

Simple comparison between deep learning algorithm and traditional ML algorithm on image recognition problem

**Linxin Sun 510020837
Michael (Jiachong) Lai 510124319**

Oct, 2021

1 Abstract

Image recognition is one of the most important aspects in machine learning. However, it is sometime difficult to decide whether we should adopt deep learning or traditional machine learning algorithm. Therefore in this project, three image recognition models were trained on cifar100 to determine if deep learning algorithm would out-perform traditional machine learning method in complicated image recognition problem. The three algorithms trained included k nearest neighbor (KNN), support vector machine (SVM) and convolutional neural network (CNN). At test time, CNN model achieved the highest prediction accuracy and have the best overall performance. Showed that for recognizing image in cifar100 data set, deep learning algorithm specifically CNN indeed out-performed the traditional machine learning algorithms.

2 Introduction

Image recognition is one of the most important aspects in machine learning that needs to be studied. Its high value is reflected in all aspects of life, such as video surveillance, autonomous driving and smart healthcare. However, there are many challenges that people are facing when implementing image recognition technology. For example, given a complicated image, it is sometimes difficult to decide whether we should adopt deep learning or traditional machine learning algorithms.

In this project, several image recognition models were trained and the main goal was to compare image recognition performance between deep learning model such as convolutional Neural Networks (CNN) and traditional supervised learning models such as support vector machine (SVM) and k nearest neighbor (KNN).

In this project, a data set namely cifar100 was used, which consisted of 50000 samples in training set and 10000 samples in test set, each of the sample was a 32x32 RGB image, which after flatten, the data set was a 50000x3072 matrix and the label of each sample would fall into 1 of the 100 different labels.

For the CNN model, multiple techniques that might increase accuracy were adopted such as dropout (Srivastava, et al., 2014) as well as Adam (Kingma & Ba, 2015) in order to conserve computing power as well as to increase accuracy; for SVM and KNN, cross-validation was adopted for fine tuning.

3 Related work

3.1 Sharpness-Aware Minimization

For this particular data set, the state-of-the-art algorithm proposed by Foret and his team provides a prediction accuracy of 96.08% (Foret et al., 2021). In Foret's article, a new technique

called Sharpness-Aware Minimization (SAM) was introduced. Rather than finding the parameters with lowest loss value as in our project, SAM could find the parameters with uniformly low loss in neighbourhood (Foret et al., 2021). With SAM, Foret and his team was able to reach 96% accuracy in cifar100 data set. Comparing to the algorithm used in this project, adopting SAM would likely to increase the computing complexity as well as require more computing power.

3.2 Transformer Architecture

Another state-of-the-art algorithm proposed by Dosovitskiy and his team provides a prediction accuracy around 94% (Dosovitskiy, et al., 2021). In Dosovitskiy’s algorithm, transformer architecture which normally used in natural language process was adopted rather than using CNN. Comparing to our method, the algorithm Dosovitskiy proposed was more efficient and could provide a much higher prediction accuracy, yet our method used CNN which was the tradition way of doing image recognition and easier to implement.

4 Pre-processing

4.1 Principal Components Analysis (PCA)

The PCA technique uses linear projection to map high-dimensional data to low-dimensional space, using the principle of maximizing the variance of the data in the projected dimensions and using fewer dimensions. At the same time, the characteristics of the original data are greatly preserved. PCA technique is a linear dimensional reduction method with minimum loss of data information. Therefore, this technique is used in this study to pre-process the data to achieve the purpose of reducing the complexity of the data. The specific implementation formula is as follows.

$$g\left(\theta^T x\right)=\frac{1}{1+e^{-\theta^T x}} \quad (1)$$

In the implementation process, the points with n_component = 0.80, dimensions = 25 were first selected for prediction. As shown in figure 1, when dimensions was chosen to be 25, 80% of the variance of the data retained

