Assignment 2

Tutors: Dongang Wang, Raghavendra Chalapathy, Seid Miad Zandavi
Group members: Jielong ZHONG (430548048: jzho2448), Zhizhi CHAI (470259267: zcha3804), Ruiqiang LI (460241999: ruli9351)

Abstract

Machine learning (ML) and deep learning (DL) have been widely used to solve different problems across many industries. Particularly, related algorithms like support vector machine (SVM), random forest (RF) and deep convolutional neural networks (CNN) are successfully applied to address problems in computer vision tasks e.g. image detection, segmentation and classifications. In this article, we mainly introduced four architectures that were commonly used for images classification tasks and compare their performance based on images from CIFAR-100 dataset. Specifically, we compared two traditional machine learning algorithms (SVM and RF) against two state-of-the-art convolutional architectures (Wide-ResNets and DenseNets). All these approaches are evaluated by the metrics of precision, recall, accuracy and confusion matrix. Through the experiments, we have found that the CNN architectures outperformed the traditional ML methods in image classification tasks.

1 Introduction

Traditional machine learning (ML) algorithms such as support vector machines were the dominant approaches for classification in images processing tasks [ref]. Although there are constantly emerging new methods related to conventional ML with improvements that can complete various object classifications, they are not capable of primely resolving image classifications once the images become complex or the big set of images have significantly variant labels [ref]. In addition, a highly acceptable classification results always require traditional ML approaches to be implemented with a sound feature engineering on data features which is time-consuming and tedious [ref]. Moreover, the trained models are not well suited for unseen data [ref] which means they have high possibility of wrongly classifying the objects in the images.

To overcome these limitations, researchers, data scientists and engineers increasingly adapt deep learning algorithms (DL), more advanced machine learning approaches, in their classification applications. Unlike conventional ML, DL architectures train the hypothesized model end-to-end without any need of feature engineering [3, 6, 10]. More recently, especially the introduction of deep convolutional neural networks (DCNN) push the performance of images classification tasks to a new level. For example, DCNN architectures like VGG-Net [10], Residual Networks (ResNets) [6] and Inception Networks [3] have competitive abilities in dealing with images object recognition on various dataset e.g. ImageNet, CIFAR-100 and SVHN.

In this article, we aimed to compare four existing methods of their performance on recognizing objects based on CIFAR-100 [1] images dataset. These four methods are split into 2 main different categories: traditional ML algorithms and DL ones. In terms of ML, support vector machine (SVM) and random forest are introduced. For DL algorithms, we intuitively re-implemented Wide Residual Networks (WRNs) [9] and Densely Connected

Convolutional Networks (DenseNets) [4], which are latest DCNN architectures that achieved competitive results on image classification tasks. As being the state-of-the-art DL models, they are more parameter-efficient compared to existing DCNNs as they require fewer parameters for training and this helps train deeper networks with improved information flow over the whole network [4]. All four models were evaluated by precision, recall and confusion matrix as well as overall classification accuracy. Through comparison, we can see how each of the method works for image classification and how DL approaches outperform ML ones.

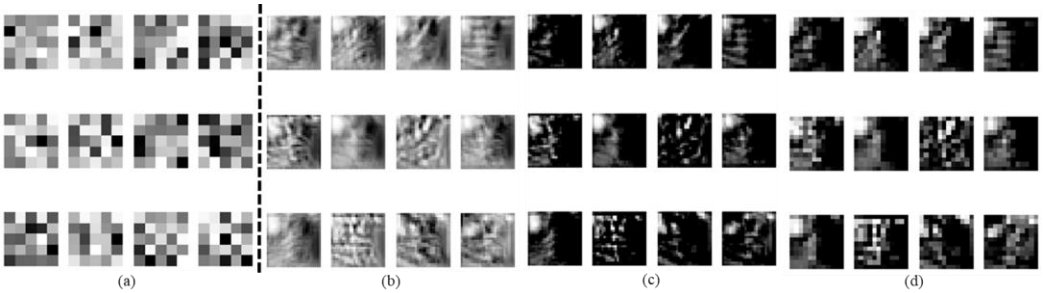

Figure 1. Convolutional Operations. (a) weights matrix used to conv images; (b) – (d) feature maps generated by convolutional operations over images.

