logistic Regression and Convolution Network (CNN). CNN model was the best between the three classifiers that were used this result was consistent with the researcher's finding where CNN outperforms most of the machine learning model. Moreover, due to the large data size of CIFAR-100 dataset, we found that training a model for CIFAR-100 dataset is computational expensive the time taken to train a model would require an average of more than an hour at least to train a a machine learning model like SVM. Apart from that, we also found a similar trend between the classifiers, the class 55: Otter was easily being misclassified. In addition, we made a contribution by investigating the best CNN model parameters to train CIFAR-100 image dataset. The combination of ADAM optimiser with ELU activation function reduces the training time from 3 hours down to 18 minutes for 20 epochs.

2 RELATED WORK OR PREVIOUS WORK

F-SVM proposed by Wu, et al. (9) is significantly better than the other SVM versions. Support Vector Machines (SVMs) and their extensions have been some of the most successful and promising machine learning methods and are being used in different applications and fields. Though SVM has been successful, its generalization margin error is a bit problematic. The error itself is a function of the ratio of the radius and margin. When feature mapping is used though, the radius is fixed and can be ignored, and thus SVM can safely minimize the generalization error by maximizing the margin. However, for joint feature learning, the radius is needed and cannot be ignored. For this problem, there were some solutions that were proposed such as:

1. Relative margin machine (RMM) only considers the spread of the data along the direction perpendicular to the classification hyperplane. Radius-margin-based SVMs (MR-SVM), metric learning-based radius-margin SVM (R-SVM+), and radius-margin SVM for feature selection (RSVM+), are only applicable to feature weighting and selection.

2. Metric learning with SVM. Metric learning can be adopted to learn a better linear transformation matrix. However, this usually leads to unsatisfying performance.

There were more proposed approaches as well, but in this paper, the authors proposed F-SVM which is a novel radius-margin-based SVM model for joint learning of feature transformation and the SVM classifier. They derived novel lower and upper bound limits for the relaxation of the radius, hence allowing the F-SVM to simultaneously learn feature transformation and the classifier. A generalized coordinate descent (GBCD) algorithm is proposed to solve our F-SVM model, which iterates alternately by updating the feature transformation and classifier.

The experiments were conducted on Image Data Sets such as CIFAR-10, CIFAR-100, MNIST, the results shows that the use of F-SVM decreased the marginalized error and provided a higher accuracy.

As a comparison with our model's performance, our SVM model was only trained by tuning C and kernel on subsample of the CIFAR-100 dataset. The author (9) achieved higher performance and outperform us because it uses an enhanced version of SVM. One of it is RMM that are optimised for feature weighing and selection. Second one was F-SVM that automatically learns the feature transformation of the classifier. F-SVM model are not using gradient descent which is the common optimisation technique to optimise a model but they are using generalised coordinate descent which updates both feature transformation and the classifier.

In the article by Christoph (11), the authors are trying to experiment on the neural networks by integrating a random forest method into it to make it a Neural Random Forest. They note that CNNs are very good and give some of the best accuracies, but the advantages of using Random Forest are:
(1) All components can be combined in a single pipeline and existing high-performance deep learning frameworks can be used directly. This accelerates the processing and enables readily GPU processing.
(2) The mapping initializes neural networks with very few training examples enabling a warm-start.
(3) The resulting network can be further fine-tuned end-to-end jointly optimizing the features as

well as the classifier.

In their experiments, they claim that they provide a transformation of random forests into neural networks creating an efficient neural network. They do this by introducing a method for generating data from a random forest which creates any amount of input data and corresponding labels. With this data, a neural network is trained that learns to imitate the random forest. The different experiments on various datasets show that the accuracy of the imitating neural network is equal to the original accuracy or even outperforms the original random forest. This model was implemented and evaluated on 5 different Datasets: MNIST, CIFAR-10, CIFAR-100, GTSRB and Caltech 101.

As a comparison with our work, the author managed to achieve outstanding performance because they used imitation techniques from random forest for their CNN network. The trained features imitated the actual learned random forest model, this allows them to use only less amount of input image and obtain a very high accuracy. In terms of our best model - CNN, we did not managed to achieve a high result like the author, we did perform feature extraction by using feature on the 10th layer of the CNN network (right after maxpooling of the 2 convolutional layers) as the input our CNN network to train, maxpooling layer narrows down the features from convolution layer this leads to potential loss of important information for the CNN network to train.

This next article talks about the Extreme Learning Machine (ELM) and how it is a powerful and favored classifier and how it is used across many applications due to its fast speed and good generalization capability.

Extreme Learning Machine (ELM) is a variant of single-layer feedforward neural networks (SLFNs) with randomly assigned and fixed neurons between the input and hidden layer. When compared to convolutional deep neural networks, ELM is much faster in training as it doesn’t need any gradient-based iterative tuning i.e. Once the input weights and hidden node biases are randomly generated, the optimal solution for the output mapping matrix can be directly calculated.

However, the architecture of an ELM is not suitable for visual dataset classifications, hence many different ideas and extensions were proposed. However, Neural Networks are gaining more popularity and are becoming the foundation of many artificial intelligent techniques used nowadays, covering various fields. Hence, the authors focused on creating an ELM classifier model with focus not only on the depth of a neural network, but also its width.

In terms of Testing and Experimentation, the authors have used 5 image datasets: CIFAR-10, CIFAR-100, STL10, Fashion MNIST, 102 Category Flower. The performance result shows that CIFAR-100 dataset the accuracy gain is over 10% in the first 60 epochs and the testing accuracy has been boosted by 2.13%.

Here in our implementation of classifiers, we will be using the idea of Neural Networks as part of our classification as that has proven to give better accuracy outputs according to the Neural Random Forests and the ELM articles. The use of a neural network has been implemented in our CNN classifier. We noticed that when compared to our work, the author managed to achieve higher result because they have different tuning process such as tuning on depth and width of network. Whereas, for our best classifier- CNN, we only managed to tune the optimiser, activation function and the number of epoch, all of these can be extended in the future where we can focus on tuning the depth and width of the network, such as the number of neurons and the number of convolutional layers.

The article by Touvron (1), the vision transformer (ViT) introduced by Dosovitskiy et al. (8) is an architecture derived from Natural Language Processing Vaswani et al (26), but applied to image classification task with raw image patches (image pixels) as input. ViT model demonstrated excellent results with transformers trained with a large private labelled image dataset containing 300 millions images from Cifar-100 dataset, Imagenet and more. The paper concluded that vision transformers "do not generalize well when trained on insufficient amounts of data". The training of these models involved extensive computing resources. The main component for ViT are:

1. Mult-head self attention layers which are the inner product of query and key vectors that are then scaled and normalized with activation function like softmax function to obtain its weight.

2. Transformer block for images the researcher added a feed-forward network on top of multhead-self attention to get a full transformer block.

3. To classify images, the researcher build on top of the ViT model and add a simple architecture that process input images by treating those images as a sequence of input token.

4. Class token which as trainable vectors are combined with patched tokens to input in to subsequent transformer layers. These tokens are normally taken from NLP (8), and are different than the usual pooling layer for computer vision.

5. Fixing the existing positional encoding across resolutions. The author use a images of lower resolution and fine tune on the network to obtain a larger reolsution. This method has speedsup the full training and increases the accuracy .

