COMP90051 Project Report

sml - 2016 - gangofthree

Zean Qin (604030), Minghao Wu (798000) and Yi Ren (745877)

1 Introduction

Optical Character Recognition (OCR) for historical documents remains a very challenging case due to the unique characteristics those documents possess such as features wandering baselines, ink splodges, use of odd fonts and calligraphic capitals and the use of characters that are no longer being used in modern documents. In this project, we present two models based on Support Vector Machines (SVM) and Convolutional Neural Network (CNN) to solve the problem. And the Convolutional Neural Network is our main focus.

The dataset we use contains 50282 training images stored in the "train.csv" file and 8715 test instances stored in the "test.csv" file. All images are in greyscale with each pixel represented by a number in range [0, 255].

2 Data Augmentation and Feature Manipulation

Inspired by Simard[1] and Krizhevsky[2], we use label-preserving transformation to expand the dataset artificially in order to improve the accuracy of the models. Numerous techniques have been considered and the results are as following:

- **Denoising/Sharpening**: We initially expected that sharpening/denoising the images would improve the accuracy. To do so, we subtracted a number (20, 30, 50 etc.) from each pixel and set all negative numbers to 0. However, the models make worse predictions. After further analyzing the dataset, we concluded that it's because many characters, such as ":" and ";", "ä" and "ã", are so similar and doing so also removes their own distinguishable characteristics.

- **Elastic Distortion**: We had the intuition that applying elastic distortion will increase the number of training instances with class labels preserved,and hence, the model should make more accurate predictions. But after a number of tests, the accuracy didn't change much. Further analyzing the data, we realized that, unlike handwritten characters where the same character can take various shapes/sizes, printed data (in our example) will all have the same dimensions within the same class, so that elastic distortion should not be applied.

- **Affine Transformation**: We shifted the training images with 1 or 2 pixel displacement horizontally and vertically to generate some new training instances. After being trained on both the old dataset and the new dataset, the model makes about 2% better predictions. The drawback is that under the same configuration, the time complexity increases linearly with the number of the training instances. In addition, this method should not have unlimited use, since a lot of characters cut the edge. And essentially, this method does not create new images, so that overuse of this method will result in overfitting.

- **Dimensionality Reduction**: We attempted to reduce the dimension of the data which reduces the time complexity and increases the error tolerance by Principle Component Analysis that essentially makes the images blurred. However, since the image size is small and the frequency distribution of classes is too skewed, this method only makes the classifiers concentrate on highly frequent classes, instead of making accurate predictions.

However, we are unable to employ the above techniques with SVM classifier due to our limited computational resources.

3 Support Vector Machine

Support vector machines is widely applied to numerous real-world problems and wins great reputation in terms of classification. Inspired by Yann LeCun's work [3] which achieves 1.4% error rate using a generic SVM with Gaussian kernel, we apply SVM to our dataset.

Assuming the data are not linearly separable, the Soft-Margin SVM is implemented and we aim to minimize

$$\left[\frac{1}{n}\sum_{i=1}^{n} max\left(0,\ (1-y_i(\vec{w}\cdot\vec{x_i}+b)) + \lambda\|\vec{w}\|^2\right)\right]$$

where $\lambda\|\vec{w}\|^2$ is the regularization term and λ is the parameter determining the trade-off between increasing the margin-size and ensuring that the $\vec{x_i}$ is correctly classified. Besides the Soft-Margin SVM, we also introduce the Gaussian Radial Basis Function (RBF) to the classifier, such that

$$k(\vec{x_i}, \vec{x_j}) = e^{-\gamma(\|x_i - x_j\|^2)}\ for\ \gamma > 0$$

In this Soft-Margin SVM classifier with RBF kernel, we have to determine the λ in the objective function and the γ in the kernel function. However,when tuning parameters, with limited computation power, it is impractical to create a grid of λ and γ to test all combinations using cross-validation. Thus, we manually determine $\gamma = \frac{1}{m}$, where m is the number of attributes, which is the default setting in the package. With cross validation, we finally determine the relative optimal $\lambda = 1.9$. We generate the model by applying this model setting to the training set and it achieves an accuracy of 79.723%.

Since SVMs always make the safest decision due to the large margin principle, it is ideal for binary classification tasks. The approach we use to upgrade the SVM to the multi-class case is called **one-versus-one strategy**, in which we train $\frac{C(C-1)}{2}$ binary classifiers to discriminate all pairs of classes, where C is the number of classes. It takes $\mathcal{O}(C^2N^2)$ time for training the model and gives us desirable accuracy [4].