2 Related Work

The exploration of object recognition in relevant images dataset like CIFAR-100 using ML and DL methods has been a critical research since their initial introduction. Before the popularity of DL approaches, mots of researchers, data scientists and engineers mainly solve image classification problems focusing on traditional ML methods. For example, K-Nearest Neighbor algorithm (KNN) was used to recognize hand-written digits (0-9) by calculating Euclidean distance between training and test images and then assigning corresponding labels [11]. Berthod et al. proposed a method using Bayesian classifier with Markov Random Field to recognize simple objects such as rectangles, stars and triangles [8]. Also, Bosch et al. tried to identify objects from Caltech-101 and Caltech-256 data sets by using improved Random Forest [3]. More recently, these existing algorithms like KNN, Decision Tree, Linear SVC/SVC and Random Forest were used to classify objects from Fashion-MNIST which was introduced by Xiao et al. [5]. In addition, Netzer et al. utilized K-Means to recognize street numbers by using dataset SVHN and achieved an accuracy of 90.06% with carefully designed handcrafted features [12]. We can see that all these methods applied on different data sets gained a reasonable performance. However, these existing ML methods have limitations in following aspects: (i) they are not computational efficiency once the size of input data become huge as they significantly rely on CPU processing abilities; (ii) the performance of these algorithms will fall once they are used to recognize complicated objects in the images; (iii) Most importantly, we must pay attention to features selection as this tremendously affect the performance of most conventional ML algorithms.

Table 1. Examples of performance on related ML algorithms from previous works based on various datasets.

Dataset	Method	Accuracy	Feat. Eng. Required
MNIST	K-Nearest Neighbors	97.60%	No
Caltech-256	Random Forest	43.50%	Yes
	Decision Tree Classifier	79.80%	Yes
	K-Nearest Neighbors	85.40%	No
Fashion-MNIST	Linear SVC	83.60%	Yes
	Support Vector Classifier	89.70%	Yes
	Random Forest	87.30%	Yes
SVHN	K-Means	90.60%	Yes

The introduction of deep neural networks has effectively resolve above limitations. AlexNet

was proposed by Krizhevsky et al. in 2012 illustrating that deeper neural networks have better performance over image classification [7]. They evaluated the architecture by using ImageNet dataset, which is a large scale of data set over 1 million images with 1,000 categories. This is more complicated than previous mentioned dataset like MNIST. After that, Simonyan and Zisserman introduced a network by stacking 3 * 3 convolutional layers together to gain deeper network while using max-pooling to reduce the size of feature maps in 2015 [19]. This network was also assessed by using ImageNet dataset. In 2015, more state-of-the-art architectures e.g. Inception Networks [3] has layers over 20 layers and ResNets [6] has layers over 100 layers. There are many other networks based on the transformation of these advanced architectures. For example, motivated by Residual Networks, Huang et al. had developed a convolutional neural network that each convolutional layer was densely connected by each other so that maximizing the usefulness of the information flow through networks. This model was tested on CIFAR-100 dataset and gained the best performance of 17.18% error rate. We re-implemented this model as one of the benchmark comparison against other algorithms in our experiments.

Table 2. Examples of state-of-the-art scores evaluated on different datasets using DCNNs.

Dataset	Method	Accuracy	Feat. Eng. Required
MNIST	CNN or CNN-transformed architectures	99.79%	Networks were trained end-to-end without any feature engineering needed.
Caltech-256		74.42%	
CIFAR-100		84.80%	
SVHN		98.70%	

3 Methods

3.1 Preprocessing

Overfitting is a main problem in both ML and DL models. In order to prevent such a problem, we pre-processed the images before inputting them into each training model so that we can minimize the generalized error during evaluation stage.

Grayscale and center-cropping. We transformed all colored training and test images into grayscale images for the purpose of lowering the dimensionality of the input data, reducing channels from 3 (RBG channels) to 1. Doing so can effectively speed up the training procedure as the number of parameters needed for training was decreased. In addition, the image was cropped from size 32 * 32 to 24 * 24 by cutting off 4 pixels for each side as the edge of images provided no useful information. This further caused the reduction of training parameters and accelerated the training speed.