The result obtained by the researcher achieved accuracy of 99.1 for DeiT-B% models and 98.9% from CIFAR-100 and was remained as the model that has the highest accuracy for CIFAR-100 dataset. Nevertheless, according to Dosovitskiy et al. (8), a pre-training phase on a large volume of curated data is required for the learned transformer to be effective.

As compared to our best model - CNN, the author outperform our model by in temrs of two aspect 1. computational time/complexity: Transformer allows parallelization which gives the ability ofor much more data to be processed in the same amount of time with transformer models. 2. Better representation of feature. By adopting attention it is able to depicts and identify the most important information among images, by computing the similarity score between input data (query and key vectors). This allows better representative of input data to train.

In Mangalam et al. (2) presented fully trained shallow machine learning models. The author first train easy learnable (shallow) example in early iteration or epochs and train the harder examples in the later epochs. This method of partitioning data achieves an improving in the overall training process as compared the method without separating into easily and hard samples. The model trained by the author are SVM, Random Forest, CNN, DenseNet, Resnet on MNIST CIFAR10 CIFAR100 datasets. The result is presented as follow:

As mentioned in the abstract and can be seen in the continuation of the report, the CNN is the most widely used model to classify the images of CIFAR100 and have been known to give the best

Table 1. Author's model Results

Models	MNIST	CIFAR10	CIFAR100
SVM	97.92 %	40.08 %	14.42 %
Random Forest	96.14 %	35.86 %	14.26%
Deep Network	98.8 % (CNN)	95.04 % (DenseNet121)	77.78 % (ResNet 101)

accuracy results. The CIFAR100 dataset is widely selected to test the various CNNs being created and modelled due to its plethora of data images and how highly functional it is to train the models. Some authors who have used the CNN to model the CIFAR100 dataset are compiled as follows:

1. The "Network in Network" model created by (19), replaced the original convolutional layer with multilayer perceptrons and yeilded a score of 65%.

2. algorithm from (20) exhibits the best known result as of now - 80.75% which was a great increase in comparison to others.
3. The current Resnet 101 model developed by Mangalam (2) achieved the highest accuracy (77.78 %) when compared to SVM and Random Forest. Empirically, the author observe that during training process, Deep Network like Dense Net, CNN learns the fastest on shallow classifiable easy examples first and then learns the hard examples in the later epochs.

The SVM model is also another widely recognized and used classifier for image classification. This model has also not been implemented as much as the CNN and other neural networks, but have been considered and used many a times for image classification in other datasets as well. Some of the widely known implementations of SVM are:
1. Lin et al. who trained the SVM over a dataset known as ImageNet 1000 - which is a difficult dataset to use. After their training, they achieved a state of the art performance of around 53%.

2. Bosch, Zimmerman and Munoz had implemented the SVM as a multi kernal SVM (M-SVM) and trained it over the Caltech101 dataset and achieved impressive accuracy of 81.3%.

3. The above SVM by Managalam (2) had achieved a 14.42% in accuracy.

In the article Dahiya (4), The authors build SVM, Multinomial Logistic Regression, Multi Layer Perceptron , Random Forests, Naive Bayes, K Nearest Neighbors , convolutional neural network, ADA boost. on MNIST and CIFAR-10 dataset. Among all, Convolutional Neural Network outperforms all the models. The highest result achieved was by using CNN that gives 80.50% validation accuracy and 65.54% test set accuracy. The author emphasize how well CNN outperforms other machine learning model in this paper, however the author still mentioned one drawback from using CNN model - CNN will scale up to thousands or more parameters make it highly computational extensive. The author begin with sub-sampling method by selecting 40000 images for training data and 10000 images for validation and test data sets.

Dahiya (4) also concluded that feature extraction and feature selection are important prepossessing techniques for image classification. In this article, the author changes RGB image to gray scale and then perform feature extraction. This is because the author found that changing image to gray scale reduced the data by 67% this accelerates the training process , this is crucial as deep neural network like CNN would require a long time to train. Although the author had done this, they have listed out that a drawback of gray scaling an image would mean that it might loose important features seen by color.

In addition, Data Augmentation was used for solving overskewed data or being biases towards a particular class and it also compensates for the variation in the data as well as helping feature extraction to obtain larger and wider information about images. The researcher proposed that feature extraction should be done in a customised manner, by understanding the data before conducting feature extraction it and design a good feature extractor to extract features from images.

In this assignment, We plan to experiment using similar sub-sampling method, data augmentation techniques, and feature extraction using CNN network by Dahiya (4). Also we will be taking into consideration her experiments and executions of the Logistic Regression,. The outcome are discussed in section of experiments and discussions.

The Logisitc Regression classifier is a classifier that is not as widely used for the CIFAR100 dataset in comparison to the CNN, SVM or even Random Forest models. Though the LR model is advantageous over some of the other classifiers due to the fact that it supports multiclass classification and has interpretability of features which is used to interpret and classify the features of an image.

Balaji et al. (21) had developed three different models: the classic CNN, CNN with gradient boosted trees (GBT) and CNN with LR. They executed and trained all three models on the CIFAR100 dataset and their CNN plus LR model earned them an accuracy of around 47%. Dong et al. (22) had used an LR model for single label multi class image classification and have achieved great results on the CIFAR100 dataset while using a Resnet-32 created by He et al. (7) which was around 64%.

In our implementation of the three classifiers that we chose: CNN, SVM and LR, we were able to see that the above mentioned researchers outperformed us by a huge difference.

The CNN was the one with the most amount of difference even though it was our most well performed classifier. The output of the researchers' had almost a whopping 40% difference than ours.

The SVM gave a score just 1% lower than the Mangalam (2) SVM implementation and this was also becuase the implementation of the SVM is a bit different and also because the training might have been different than what was done by us. This might be because they were more prepared and had more time to explore the data and preprocess it accordingly. For example, the proposed preproecssing method by (2) allows the model to identify easy trainable and hard trainable sample, as a result this allows the model to differentiate between samples of images and pick the one that is hard to train and train it with more iteration. When compared to Dahiya (4), the author achieved a slightly higher performance compared to us which is about 10% mroe in accuracy. The additional preprocessing method used by the author to achieve better performance in classifying images are

larger subsampling percentage, converting images to grayscale that allows them to obtained a larger sample to train.

As for the LR model, we have decided to use a pure LR model without any neural networks, just the One vs Rest classifier used for multiclass classification, hence the accuracy was lower than the previous works.

3 INTRODUCTION

The problem that we are trying to solve was the classification analysis for the data set CIFAR100 and to label the images from the target data set to the image data set as accurately as possible. The CIFAR100 dataset consists of 100 classes and 60000 images of 32x32 size. The training dataset given contains images in random order and training samples might contain more images from one classes compared to other classes (16). Moreover, all classes are mutually exclusive, for example, there will not be overlap between automobile and trucks. Our goal is split this dataset into three sets: train, test and validation and then classify the labels of these sets to their respective images as accurately as possible using different classification methods.

The problem is important as it shows how well the students have understood the assignment and what must be done. It also shows how well the students know their classifiers and how they implemented the classifiers well and how they understood the data and their exploration of the data.

4 METHODS

Methods:
The pre-processing techniques that were used:

1. Flipping the Image upside down: Flipping an image means rotating the image on its horizontal axis. All the rows and columns of the pixels of the image are rotated on the vertical axis reversing them. This is a form of Image Augmentation and is a method of image processing. This is done to help increase the accuracy of the machine learning method being implemented onto the image. This is also the most popular method of data augmentation due to the simplicity of the code and how much value it can add to the model.

