Qintong Wu
8421493946
qintongw@usc.edu

Leaf Classification: Feature Extractions and Ensemble Models

EE660: Final Project

2017-12-01

1 Abstract

In this paper, we explore various machine learning models and techniques to build a classifier that predicts leaf labels accurately. The novelty of my approach is to extract extra features from images then combine new features with numerical features already given in the dataset. Different models are evaluated and compared, besides all other base models, an ensemble method built upon them.

Overall, K Nearest Neighbors(KNN) achieves the best result both in the training and test set compared to other base model. However, the ensemble model performs much better than KNN in test set, even though its train accuracy is a little bit lower. Moreover, extra features extracted from images also boosts accuracies. Finally, standalone Convolutional Neural Network (CNN) outperformed all other machine learning models.

2 Problem Statement and Goal

This project mainly focuses on classification problem, that is, leaf classification. To better understand machine learning techniques, a variety of methods will be experimented.

This dataset is provided by a competition held on Kaggle 8 months ago.[1] The dataset consists approximately 1,584 images of leaf specimens (16 samples each of 99 species) which have been converted to binary black leaves against white backgrounds. Three sets of features are also provided per image: a shape contiguous descriptor, an interior texture histogram, and a fine-scale margin histogram. For each feature, a 64-attribute vector is given per leaf sample.

The difficulty of this classification problem is that the features provided including numerical measurements and raw images. Thus, utilizing both different type of inputs is critical to the success. As a further matter, there will be 99 classes of leaves to be predicted, which is a unsurprisingly difficult task for humans, even for experts. According to the description above, there are limited number of training samples (1584 in total, 10 per class).

This project's final goal is to accurately classify the leaves, and to achieve as low logloss as possible.

3 Prior and Related Work - None

4 Project Formulation and Setup

My approach to this problem can be divided into 3 main categories:

- Visualization through PCA and t-SNE methods
- Feature extractions and fusions
- Base models and the ensemble model

The original feature set will be experimented as well to check the performance of feature extractions and fusions.

[1] Leaf Classification: https://www.kaggle.com/c/leaf-classification/data

4.1 Visualization

In general, the goal of this subtask is not to enhance the final performance of a particular model but to give a better understanding about the dataset itself, and to help validate the intuitive guesses and further explorations. The most popular methods of data visualization are Principle Component Analysis (PCA), and t-Distributed Stochastic Neighbor Embedding (t-SNE). Originally, both methods are used for dimensionality reduction, which projects data from high dimensional space to a subspace without too much information loss. Due to the fact that human can only recognize 3D world at most and often most of information can be preserved in 3 or less top eigenvectors, PCA becomes an effective method for data analysis and visualization. t-SNE is particularly well suited for the visualization of high-dimensional datasets, as its name indicates, through t-Distributed Stochastic Neighbor Embedding and Barnes-Hut Approximation.

PCA can be understood and implemented this way: given a bunch of data points in $\mathbb{R}^N$, a covariance matrix $\mathbf{P}$ is constructed

$$\mathbf{P} = \mathbf{X}\mathbf{X}^{\mathbf{T}}$$

then, by eigendecomposition

$$\mathbf{P} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-\mathbf{1}}$$

where $\mathbf{Q}$ is the square matrix whose ith column is the eigenvector q_i of $\mathbf{P}$ and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues. After this construction and decomposition, for convenience, eigenvalues and corresponding eigenvectors are sorted descendingly, then pick up top 2 or 3 eigen vectors to reconstruct the original dataset. Although there are lots of math under the hood, such as Johnson–Lindenstrauss lemma and Karhunen–Loève theorem, the basic ideas above can be sufficient for the practical purposes.

t-SNE was proposed in 2014 and quickly became popular due to its effectiveness and elegance. However, the mathematical formulation can be more complex. According to the original paper, t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding. The distribution in the input space is constructed by Gaussian kernel and in the low-dimensional space is constructed by t-distribution. To measure the dissimilarity, KullbackLeibler divergence is used

$$KL(P||Q) = \sum_{i \neq j} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

where $p_{ij} = \frac{exp(-\|\mathbf{x_i} - \mathbf{x_j}^2\|/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|\mathbf{x_i} - \mathbf{x_k}^2\|/2\sigma_i^2)}$ and $q_{ij} = \frac{(1+\|\mathbf{y_i} - \mathbf{y_j}^2)^{-1}}{\sum_{k \neq l} (1+\|\mathbf{y_i} - \mathbf{y_j}^2)^{-1}}$. This completes the projection method and is sufficient for practical purpose.

