Classifying CIFAR-10 Images Using Unsupervised Feature & Ensemble Learning

Truc Viet "Joe" Le
Heinz College
Carnegie Mellon University
tjle@andrew.cmu.edu

Naassih Gopee
School of Computer Science
Carnegie Mellon University
ngopee@andrew.cmu.edu

ABSTRACT

We perform the image classification task on the CIFAR-10 dataset, where each image belongs to one of the ten distinct classes. The classes are mutually exclusive and are mostly objects and animals. The images are small (32×32 pixels), of uniform size and shape, and RGB coloured. We implement an proposed image preprocessing framework to learn and extract the salient features of the images. The method was demonstrated to increase the classification performance significantly. We testify such claim and see a considerable improvement of more than 15% from the baseline (i.e., without preprocessing). We further experiment with various parameters and settings of the proposed method to tune the preprocessing frameworks. We also experiment with a variety of linear classifiers on the preprocessed images. We find out that a simple SVM classifier with linear kernel performs the best. We finally experiment with ensemble learning by combining a linear SVM with a multinomial logistic regression. The ensemble learning marginally improves on the simple linear SVM at a high computational cost.

1. INTRODUCTION

In this project, our task is to classify the 15,000 colour images randomly extracted from the CIFAR-10 dataset [3]. The CIFAR-10 dataset consists of 60,000 "tiny" colour (RGB) images, where each is of dimension 32×32 pixels. These images can be classified into 10 classes of equal size, where each contains 6,000 images and the classes correspond to mostly objects (such as ships, cars, airplanes, etc.) and animals (cats, dogs, frogs, etc.). A hundred random sample images from the dataset are shown in Fig. 1 together with their correct classifications. These 10 classes are completely mutually exclusive, e.g., there is no overlapping between automobiles and trucks. Hence, the goal of the project is to label the 15,000 test images with their correct classes.

To this end, we are provided with a training set of 4,000 random images from the same CIFAR-10 dataset (that are non-overlapping with those in the test set) to develop a predictive model for the classification. We reimplement the image preprocessing method proposed by Coates *et al.* [1] with some minor modifications to improve the classification. The preprocessing relies two separate sequential phases: feature learning and feature extraction. Feature learning extracts random patches from the training images, where each patch is a smaller sub-image of a predefined dimension, and learns the features from the extracted patches using an unsupervised clustering algorithm such as K-means or GMM. Thus, each feature corresponds to the centroid of each cluster. Feature extraction then systematically extracts overlapping patches (convolutional extraction) from each image in the entire dataset. Each extracted patch is re-represented using the learned features (centroids) through a non-linear mapping function. Finally, each image is reconstructed by collaging their patches and "pooling" (or summing) those collaged patches together over the 4 quadrants (divisions) of the image. The paper claims to have obtained correct classification of over 79% using such preprocessing and a simple final linear classifier (e.g., SVM with linear kernel).

In our implementation of the proposed preprocessing framework, we slightly modify the pooling procedure from over 4 quadrants to over 9 divisions. This has increased the prediction accuracy by 1.5% without incurring much additional computational complexity. We also experiment with several parameters of the preprocessing (i.e., the number of random patches to extract for feature learning, the dimension of each patch, and the number of clustering centroids) and

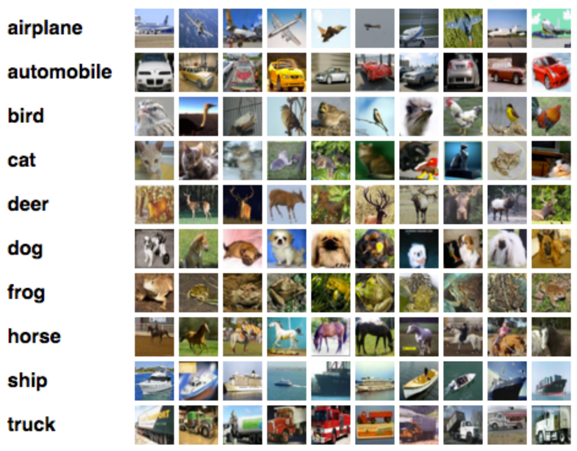

Figure 1: Sample images from the CIFAR-10 dataset and their correct classifications [3].

select the best set of parameters for our classification. For classification, we additionally experiment with a variety of linear classifiers on the preprocessed data: linear SVM, naive Bayes, multinomial logistic regression (MLR), and random forest. Finally, we propose an ensemble learning approach by combing our two best classifiers: linear SVM and MLR. Our final prediction accuracy on the test set is over 57% (and almost 58%) with ensemble learning, which is a marginal improvement over a simple linear SVM classifier.