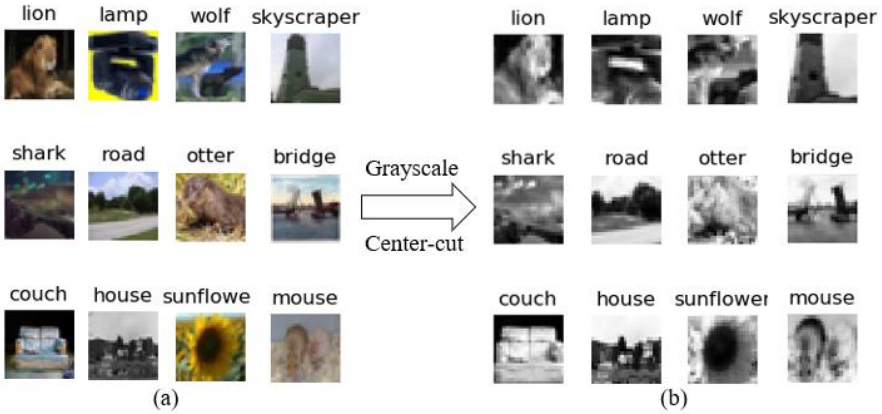

Figure 2. Images from CIFAR-100. (a) the original color images with size 32 *32 pixels; (b) the center-cropping and grayscale images with size 24 * 24 pixels.

Normalization. Apart from transforming the images into grayscale, we also normalized each data sample by subtracting the mean and dividing by the standard deviation so that the pixel values of the images can stay in the same scale and zero-centered. The most important reason

is to ensure that the gradients would not fluctuate dramatically during back-propagation.

$$X = \frac{X - mean}{std}$$

Principle Components Analysis (PCA). PCA is another common technique for pre-processing data so as to reduce feature dimensions, particularly useful for ML models. The data is first centered as mentioned above. Then the covariance matrix was computed that we can know the correlation structures of the data. This procedure is valuable as what features being kept after this operation was according to the most variance. Moreover, applying PCA can effectively and efficiently save us time on both space and training time, especially for training linear. Also, this generally resulted in good performance.

3.2 Traditional Machin Learning Classifiers

3.2.1 Random Forest

The Random Forest is a traditional model combining tree predictors to train classifiers. Each tree is depending on randomly sample vectors which has a similar distribution. The generalization error [17] of the random forest will be affected by the number of trees, the strength and the correlation of each tree. The convergence of random forest is similar to bagging. However, it will have lower generalization error when it has a host of trees. The Random Forest classifier can be described as: $\{h(x, \theta_k), k = 1,2 \dots\}$ where θ_k refers to independently identically distributed random vector [17] and x refers to training sample. Each based leaner $h_k(x)$ in random forest will casts a vote for the most popular class.

Decision tree. The decision tree is a tree structure model consists of one root, some interior node and some lead nodes. Each interior node and lead represent the feature testing and testing results respectively. The goal of decision tree function is to generate a tree with strong generalization.

Gini index. The Gini index are used to find the optimal division feature of samples. It can be derived from: $Gini(D) = \sum_{i=1}^{m} \sum_{k \neq k'} p_k p_{k'}$. Then the optimal division feature is:

$$a = arg\min_{a \in A} \sum_{v=1}^{V} \frac{|D^v|}{D} Gini(D^v)$$

3.2.2 SVM

For the traditional model, we tried the SVM by using the scikit-learn library. Basically, SVM are used to find the hyperplane to separate the sample data. Assume the hyperplane already can divide the sample data. The model is described below in figure (a). Those sample points on the hyperplane boundary are the support vectors. The margin γ refers to distance between two boundaries. As a result, finding such hyperplane equals to find the optimal results from:

$$\min_{w,b} \frac{1}{2} \|w\|^2 \ , s.t. \, y_i(w^T x_i + b) \geq 1, i = 1,2, \dots m.$$

Since this question is difficult to solve, it was converted to the dual problem by using the Lagrange multiplier method:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_i {x_i}^T x_j \ , s.t. \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1,2, \dots, m.$$

Moreover, the optimal result α should satisfy the Karush-Kuhn-Tucker (KKT) condition:

$$\begin{cases} \alpha_i \geq 0 \\ y_i f(x_i) - 1 \geq 0 \\ \alpha_i (y_i f(x_i) - 1) = 0 \end{cases}$$