4.2 Feature Extraction

The goal of this subtask is to generate new features from the raw images for the purpose of further training. Traditionally, Scale-Invariant Feature Transform (SIFT) or wavelet methods are effective for images processing. However, these methods need intricate design and cumbersome babysitting to achieve good outputs. Compared to those traditional methods, Convolutional Neural Network (CNN) proves itself a legendary method and almost dominates current image processing field. Hence, CNN will be practiced here to automatically learn some new features from the raw images as feature extraction step.

Convolutional Neural Networks are composed of several gradients: Convolutional Layer, Pooling Layer, Normalization Layer, and Fully-Connected Layer which is not necessary. To mathematically characterize CNN can be lengthy and not the goal of this project, so CNN will be used as a black box here without providing too much details.

In a nutshell, CNN will take the raw image as input, then automatically learn a set of weights and bias as its own parameters. It will also output new features when images come in, in which features are transformed by the learned parameters and pixel data. That is, training data are used to train the network to learn from different leaves' then generalize to similar leaves it never see before, and the output is a set of features associated with the leaf image fed in.

4.3 Base models and the ensemble model

This subsection describes models that will be used in detail.

4.3.1 K Nearest Neighbors

K-Nearest-Neighbors (KNN) belongs to the family of supervised learning, also refers to one of non-parametric algorithms. KNN avoids modeling the underlying data distribution by not making any assumptions on the dataset while it minimizes the pairwise distances between data points in feature space. The distance metric is often euclidean distance (L2 distance)

$$d(x, x') = \|x - x'\|_2$$

but other metric can also be used in the appropriate context, such as L1 or Hamming distance.

The hyperparameter is the value of k, which have the model find the top k closest images according to the distance metric and have them vote on the label of the test images.

4.3.2 Decision Tree

Decision Tree is the building block of Random Forest algorithm, which will be introduced next. The performance of single Decision Tree can be not good in high dimensional input space, although the result is somewhat interpretable. The goal of practicing Decision Tree is to validate the performance of Random Forest.

The model of Decision Tree is simple: recursively split regions into multiple sub-regions and assign corresponding weights

$$\hat{f}(\mathbf{X}) = \sum_{m=1}^{M} c_m \mathbb{I}\{(X1, X2) \in R_m\}$$

where $\mathbb{I}$ denotes the indicator function and c_m is the corresponding weight in each region R_m.

The hyperparameters are max depth and criterion. Max depth of the tree decides how many splits at most, and in high dimensional feature space, it is not easy to decide this hyperparameter. Criterion can be chosen as misclassification error, gini index or cross-entropy.

4.3.3 Random Forest

The Random Forest starts with a standard machine learning technique Decision Tree which, in ensemble terms, corresponds to the weak learner. The Random Forest combines trees with the notion of an ensemble. Thus, in ensemble terms, the trees are weak learners and the forest is a strong learner. Besides, Random Forest draws multiple subsets repeatedly from the whole dataset with replacement. In machine learning literature, this method is called Bagging. By this way, the variance of the final model can be decreased, leading to a consistent estimator.

The predicted outputs is defined as following:

$$\mathbb{P}(\hat{y} = c | x, D^*) = \frac{1}{B} \sum_{b=1}^{B} \mathbb{P}_c^b(x)$$

where B denotes the number of trees in the forest and $\mathbb{P}_c^b(x)$ denotes the probability of label c in the bth tree, and D^* is the corresponding data subset.

4.3.4 Support Vector Machine

Support Vector Machine, as its name suggests, optimizes over support vectors and associated hyperplanes. This method is often called large margin classifier.