2. Parameters inside ImageDataGenerator: The ImageDataGenerator is a method in the keras library which is used for the preprocessing of image datasets. It takes up many different arguments such as rotation, height shift, zca whitening, brightness, zoom, etc. It is used to preprocess images in many different ways. Here, we decided to make use of some of the parameters. The ones that we used for preprocessing are:
a. ZCA Whitening: This is way of transforming an image in a way that is defined by a covariance matrix. It is called “whitening” in relation to the transformation of white noise. This is a Boolean argument and it was made True.
b. Brightness Range: This is another parameter which is used for changing the brightness of the image. It is a tuple argument and is given as a range.
c. Zoom Range: This attribute is for random zoom of the image and it is represented by a single value or a tuple for lower and upper ranges for the zoom. We initialized it to the maximum which was ‘1.0’.

3. Reduction of dimensionality from 4d to 2d: The CIFAR100 dataset when loaded and initially preprocessed is in 4d dimensionality. Classifiers like SVM and Logistic Regression cannot accept the 4d dimensionality and only accepts 2d. Hence, the shape and size of the datasets – xtrian, xtestand xvalid, needed to be reshaped and reduced into 2d arrays.

4. Splitting training dataset into Train, Validation datasets and keep Test datasets for evaluation: We splitted the training data into 30% for validation and 70% for training and did not split

test dataset we remained as it is. We found that the training dataset given contains images in random order and training samples might contain more images from one classes compared to other classes (Krizhevsky, 2021). Therefore we use stratified method of sampling to split training data into training and validation data sets. Training dataset will be used to train machine learning models, validation dataset will be used for hyperparameter tuning and test dataset will be used to evaluate performances of the machine learning models that we built.This gave us a total of 70000 images for training, 30000 for validation and 20000 for testing.

The classifier methods that have been used were chosen due to the intensive previous work literature review that we have done. During the literature review, we have come across the idea that CNNs are the most widely used and popular method while classifying the CIFAR dataset collection – whether it was CIFAR10 or CIFAR100. In the reports and articles that we have read, we have also seen that the CNN gave the best results. Hence, we decided that the CNN was definitely going to be used as one of the classifiers.

The next two classifiers that we decided on were based on the other classifiers that were mostly mentioned in the articles of the literature review. The SVMs are classification method that is widely used since it is a good method for classification. The SVM builds a hyperplane and divides the dataset into different classes to classify the dataset. This was the next most widely used the classifier that was written about in the articles, hence it was decided to implement.
The last, but not least, third classifier was decided by trying out different classification methods – KNN, Naïve Bayes and Logistic Regression. The Logistic Regression was the best performing out of the 4 and hence was decided for complete implementation. (The classification reports of KNN and Naive Bayes are given in the appendix).

Therefore, the three classifiers that were used for implementation are:

1. Convolutional Neural Networks: Neural Networks are a form of machine learning algorithms based on the biological neural network and how neurons are connected to each other to pass on information. A convolutional neural network is a special type of architecture under the umbrella of neural networks which uses aspects of the visual vortex, it is usually applied to images for image classification problem. It "learns spatial hierarchies of features adaptively from low to high level patterns" (17). The typically layers of CNN include:

a. The convolutional layer acts as a feature extractor that have a set of filters that help to derive feature maps from input data like images, the output of this layers typically pass into a non-linear activation function.

b. The pooling layer is the third type of layer which works on the width and height of the image to perform a downsampling operation on them that reduces dimensionality of feature maps.

c. Fully connected layers performs classification based on the output of pooling layer. The task of this layer is to assign a class label to each image.

We implemented a 2 convolutional layers with max pooling layer at the end of each convolutional layer, followed by 1 layer of dense layer. ELU was choosen as the activation function used in our CNN model. We decided on this architecture because it gave the highest accuracy result, less computationally intensive, and decrease training time.

2. Support Vector Machines (SVMs): These are good classification methods used for the classification of different datasets which also include image classification. An SVM constructs a hyperplane

in a multidimensional space to separate different classes in the dataset.

Its goal is to generate an optimal hyperplane which minimizes the error. The foundational idea of the SVM is to find the maximum marginal hyperplane that best divides the datasets into the different classes.

During our implementation of SVM, we have also used the One vs Rest classifier. This is a strategy that fits one classifier per class. This is the most commonly used strategy for multiclass classification and is usually a default. We also use Radial basis function kernel to maps image data into high dimensional space in order to find perfect linear separation in an infinite high dimensional feature space (kernel trick).

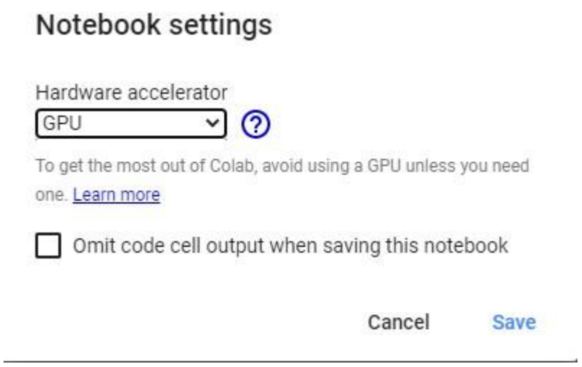
3. Logistic Regression (LR): The Logistic Regression classifier is a very common in machine learning and it is a predictive analysis algorithm based on the concept of probability. The logistic regression uses a more complex cost function which is known as the Sigmoid Function and this sigmoid function is limited to 0 and 1. This function is used to map the predictions to their probabilities.

The LR classifier works well with image classification as well since it works with multiclass classification very well. The benefit of logistic regression for image classification is its interpretability — we can use the coefficient values to understand what features (i.e. which pixels) are important in determining what class a sample belongs to.

In our implementation of the LR classifier, we had used the One vs Rest classifier for multiclass classification.

5 EXPERIMENTATION AND DISCUSSIONS

5.1 Runtime


Notebook settings

Hardware accelerator: GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

☐ Omit code cell output when saving this notebook

Fig. 1. Mapping input data and its label on a scatterplot

We have set runtime to be execute on GPU level, so all of our code are executed on GPU level.

5.2 Hardware specification

The models were run on Jupyter Notebook software. GPU: 1 x Latitude 7420 - 32GB RAM CPU: 1 x 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz 1.80 GHz RAM: 32 GB DISK: 64 GB

5.3 software specification

Matplotlib Keras tensorflow sklearn statsmodels numpy statistics

5.4 Data Exploration

We conducted Box-Tidwell test to test on linearity assumption for our image data, we observed that all 3072 features formulated have a p-value below 0.05, hence linearity assumption is violated. Detailed code and execution can be found in our data exploration ipnyb notebook file.

In addition, to get a clear understanding we also visualise image data on hyperplane

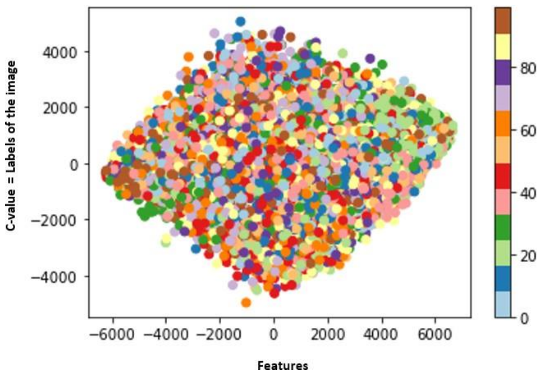


Fig. 2. Mapping input data and its label on a scatterplot

We used python sklearn scatterplot function to plot our image data against its label. Based on the plot obtained, it clearly shows that the image data is non linearly seperable since all are cluttered together. Hence we will be using non-linear method to classify images in CIFAR-100. Logistic regression was an exception because when we conduct analysis it somehow performs better than other non linear model like K-Nearest Neighbours and Naive Bayes. The result of those model can be found under classificaiton report in appendix.