However, we hit the performance bottleneck when there are a large number of classes, since the time consumption grows exponentially with the number of classes. With limited computation power and time, it is not an ideal method in terms of efficiency. In addition, SVM seems to have a poor ability in terms of feature extraction, since it is unable to identify instances that are in different classes but has similar appearances. For example, it is very hard for SVM to discriminate semicolons from colons.

4 Convolutional Neural Network

4.1 Motivations

Convolutional Neural Networks, unlike SVM or normal ANN, make explicit assumption that the inputs are images. And this assumption allows us to restrict the architecture in a more sensible way[5]:

- it is made up of layers (like normal neural networks), but neurons in each layer are arranged in 3 dimensions: width, height and depth.
- neurons in convolutional layers are only connected to a local region of the input and many neurons in the same convolutional layer share the same parameters which drastically reduces the number of parameters.

These properties will make the forward function easy to implement and the model should be faster to train and make more accurate predictions in OCR applications. Therefore, after trying SVM, we also attempted the problem using CNN.

4.2 Model Design and Algorithm

The model [6] has two sets of convolutional, activation and pooling layers, then a fully-connected layer and activation, another fully-connected and finally a softmax classifier. See Figure 1 for details. And it is implemented in the "LeNet" class (located in the "lenet/pyimagesearch/cnn/networks/lenet.py" file).

The main steps (implemented in the "lenet/lenet.py" file) involve loading the training data from "Train.csv", manipulating features (as discussed in section 2), fitting the model and validating it on a subset of training set, and making predictions over testing set.

4.3 Model Configuration

Some important configurations of the model and their reasons are:

- **Two Sets of CONV ⇒ RELU ⇒ POOL Layers and One Fully Connected Layer**: As the number of layers grows, the amount of parameters to fit also increases which can cause overfitting. During tests, we observed that using too much layers can cause the validation accuracy to drop. To reduce overfitting, we attempted to randomly drop a number of neurons in the network. However, it does not seem to work well as decreasing the number of

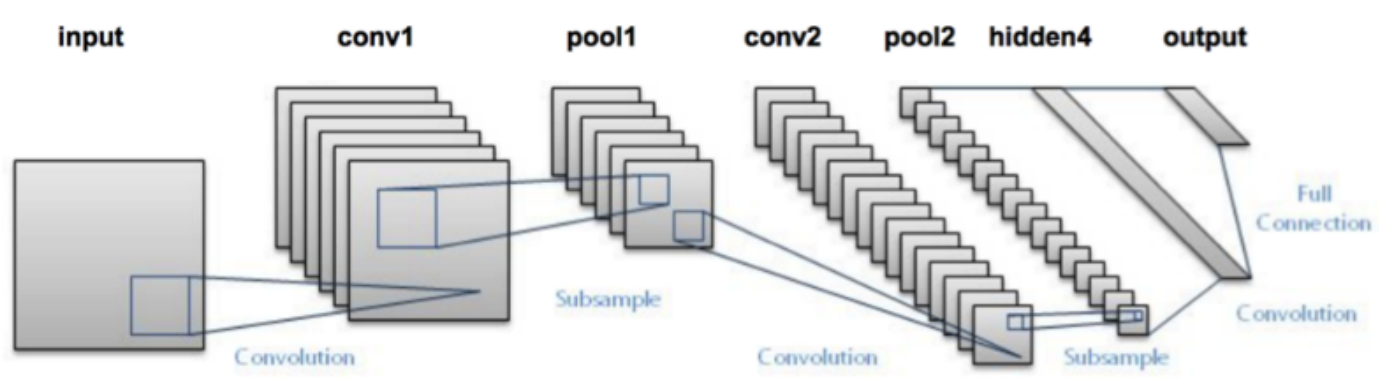

Figure 1: CNN Architecture

layers in this context. In the end, we set the model to use two sets of CONV $\Rightarrow$ RELU $\Rightarrow$ Pool layers and one fully connected layer.

- **Number of Neurons in the Output Volumes**: The number of neurons can be calculated using the following formula[5]:

$$\frac{W - F + 2P}{S + 1} \tag{1}$$

where W is the input volume size, F is the perceptive field size of the Conv Layer neurons, S is the stride and P is the amount of zero paddings on the border. We set the stride = 1 and do not use any paddings, and determine the number of neurons based on the formula 1.

- **Epoch**: during training, we measured the performance of the model against Epoch (one forward pass and one backward pass of all the training instances). Cross validation shows that the optimal number of Epoch is 20.