Given train data $x_i \in \mathbb{R}^d$ and class labels $y \in 0, 1^n$, the classification problem can be formulated as an optimization problem:

$$\begin{aligned} \underset{\beta, \beta_0}{\text{minimize}} \quad & \frac{1}{2}\|\theta\|_2^2 + C \sum_{i=1} N\zeta_i \\ \text{subject to} \quad & y_i(x_i^T \beta + \beta_0) >= 1 - \zeta_i, \quad \zeta_i >= 0 \end{aligned}$$

Besides this vanilla formulation, a variety of kernels can also be chosen to map data into higher dimension, thus, a better hyperplane solution may be available. The common choices of kernels are

$$dth - Degreepolynomial : K(x, x') = (1+ < x, x' >)^d,$$
$$Radialbasis : K(x, x') = exp(-\gamma \|x - x'\|_2^2),$$
$$Neuralnetwork : K(x, x') = tanh(\kappa_1 < x, x' > +\kappa_2).$$

According to problem formulations above, the hyperparameter C gives the amount of regularization. Also, the choice of kernel can greatly affect the final result.

4.3.5 Ensemble

The effectiveness of aggregating a bunch of base models has been proved by better performance of Random Forest compared to single Decision Tree. Therefore, an ensemble model built upon all base models may have better performance than any single model as well. The idea of this approach is to combine different classifiers and average their predicted probabilities to predict class labels.

Given a fixed ensemble $h = [h_1(x), ..., h_K(x)]$, the final output is defined as

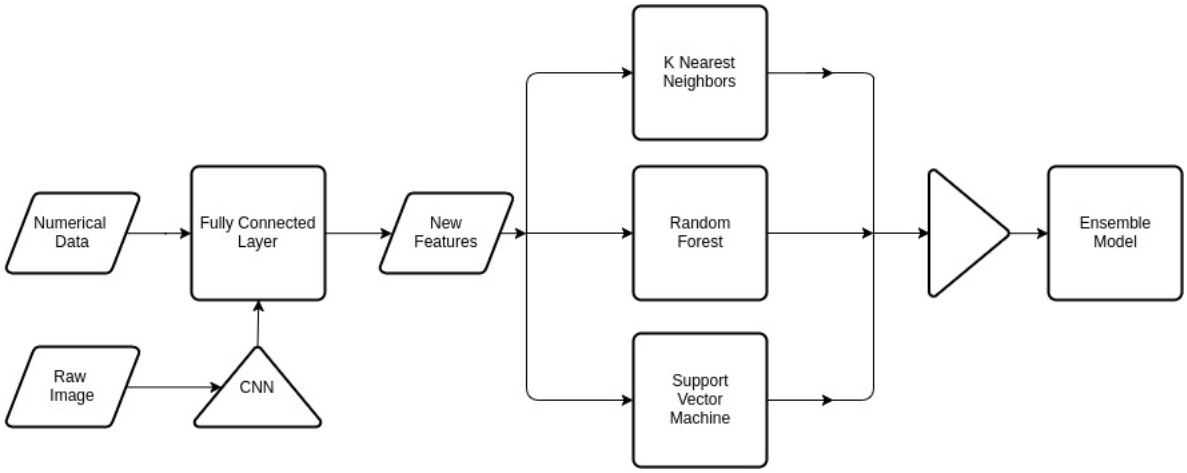
$$\mathbb{P}_{ensemble}(x) = \sum_{i=1}^{K} w_i h_i(x)$$

where w_i is the parameters learned during training. This new ensemble model simply outputs the weighted average of all base models. The final class label is then derived from the class label with the highest average probability. Similar to other ensemble methods, the new model may be more robust and overcome the weaknesses of the original base models.

4.4 Comment and Architecture

The novelty of my approach is to extract extra features from raw images, and generate a set of new learned features instead. The advantages of this approach are to use as much information as possible and reduce the dimensionality of feature space. Thus, overfitting problem is potentially prevented and underfitting problem becomes impossible. Further, an ensemble model is built upon base learners, the performance on the test dataset will be more robust in theory due to the voting behavior.

