CNN-Based Model for Recognizing Street View House Numbers

Group Members: Shiwei Yu, Jiaying Zhuang, Xiaomeng Chen, Zidong Xu, Zixuan Wang

Introduction & Dataset

Learning and recognizing digits in images of natural scenes is an important branch of deep learning image recognition, which has a wide range of uses. So there are two main purposes of this project. One is to use the SVHN dataset to develop better models and explore better ways to solve digit recognition problems.[1] The second is to explore the practical application of these models. For example, a possible application may be the update of smart maps. After automatically identifying the house number information in the pictures collected from the Street View car, the system can synchronize the map information in real time.
SVHN is a real-world image dataset for developing machine learning and object recognition algorithms, containing more than 600,000 digit images, taken from house numbers in Google Street View imagery. In this project, we use 73257 numbers for training. The numbers contain 10 classes, 1 class for each number, e.g. number '1' has label 1, '9' has label 9. According to EDA, we found that this dataset has an uneven distribution of digits, with more digits '1' than other digits, and the data we used after preprocessing still had some distracting digits on either side of the digit of interest. And these noises also put forward higher requirements for the training of our model.

Models & Results

- PCA & KNN

Considering that the SVHM dataset is image data, we think of dimensionality reduction. And in our opinion, PCA is one of the most important methods. In our expectation, PCA can reduce the problems caused by the disaster of dimensionality (high dimension); it can be used to compress data to minimize data loss; and it can reduce high-dimensional data to low-dimensional for visualization. After PCA dimensionality reduction, the first 173 features can represent 99% of the original features. After we calculated the eigenvalues and eigendigits using principal component analysis, we used k-nearest neighbors to classify the test dataset. In other words, we used PCA and KNN during feature extraction to obtain dimensionality-reduced salient features when building the model. However, the results we tried were only 49% accurate.
We guess this is because of the large parameter space of images. In other words, the domain of definition of the parameters is so large that the desired clusters cannot be found during the clustering process, resulting in out-of-focus features.

- K-Means

K-means focuses on the number of clusters and the dimensionality of the dataset. If we used the original dataset without dimensionality reduction, we only got 18.92% accuracy when clusters equal 10 and 27.61% when clusters increased to 500. So in order to improve, we compressed the data using PCA to a degree that preserves 95% variance of the data and only lost 5%. Then we found that only 54 out of 3072 features can preserve 95% of the data, which means SVHN is originally very sparse and most of the data is rather present at a much lower dimension.

[1] Datasets: http://ufldl.stanford.edu/housenumbers/

However, if we plot again we would see that actually the accuracy rate increased only by 0.1% when clusters equal 500.

- VGG-16

In this part we will apply transfer learning based on VGG-16 to further explore the SVHN dataset. VGG-16 is a CNN model with 16 weight layers and aims at recognizing large-scale images.[2] It has been trained with more than 14 million images of 10 classes in the ImageNet database. It is a power computer vision model, so we used a pre-trained version of it and trained our own layers to fit the dataset.

The first step is data preprocessing. We set the label "10" in the dataset as "0" and convert the labels to categories to meet the demands of training. To improve the data quality, we standardized the training set and test data based on the mean and std of the training data and adjusted the brightness of the images. Also, we applied angle and location shifts to make the dataset more diverse.

Secondly, we tried the model without fine-tuning. We removed the "top" part of VGG16, which includes 3 linear layers and 1 output layer. Then we defined our own fully-connected layers and set the output layer as 10 classes. We utilized the "rmsprop" optimizer and early stopping and set the learning rate as 0.001. After 8 hours of training, the model achieved an accuracy of 40.13%.

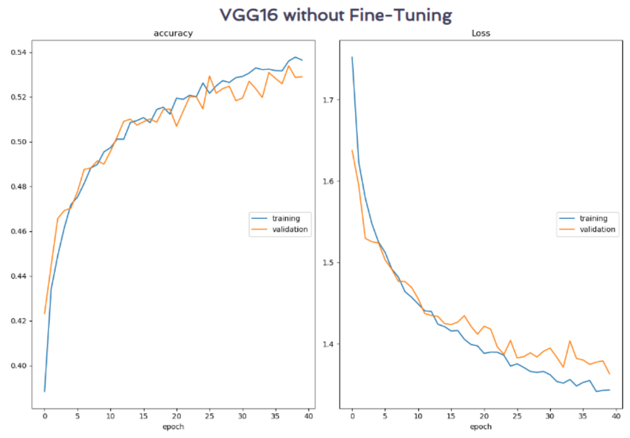

Fig. 1 Accuracy and Loss Plot of the VGG-16 model without fine-tuning

Since the accuracy is not that good, we tried to fine tune the model thirdly. Before training, we searched for some information on digit detection and found that digit detection is not sensitive to brightness, and location shift is also not necessary. So we deleted these preprocessing steps and kept only the standardization and angle rotation part. This time we only utilized the first 10 layers of VGG-16. We froze the first 8 layers and retrained the last convolutional layer and max-pooling layer. Linear layers are similar to those in the model without fine-tuning. The training time for each epoch increased a lot, so we tried only 10 epochs for this fine-tuning model. The test accuracy reached 92.42% and was a relatively good result.