We choose the polynomial kernel: $\kappa(x_i, x_j) = ({x_i}^T x_j)^d$ as our kernel function. The kernel function is to map the previous separation plane to a high-dimensional hyperplane. After

solving, the final model can be expressed as:

$$f(x) = w^T\Phi(x) + b = \sum_{i=1}^{m} \alpha_i y_j \kappa(x_i, x) + b$$

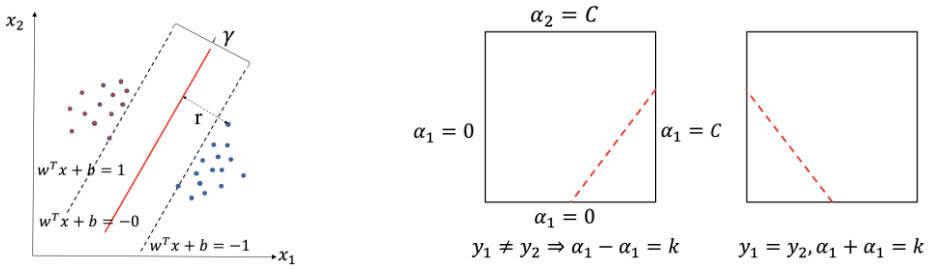

Figure 3. (a) Hyperplane from SVM; (b) conditions of α that affects hyperplane

The Sequential Minimal Optimization. The algorithm to speed up the training process of SVM classifier. Basically, it randomly chooses two variables α_1, α_2 and fixes other variables [14]. Then the sub problem of the previous optimization problem can be described as:

$$\arg\min_{\alpha_1,\alpha_2} W(\alpha_1, \alpha_2) = \frac{1}{2}\kappa(x_1, x_1){\alpha_1}^2 + \frac{1}{2}\kappa(x_2, x_2){\alpha_2}^2 + \kappa(x_1, x_2)y_1y_2\alpha_1\alpha_2 - (\alpha_1 + \alpha_2) + y_1\alpha_1\sum_{i=3}^{m}\alpha_i y_i\kappa(x_i, x_1) + y_2\alpha_2\sum_{i=3}^{m}\alpha_i y_i\kappa(x_i, x_2)$$

$$s.t. \quad \alpha_1 y_1 + \alpha_1 y_2 = -\sum_{i=3}^{m}\alpha_i y_i = \varsigma\ , \qquad 0 \le \alpha_i \le C$$

The variable ς and C are two constant terms. This step is to convert the two variables quadratic programming problem to the single variable optimization problem [15]. The previous restriction can be described by figure (b). Assuming that the initial feasible solution is: ${\alpha_1}^{old}, {\alpha_2}^{old}$ and the optimal solution after optimization is: ${\alpha_1}^{new}, {\alpha_2}^{new}$. Assuming that we take the α_2 as the optimization direction. Then the unclipped result of α_2 can be derived from:

$${\alpha_2}^{new,unclipped} = {\alpha_2}^{old} + \frac{y_2(E_1 - E_2)}{\eta},\ \eta = \kappa(x_1, x_1) + \kappa(x_2, x_2) - 2\kappa(x_1, x_2).$$

According to conditions above, the optimal result ${\alpha_2}^{new}$ should be clipped using the constraining direction: $L \le {\alpha_2}^{new} \le H$ where:

$$\begin{cases} L = \max(0, {\alpha_2}^{old} - {\alpha_1}^{old}) \quad H = \min(C, C + {\alpha_2}^{old} - {\alpha_1}^{old}) & if \quad y_1 \neq y_2 \\ L = \max(0, {\alpha_2}^{old} + {\alpha_1}^{old} - C) \quad H = \min(C, {\alpha_2}^{old} + {\alpha_1}^{old}) & if \quad y_1 = y_2 \end{cases}$$

After getting the ${\alpha_2}^{new}$, the optimal result of α_1 can be calculated by: ${\alpha_1}^{new} = {\alpha_1}^{old} + y_1y_2({\alpha_2}^{old} - {\alpha_2}^{new})$. After optimization, we can use α to make predictions.

3.3 Deep Learning Architecture

3.3.1 Wide Residual Networks

WRNs is a modern DCNNs structure motivated by [6] and it has more convolutional layers per block, denoted as widened convolutional layers with more features planes and larger filter sizes in convolutional layer. All these properties can increase the representational power of residual blocks [9]. This network shows that the performance is not good with the layer increasing. The network has widened convolutional layers, and the result of this network is better than the thin convolutional layers network, because the residual block in the widened convolutional layers can learn more during training. And dropout is introduced into the residual block first time, which is used to disable the residual block during the block.

This method increasing more useful representations learned by residual block. For our dataset, the WRNs has 3 layers containing Residual blocks with each having 4 bottleneck layers (1×1 conv-3×3 conv-1×1 conv). Each residual block was followed by an average pooling layer so that the size of feature map can be decreased. Noted that the introduction of bottleneck layers is to reduce computation expense.