The final machine learning pipeline flowchart is shown below:



5 Methodology

5.1 The Hypothesis Set

At the first stage, there are 192 numerical features and a raw image with size 96x96 in gray scale. So the total features will be $96 \times 96 + 192 = 9408$ features in total. The output will be 99 categorical labels. This obviously is not an easy problem to learn.

At the second stage, a CNN is used to extracting features from the raw images, and a following fully-connected layer compress learned features and original numerical features into a set of new features. The number of new features is 1500 in total. Still, The output is pickup from 99 categorical labels.

5.2 Data usage

There are 1584 records with training and test data split in advance. The number of training data is 990 and that of test data is 594. Every model mentioned above is trained by 990 training data and tested by 594 test data. During training step, 5-folds cross validation is used to deal with the problem of insufficient data.

5.3 Feature Space

The training set consists of 990 records in total while test set consists of 594 records in total, in which each record is associated with 192 numerical data and a raw image.

The numerical features can be seen as three sets of features: a shape contiguous descriptor, an interior texture histogram, and a fine-scale margin histogram. For each feature, a 64-attribute vector is given per leaf sample. All these features are continuous and hard to explain the physical meaning since leaf expert is not available at this time. However, statistical results reveal the characteristics of two set of features: the first and second moments of both shape contiguous descriptor and fine-scale margin histogram are concentrated, and not surprisingly, any higher moments are concentrated as well. This fact may indicate the existence of heavy tail probability distribution. But the frustrating part is that the descriptions and sources of features are not available, so the guess above can not be validated.

5.4 Training Procedure

Training procedures are divided into three stages:

1. use images to train CNN for feature extraction
2. use new features to train different base models
3. use new features and base models to train the ensemble models

The final measurement is categorical entropy, a.k.a logarithm loss. However, during the train step, the accuracy measurement is used.

A simple error measurement is defined

$$misclassified = \frac{\sum_{i=1}^{N} \mathbb{I}(argmax_k h_k(x_i) \neq y_i)}{N}$$

where $h_k(x_i)$ is the probability of kth class in the observation i. This error measurement gives the percentage of mismatched predictions over all classes.

5.5 Test Procedures

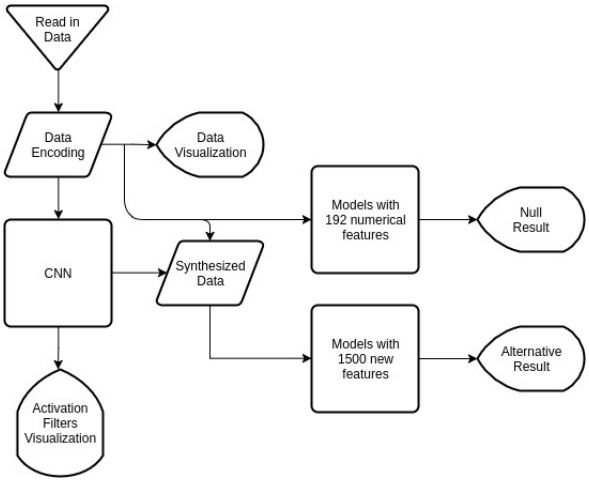
To measure the performance of the novel approach, two sets of features are used for training and testing. One is only 192 numerical features, the other is the 1500 new features.

The multi-class logarithmic loss of the form

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij})$$

is used, where N is the number of images in the test set, M is the number of species labels, log is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j. In order to avoid the extremes of the log function, predicted probabilities are replaced with $max(min(p, 1 - 10^{-15}), 10^{-15})$.

During test step, the evaluation for the final submission is followed. Every model used logarithmic loss as the final measurement, and for every input record, standardized probabilities corresponding to each labels are assigned.



6 Implementation

6.1 General Description

The entire process is described thoroughly in previous section. To avoid redundant repetitions, an implementation flowchart is represented below

In this implementation, two set of features are provided and the same models are used. The difference between inputs shows the effectiveness of the feature extraction. Besides, some trivial experiment are conducted, including Random Forest versus Decision Tree and base models versus the ensemble model.