5.5 Preprocessing method

In this section we present the results that we obtained by executing the code with various different model parameters, preprocessing methods as well different subsets of images and epoch numbers.

Table 2. Results with different pre-processing method tested using CNN model

Type of preprocessing	Validation Accuracy(%)	F1-Score (%)
Divide 255, Flip,add to original data and no feature extraction	40 %	39 %
Divide 255, Flip,add to original data with feature extraction	42 %	41%
Normalise divide 255 only	41 %	40 %
Import data with no pre-processing	35 %	35 %
Feature extraction - mean value of all colour pixels	34 %	33 %
Divide 255 and conduct PCA	23 %	21 %

We picked normalisation that divide image pixels by 255, flip the images upside down and add back to original data without feature extraction to be the base preprocessing method, which is the pre-processing method that all classifiers used for its preprocessing stage. We selected the top 2 best performing preprocessing method and did not include feature extraction because we noticed some classifiers will not work well with features extracted form CNN network.

This preprocessing method was used in the base classifier-SVM for comparison. On a contrary, CNN and Logistic regression have added additional methods on top of this base pre-processing method. CNN used features from CNN, Logistic regression uses data with additional ZCA whitening and ZCA epsilon because it ran the fastest so we are able to conduct more experimentation with less computational complexity (O(nd)) time complexity for training a logistic regression according to big O theory).

5.6 CNN CLASSIFIER

Table 3. CNN - Analysis of time for each model's paramter

Optimisation techniques	Training time	Prediction time
SGD optimiser with RELU activation function	3 hr 26 mins	2 mins
ADAM optimiser with RELU activation function	2 hr 33 mins	3 mins
ADAM optimiser with ELU activation function	18 mins	55 milliseconds

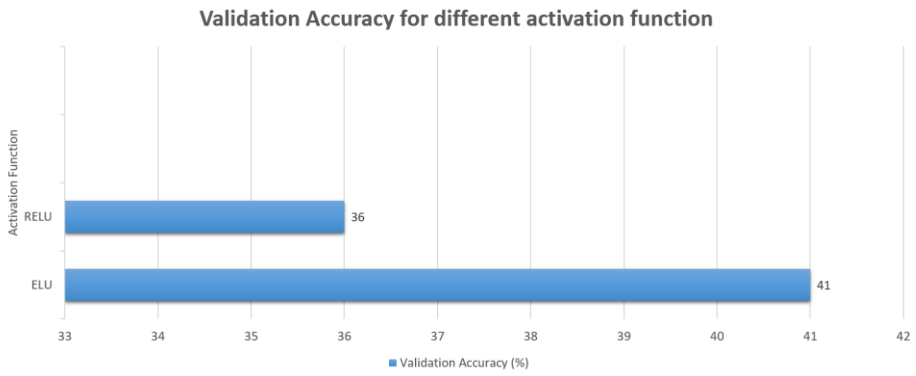


Fig. 3. Results for different activation function

Besides normalising image pixels by dividing 255 to express those images data pixels in 0-1 range and conduct PCA to keep 95% summative variance. We are also interested to try on data augmentation method like flipping images upside down, this gave an increase of 1% in validation accuracy and 1% in F1-Score. We conducted a feature extraction technique which is to average the pixels' number for each colour channel. For instance we take the mean value of the 3 colour channels for each image. However, this only allows us to obtain 34% validation accuracy and 33% F1-Score.

Then we move on to try extracting features from the 10th layer of our CNN network, we picked the 10th layer because that layer located directly after maxpooling of the result of the 2 convolution layers. We choose this method because when we tried to extract the 1st and 2nd convolution layers it gave us memory error, our current PC was unable to handle the huge load of features extracted from the convolutional layers, therefore we picked the layers after max pooling which reduces the dimension of convolutional layers.

In reference to Table 2, the choosen pre-processing method for CNN is the combination of normalisation (by dividing 255),flipping image upside down add to original dataset and model a CNN network to extract the features as the input for the best CNN network. The best result obtained from the chosen pre-processing method for CNN is 42% for validation accuracy and 41% for F1-Score. This indicated that by using extracted features from a CNN layer it increased the result up to 42% for validation accuracy, which equates to 2 percent lift in validation accuracy compared to without using feature extraction, and 41% for F1-Score, up to 2 % increase in F1-score compared to without using feature extraction.

Reason for the increase is due to the relevance of the features extracted from CNN model, the layers was trained for 10 epochs on a CNN network that has 2 convolutional layers + 1 dense layers, ELU as activation function and ADAM as optimiser. This parameter was decided based on hyperparamter tuning, detailed results will be explained subsequently.

Based on Fig. 3, we tried different optimisation techniques to determine which can give us the fastest training time and prediction and we found out that the combination of ADAM optimiser and ELU activation function gave us the best performance, that approximately reduces a total of 3 hours compared to the worst performing one SGD optimiser and RELU activation function. This is because ELU activation function fixes some of the problems associated with RELU activation function, it keeps some positive numeric values. For example, if the input value is less than 0, we get a value slightly below zero, it will not be exactly zero. Adam optimiser on the other hand has been proven to achieved least computational cost compared to other optimisers, this result was consistent with the findings from Ruder (23), where the research found that ADAM optimiser is computationally less costly when tested with other optimisers for MNIST image dataset.

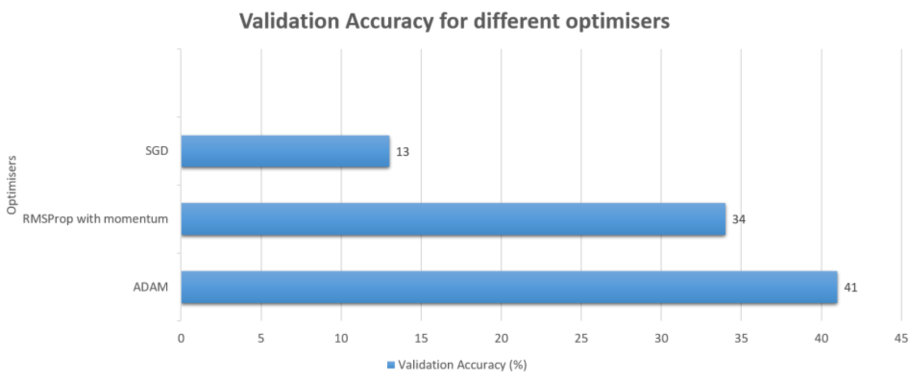


Fig. 4. Results for different optimisers

In reference to Fig 4., the idea of applying momemtum on learning rate via RMSprop was inspired by Smith (3). Instead of having a fixed learning rate, it learning rate for each epoch is updated using momentum (m), when the gradient changes its direction, momemtum (m) will smooth out the variations. However, based on the result in Table 2, Adam optimiser achieve better result than using SGD or RMSprop with momemtum. Therefore, Adam optimiser was choosen as the optimiser for our CNN model.