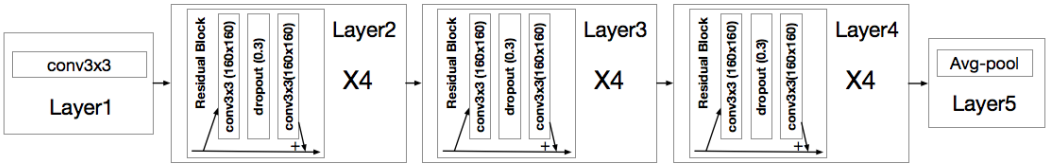

Figure 4. Wide Residual Networks with 3 residual blocks.

3.3.2 Densely Connected Convolutional Networks

DenseNets is a state-of-the-art DCNNs architecture evaluated under many benchmark data sets like CIFAR-100. Traditional DCNNs comprise stacking convolutional layers followed by non-linearity functions e.g. ReLU or Tanh. There are many problems associated with such a structure. For example, it is easy to be overfitting and requires a huge amount of parameters to be trained when the networks go deeper. However, DenseNets overcomes such problems by conveying the information flow through densely connected layers [4]. This technique farthest keeps useful features. We re-implemented it taking advantage of Keras and modified the number of dense blocks so as to make it suitable for our dataset. The whole networks can be composite of dense block, transition layers and bottleneck layers. A 3*3 convolution operation with 16 filters channel was performed on the input images before entering the first dense block.

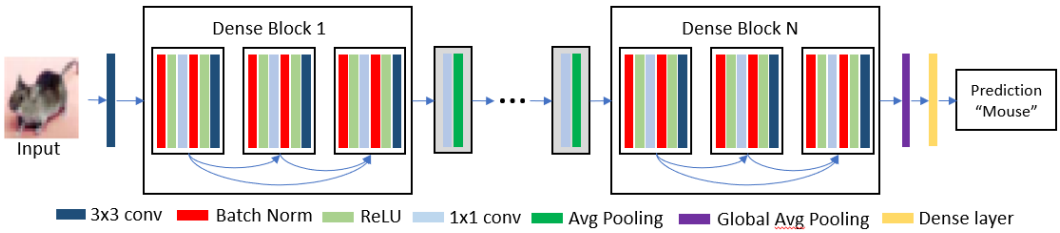

Figure 5. Overall structure of Densely Connected Convolution Networks. For our dataset, we have three dense blocks for the whole network with transition layer between dense blocks.

Dense Block. The basic components in each dense block are 1×1 convolution and 3×3 convolution. However, it should be noted that 1×1 convolution should be before 3×3 convolution as it has been showed that this is a technique that effectively reduce the number of feature maps and was described as bottleneck layer [3, 6]. Between convolutions, batch normalization (BN) and ReLU are performed. The structure of such a component: BN-ReLU-1×1 conv-BN-ReLU-3×3 conv. One dense block consists of many above components with a dense connection. This pattern illustrates that the input of current layer is the concatenation of all feature maps from previous layers (arrows in Figure 5 describes the connected directions):

$$x_c = f_c([x_0, x_1, x_2, \dots, x_{c-1}])$$

Where x_c is the current input and f_c is the non-linear function performed on these input feature-maps.

Transition Layer. One of important part in DCNNs is to down-sample the feature-maps by changing their sizes. Down-sampling layer was denoted as transition layer in original paper [4] which has 1×1 convolution followed by a 2×2 average-pooling layer. Transition layer is the part between dense blocks.

There is a global average-pooling layer at the end of last dense block and a dense layer before output. The softmax function was applied to final feature matrix to produce prediction results.

$$\text{Softmax: } f(x_i) = \frac{\exp(x_i)}{\sum_j^k \exp(x_j)}$$

Where k represents the number of classes and j is the j^{th} category.

4 Experiments and Results

4.1 Dataset

CIFAR-100 [1] has been regarded as a competitive benchmark dataset which is widely used to evaluate different ML and DL models. CIFAR-100 consists of total 60,000 colored natural images with size of 32 * 32 pixels. The whole dataset was split into two batches with 50,000 training samples and 10,000 test images. There are 100 classes for all these images with 600 images each. Data augmentation such as flipping, shifting and mirroring was applied to input data.

4.2 Training (Implementation Details)

We empirically compared two DCNNs to SVM and Radom Forest by setting up variant hyper-parameters for them.