5 Comparative Analysis and Reflections on Methods

Essentially, the ideas behind SVM and CNN are different.

SVM always tries to insert a hyperplane between a pair of classes. If the instances are not separable under current dimensions, we can define the kernel functions to push the instances into higher dimensions and eventually those instances will be separable.

CNN chooses a different path. It tries to extract features of each class by defining convolutional layers. CNN extracts the features of training instances and compares them with testing instances, which is simulating the learning process of human.

We believe CNN is a more promising machine learning algorithm comparing with SVM. On the one hand, CNN is closer to the human learning process. On the other hand, when the data is not separable, we have to use kernel functions to push the data into higher dimensions. However, there are infinite kernel functions and we do not have a clue about which one is the best. In addition, the optimal kernel function varies case-by-case, so that it is not practical to spend a large amount of time on finding optimal kernel function for each case.

In this project, we introduced the Gaussian Radial Basis Function as a kernel function into the Soft-Margin SVM, and thus, more features are introduced into the dataset. The SVM model with this kernel function gives us a better accuracy compared with the classic SVM model. However, introducing the kernel function limits the upper bound of this algorithm. Because the complexity of the dataset grows and it can not afford more training instances. The accuracy of SVM with RBF kernel stops growing at 79.723%.

CNN outperforms SVM with an accuracy of 82.289%. The reason is that CNN has a better capability on feature extraction and we are unable to find a better kernel function for SVM with limited time and resource. This is the main reason why the accuracy rate of SVM stops at 79.723% and CNN achieves 82.289% and still has potential to grow.

Another reason why CNN is a more suitable approach is its efficiency. With the use of kernel function, the time complexity grows exponentially, and hence, we are unable to train the SVM model with the expanded training set. Within the same time limit, CNN is benefited from the larger training set.

6 Miscellaneous Reflections

We also gain a few miscellaneous reflections from this project, besides those previous ones.

- **Validation Set Selection**: The model is tuned by 10-fold cross validation. During the first few experiments, we noticed that the accuracy reported by cross-validation is always about 10% lower than the accuracy we get from the Kaggle website. It seems that the training instances are from different source after convert a few instances from pixel values to images. We shuffled the training instances and the accuracy gap is eliminated. Therefore, the instances in validation set should be randomly selected.

- **Frequency Distribution of Classes**: The class of the training instances should be uniformly distributed. If the frequency distribution is too skewed, the model will put heavy weights on those highly frequent classes. It is a reasonable decision based on the probability of each class, but this strategy does reduce the model's capability to predict those rare classes.

- **Dimensionality Reduction**: When the number of attributes is not large enough compared with the number of classes, we should be cautious to reduce the dimensions. Because this strategy could remove some characteristics of instance, if there is a large number of classes compared with the the amount of attributes, like this project, the model will only focus on those highly frequent classes. For example, after performing Principle Component Analysis, the model predicts most testing instances as "a" or "i" which are very frequent in the training set.

- **Data Augmentation**: Theoretically, we can eventually obtain the true value as the number of trials goes to infinity. There are a few method to expand the training set, but we can only carefully choose some of them on case-by-case basis.

- **Utilization of all available information**: We should try to utilize all the information available. For this project, borrowing an idea from cryptography, if we can identify 80% characters correctly in a word, it will be easy to guess the rest of characters in this word.

7 Conclusion

Optical Character Recognition for printed historic documents still remains a challenge. We build two models based on Support Vector Machines (SVM) and Convolutional Neural Networks (CNN). During tests, we are able to achieve 79.723% accuracy using SVM with $\gamma = \frac{1}{m}$ where m is the number of attributes and $\lambda = 1.9$. And in the case of CNN, we are able to train the model to predict 82.289% of the test instances correctly by using various data argumentation techniques.

Due to some challenging characteristics of the data, such as similar but hard to distinguish characters such as "ä" and "ã" and other issues like features wandering baselines, ink splodges, use of odd fonts, plus some mislabeling of the training data, it is nearly impossible for any classifier to achieve 100% accuracy.

However, we believe the accuracy can be further improved by taking into account the frequency distribution of the classes and by incorporating semantic analysis over the whole paragraph when making predictions.

References

[1] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis.," in *ICDAR*, vol. 3, pp. 958–962, 2003.

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

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[5] "Cs231n convolutional neural networks for visual recognition." http://cs231n.github.io/convolutional-networks/.

[6] "Lenet – convolutional neural network in python." http://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/.