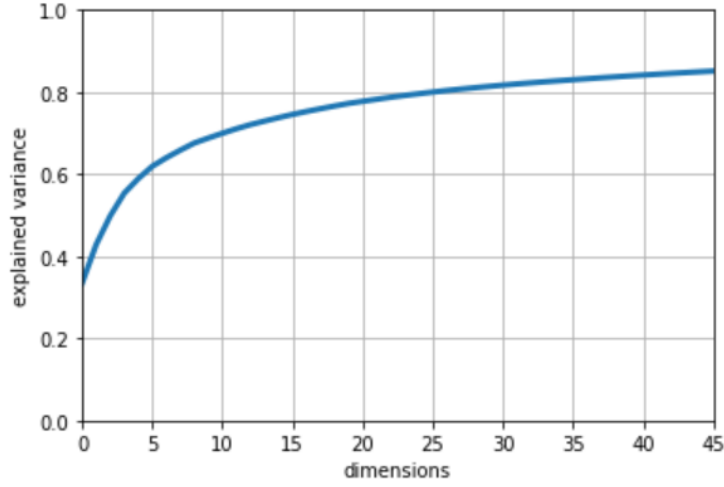


Figure 1: PCA dimensions

5 Method

5.1 Support Vector Machine (SVM)

The SVM algorithm classifies the data according to the supervised learning method and the inverse binary classification method. The maximum marginal hyperplane in learning the samples is the bounded edge curves after the SVM algorithm. The sklearn default SVC used in this study in order of 2, i.e. the kernel function used is a polynomial kernel function of order 2 and is a one verse rest classifier.

The SVM algorithm is very effective in handling classification problems because there are a large number of kernel functions available and it can be very flexible to handle some classification problems. Moreover, it only uses a part of the data to support vectors to make planar decisions and does not use all the data. (And the classification accuracy is high and the generalization ability is also strong. So try to implement it in this research).

Specifically for this study, the n-2 type of running linear and polynomial models were used. 50,000 training data from the dataset were used for training and testing. 75% of the training data were used as new training data and 25% of the training data were used as new test data. It can be seen that the accuracy of the linear model was 17.6% and the accuracy of the polynomial model was 19.4%. The results are shown in the following table.

Type	Accuracy
Linear SVM	0.176
Quadratic SVM	0.194

5.2 K Nearest Neighbour (KNN)

The basic idea behind KNN algorithm is to select the K most adjacent samples of a given sample in the feature space calculated according to a certain distance formula, by majority algorithm, the label that occur the most would be voted as the output.

The KNN algorithm is suitable for forming classification models, especially for rare event classification problems, and has the characteristics of being easy to understand and implement, suitable for analyzing multiple classification problems and even with large number of labels. Therefore an attempt was made to implement KNN algorithm in this study.

Specifically in this study, sklearn was first used to implement algorithm. Secondly the K parameter of KNN was first chosen to be 1. Out of the 50,000 training data, 75% of the data were classified as new training data and 25% were classified as new test data. It can be seen that the accuracy was 17% at this point. The result is shown in the figure below.

```
Size of train and train label set: (37500, 202), (37500,).  
Size of test and test label set: (12500, 202), (12500,).  
Accuracy on test set: 0.17
```

Figure 2: KNN study

5.3 Convolutional Neural Networks (CNN)

CNN is a neural network that specialize in image recognition, CNN usually takes 3 channels (RGB) image or 1 channel (black and white) image as input. By applying filter which usually is a matrix on the image to producing feature maps. Where the size of feature maps depend on the size of the filters. The values in filter matrix was usually unknown, every time a train data pass through, the values in filter matrix would be updated by back-propagation procedure. To prevent the model from being linear, a non-linear activation function was usually applied on the feature maps. Then by pooling and flattening the feature map, a 1 x (width x height x channel) matrix would be formed to pass into the fully connected network.

In this project, a CNN was trained and used to make prediction. GPU was used in order to train the model. Due to the lack of GPU resource on local machine the model was trained on Kaggle kernel. For which provided a free-to-use GPU resource that could significantly shorten the computation time.

Unlike the other two algorithm implemented, the cifar100 data set was imported using keras.datasets package. Cifar100 data set consisted 50000 training data and 10000 test data. Each samples in cifar100 data set consisted a RGB image, 32 in height and 32 in width, leading each of the image to be a 32x32x3 tensor. For pre-processing, the 32x32x3 tensor in both train and test set were first converted to 32 bits floating point number, then each value in the tensor was normalized by dividing the value by its standard deviation of the tensor, such that the every floating point number was restrained to be within 0 and 1. The label data were also converted

into 32 bits integer which could be potentially easier to manipulate.