Random Forest. When implementing the random forest model, we first tune the number of tree estimator and the depth of each tree. The result shows that although the test accuracy is growing steadily, the training time is expected to become longer. When the number of trees exceeds 1000, the correct rate has already tended to converge. Due to ensemble learning, RF is hard to be prone to overfitting. Therefore, we did not implement *10-fold cross validation*.

SVM. There are many alternative hyper parameters when implement the SVM. We tried two kernel functions: polynomial kernel and Radial Basis Function (RBF) kernel. Although polynomial function will gain a shorter training time, the corresponding test accuracy are lower than RBF kernel. Also, we tried the range from 2 to 150 for the number of reduction components when implement the PCA and decide to use 60 since it gains a best performance for RBF kernel. We also tuned and set the margin constant C as 10 since when C equals to 10, both training time and test performance will become better. Finally, in order to prevent overfitting, *10-fold cross validation* was implemented while training SVM

WRNs. In the experiment of Wide Residual Networks, the network has been trained through 150 epochs and the average running time for each epoch was 90 seconds. The total running time for all process was almost 4 hours. The initial learning rate is 0.1. when epochs are more than 60, the learning rate was decayed to 0.02. when epochs are more than 120, the learning rate was changed to 0.004. The optimizer for this network was SGD with the loss function using cross entropy loss function. The network has a depth of 28, and the dropout [13] rate was defined as 0.3 through all processes to randomly drop some neurons out so as to prevent overfitting. Widen factor for the network is 10 and class number was set to 100.

DenseNets. We have changed the optimizer from SGD to Adam compared to original method mentioned in [4]. The initial learning rate of the networks was set to 1e-3 and this was decayed to 1e-4 and 1e-5 respectively when the training epoch reached 100 and 150. The loss function used in this architecture is the same as WRNs, categorical cross entropy. The whole model was trained with a batch size of 64 for 200 epochs with 59s per epoch and a dropout [13] rate with 0.2. The filters were initially set to 16 and this increased with a growth rate of 12. In addition, we applied l2 penalty (1e-4) to the batch normalization. In order to improve performance, we also used data augmentation (horizontal flip, image shift, rotation) in our dataset. This action helped making the model more robust and preventing overfitting. DenseNets trained with augmentation was denoted as **C100+**.

Hardware Specifications. All experiments had been evaluated and run on a computer with hardware and software specifications as shown in Table xxx. In our experiments, ML related algorithms were simply run by using CPU while DCNNs were run by taking advantage of computation power from graphic cards (GPU) so as to accelerate training speed. This is because CPU is unable to train the deep networks compared to GPU. However, it should be

noticed that running these models on lower or higher specifications of different computers might result in slight accuracy divergence but it could lead to dramatic difference in time-efficiency.

Table 3. Hardware and Software Specifications.

Items	Specifications
Computer Model	GIGABYTE Z370 AORUS ULTRA GAMING 2.0
Processor	Intel Core i7-8700K CPU @ 3.70 GHz
Running Memory	4 * 8GB CORSAIR 3000 MHz LDDR4
Graphic Cards	EVGA GeForece GTX 1080 Ti
Operating System	Microsoft Windows 10 Home 64-bit
Developing IDE	JetBrains PyCharm & Jupyter Notebook

4.4 Results on Cifar-100

Precision, Recall and Accuracy. Precision, recall and accuracy have been commonly used in measuring the performance of classification tasks. Precision is the proportion of items classified correctly over the total items assigned to that category. Recall is the proportion of items assigned to that category over the total items in that category. Accuracy is the total items classified correctly over the all samples. Table 4 shows the results for each measurement index.

$$\text{Precision} = True\ Positives/(True\ Positives + False\ Positives)$$
$$\text{Recall} = True\ Positives/(True\ Positives + False\ Negatives)$$
$$\text{Accuracy} = correct\ items/total\ items$$

Table 4. Scores for precision, recall and accuracy as well as running time for each classifier.