Table 4. CNN - Applying different tensor decomposition techniques

Type of decompression techniques	Validation Accuracy(%)	test set accuracy (%)
subsample 10% + CP decomposition	16 %	15 %
subsample 10% + Tucker decomposition	15 %	15 %
subsample 10% + no compression technique applied	16 %	17 %

Based on Table 4, we also tried on different tensor decomposition methods to reduce the number of variables or terms used to represent multidimensional data - image pixels in the case of this assignment, as well as to identify the underlying relationship and structure among those variables. However, we noticed a drop in the result, from 41% accuracy decreased until approximately 16% in accuracy. This is because the decomposition methods might have removed important features for image classification, this leads to a drop in the performance for image classification.

In reference to table 5, we tried increasing the number of epoch up to 20 epochs and it gave an increase of 4 % in terms of accuracy, precision, recall and F1-Score also increased approximately up to 4 %. In addition, we train 20 epochs we noticed it only took 18 minutes so we decided to stick to 20 epochs due to the increase in result and the time taken to train on 20 epochs was not that long.

Table 5. CNN - Testing on different epochs based on the best model parameter

Number of epochs	Accuracy(%)	Precision(%)	Recall (%)	F1-Score (%)
ELU 10 epoch	40 %	41 %	40 %	39 %
ELU 20 epoch	44 %	45 %	44 %	43 %

Table 6. Preprocessing for SVM

Type of pre-processing method	Validation set accuracy (%)
Subset 20% train,valid and test dataset	7.68 %
Subset 40% train,valid and test dataset	8.46 %
Subset 40% train,valid and test dataset without HOG	13 %
Subset 40% train,valid and test dataset with HOG	12 %

5.7 BASE CLASSIFIER: SVM CLASSIFIER

Time taken to train SVM model with the best model parameter using different subsampling amount (%).

Table 7. Speed for different subsampling method applied on SVM

Subsample (%)	Time taken to train SVM
20%	43 mins
40 %	1 hr 33 mins

As a general rule of thumb, it is best to have big sample of data to train. We realised that the accuracy was too low when we only sample 20% of training,validation and test datasets, which is approximately 7% for both validation and test set accuracy. However, when we decided to sample 40% of training,validation and test datasets the accuracy went over 8%, as shown in Table 6, the validation accuracy was at 8.45%. Besides that, in order to gain a more representative sample, it will take additional 50 minutes to train with 40% sample compared to 20% sample, this is the tradeoff we willing to undergo for the increased accuracy rate. We also found out that Histogram of Oriented Gradients (HOG) did not work well with our SVM classifier.

The results obtained when we performed HOG on three datasets - train,valid and test was 12% for both validation and test dataset accuracy, 11% for precision, 12% for recall and 11% for F1-score. On a contrary, the accruacy did not deviate that much without HOG, it only managed to increase approximately 1%, however, the precision score went about 3% and the scores for other metrics remained almost the same. This shows that without HOG, SVM is able to classify true positive samples better, this might be potentially because HOG is not robust enough to identify and extract the right Histogram descriptors (features) from the 40% sample of traning, validation and test datasets, this finding was similar to Fleyah (24) where the author found out that HOG performs poorly to extract the right HOG descriptors from images about sign boards and traffic signs.

In reference to Table 8, we noticed that when we increase the C parameter of SVM model, the validation accuracy increased up to 5% because the C parameter helps to prevent misclassifying data by narrowing the margin in hyperplane. Besides that, RBF kernel increased the validation

Table 8. Parameter tuning for SVM

Model Parameters	Validation Accuracy(%)
C = 1.0 , Kernel = ’rbf’	13 %
C = 0.1 , Kernel = ’rbf”	8.46 %
C = 1.0 , Kernel = ’poly”	6 %

accuracy up to 7%, this is because for image classification, RBF is highly recommended by many researchers because it usually gave better results compared to other kernels (25), this because it is able to plot curvature around data points in hyperplane, the decision boundaries can be defined by a topology condition, such as value above 0.5, which is the typical value for rbf kernel.

5.8 LOGISTIC REGRESSION

Table 9. Pre-processing for Logistic regression

Pre-processing techniques	CV-Accuracy(%)
Base preprocessing+ZCA epsilon,ZCA whitening+increase brightness	18.43 %
Base preprocessing method	13.51 %

In the aforementioned paragraphs, we mentioned that because logistic regression is less computational expensive, in terms of training is only took 15 minutes to train the entire CIFAR-100 dataset. Therefore we managed to experimented other pre-processing methods like ZCA epsilon that extract colour spectrum for each images and ZCA brightening that brightens the images. This gave us an increase of approximately 5 % in terms of cross-validation accuracy (performing on training and validation set excluding data from test dataset). By using this preprocessing methods the classifier able to predict more accurately because the corner of the image was blurred out making the classifier to focus on the center point of image where most important features lies, and brightens the image enable the classifier to get a clearer view of the image.

The following are the best model parameter choosen for logistic regression based on grid search cross validation k=10.

Table 10. Pre-processing for Logistic regression

Hyperparameters	C	Penalty
Values	0.01	l2

5.9 Summary of methods and experimentation

base preprocessing techniques include:
applying normalisation 255 + flip images + add on to original data , convert y target variable to onehot encoding

Final model parameters:

1. Best model: parameter chosen for CNN - elu - 2 layer cnn + 1 layer max pooling - adam activation function - base preprocessing plus feature extraction - 20 epochs

2. Selected base model for comparison: parameters choosen for SVM - rbf,c = 1.0, subset 40 % of training, validation and testing data

3. logistic regression - base preprocessing add ZCA epsilon, ZCA whitening, and increase brightness, best model parameter choosen 'C': 0.01, 'penalty': 'l2'

5.10 Comparison between classifiers

The final three model performance and result are shown in Table 12 and Table 13.

Table 11. Summary of results

Classifier	CV-Accuracy (train+valid) (%)	Test set Accuracy(%)	Precision (%)	Recall (%)	F1-Score (%)
CNN	44 %	44 %	45 %	44 %	43 %
SVM	14 %	13 %	14 %	13 %	10 %
LR	18 %	11%	10 %	11 %	11 %

Table 12. Time taken to train each classifier and conduct predictions

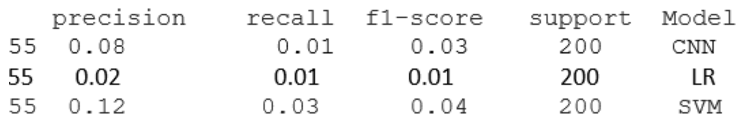
Classifier	Time taken to train	Time taken for prediction	Data size
CNN	18 mins	55 milliseconds	Entire dataset(train,valid and test)
SVM	1 hr 33 mins	20 mins	40% of entire dataset(train,valid and test)
LR	15 mins	3s	Entire dataset(train,valid and test)

In reference to Table 11, it is clearly seen that CNN outperforms the other classifiers - base classifier(SVM) and Logistic Regression in terms of test accuracy, precision , recall and F1-Score.

This findings was similar to the trend we found in our previous work research, deep neural network like CNN usually outperforms most machine learning models. Since it is able to train more extensively by having more epochs and we are able to tune the models in various ways compared to machine learning models like Logistic Regression and SVM that have limited parameter tuning capabilities. From our experimentation we noticed that training CNN model takes a long time (on average it takes 3 hours to train a CNN model) due to its computational complexity. However, we managed to contribute to find the parameter that reduces time for CIFAR-100 image classification which is using ELU activation function and ADAM optimiser.