2. BACKGROUND

The current state-of-the-art classification on the CIFAR-10 dataset has attained more than 90% performance using deep learning techniques [5, 2]. Convolutional neural networks have also shown performance near that of human's manual classification in the high 80% and reaching 90% [4]. However, those methods often make use of heavy-duty machines with expensive hardware (e.g., a dedicated GPU) and requires a very long learning time, in the matter of days [4]. Therefore, in this project, we are not seeking performance or methods that are comparable to those of deep learning and convolutional neural networks[1] due to limited resources. We are rather looking for an approach that is relatively simple to implement, that can be run on a laptop computer in the matter of hours, and at the same time gives results that are significantly better than a naive approach (which could be thought of as running a straightforward linear classifier on the raw data or just random guessing).

To this end, we propose to implement and experiment with the method proposed by Coates *et al.* [1], which we deem simple and implementable. The approach leverages the bulk of its work on data preprocessing to learn and extract image features, which is not typically computationally expensive. Yet, it was claimed to have attained an impressive performance of 79.6% by running a simple SVM classifier on the preprocessed data. Through this, we also wish to test if more sophisticated classifiers (e.g., ensemble learning) could be used to improve on a simple linear classifier.

3. UNSUPERVISED FEATURE LEARNING & EXTRACTION

Here, we implement the preprocessing framework proposed by Coates *et al.* [1] that extracts and learns the image features in an unsupervised fashion. We also customise the proposed framework to slightly improve the classification accuracy without adding much computational complexity.

The framework consists of two main components: the (unsupervised) feature learning and the feature extraction component. Fig. 2 illustrates the high-level pipeline of the preprocessing framework. The feature learning component learns features from random patches extracted from the training images. The feature extraction component systematically extracts overlapping patches (convolutional extraction) from each image in the entire dataset to re-represent them in a sparser feature space, which in turn makes training a linear classifier more effective.

[1]A resourceful website maintained by Rodrigo Benenson that tracks state-of-the-art classification results on the CIFAR-10 dataset and others is at: `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`.

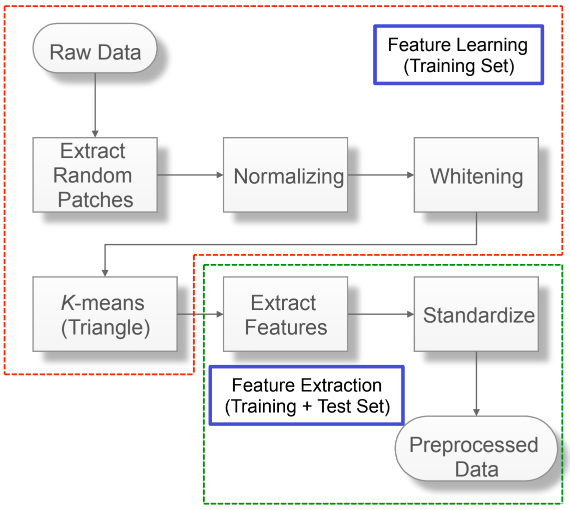

Figure 2: Steps performed by the preprocessing framework to learn and extract image features.

3.1 Feature Learning

For each image in the training set, we randomly extract P unique patches that may or may not be overlapping. Each patch has dimension $w \times w$ and d channels ($d = 3$ for RGB) where w is the *receptive field* (RF) size. Each patch is thus represented by a $\mathbb{R}^N$ ($N = w \times w \times d$) dimensional vector. We then normalise each patch by subtracting from the mean and dividing by its standard deviation, which helps adjust the local brightness and normalise the contrast.