Afterward, in order to increase the prediction accuracy, image pre-processing technique was implemented. Image data generator package from keras was imported to implement image augmentation. The rotation range was set to be 20 degree, meaning that the generated image might have a rotation as much as 20 degree; the horizontal flip argument was set to be true, allowing to generate horizontal-flip images of the original images; the width and height shift range were both set to be 0.1, allowing the new generated image to move horizontally or vertically. After applying the image data generator, a new set of training data containing the original data and their transformations were used to train the data. Due to the increase in size, it might consume more computing power and computation time to train the model, yet this data augmentation technique could potentially increase the prediction accuracy.

As shown in figure 3, when constructing the convolutional layers, three 2-dimensional convolutional layers were constructed. The first 2-dimensional convolutional layer accepted the (32, 32, 3) training data, applying 128 filters on it, with 3x3 convolution window. Padding was added evenly around each images. The activation function was set to Rectified Linear Unit (relu), which convert all negative input into 0. Between the first 2-dimensional convolutional layer and second 2-dimensional convolutional layer, a normalization layer was added to keep the data normalized.

The second and the third 2-dimensional convolutional layer both had 64 filters and had 3x3 convolution window, the padding was set to be same and the activation function used was relu. Normalization layers were added after each convolutional layer to normalize the data. After the normalization layer, two max pooling layers were added with kernel size of 3x3. Meaning that the 3x3 matrix would be condense into a single value which would be the largest value in the matrix. Padding for both pooling layers were set to be same.

After flattening the matrix, the train data were passed to three dense layers. The first layer was consisted of 512 units, using relu as activation function. After the first dense layer, a normalization and dropout layer was added, in order to normalized the training data. The dropout was set to be 0.5, meaning that for each training, 50% of the units would be in-activated, which could theoretically improve the performance of the model (Srivastava, et al., 2014). The second dense layer was consisted of 256 units, using relu as activation function, a 50% dropout layer and a normalization layer were also added after this dense layer. For the last fully connected layer, number of units were set to be 100, where each of the units represent one label, soft max was used as activation function, which could convert vectors of number into probability of labels.

When training the model, rather than having the optimizes using Stochastic gradient descent, Adam was adopted, which could theoretically accelerate computation time and lower memory usage (Kingma & Ba, 2015).

When training the model, epochs was set to be 500 and batch size was set to be 256. When training the mdoel, the validation set used was the test data from cifar100, therefore, the validation accuracy would simply be the prediction accuracy of the CNN model. After training, weights for each of the parameters were saved in a .h5 file for future usage.

After training the CNN models, in order to distinguish the impact of learning rate and image augmentation technique, the code was executed twice more with 50 epochs each, and with different parameters. Model obtained from run 1 was considered as the primary model, with learning rate set to 5e-3 and image generator enabled; run 2 was the execution that with learning rate 1e-3 and image generator enabled; run 3 was the execution that with learning rate 5e-3 and image generator disabled. The run time, test and validation accuracy were then used to identify the impact of different parameters. A separate execution was done without GPU at the end, in order to identify if the usage of GPU would increase the performance and decrease the model training time.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 128)	3584
batch_normalization (Batch Normalization)	(None, 32, 32, 128)	512
conv2d_1 (Conv2D)	(None, 32, 32, 64)	73792
batch_normalization_1 (Batch Normalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 11, 11, 64)	0
batch_normalization_2 (Batch Normalization)	(None, 11, 11, 64)	256
conv2d_2 (Conv2D)	(None, 11, 11, 64)	36928
batch_normalization_3 (Batch Normalization)	(None, 11, 11, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
batch_normalization_4 (Batch Normalization)	(None, 6, 6, 64)	256
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 512)	1180160
batch_normalization_5 (Batch Normalization)	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
batch_normalization_6 (Batch Normalization)	(None, 256)	1024
dense_2 (Dense)	(None, 100)	25700
Total params: 1,456,100		
Trainable params: 1,453,796		
Non-trainable params: 2,304		

Figure 3: Detailed layers configuration for CNN model

6 Experiments & Discussion

6.1 Support Vector Machine (SVM)

6.1.1 10-fold CV for SVM

The 10-fold grid search method is used in this study to find the optimal parameters in the SVM algorithm. The parameters set to be tested are specifically two-dimensional polynomial

multifunction, three-dimensional polynomial multifunction and four-dimensional polynomial multifunction. The following table shows the performance of the SVM algorithm based on each parameter run. From the table, it can be seen that the best parameter determined after the 10-fold grid search method is the two-dimensional polynomial multifunction.