On the other hand, when we compare base classifier - SVM with logistic regression, we noticed that LR performs poorly than the base classifier in terms of test set accuracy, precision and recall. This result is expected because we have tested on Box-Tidwell test and all features does not satisfy linearity assumption. Having non-linearity component in logit would affect the classification performance, this leads to a poor result. However it trains the fastest among all classifiers such the time complexity to train LR is only O(nm), this enable us to explore more on data preprocessing method, which we then learned that ZCA epsilon, ZCA whitening and Image Whitening gave a better result for image clasification.

Another findings we note is that, the result from Table 12 we noticed that all three classifiers managed to balance the precision and recall. Usually these metrics measure different aspects and normally would obtain different results. However, we noticed that it is relatively similar in our case. One possible reason is because we have distributed our sample in a stratified manner according to each class, each class would have same number of samples 200 of image data to train. There will not be much differences or variation between the number of images correctly classified compared against the total number of samples in each class.



	precision	recall	f1-score	support	Model
55	0.08	0.01	0.03	200	CNN
55	0.02	0.01	0.01	200	LR
55	0.12	0.03	0.04	200	SVM

Fig. 5. Result for Class 55: Otter

When we compared the classification report from three of the classifiers, interestingly we noticed a commonality between the classifiers, the class 55: Otter tend to be misclassified easily among the classifiers. In terms of the CNN classifier the recall for the class was at 1% which pulls down the average F1-score to 3%, when we looked at logistic regression we noticed that the precision for this class was 2% , recall scored 1% and F1-score only managed to achieve 1%. In terms of SVM classifier, the recall for this class was only at 3% which pulls the F1-Score down to 4%. Although we distribute the data samples equally for each class when training the classifiers, two possible explanations:
1.The images of otter in this dataset was in low quality and was captured in a way that makes the model unable to depict the differences with the other classes, for example otter in this dataset might look like the animal seal.

2. The class otter might be very similar to other classes like the animal seal.
These two reasons might be the factor behind that makes the general classification of the image otter slightly difficult to be classified correctly.

In terms of speed, we noticed that SVM took a long time to train, it exceed 1 hour for training time and we have to run for more than 1 day for cross l validation. In future, we plan to optimiser the speed for svm model by combining with other machine learning model like to increase performance. According to Advani (26), training SVM for CIFAR-100 dataset usually requires a longer duration, the author took about 10 hours to train a SVM model for CIFAR-100 dataset. Gopee et al. (28) on the other hand also concluded the same thing training SVM for image classification is computationally expensive and the author suggest to use ensemble, combining SVM with multinomial logistic regression to reduce the computaiton time. The author experimented this method and noticed that by using this combination, multinomial logistic regression managed to improves the computational time of SVM, and the accuracy performance increased by about 15%.

1

6 CONCLUSION AND FUTURE WORK

Conclusion: The conclusion that we have achieved for the classification is that the CNN works better than any other classifier and this was because it trains more and have more parameters to tune to achieve higher accuracy. The other two classifiers – SVM and LR are also good classifiers, but they underperform in comparison to CNN as they only train in 1 epoch of the training data and have limited parameter tuning capabilities(26). We noticed that it normally takes 3 hours to train a CNN network, however, we managed to made a contribution, which is we found that ELU activation function and ADAM optimiser reduces the computation into less than an hour for 20 epochs.

Future Work and Improvements: The future work and improvements that we have concluded that might increase the accuracy of the classifiers are:
1. For CNN: Since the accuracy is only about 45%, in future we can increase this by running the CNN model with more epochs – 50 or 100 so that the model can train with more iterations and hence in turn increase the accuracy.
2. The Preprocessing: We can do more image preprocessing and image augmentation methods that include rotation, transitions, brightness and zoom as well as getting out the pixels for feature extraction.
3. Ensemble Methods: We were also thinking of combining a few classifier methods together such as the SVM with the LR to create an ensemble method which might be able to increase the accuracy and reduces the computation cost for a faster training time.

[1]COMP5318 Machine Learning Assignment 2

REFERENCES

[1] Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H. Training data-efficient image transformers distillation through attention. InInternational Conference on Machine Learning 2021 Jul 1 (pp. 10347-10357). PMLR.

[2] Mangalam K, Prabhu VU. Do deep neural networks learn shallow learnable examples first?.

[3] Smith LN. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. arXBoneyiv preprint arXiv:1803.09820. 2018 Mar 26.

[4] Dahiya S, Tyagi R, Gaba N. Comparison of ML classifiers for Image Data. EasyChair; 2020 Jul 11.

[5] Boney L, Tewfik AH, and Hamdy KN. “Digital Watermarks for Audio Signals," *Proceedings of the Third IEEE International Conference on Multimedia*, pp. 473-480, June 1996.

[6] Goossens M, Mittelbach F, Samarin. *A LaTeX Companion*, Addison-Wesley, Reading, MA, 1994.

[7] Kopka H, Daly PW. *A Guide to LaTeX*, Addison-Wesley, Reading, MA, 1999.

[8] Pan D. ‘*A Tutorial on MPEG/Audio Compression*‘," *IEEE Multimedia*, Vol.2, pp.60-74, Summer 1998.

[9] Wu X, Zuo W, Lin W, Jia W and Zhang D. *F-SVM: Combination of Feature Transformation and SVM Learning via Convex Relaxation,"*" in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, pp. 5185-5199, Nov. 2018

[10] Qing Y, Zeng Y, Li Y, Huang GB. Deep and wide feature based extreme learning machine for image classification. Neurocomputing. 2020 Oct 28;412:426-36.

[11] Reinders C, Rosenhahn B. Neural Random Forest Imitation. arXiv preprint arXiv:1911.10829. 2019 Nov 25.

[12] Pant A. Introduction to Logistic Regression. [internet]. Medium.2019 [cited Nov 7]; Available from: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148, 22 January 2019

[13] Rahman F. *Logistic Regression for Image Classification [Internet]*. Medium. 2021, 31 July 2020]. Available from: https://medium.com/swlh/logistic-regression-for-image-classification-e15d0ae59ce9

[14] Sorokina K. *Image Classification with Convolutional Neural Networks [Internet]*. Medium. 2017 [20 November 2017]. Available from: https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8

[15] Taha B. What is SVM | Build an Image Classifier With SVM [Internet]. Analytics Vidhya. 2021 [18 June 2021]. Available from: https://www.analyticsvidhya.com/blog/2021/06/build-an-image-classifier-with-svm

[16] Krizhevsky A. CIFAR-10 and CIFAR-100 datasets [Internet]. Cs.toronto.edu. 2021 [cited 6 November 2021]. Available from: https://www.cs.toronto.edu/ kriz/cifar.html

[17] Yamashita R, Nishio M, Do RK, Togashi K. Convolutional neural networks: an overview and application in radiology. Insights into imaging. 2018 Aug;9(4):611-29.

[18] Shearer P. Kernel SVM: I want an intuitive understanding of mapping to a higher-dimensional feature space, and how this makes linear separation possible [Internet]. stackexchange. 2016 [cited 6 November 2021]. Available from: https://stats.stackexchange.com/questions/168051/kernel-svm-i-want-an-intuitive-understanding-of-mapping-to-a-higher-dimensional

[19] Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv:1312.4400. 2013 Dec 16.