Coates *et al.* [1] suggest that further *whitening* the patches would improve accuracy. Image whitening is basically a decorrelation transformation that transforms each patch into a matrix whose covariance matrix is the identity matrix. The rationale of whitening is that the raw input is usually highly redundant because adjacent pixels are typically highly correlated. Thus, the goal of whitening is to make the pixels less correlated and have the same variance. According the authors, whitening improves performance because clustering algorithms are blind to correlations. Therefore, we also make use of image whitening in our framework.

For unsupervised learning on the random patches, Coates *et al.* [1] concluded that K-means clustering is the best option (compare to alternatives such as GMM). This is because not only is it much faster, it also produces higher accuracy through their experimented cross-validation. Hence, we also adopt K-means clustering in our preprocessing to learn the features from the random patches.

We perform K-means clustering on the set of randomly extracted patches using a specified K. We then represent each patch using the following non-linear mapping (refer to as "K-means (triangle)" or "soft K-means" in [1]). Given the K centroids C_i ($1 \leq i \leq K$) of the K clusters, we map each input patch x into a K-dimensional vector $f : \mathbb{R}^N \mapsto \mathbb{R}^K$ using the following non-linear function:

$$f_i(x) = \max\{0, \mu(z) - z_i\}, \qquad (1)$$

where $z_i = ||x - C_i||^2$ and $\mu(z)$ is the mean of vector z.

Fig. 3 shows the visualisation of the $K = 400$ centroids learned from the training set with $P = 50$ random patches per image. Because our training set has 4,000 images, there

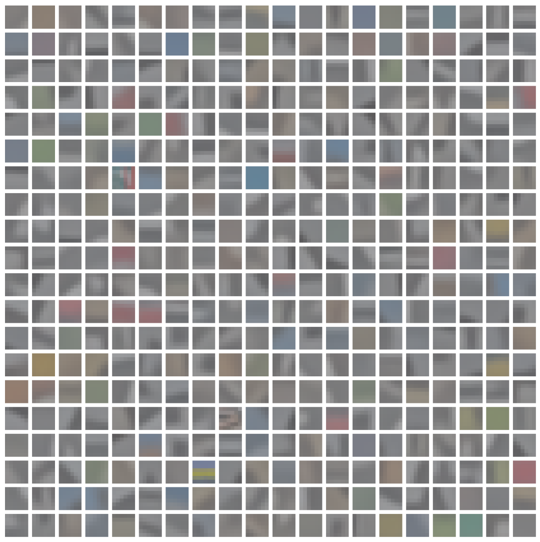

Figure 3: Visualisation of the 400 centroids of the K-means clustering performed on 200000 random patches extracted from the training images.

are 200,000 random patches in total. It shows that the learned centroids correspond to certain characteristic features of the images such as background colours (red, blue, green, yellow, etc.) and separating edges or corners.

3.2 Feature Extraction

For each image in the training and test set, we perform convolutional extraction by extracting $(n-w+1)^2$ patches, where each patch is of dimension $w \times w$ and $n = 32$ is the original dimension of the input image. Convolutional extraction systematically extracts overlapping patches such that any two adjacent patches (horizontally or vertically) have $(w-1) \times w$ pixels in common (i.e., they are different by w pixels horizontally or vertically). For each extracted pixel, we represent it in the K-dimensional feature space via the non-linear mapping described by Eqn. (1) above.

We then re-represent each image by collaging their extracted (overlapping) patches as described in Fig. 1 of [1]. That is, for such collaged image, Coates *et al.* [1] pooled over the 4 equal quadrants of the image to form the final feature vector representation, where "pooling" means taking the sum of all the elements in each quadrant. This is achieved diving the image into four equal parts (or quadrants) and then summing up the patches in each part. Pooling helps reduce the dimensionality of the final image representation and makes classification more efficient and effective. We call the number of such divisions Q. After experimenting with pooling, our experiments showed that by increasing the number of divisions to $Q = 9$ (from $Q = 4$), we gain about 1.5% in classification accuracy. We thus use the 9-division pooling in our subsequent experiments.