Degree	Accuracy
2	0.198
3	0.196
4	0.152

6.1.2 Evaluation metric for SVM

The accuracy rate refers to the number of correctly classified samples after classification as a principal example of the total number of samples, with the following formula:

$$Accuracy = \frac{n_{correct}}{n_{total}} \quad (2)$$

In the above equation, ncorrect is the number of correctly classified samples in the classification, and ntotal is the total number of samples in the database.

After training the model with hyper parameter, the best accuracy the SVM algorithm can achieve was 0.208.

Precision refers to the percentage of samples that need to be predicted as positive that are also predicted by the model to be positive in their true value. The general formula for deriving precision are as follow:

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

Where TP is the number of data with true positive values and FP is the number of data with false positive values.

By using the poly SVM and two-dimensional polynomial function as the selected parameters, the accuracy obtained in the test data is shown in the following table.

Average	Accuracy
Macro	0.205
Micro	0.180

In this study two values were selected to calculate the precision rate. That is, macro average and micro average. Where macro average means that each category will have the same weight when calculating the mean, and the final result will be the arithmetic mean of the indicators of each category. micro average means that each sample of all categories will be given the same

weight when calculating the multi-categorical indicators, and then All samples are put together to calculate each indicator. The range of the accuracy index is from 0 to 1.

Recall is used to describe the ability of the classifier to find all positive samples which can be calculated by the following equation:

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

Where the TP stands for true positive and FN stands for false negative. The recall obtained from SVM model was 0.187

6.2 K Nearest Neighbour (KNN)

6.2.1 10-fold CV for KNN

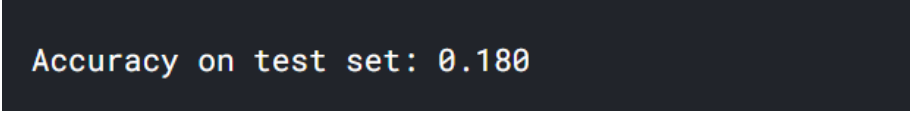
The parameters set for the 10-fold CV grid search in the KNN algorithm are n neighbors as 3, 5 and 7, respectively. and p is chosen as 1, 2, which represent the distance algorithm taken as the Manhattan distance algorithm and the Euler distance algorithm, respectively. Then there are a total of 6 cases for screening, and the following table is a summary of these 6 cases. It can be seen that the best parameter selected after 10-fold CV is p taken as 1, i.e., Manhattan distance algorithm and n neighbor as 7.

Number of neighbor	p	Accuracy
3	1	0.1626
3	2	0.1571
5	1	0.1699
5	2	0.1668
7	1	0.1744
7	2	0.1735

6.2.2 Evaluation metric for KNN

Accuracy is used to represent the number of samples correctly classified using the KNN algorithm as a percentage of the total number of samples.

The parameters selected after the above study were neighbors of 7 and p of 1. Then the KNN algorithm using these two parameters was applied to the test dataset and the accuracy obtained was 0.18, and the results are shown in the figure. The macro and micro accuracy is shown in the table below.



Accuracy on test set: 0.180

Figure 4: The Accuracy for KNN

Type	Accuracy
Macro	0.209
Micro	0.187

6.3 CNN

CNN model provided a prediction accuracy of 60.47%, meaning that the 60% prediction made by using this model was correct. The average precision, recall and f1-score for this model was 62%, 60% and 60% respectively, meaning that on average 62% of the predictions were correct and for a given class, 60% of the data in that class was correctly classified. Macro average method was implemented, which used the recall and precision for each class and using the average of them as the model average. The f-1 score, which was a combination of precision and recall was calculated by using formula:

$$f1 = \frac{2 \times (precision \times recall)}{precision + recall} \quad (5)$$

The macro average f1-score for this model was 60%, which indicated that this model was not perfect, yet acceptable.