[20] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4700-4708).

[21] Balaji A, Wu Y, Yoon J. CIFAR100 Convolutional Model Based Classification Benchmark.

[22] Dong Q, Zhu X, Gong S. Single-label multi-class image classification by deep logistic regression. InProceedings of the AAAI Conference on Artificial Intelligence 2019 Jul 17 (Vol. 33, No. 01, pp. 3486-3493).

[23] Ruder S. An overview of gradient descent optimization algorithms. arXiv 2016. arXiv preprint arXiv:1609.04747. 2016.

[24] Fleyeh H, Roch J. Benchmark evaluation of HOG descriptors as features for classification of traffic signs. Högskolan Dalarna; 2013.

[25] Yekkehkhany B, Safari A, Homayouni S, Hasanlou M. A comparison study of different kernel functions for SVM-based classification of multi-temporal polarimetry SAR data. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 2014;40(2):281.

[26] Advani V. Deep Learning vs Machine Learning | Know the Difference [Internet]. GreatLearning Blog: Free Resources what Matters to shape your Career! [Internet]. 2021 [cited 7 November 2021]. Available from: https://www.mygreatlearning.com/blog/deep-learning-vs-machine-learning/

[27] Xu H, Ainsworth M, Peng YC, Kusmanov M, Panda S, Vogelstein JT. When are Deep Networks really better than Random Forests at small sample sizes?. arXiv e-prints. 2021 Aug:arXiv-2108.

[28] Gopee N. Classifying CIFAR-10 Images Using Unsupervised Feature Ensemble Learning

Appendices

.1 How to run the code file?

The following depicts the steps that can be taken to run out code files

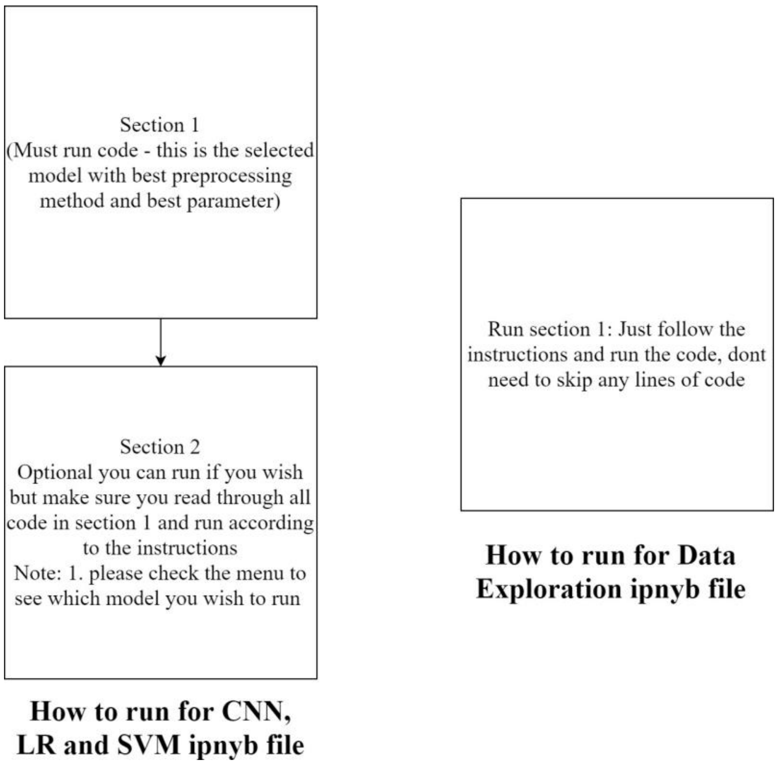
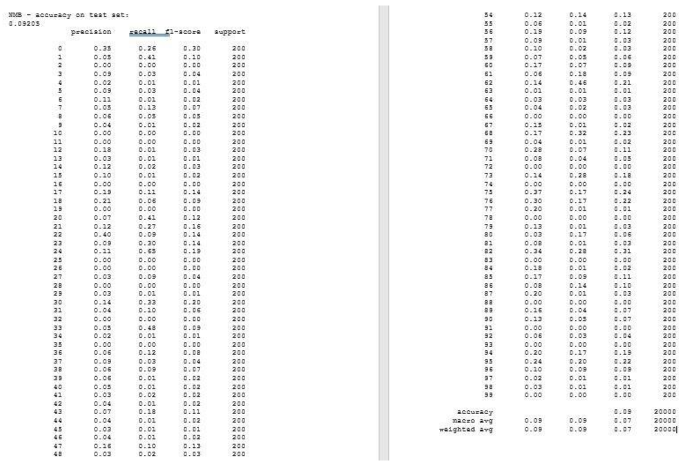

Section 1
(Must run code - this is the selected model with best preprocessing method and best parameter)
Section 2
Optional you can run if you wish but make sure you read through all code in section 1 and run according to the instructions
Note: 1. please check the menu to see which model you wish to run
How to run for CNN, LR and SVM ipnyb file
Run section 1: Just follow the instructions and run the code, dont need to skip any lines of code
How to run for Data Exploration ipnyb file

