Comparative analysis of image classification algorithms using machine learning and deep learning

Qing Guo
V00930634

GongHeng Liu
V00949607

YuMing Lu
V00885681

YingTao Zhang
V00943483

Abstract

Image classification is a machine learning task that involves assigning a label to an input image, such as "cat" or "dog". The goal of this task is to train a model to accurately predict the correct label for a given image. In this study, we compared the performance of several common machine learning models for image classification, including support vector machines (SVM), AdaBoost, XGBoost, and k-means. They also used a hybrid model combining convolutional neural networks (CNNs) with other algorithms for comparison. We conduct experiments with datasets containing up to 60,000 images of cats and dogs. We found that the normal model is about 50% accurate, while the hybrid model CNN-SVM is 97.02% accurate. The accuracy rates of the CNN-AdaBoost and CNN-XGBoost models are also above 90%, and the accuracy rate of k-means can reach up to 76%. This study shows the Role of CNN in Enhancing Accuracy and Efficiency.

1 Introduction

Image classification is a subfield of artificial intelligence that deals with the use of computer algorithms to identify and categorize different patterns in images. This is typically done using techniques from computer vision, machine learning, and image processing, which allow the algorithms to extract features from the images and use them to make predictions about their content. Image classification can be applied to a wide range of tasks, such as object recognition, scene understanding, and image restoration. By applying these techniques to images, it is possible to improve their quality, extract useful information from them, and make them easier for humans to interpret.

This statement suggests that image recognition and classification technology has become increasingly important in recent years due to the limitations of human perception and the growing need for more efficient and accurate ways of processing and analyzing images. As the amount of digital information available to us continues to increase, it has become increasingly difficult for humans to keep up with the sheer volume of data, and this has led to the development of new technologies that are better suited to dealing with large amounts of information. Image recognition and classification algorithms can process and analyze images much faster than humans, and they can do so with greater accuracy and reliability. By using these technologies, it is possible to solve problems that would be impossible or impractical for humans to tackle on their own.

In this final project, we propose the use of four mixing models based on Convolutional Neural Networks (CNN) for image classification. The mixing models are constructed by combining multiple CNN, each of which is trained on a different subset of the data. This allows us to obtain a model with a strong feature extraction ability for classifying pictures of cats and dogs. The CNN are used as feature extractors, providing more refined data to the other models in the mixing model, which improves their accuracy. We apply these four mixing models to classify pictures of cats and dogs, but our approach can be extended to other classification tasks as well. Overall, we aim to demonstrate the effectiveness of using mixing models based on CNN for image classification.

NeurIPS 2021 AI for Science Workshop.

2 Experiment I

In this section, we will discuss the methods we used in the project, including how we did the data preprocessing, the model we used for training, and the loss function applied to the model. After that, we'll explain the results we got from the experiment.

2.1 Dataset

To load the cat and dog image dataset from Kaggle into our project, we will first need to download the dataset and extract the files. We can then place the images in the appropriate folders (e.g. the "cat" folder for cat images and the "dog" folder for dog images) under the target folder. We can use the TensorFlow and Keras libraries to read and preprocess the images. Specifically, we can use the ImageDataGenerator class in Keras to read the images in batches, and we can specify the size and color channel of the images. Since we only have two categories (cats and dogs), we can set the class_mode parameter to binary to indicate that the images belong to one of two classes. This will allow us to easily train and evaluate our model on the dataset.

2.2 Data Augmentation

DataAugmentation (DA) is a technology that prevents overtraining of the model by the augmentation of data. Regularization through DA is known as a simple technology to enhance generalization performance because it does not involve the content of the model.[4] Especially in the field of imagery, there have been a large number of DAs that have been studied and explored for each task. The optimal DA varies from task to task or the nature of the data. Since we had a large enough data set, we didn't choose to use data enhancement to augment our data, Simply using tf.Keras.Preprocessing, image.ImageDataGenerator from Tensorflow random flip horizontal, data for cutting, zooming, rotate, in order to prevent overfitting. And we set the rescaling factor to 255 so that all the data would lie in the range of 0 to 1.

2.3 PCA

To reduce the dimensionality of the training dataset (with dimensions 64x64 for each data point), we employed the principal component analysis (PCA) method from the SKlearn decomposition module. This step removes unimportant data from the images to prevent high sparsity caused by excessive dimensions, thereby reducing the memory requirements and computational burden of learning on a large dataset. In this study, we selected 400 `n_components`, which were able to successfully interpret 94.8% of the original data.