Figure 5 showed the training and validation accuracy for each run. In this figure, red and pink lines represented model 1 results obtained from run 1; green and light blue lines represented model 2 results from run 2; blue and black lines represented model 3 results from run 3. From the figure, accuracy obtained from model 3 which had relatively low learning rate (5e-3) and not using image augmentation, had the highest training accuracy among all three runs yet the lowest validation accuracy. By comparing result from model 3 and model 1, it seems that with the same learning rate, image augmentation technique would help to increase the validation accuracy, which would likely to produce a model that could better classify the data set. However, as shown in figure 6, running time for model 3 was the lowest, showing that image augmentation would cost more computing power due to the increase in data size. With the increase in the data size, the trained model would be more likely to find the pattern of pixel layout in image and more likely to have a relatively well-perform weights for each parameters leading the trained model to have better prediction accuracy

Also from figure 5 and 6, model 2 which was trained with image generator enabled but higher learning rate (1e-2) had similar train and validation accuracy to model 1. Yet, as the training continue, the validation accuracy remained around 50%, leading the model to be less

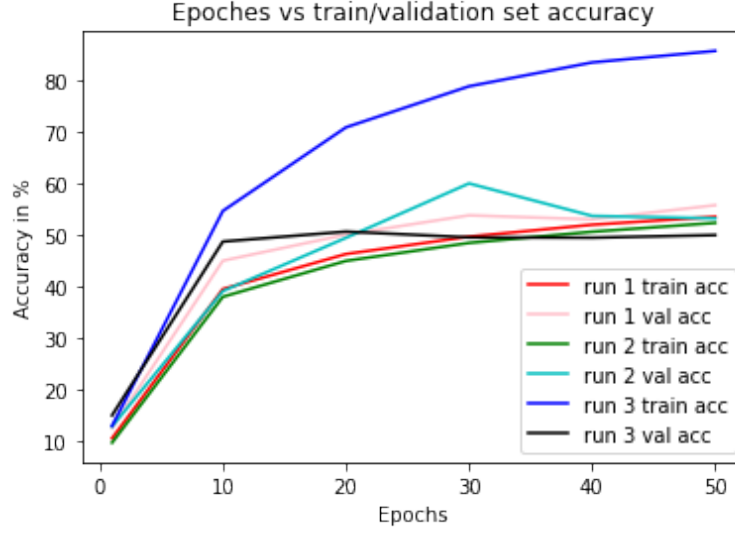


Figure 5: Training & validation accuracy in each run

accurate. One potential reason for that was with higher learning rate, the model would be more likely to overshoot the global minimal and bouncing around the global minimal.

Therefore, the model 1 with image generator enabled and learning rate around $5e-3$, was chosen as the best model, which had prediction accuracy around 60%. However, even model 1 was the best of the three, it was not perfect. As shown in figure 5, the trained accuracy was as well around 60%, indicating that this model was apparently under fitting. The whole process was relatively time consuming as well, the time spent on training this model was exactly 4 hours, yet the training accuracy was only around 60%. The validation accuracy might increase as to increase the number of epochs, yet this model provided an acceptable prediction accuracy, therefore, this model was chosen to be the best model.

In order to compare if computer infrastructure would have impact on the model, as mentioned in method section, the model was also trained in a CPU only environment. As shown in figure 6, it was obvious that compare to running on GPU environment, running only on CPU might cost as much as 4 times the computation time. The potential reason for that was GPU were good at data-level parallel, such as matrix multiplication problem. For which the CNN model was almost nothing more than a matrix multiplication problem.

Since the data was consisted of 100 labels, the confusion output was so large, therefore the heat map for confusion matrix was placed in appendix section. Please see figure 7 in the appendix.