All parameters are chosen based on heuristics and manual adjustments. Although grid search or model selection applies here, the accuracy of the ensemble model turns out to be fairly good using heuristic settings, so parameter tuning or model selection is not necessary.

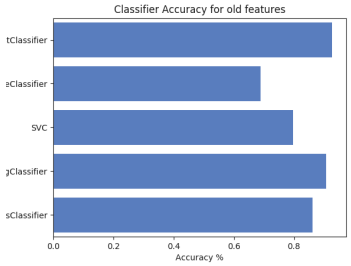
6.2 Cross Validation Result

As aforementioned, 5-folds cross validation is applied to conserve data. Two sets of results with different inputs are presented below: the first table shows the accuracy and the second shows the logarithm loss

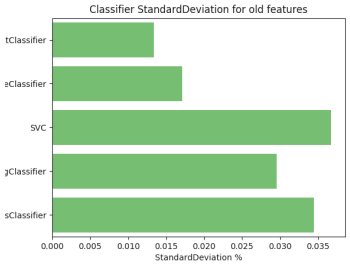
fmt: acc (+/- std)	KNN	DecisionTree	RandomForest	SVM	Ensemble
192 numerical	0.86 (+/- 0.03)	0.71 (+/- 0.01)	0.93 (+/- 0.01)	0.80 (+/- 0.04)	0.91 (+/- 0.03)
1500 new	0.98 (+/- 0.01)	0.66 (+/- 0.06)	0.91 (+/- 0.01)	1.00 (+/- 0.01)	0.99 (+/- 0.01)

fmt: log (+/- std)	KNN	DecisionTree	RandomForest	SVM	Ensemble
192 numerical	1.29 (+/- 0.27)	10.82 (+/- 0.80)	2.49 (+/- 0.02)	4.62 (+/- 0.01)	1.36 (+/- 0.04)
1500 new	0.15 (+/- 0.08)	13.22 (+/- 1.19)	3.11 (+/- 0.02)	2.36 (+/- 0.02)	1.03 (+/- 0.01)

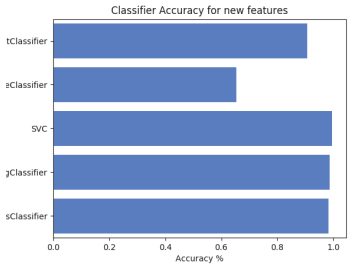
At first, other models, such as Logistic Regression, Gradient Boosting and AdaBoost were in the training pool as well. However, their performances are surprisingly bad (accuracies were around 20%). Since the goal is to generate a good ensemble model by averaging all base models, those models unable to fit well are simply discarded.



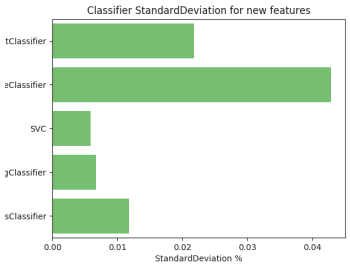
(a) Accuracy for numerical



(b) Standard Deviation for numerical



(c) Accuracy for new

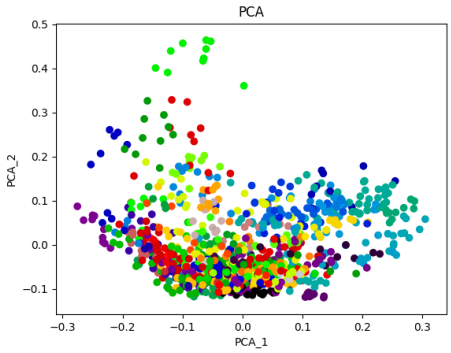


(d) Standard Deviation for new

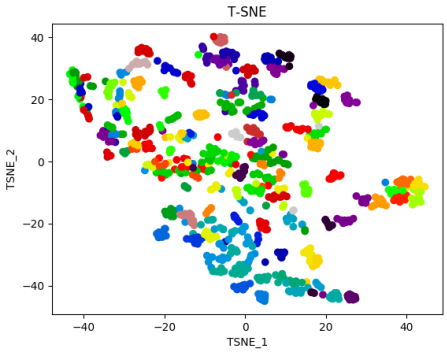
7 Final Results

7.1 Visualization

In this section, the numerical features are analyzed by unsupervised clustering algorithm. PCA and t-SNE algorithms are used for projecting high dimensional data onto 2D subspace to visualize the structure of data. The clusters of data points are shown below