The final number of features of each re-represented image is $9K$, where K is the number of centroids of the K-means clustering. Finally, we standardise all the feature vectors in the training and test set separately, which is a common practice in object recognition.

4. CLASSIFICATIONS

In this section, we vary the parameters P, Q and K and report the complexity and performance (i.e., prediction accuracy) tradeoff. We also test several simple linear classifiers on the preprocessed images and report their cross-validation performances. We finally propose ensemble learning models that marginally improve the performance by combining certain simple linear classifiers together.

4.1 Feature Learning Complexity

Table 1 shows the feature learning complexity vs. performance tradeoff, where the performance was recorded by classifying the test data using a simple L2 linear SVM model with cost $\lambda = 300$. The complexity measures the time (in seconds) required to run each sub-component (i.e., each "box") in the preprocessing pipeline (depicted in Fig. 2) as well as the whole pipeline ("Total"). The following machine specifications were used to run these experiments:

- Processor: 2.3 GHz Intel Core i7
- Memory: 8 GB 1600 MHz DDR3
- Operating system: Mac OS X
- Development environment: Preprocessing – MATLAB, Cross-validation Classification – R.

In Table 1, "P. Ext" stands for patch extraction time, "Norm." stands for normalisation time, "F. Ext." stands for feature extraction time, and "Std." stands for standardisation time. Table 1 suggests that RF size of $w = 6$ is just right for the classification task, and making it larger doesn't really help. This is also what was concluded by Coates *et al.* [1]. We get the best accuracy when the $K = 1600$ and $P = 400000$. We finally see that increasing Q from 4 to 9 increases the performance by almost 1.5% while the corresponding increase in complexity is rather marginal. Thus, for the subsequent experiments, we use the following learning parameters: $w = 6, P = 400000, Q = 9$, and $K = 1600$.

4.2 Cross-validation Training

To select an appropriate classifier, we perform a 5-fold cross-validation on the preprocessed training images. We use the following classifiers to benchmark the performance of the cross-validation (CV) training.

1. **L2 SVM**: a simple SVM model using linear kernel[2] and L2 regularisation with cost $\lambda = 400$[3];
2. **RF ($n = 500$)**: a random forest with 500 trees;
3. **NB**: a simple naive Bayes classifier;
4. **Multinomial LR**: a simple multinomial logistic regression (without kernel) with cost $\lambda = 400$.

The results of the cross-validation averaged over 5 folds are shown in Table 2. The averaged prediction accuracy is shown in the second column and the training and prediction

[2]Because the preprocessed images already have a large number of features, we intentionally do not use any kernels for the SVM. Our experiments also show that SVM without kernel outperforms that with kernel.

[3]We also experimented with various values of λ ranging from $\lambda = 300$ to $\lambda = 500$ and found out that $\lambda = 400$ gives the best performance.

RF Size (w)	K	P	Q	P. Ext.	Norm.	Whitening	K-means	F. Ext.	Std.	Total	Accuracy
6	1,600	400K	4	155.19	1.28	0.84	274.70	127.91	1.10	561.02	54.71%
6	1,600	400K	9	162.40	2.70	1.20	262.39	124.68	2.73	556.10	56.20%
6	500	200K	9	81.58	0.44	0.44	44.69	46.09	0.34	173.57	54.13%
8	1,600	400K	9	160.98	6.78	1.77	356.05	142.17	4.56	672.31	53.75%

Table 1: Time complexity and performance tradeoff in choosing learning parameters.

time totalled over 5 folds (measured in seconds) are shown in the third column. Table 2 shows that simple L2 SVM performs the best with more than 50% accuracy, which is followed by the Multinomial LR model. On the other hand, the two most time-consuming methods (to train and predict) are RF and Multinomial LR, respectively.

Methods	5-Fold CV	Training & Pred.
L2 SVM	50.75%	922.85
RF ($n = 500$)	44.35%	6,104.64
NB	39.27%	605.96
Multinomial LR	45.52%	5,614.79

Table 2: Cross-validation results on the training set.

For all these classifiers, the three most commonly confused classes are $(4, 6)$ or (cats, dogs), $(1, 9)$ or (airplanes, ships), and $(2, 10)$ or (cars, trucks) (both ways each).