2.4 Experimental Set Up and Result Analysis

Table 1: Accuracy of Each Model with 400 n components

Models	Running Time (seconds)	**Accuracies (%)**
Linear SVM	1	54.8305
KMeans	30	53.65854
RBF SVM	180	52.91304
XGBoost	12720	51.00231
AdaBoost	24960	50.59223

We trained four models using enhanced image data and optimized their hyperparameters. The results, shown in Table 1, indicate that despite retaining 94.8% of the image data, the accuracy of the models was still very low, with all models achieving an accuracy between 50-55%. Additionally, the running times of the models varied greatly, with the faster models such as Linear SVM taking only about 1 second, while the slower models like XGBoost and AdaBoost took 212 minutes and 416 minutes, respectively. Overall, the performance of the models was disappointing and the extreme differences in running time highlight the need for further optimization.

In summary, none of the models demonstrated satisfactory performance in the classification of pictures of cats and dogs. This may be due to the fact that cats and dogs have highly similar features and share many identical textures. In such a scenario, the two labels may not exhibit significant differences, making it difficult for the models to identify key features that are sufficient to distinguish between the two species. As a result, the models were only able to achieve probabilities equivalent to random flipping.

2.5 Drawbacks and Limitation

Although principal component analysis (PCA) was able to retain only the key information in the original data, it significantly improved the training speed and reduced the use of memory and computing resources. However, it was unable to extract the key features that would enable us to distinguish between cats and dogs. In order to extract these features, we require a different approach. A Convolutional Neural Network (CNN) can be an effective method for extracting key features, so we will construct a CNN as a feature extractor to provide more refined data for the models.

3 Experiment II

Images in computers are represented as numerical values, and each image can be considered as a matrix of pixel intensities. These pixels have finite, discrete digital representations that reflect their intensity or grayscale value. Machine learning algorithms allow the machine to understand the patterns of these pixels and classify the image based on those patterns. For tasks with distinct categorical identities, such as number or item classification, the model can perform the classification by analyzing the relationship between the numbers. In the case of cats and dogs, we need the model to identify the hidden identities of the two species. As the leading image classification technology in recent years, Convolutional Neural Networks (CNN) are the ideal tool for this task.

In general, Convolutional Neural Networks (CNN) possesses two key features. The first is called feature extraction, in which the input of each neuron is connected to the local receptive field of the previous layer, allowing local features to be extracted. The second is feature mapping, in which each computational layer of the network consists of multiple feature maps, each of which is a plane. Through multiple iterations of feature extraction and feature mapping, we obtain data that is significantly smaller than the original image. This data is then connected to a fully-connected layer, and the weights of the model are updated through the backpropagation algorithm. After repeated training, we will have a feature extractor that is capable of identifying the key features of cats and dogs.

3.1 Experiment Set Up

In this experiment, we will not repeat the procedure of the previous experiment. Instead, we will replace the use of principal component analysis (PCA) for feature extraction with a trained Convolutional Neural Network (CNN) model. Additionally, we have decided not to use Data Augmentation to increase the size of the training data, as we have determined that the existing dataset is large enough and the images in the dataset are distinct enough. The hyperparameters for each model will be optimized by changing the parameters, which is expected to improve the accuracy and performance of the models.

3.2 Proposed CNN Model Structure

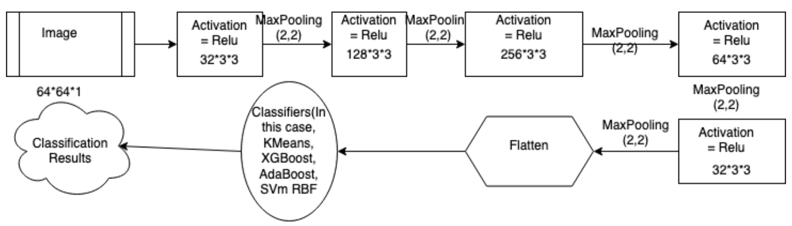
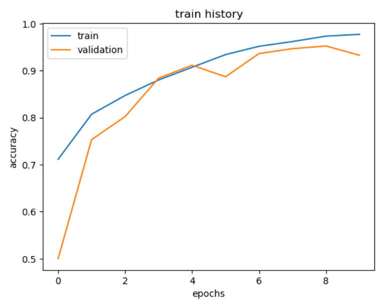
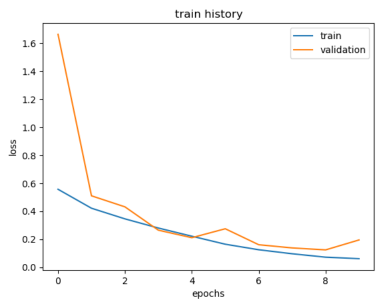