(a) PCA

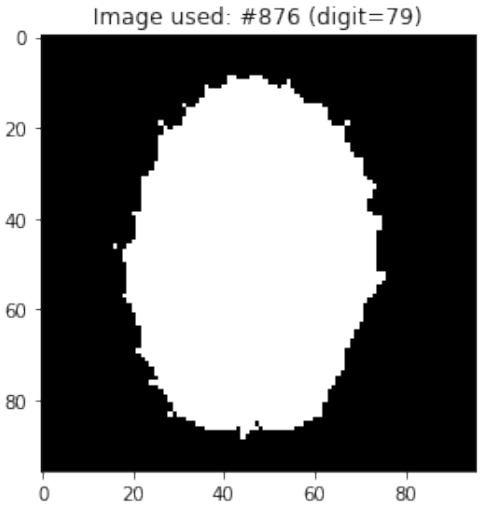


(b) t-SNE

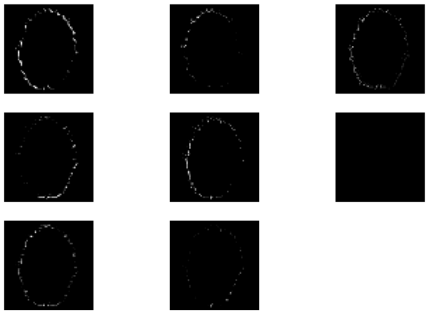
In the figures above, data with the same labels have the same colors. From the results of both algorithms, especially of PCA, the same labels are well grouped in 2D subspace, even though population in the bottom is relatively dense. Hence, the numerical features can be informative for classification tasks since there may be much more room for separating clusters in higher dimension. According to the result of t-SNE algorithm, a similar pattern can be found as well.

7.2 CNN Filters Visualization

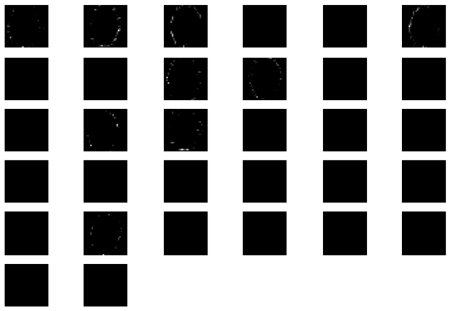
Convolutional Neural Network's filters could offer much information and visualization of these filters' content is a key to understand what is under the hood. One of common strategy is to visualize the outputs of activation layers since it underpins the learning process. Here, a random selected image by program is shown and its features learned are also provided



(a) raw image of a random leaf



(b) filters on the first activation layer



(c) filters on the second activation layer

These results are reasonable since the output is extremely sparse and only edges of the leaf is shown, which means CNN is detecting the shape and trying to use this information for classifying. This is intuitively right because humans recognize leaves by its shape as well. In the second activation layer, activations become even sparser and focus on locally edges rather than global shape in the first layer. This human-like learning process is really amazing. However, the activation is not as strong as other image classification tasks because of the lack of images. Overall, this is good enough to show the effectiveness of feature extractions.

7.3 Kaggle Evaluation

This competition gives the test dataset offline, which requires trained models output results as CSV format then uploads to the online judging platform. Here, all models' results are evaluated by online judging platform and logarithm losses are given as final scores

fmt: logloss	KNN	DecisionTree	RandomForest	SVM	Ensemble	Baseline Benchmark
192 numerical	1.15147	10.75702	3.36868	4.60760	1.28493	4.59511
1500 new	1.15147	12.73455	3.37984	2.47307	0.54632	4.59511

According to the leaderboard, there are 24 competitors who achieved 0 logloss, which means they got the perfect result. However, I do not believe their results at all. These perfect results may come from severely overfitting on the test dataset or their model may perform much worse on new data other than test data. Some reasonable results achieved around 0.005 logloss. I checked one of kernels with 0.00544 logloss, the model and machine learning pipeline is surprisingly simple: there is no data preprocessing at all, so 192 numerical features are simply fed into a 3-layer neural network with dropout regularization. I tried to reproduce his work using mxnnet package and got 0.01550 logloss as well. It seems that Neural Network is extremely effective in this classification task.