4.3 Ensemble Learning

We experiment with ensemble learning by combining the individual classifiers in Sect. 4.2. We particularly use the following three classifiers: L2 SVM, NB, and MLR (multinomial logistic regression). We decided not to use RF because from our experiments, RF is very time-consuming and yet slightly worse than MLR. The ensemble learning was combined using a simple multinomial logistic regression without regularisation ($\lambda = 0$). Table 3 shows the averaged results of the 5-fold CV ensemble learning.

Methods	5-Fold CV	Training & Pred.
L2 SVM + NB	52.78%	1,823.81
L2 SVM + MLR	54.47%	5,819.87
L2 SVM + NB + MLR	55.62%	8,627.98

Table 3: 5-fold cross-validation (CV) of ensemble learning on the training set.

Table 3 shows that a 3-way ensemble of linear L2 SVM, MLR, and NB gives the best and an ensemble of L2 SVM and NB gives the lowest CV performance. The 3-way ensemble is also the most costly model to learn. The ensemble of L2 SVM and MLR stands in the middle.

Table 4 shows the performance of the ensemble models on the actual test set compared to that of the best simple linear classifier (L2 SVM) on the same test set. Interestingly, Table 4 shows that a 3-way ensemble performs worse on the test set than the simple SVM. This is probably because the NB classifier has significantly overfitted the training data. On the other hand, an ensemble of SVM and MLR marginally improves the performance by 0.02%.

For comparison purpose, Table 4 also shows the performance on test set using the raw original images (i.e., without preprocessing) and an SVM classifier using RBF kernel ($\gamma = 0.00033$ and $\lambda = 1$). It shows that preprocessing has significantly increased the accuracy by more than 15%.

Method	Accuracy
Ensemble: SVM + MLR	57.810%
Ensemble: SVM + MLR + NB	57.752%
L2 SVM ($\lambda = 400$)	57.790%
SVM (RBF kernel on raw data)	42.781%

Table 4: Performance on the actual test set.

5. CONCLUSION

In this project, our main motivation is to implement a feasible (and non-trivial) classification method for the CIFAR-10 dataset that fits well with the timeframe and resource constraints of the project execution. Through that, we also wish to experiment with the various parameters and settings of such implementation to investigate the tradeoff between model complexity and improvement in performance. To this end, we leverage on an image preprocessing approach proposed by Coates *et al.* [1] that has been shown to significantly improve on the CIFAR-10 classification. Our implementation of the preprocessing has indeed considerably improved on the baseline (i.e., SVM on the raw data) by more than 15%. For classification on the preprocessed images, we conclude that a simple SVM with linear kernel is good enough for learning and prediction. A more sophisticated ensemble learning doesn't add much benefit in terms of prediction accuracy on the test set, and yet incurs significant cost. This is in accord with the conclusions in [1].

On the other hand, our main contribution is the modification of image re-representation method at the end of the preprocessing pipeline. That is, instead of pooling over 4 quadrants as in [1], we pool over 9 divisions, which in turn shows an increase in performance of 1.5% without adding much computational cost to the feature learning.

The preprocessing works because it has extracted and summarised the salient features of the images that are typically captured in small "patches", e.g., edges, corners, and contours. Patch size seems to matter (as most features are rather small, such as eyes, ears, and body features of an animal) and the number of patches also matters (typically, the more, the better). Pooling over 9 divisions instead of 4 helps because more features are conserved rather than subsumed by such pooling. Finally, the ensemble learning only improves marginally on the test set probably because the individual classifiers have significant overlapping predictions (both correct and wrong labels) such that combining them doesn't add much advantage. In retrospect, we could also have used a different approach to combine SVM and MLR, e.g., using an SVM with RBF kernel and tuned parameters rather than a multinomial logistic regression. This might have improved the accuracy of the ensemble learning.

6. REFERENCES

[1] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[2] I. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1319–1327, 2013.

[3] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

APPENDIX

Naassih was in charge of coding and running the experiments on image preprocessing in MATLAB. Joe was in charge of coding and running the experiments on classification and ensemble learning (both CV and on the test set) in R. Joe was in charge of compiling and writing the final report.