Methods	Precision	Recall	Accuracy	Training Time
SVM + PCA	30.01	30.10	29.54	5 hours
Random Forest + PCA	25.45	26.53	25.45	1.2 hours
Wide Residual Network	75.82	75.64	75.00	4 hours
DenseNet	55.32	57.59	55.32	7.25 hours
DenseNet (C100+)	66.64	64.24	64.24	2.6 hours

Confusion Matrix. This technique is used to describes the performance of a classification model. The diagonal entries of the matrix are the correctly assigned labels while entries out of the diagonal are those labels wrongly classified. The matrix entries of diagonal were denoted as red color if the number of classification results in that category did not surpass threshold value with 50. Also, entries out of diagonal marked with red color means the number of misclassified items exceeded 5 as this indicates poor classification ability. The following tables show part (first 10 categories) of classification results from each classifier.

Table 5. confusion matrix from SVM

	apple	aquarium_fish	baby	bear	beaver	bed	bee	beetle	bicycle	bottle
apple	58	2	2	0	1	0	0	1	0	0
aquarium_fish	1	49	1	1	0	0	0	0	0	0
baby	0	3	20	3	1	1	1	0	4	1
bear	0	0	0	17	2	0	3	0	0	1
beaver	0	0	0	7	20	1	0	2	0	0
bed	0	1	3	1	2	28	0	1	3	1
bee	1	0	1	0	1	0	25	5	2	0
beetle	0	1	0	4	3	0	5	35	0	0
bicycle	0	0	1	0	1	0	0	2	29	1
bottle	0	0	0	0	0	0	0	0	0	45

Table 6. confusion matrix from Random Forest.

	apple	aquarium_fish	baby	bear	beaver	bed	bee	beetle	bicycle	bottle
apple	50	2	2	0	0	0	1	1	0	1
aquarium_fish	3	47	1	0	0	1	1	1	2	0
baby	0	0	14	0	0	0	1	0	3	2
bear	0	0	0	8	5	1	1	0	0	1
beaver	0	0	0	5	15	1	1	0	3	1
bed	1	0	3	0	0	14	0	1	0	0
bee	1	0	1	2	0	0	18	1	0	0
beetle	0	0	0	11	3	0	6	33	0	0
bicycle	0	0	1	0	0	0	0	0	19	0
bottle	0	0	0	1	0	0	0	0	1	32

Table 7. confusion matrix from WRNs.

Category	apple	aquarium_fish	baby	bear	beaver	bed	bee	beetle	bicycle	bottle
apple	85	0	0	0	0	1	0	0	0	0
aquarium_fish	0	89	0	0	0	0	0	0	0	0
baby	0	0	64	0	0	0	0	0	0	0
bear	0	0	0	65	7	0	0	0	0	0
beaver	0	0	0	1	70	0	0	0	0	0
bed	0	0	0	0	0	78	0	0	0	0
bee	0	0	0	0	0	0	82	2	0	0
beetle	1	0	0	0	0	0	2	75	0	0
bicycle	0	0	0	0	0	0	0	0	91	0
bottle	0	0	0	0	0	0	0	0	0	81

Table 8. confusion matrix from DenseNets

	apple	aquarium_fish	baby	bear	beaver	bed	bee	beetle	bicycle	bottle
apple	79	0	1	0	1	0	0	0	0	0
aquarium_fish	0	57	0	0	1	0	0	0	0	0
baby	1	1	51	0	1	1	0	0	0	0
bear	1	0	2	42	6	0	0	0	0	0
beaver	0	0	0	2	45	0	1	0	0	0
bed	0	1	1	0	0	62	0	0	0	0
bee	0	0	0	1	0	0	67	1	0	1
beetle	0	0	0	2	2	0	8	28	2	0
bicycle	0	0	1	0	0	0	2	0	75	0
bottle	0	0	0	0	0	3	0	0	0	66

Table 9. confusion matrix from DenseNets with data augmentation

	apple	aquarium_fish	baby	bear	beaver	bed	bee	beetle	bicycle	bottle
apple	84	1	0	0	0	0	0	0	0	0
aquarium_fish	0	77	0	0	0	0	0	0	0	0
baby	0	0	71	0	1	1	0	0	1	0
bear	0	0	0	21	4	0	1	0	0	0
beaver	0	0	0	0	34	0	0	0	0	0
bed	0	0	0	0	0	71	0	0	1	0
bee	0	0	0	0	0	0	66	1	0	0
beetle	1	0	1	0	0	0	1	54	0	0
bicycle	0	0	0	0	0	0	0	0	89	0
bottle	0	0	0	0	0	0	0	0	0	67