Figure 1: Proposed Feature Extract Diagram

The CNN constructed by our group consists of six blocks. The first block consists of a 2D convolution layer with 32 output filters and a 2x2 max pool layer. The second block consists of 2D convolution layers with 64 output filters and 2x2 max pools. The third block consists of 2D convolution layers with 128 output filters and 2x2 max pools. The fourth block consists of a 2D convolution layer with 32 output filters and 2x2 max pools. The fifth layer consists of a 2D convolution layer with 32 output filters, 2x2 max pools, and a flattened layer. The last block consists of a densely connected layer. The model includes three dense layers, with the first and second dense layers containing 1024 and 128 units with a Rectified Linear Unit (ReLU) activation function. The dropout rate is 10%. The final dense layer has a single unit with a 'Softmax' activation function, which transforms a feature map into a one-dimensional vector, serving as the feature extractor for the CNN model. The convolution layer's kernel size is 3x3. The CNN model has a total of 449633 parameters.



(a) CNN accuracy



(b) CNN loss

Figure 2: CNN Accuracy and Loss

3.3 Loss Function

The loss function used for the CNN model is binary cross entropy. This function compares each predicted probability to the actual class output, which can be either 0 or 1. It then calculates a score that penalizes the probabilities based on their distance from the expected value.

$$Y_i : Realvalue \quad \widehat{Y_i} : Predictedvalue$$

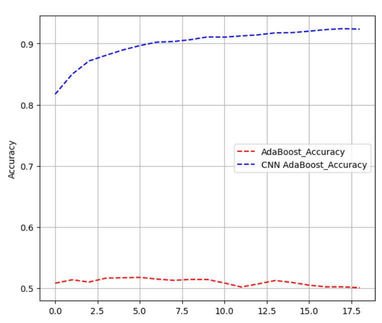
$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y_i} + (1 - Y_i) \cdot \log(1 - \hat{Y_i}))$$

Figure 3: Binary Cross Entropy

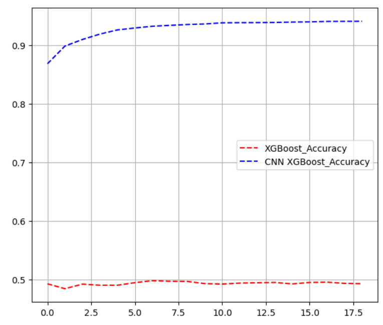
3.4 Parameter Adjustment of the Model

In our experiment, we adjusted the number of estimators (n_estimators) for AdaBoost and XGBoost from 0 to 275. We found that the number of estimators grows very slowly after reaching 200. Furthermore, we observed that increasing the number of weak learners improves the performance of both AdaBoost and XGBoost, as these algorithms are known to be strong classifiers. However, it is important to find the optimal number of estimators that provides the best performance on unseen data. Increasing the number of estimators beyond this point may lead to overfitting and degrade the performance of the model on new data.

In our experiments with support vector machines (SVMs), we found that using an RBF (radial basis function) kernel improved the accuracy of our model compared to using a linear kernel. Therefore, we decided to use the RBF kernel for the remainder of our experiments. We have also selected three additional models (AdaBoost, XGBoost, and KMeans) to combine with our SVM model. We will experiment with different methods of combining these models, such as ensembling or majority voting, to find the approach that provides the best performance on our dataset.



(a) AdaBoost



(b) XGBoost

Figure 4: AdaBoost and XGboost result

3.5 Experiments Results

After 10 epochs of training, we successfully built our Convolutional Neural Network (CNN) model and achieved an accuracy of 94.82 % and the training time is 1170 seconds. We then used this model as a feature extractor to train the extracted features for each model. To do this, we removed all the dense layers and saved the model with 128 dimensions after 10 epochs, then the weights of the trained model are retained. These models were then used as the feature extractors for each classifier. The data obtained after 10 epochs of training was recorded and visualized. We chose to train for 10 epochs because some models can already achieve high accuracy within this number of iterations, and the improvement in accuracy by continuing to train is often minimal. Therefore, continuing to train beyond 10 epochs would not be beneficial. The results, shown in Figure 3, demonstrate the accuracy of all combined within 10 epochs of training.