8 Interpretation

8.1 K Nearest Neighbors

This non-parametric model performs consistently with different features as inputs according to the result. One guess is that new learned features from CNN in high dimensional space is not as dense as the original features. Thus, the clustering process is not violently affected by introducing new features. However, new features do influence the predicting force, a.k.a probabilities distribution of each records. This conclusion is proven by the dramatic drop in logarithm loss and its standard deviation according to the results in cross validation step. These contrasts given different measurements show that new features significantly increase the reliability of KNN method, which means the latter predictions will be trusted although their overall accuracies are the same.

8.2 Decision Tree and Random Forest

Theoretically, Random Forest performs saliently better than its base learner - Decision Tree. From the experiment result, it is obviously true. The logloss of baseline benchmark provided by Kaggle is around 4.6, and single Decision Tree performs even worse than this baseline. However, with the same parameters of the tree set, Random Forest that consists of 500 trees achieves around 3.4 logloss.

The result that is worth to be noted shows that Decision Tree performs worse with new features learned while Random Forest's result are almost identical. This result may come from the settings of hyperparameters. That is, the number of features are different but the max depth of the single tree is the same, which causes difficulty in learning with new features. Proper parameters tuning may relieve this problem. Also, the power of bagging can be concluded from the consistent results for Random Forest.

8.3 SVM and Ensemble

With new features fed in, both Support Vector Machine and Ensemble models performs much better. This result shows the effectiveness of the feature extractions as expected.

The ensemble model achieves the best final result since it averages the predicted probabilities among KNN, RF and SVM. By proper averaging, the ensemble models overcome the weaknesses of base models and become more robust in test dataset. From the results in training dataset, all models seem equivalently good and achieve almost perfect accuracies. However, only the ensemble model can achieve logloss as lower as 0.5, which means the ensemble model generalizes better to the unseen data compared to other base models.

8.4 Comments

Even though CNN is not in the ensemble list, it is also trained and tested. The logloss of CNN is around 0.007, which proves its superiorness dealing with images as inputs. As mentioned in last section, a simple Neural Network is also practiced and achieves 0.01 logloss that is slightly worse than CNN.

The entire machine learning pipeline can be complex and cumbersome while its result is not better than the result of Neural Network. This is somewhat frustrating at least for this classification task. But the most important aspect in this project is that both numerical features and raw images are correctly processed as inputs. Moreover, the effectiveness of information in raw images is validated by the results.

9 Summary and Conclusions

In this project, a variety of machine learning methods are experimented. Furthermore, an ensemble model built upon base models by averaging their outputs is practiced. According to the final result, the ensemble model outperforms all base models. Other than the perspective of models, feature extractions by Convolutional Neural Network also help a lot, decreasing logarithm loss up to almost 50%.

However, this project is not yet perfect. The performance can be enhanced from two aspects: 1) conduct model selection or other techniques to fine-tune the hyperparameters of the base models within the ensemble; 2) augment the image data by rotating and scaling to generate more data samples for training CNN so that better feature extractions are guaranteed.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 978-0387-31073-2. URL: http://research.microsoft.com/en-us/um/people/cmbishop/prml/.

[2] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[3] Trevor Hastie, Robert Tibshirani und Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[4] Andrej Karpathy. *Understanding CNN*. 2017. URL: http://cs231n.github.io/understanding-cnn/.

[5] Laurens van der Maaten. „Barnes-Hut-SNE“. In: *CoRR* abs/1301.3342 (2013). arXiv: 1301.3342. URL: http://arxiv.org/abs/1301.3342.

[6] F. Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.