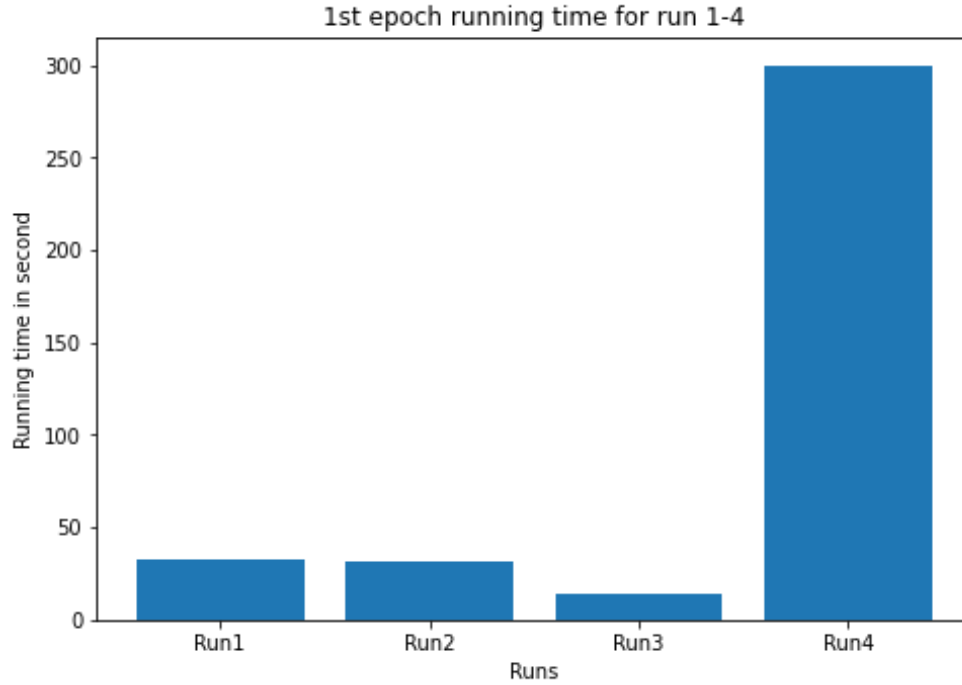


Figure 6: Running time of first epoch for each run

6.4 Models comparison

Algorithm	Accuracy
SVM	0.205
KNN	0.180
CNN	0.60

The table above showed the comparison on accuracy among three algorithms. As shown in the table, the best highest accuracy was achieved by using CNN model, with a value of around 60%. Showing that for this particular data set, deep learning model such as CNN did out-perform traditional machine learning algorithm such as SVM and KNN.

7 Conclusion & Future work

7.1 Conclusion

By constructing three classification model using different algorithm, convolutional neural network model had the best performance among the three models for this particular data set. The

CNN model out-performed the other two algorithm with a prediction accuracy of 60%. Yet due to the limitation of the project, a general conclusion can not be drawn, since only one complicated image data set was used.

7.2 Future Work

In the future, for the purpose of identifying the best algorithm to use in such image multi-class classification problem, one could try to run KNN, SVM and CNN on different data set to ferret out if a consistent result can be derived.

And when implementing the KNN and SVM algorithm code, the running time added up to over 20 hours. Because they are traditional machine learning algorithms, some reduction in runtime was required. In future work, we will try to run these two types of algorithms by improving the data variance of SVM, i.e., and then improving the accuracy of the downscaled data to achieve the runtime reduction.

In terms of improving accuracy of CNN model, there were a few approaches can be implemented. First was to use different convolutional layer layout, such as VGG and Resnet or even by simply increasing the number of layers, which would potentially improve the prediction accuracy. Another approach was that with more computing power, one could try to increase the epoch size. By increasing the epoch size, one could expect the increase in train accuracy and possible increase in validation accuracy. In terms of parameters usage, one could use grid search to find the best learning rate which could significantly decrease the training time and prevent over-shooting the global minimal.

References

1. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15.
2. Kingma, D. P., & Ba, J. (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. *ICLR Conference Paper*.
3. Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2021). SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION. *ICLR Conference Paper*.
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. *ICLR Conference Paper*.

8 Appendix

8.1 Data set link

<https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz>

please noted that by clicking the link above, a zip file containing cifar100 data set will be downloaded automatically.

The CNN model were trained using kaggle kernel, packages used: keras (ver. 2.6.0), numpy (ver. 1.19.5), tensorflow (ver. 2.6.0), pandas (ver. 1.3.3), matplotlib (ver. 3.4.3), sklearn (ver. 0.23.2).