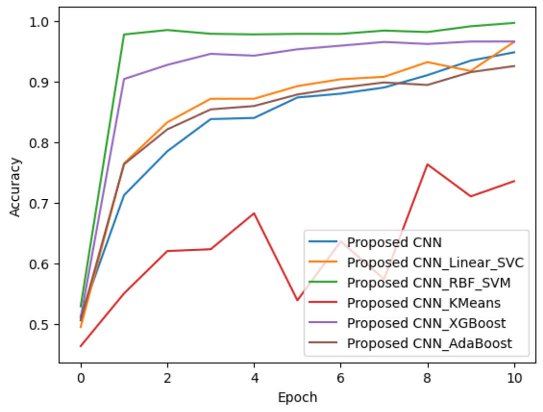


Figure 5: Combining models accuracy within 10 epochs

3.6 Results Evaluation

In this study, we trained a Convolutional Neural Network (CNN) model for image classification tasks. After the first round of training, the accuracy of the CNN model was 71.37%. Other common machine learning models, such as support vector machines (SVMs), AdaBoost, and XGBoost, all achieved an accuracy of over 90%, with the SVM model showing the highest accuracy of 95.77%.

We found that the pre-trained CNN model was able to extract key information from the input images and convert them into a 128-dimensional data output. This data was then used by other classifiers to achieve more accurate test results.

Additionally, we found that using the refined information from the pre-trained CNN model significantly improved the training speed of other models. For example, the XGBoost model only took 3 seconds to obtain the prediction results after using the refined information, compared to 212 minutes without it.

After ten rounds of training, the accuracy of the SVM, AdaBoost, and XGBoost models was over 95%, with the SVM model achieving the highest accuracy of 97.02%.

In contrast, the accuracy of the k-means model did not exceed 80%. This is likely due to the fact that k-means is an unsupervised learning algorithm, meaning that the input data is not labeled or labeled ahead of time. This means that the algorithm has to learn on its own without labeled data, which can lead to lower accuracy compared to supervised learning algorithms.

Overall, all models showed improved classification ability for the Cats and Dogs dataset when combined with the CNN model. The increase in training time and test accuracy for the CNN model also led to an increase in the test accuracy of the other models. The results indicate that the CNN model played a crucial role in feature extraction, accurately distinguishing small identity differences between similar species. Even though the specific key features used by the CNN model are unknown (it is a "black box"), we believe that it has effectively mined these features and is an effective approach for distinguishing between different classes of images.

4 Conclusion

In this project, we attempted to use four different machine learning models – K-Means, SVM, AdaBoost, and XGBoost – to classify images from cat and dog datasets. However, we faced challenges with these models, including long run times and low accuracy. Our run time goes more than 2 days while trying to keep all 4096 dimensions and we are forced to shut down the program, so we decided to use principal component analysis (PCA) to reduce the dimensions. Despite this, the accuracy of the final results was not ideal. None of the models exceeded 55% accuracy, despite the high quality of the image data (94.8%). In order to improve efficiency and accuracy, we used a Convolutional Neural Network (CNN) to construct four combined models. CNN is known for its ability to extract features from high-dimensional datasets, and its use allowed us to achieve an accuracy of 80% or higher in most models. Among all the models, the CNN RBF SVM had the best performance. After CNN training in the first epoch, the training accuracy improved from 71.24% to 95.77%. In the last 10 epochs, it outperformed all other models with an accuracy of 97.02%. K-Means, on the other hand, was the only model that did not reach 80% accuracy with the help of the feature extractor. This may be because K-Means is an unsupervised clustering algorithm, but the improved accuracy of all models demonstrates the effectiveness of using CNN as a feature extractor for image classification tasks.

5 References

[1] Lyashenko, V. (2022, December 13). Data Augmentation in python: Everything you need to know. neptune.ai. Retrieved December 13, 2022, from https://neptune.ai/blog/data-augmentation-in-python

[2] Sklearn.ensemble.adaboostclassifier. scikit. (n.d.). Retrieved December 13, 2022, from https://scikit learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[3] Wrappers for the Scikit-Learn API (no date) Scikit-learn API - Keras Documentation. Available at: https://faroit.com/keras-docs/1.0.6/scikit-learn-api/