4.5 Discussion

Apparently, DenseNet and WRNs outperformed SVM and RF through experiment comparison. We can see that WRNs and DenseNets have higher scores of precision, recall and accuracy than that of SVM and RF. Such difference proves that DCNNs architectures have more learning power as well as having excellent representation capabilities. Besides, this difference is not only described in precision, recall and accuracy, but also showing in the confusion matrix. From that, it was showed SVM and RF have poor classification ability. Nevertheless, an interesting thing was that DCNNs could also result in bad classification for categories e.g. bear, beaver and beetle results in DenseNets. With regards to ML methods, we found that if we did not reduce feature dimensions, it is extremely time-consuming to train SVM and RF relying on only CPU, while this situation is moderate when training DenseNets and WRNs with the help of GPU computational power. However, even if we used GPU to accelerate training speed, it can be seen that the models still required much time to do that (4 hours for WRNs and 2.5 hours for DenseNets). This is because they have complicated structures and they have a large number of parameters needed for training.
On the other hand, we have found that feature selection process is important to conventional ML algorithms. If we cannot remove useless features properly by taking advantage of technique like PCA, the performance of these algorithms would have even worst results than those when they are implemented with PCA (see Table 4). But for DCNNs, they need no feature selections as they were trained end-to-end. For DCNNs, we can see that data augmentation is also important during training model. From Table 8 and Table 9, we can see that the model with augmentation has a better results than that of no augmentation.
Although DCNNs architecture has strong capabilities to describe features on the images, CIFAR-100 is still a challenging dataset for classification tasks. As figures shown in Table 4, we can know that even though we re-implemented the state-of-the-art architectures to evaluate it, we did not get the same performance compared to original performance in [4, 9]. Resulting in such difference could be due to the distinct data processing methods and hyper-parameters setting.

5 Conclusion and Future Work

In conclusion, we can see that DCNN architectures, WRNs and DenseNet, outperformed SVM and Random Forest algorithms with quite competitive results through experiment studies. This indicates that convolution operation embedded in these models has more powerful abilities in tackling image classification tasks especially when the dataset is more complex. We noted that this is because the convolution is more powerful at capturing useful semantic information of the images than traditional ML approaches.
Although we have achieved competitive performance on CIFAR-100, we believe that further gains in accuracy of WRNs and DenseNet can be gained by more detailed hyper-parameters tuning and data pre-processing. Furthermore, choosing more latest or state-of-the-art DCNN architectures can also help us to gain higher evaluation accuracy based on CIFAR-100. Certainly, if we can do more feature engineering to extract useful features, then the performance of SVM and random forest might also achieve better results.

References

[1] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images”, Master’s Thesis, Department of Computer Science, University of Toronto, 2009.

[2] A. Bosch, et al., “Image Classification Using Random Forest and Ferns”, *2007 IEEE 11th International Conference on Computer Vision*, pp. 1-8, 2007.

[3] C. Szegedy, et al., “Going Deeper with Convolutions”, *CVPR 2015*, arXiv:1409.4842, 2015.

[4] G. Huang, Z. Liu & L.V.D. Maaten, “Densely Connected Convolutional Networks”, *CVPR 2017*, 2017, arXiv:1608.06993.

[5] H. Xiao, K. Rasul and R. Vollgraf, “Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms”, *CVPR*, arXiv:1708.07747.

[6] He, K., Zhang, X., Ren, S., Sun, J., “Deep Residual Learning for Image Recognition”, *2016 IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.

[7] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *NIPS*, vol. 60, no. 6, pp.84-90, 2012.

[8] M. Berthod et al., "Bayesian Image Classification using Markov Random Fields", Image Vision Computing, vol. 14, pp. 285-295, 1996.

[9] S. Zagoruuyko and N. Komodakis, "Wide Residual Network", *CVPR*, arXiv:1605.07146, 2016.

[10] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *Computer Vision and Pattern Recognition*, arXiv:1409.1556, 2014.

[11] Y. LeCun, et al., "Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition", *The Statistical Mechanics Perspective*, pp. 261-276, 1995.

[12] Y. Netzer, et al., "Reading Digits in Natural Images with Unsupervised Feature Learning", *Neural Information Processing Systems*, 2011.

[13] N. Srivastava, et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *JMLR*, 2014.

[14] J.C. Platt, "12 fast training of support vector machines using sequential minimal optimization", *Advances in Kernel Methods*, pp. 185-208, 1999.

[15] J.C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines".

[16] I. Jolliffe, "Principal Component Analysis", *International Encyclopedia of Statistical Science*, pp. 1094-1096, 2011.

[17] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.