8.2 Confusion matrix

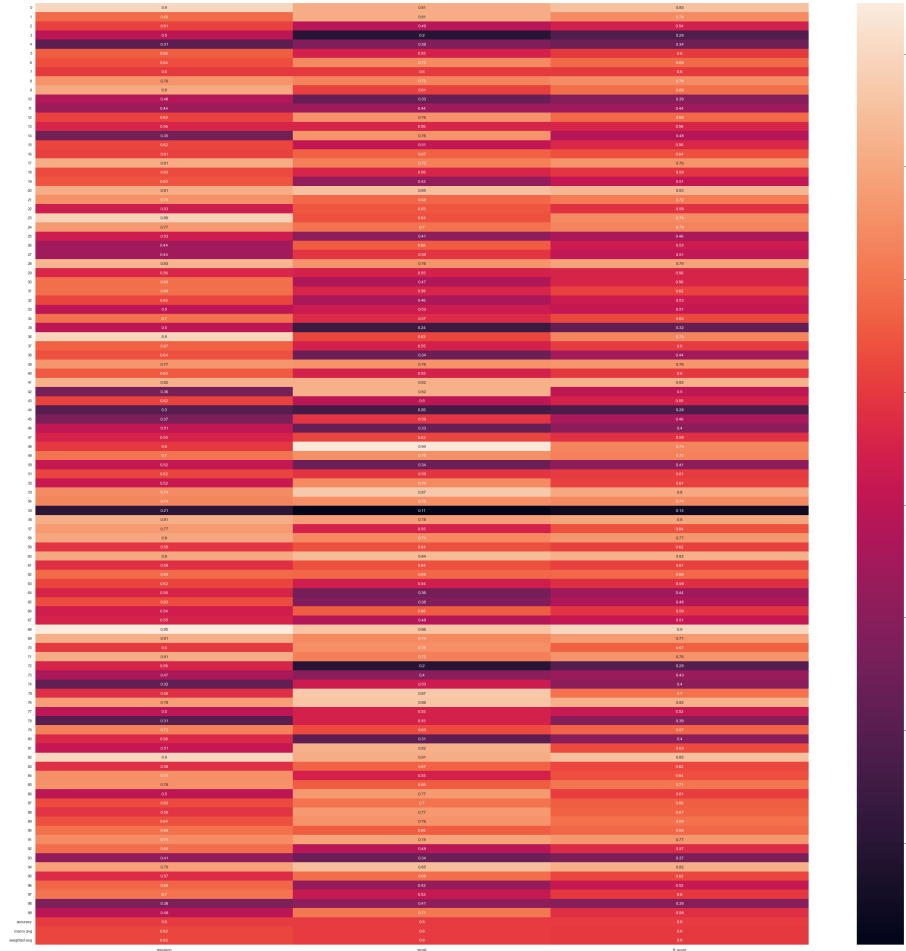


Figure 7: Confusion matrix for CNN

8.3 Code overview

8.3.1 Best algorithm

By running the `group84_best_algorithm1.ipynb`, the model will be created automatically, a h5 file called `group84_pretrained_model.h5` storing the model weight will be generated under the same directory. The last chunk of the code shows the results for all evaluation metrics.

Evaluation section can be run separately (without training the model again), simply place `group84_best_algorithm1.ipynb` and `group84_pretrained_model.h5` under the same directory, and run the validation section in the ipynb file.

The submitted ipynb has result for all chunks, if you wish to run the code again, please make sure run on a copy and not on the original file, since the train process may take up to 4 hours.

8.3.2 Other algorithm

Block 1, 2 and 3:

These parts download the cifar-100 data installer from <https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz>, which will then be unpacked to produce a file that can be read. Then it will be named as the data part that will be used in the code later.

Block 4 and 5:

These two parts are pre-processed data.

Block 6:

This panel uses training data to train the SVM and KNN algorithm model.

Block 7:

This panel is cross-validated using ten folds to pick out the best parameters. The final result of the selection for SVM is that the kernel is chosen to be poly and the degree is 2. For KNN is that the `n_neighbors` is 7, and `p` is 1.

Block 8:

This panel applies the SVM and KNN algorithm obtained from the best parameters of the ten-fold grid search to the test dataset and prints out some of the evaluations.