Fig. 6. How to run our code files

.2 Classification Reports for Naive Bayes and K-Nearest Neighbours



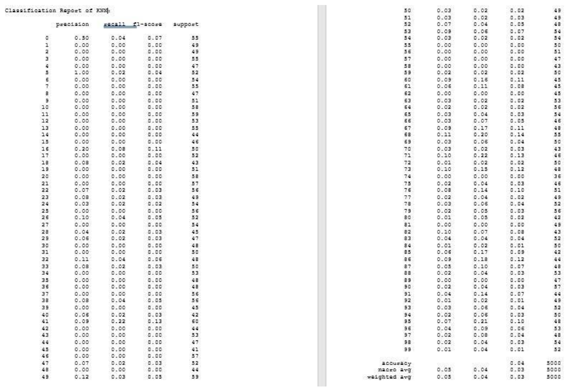
```
NMB - accuracy on test set:
0.09205
              precision    recall  f1-score   support

           0       0.35      0.26      0.30       200
           1       0.05      0.41      0.10       200
           2       0.00      0.00      0.00       200
           3       0.09      0.03      0.04       200
           4       0.02      0.01      0.01       200
           5       0.09      0.03      0.04       200
           6       0.11      0.01      0.02       200
           7       0.05      0.13      0.07       200
           8       0.06      0.05      0.05       200
           9       0.04      0.01      0.02       200
          10       0.00      0.00      0.00       200
          11       0.00      0.00      0.00       200
          12       0.18      0.01      0.03       200
          13       0.03      0.01      0.01       200
          14       0.12      0.02      0.03       200
          15       0.10      0.01      0.02       200
          16       0.00      0.00      0.00       200
          17       0.19      0.11      0.14       200
          18       0.21      0.06      0.09       200
          19       0.00      0.00      0.00       200
          20       0.07      0.41      0.12       200
          21       0.12      0.27      0.16       200
          22       0.40      0.09      0.14       200
          23       0.09      0.30      0.14       200
          24       0.11      0.65      0.19       200
          25       0.00      0.00      0.00       200
          26       0.00      0.00      0.00       200
          27       0.03      0.09      0.04       200
          28       0.00      0.00      0.00       200
          29       0.03      0.01      0.01       200
          30       0.14      0.33      0.20       200
          31       0.04      0.10      0.06       200
          32       0.00      0.00      0.00       200
          33       0.05      0.48      0.09       200
          34       0.02      0.01      0.01       200
          35       0.00      0.00      0.00       200
          36       0.06      0.12      0.08       200
          37       0.09      0.03      0.04       200
          38       0.06      0.09      0.07       200
          39       0.06      0.01      0.02       200
          40       0.05      0.01      0.02       200
          41       0.03      0.02      0.02       200
          42       0.04      0.01      0.02       200
          43       0.07      0.18      0.11       200
          44       0.04      0.01      0.02       200
          45       0.03      0.01      0.01       200
          46       0.04      0.01      0.02       200
          47       0.16      0.10      0.13       200
          48       0.03      0.02      0.03       200
```

```
          54       0.12      0.14      0.13       200
          55       0.06      0.01      0.02       200
          56       0.19      0.09      0.12       200
          57       0.09      0.01      0.03       200
          58       0.10      0.02      0.03       200
          59       0.07      0.05      0.06       200
          60       0.17      0.07      0.09       200
          61       0.06      0.18      0.09       200
          62       0.14      0.46      0.21       200
          63       0.01      0.01      0.01       200
          64       0.03      0.03      0.03       200
          65       0.04      0.02      0.03       200
          66       0.00      0.00      0.00       200
          67       0.15      0.01      0.02       200
          68       0.17      0.32      0.23       200
          69       0.04      0.01      0.02       200
          70       0.28      0.07      0.11       200
          71       0.08      0.04      0.05       200
          72       0.00      0.00      0.00       200
          73       0.14      0.28      0.18       200
          74       0.00      0.00      0.00       200
          75       0.37      0.17      0.24       200
          76       0.30      0.17      0.22       200
          77       0.20      0.01      0.01       200
          78       0.00      0.00      0.00       200
          79       0.13      0.01      0.03       200
          80       0.03      0.17      0.06       200
          81       0.08      0.01      0.03       200
          82       0.34      0.28      0.31       200
          83       0.00      0.00      0.00       200
          84       0.18      0.01      0.02       200
          85       0.17      0.09      0.11       200
          86       0.08      0.14      0.10       200
          87       0.20      0.01      0.03       200
          88       0.00      0.00      0.00       200
          89       0.16      0.04      0.07       200
          90       0.13      0.05      0.07       200
          91       0.00      0.00      0.00       200
          92       0.06      0.03      0.04       200
          93       0.00      0.00      0.00       200
          94       0.20      0.17      0.19       200
          95       0.24      0.20      0.22       200
          96       0.10      0.09      0.09       200
          97       0.02      0.01      0.01       200
          98       0.03      0.01      0.01       200
          99       0.00      0.00      0.00       200

    accuracy                           0.09     20000
   macro avg       0.09      0.09      0.07     20000
weighted avg       0.09      0.09      0.07     20000
```

Fig. 7. Classification Report of Naive Bayes



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

           0       0.50      0.04      0.07        55
           1       0.00      0.00      0.00        49
           2       0.00      0.00      0.00        49
           3       0.00      0.00      0.00        55
           4       0.00      0.00      0.00        47
           5       1.00      0.02      0.04        52
           6       0.00      0.00      0.00        54
           7       0.00      0.00      0.00        55
           8       0.00      0.00      0.00        47
           9       0.00      0.00      0.00        51
          10       0.00      0.00      0.00        58
          11       0.00      0.00      0.00        59
          12       0.00      0.00      0.00        53
          13       0.00      0.00      0.00        55
          14       0.00      0.00      0.00        44
          15       0.00      0.00      0.00        46
          16       0.20      0.08      0.11        50
          17       0.00      0.00      0.00        52
          18       0.08      0.02      0.04        43
          19       0.00      0.00      0.00        51
          20       0.00      0.00      0.00        58
          21       0.00      0.00      0.00        57
          22       0.07      0.02      0.03        56
          23       0.08      0.02      0.03        49
          24       0.03      0.02      0.02        54
          25       0.00      0.00      0.00        56
          26       0.10      0.04      0.05        52
          27       0.00      0.00      0.00        54
          28       0.04      0.02      0.03        45
          29       0.06      0.02      0.03        47
          30       0.00      0.00      0.00        48
          31       0.00      0.00      0.00        50
          32       0.11      0.04      0.06        48
          33       0.08      0.02      0.03        50
          34       0.00      0.00      0.00        53
          35       0.00      0.00      0.00        48
          36       0.00      0.00      0.00        48
          37       0.00      0.00      0.00        56
          38       0.08      0.04      0.05        56
          39       0.00      0.00      0.00        45
          40       0.06      0.02      0.03        42
          41       0.09      0.22      0.13        60
          42       0.00      0.00      0.00        44
          43       0.00      0.00      0.00        53
          44       0.00      0.00      0.00        47
          45       0.00      0.00      0.00        41
          46       0.00      0.00      0.00        57
          47       0.07      0.02      0.03        52
          48       0.00      0.00      0.00        44
          49       0.12      0.03      0.05        59
```

```
          50       0.03      0.02      0.02        49
          51       0.03      0.02      0.03        49
          52       0.07      0.04      0.05        48
          53       0.09      0.06      0.07        54
          54       0.03      0.02      0.02        54
          55       0.00      0.00      0.00        50
          56       0.00      0.00      0.00        51
          57       0.00      0.00      0.00        47
          58       0.00      0.00      0.00        43
          59       0.02      0.02      0.02        50
          60       0.09      0.16      0.11        45
          61       0.06      0.11      0.08        45
          62       0.00      0.00      0.00        45
          63       0.03      0.02      0.02        53
          64       0.02      0.02      0.02        56
          65       0.03      0.04      0.03        54
          66       0.03      0.07      0.05        46
          67       0.09      0.17      0.11        48
          68       0.11      0.20      0.14        55
          69       0.03      0.06      0.04        50
          70       0.03      0.02      0.03        43
          71       0.10      0.22      0.13        46
          72       0.01      0.02      0.02        50
          73       0.10      0.15      0.12        48
          74       0.00      0.00      0.00        36
          75       0.02      0.04      0.03        46
          76       0.08      0.14      0.10        51
          77       0.02      0.04      0.02        49
          78       0.03      0.06      0.04        52
          79       0.02      0.05      0.03        56
          80       0.01      0.05      0.02        42
          81       0.00      0.00      0.00        49
          82       0.10      0.07      0.08        43
          83       0.04      0.04      0.04        52
          84       0.01      0.02      0.01        50
          85       0.06      0.17      0.09        42
          86       0.09      0.18      0.12        44
          87       0.05      0.10      0.07        48
          88       0.02      0.04      0.03        53
          89       0.00      0.00      0.00        47
          90       0.02      0.04      0.03        57
          91       0.04      0.14      0.07        44
          92       0.01      0.02      0.01        49
          93       0.03      0.06      0.04        52
          94       0.02      0.06      0.03        50
          95       0.07      0.21      0.10        48
          96       0.04      0.09      0.06        53
          97       0.02      0.08      0.04        48
          98       0.02      0.04      0.03        54
          99       0.01      0.04      0.01        52

    accuracy                           0.04      5000
   macro avg       0.05      0.04      0.03      5000
weighted avg       0.05      0.04      0.03      5000
```

Fig. 8. Classification Report of KNN