[2] McDermott, J. (2021, March 1). Hands-on Transfer Learning with Keras and the VGG16 Model. Learn Data Sci. https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/

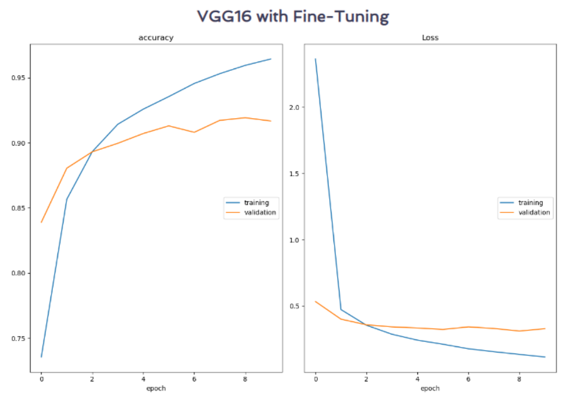

Fig. 2 Accuracy and Loss Plot of the VGG-16 model with fine-tuning. 8 layers are frozen and 1 convolutional layer and 1 max-pooling layer is retrained.

- ResNet

Based on the classic Model – ResNet-18 and some ResNet Based models[3], we made our own neural network. Because the original model was built on another dataset — ImageNet, the input, and output of the images have changed to adapt ResNet to the street view house number dataset. We also transformed the original image size from 32X32 to 28X28 to make the images(digits) consistent in clarity. As most of the images are blurred, some images with clear digits may influence the learning process of the model. For the architecture of the tuned model, the max-pooling layer was deleted from the original structure in the first block. For blocks 2 and 3, and the average pooling layers, we have followed the original ResNet structure. The linear transformation layer is rewritten to make sure that the number of output prediction classes is 10. We first attempted a loop running 6 iterations with 3 epochs each to test the stability of the neural network. After the stability test, we trained the neural network with 10 epochs. From the results of the training set, the residual network is still learning and the result can even be improving. The accuracy on the test set improves to 95% as the epochs increase. For the losses, the test loss is consistently lower than the training loss, but the gap between them shrinks over time. Both test and train losses are very high. For the present, we haven't trained it with a higher number of epochs. But we expected the accuracy can be further increased, and the loss can be decreased. Then, we trained the neural network with 30 epochs and it triggered the early stopping at the 23th epochs. The train and test losses have decreased greatly compared to the 10 epochs, while the accuracy without further increase.

[3] Ramzan, Farheen & Khan, Muhammad Usman & Rehmat, Asim & Iqbal, Sajid & Saba, Tanzila & Rehman, Amjad & Mehmood, Zahid. (2019). A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks. Journal of Medical Systems. 44. 10.1007/s10916-019-1475-2. https://www.researchgate.net/publication/336642248_A_Deep_Learning_Approach_for_Automated_Diagnosis_and_Multi-Class_Classification_of_Alzheimer%27s_Disease_Stages_Using_Resting-State_fMRI_and_Residual_Neural_Networks

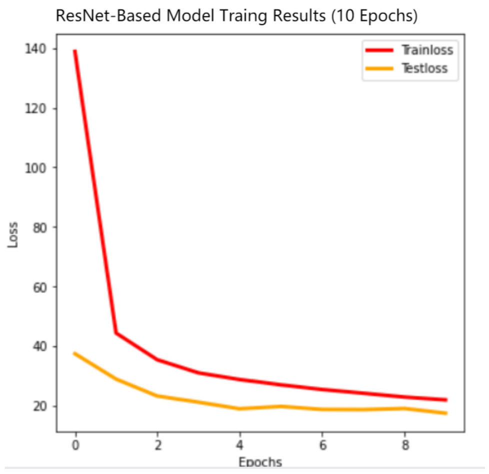

Fig. 3 Training Results in the ResNet model with 30 epochs, it triggered the early stopping at the 23th epochs.

Conclusion

To summarize, we have applied 4 models in total, where K-Means is unsupervised learning and the other three are supervised learning. K-Means had the lowest accuracy rate of only 28% even though clusters increased to 500 and the dimensionality reduction helped a little. The result of the PCA & KNN attempt was that the accuracy rate was 49%. This was still not a significant result, and one possible reason we think may be that the parameter space of the image is very sparse. In other words, the domain of definition of the parameters is very large, and therefore the clustering algorithm can't locate the clusters. Another model we tried was VGG-16, and we found that fine-tuning can make the model fit the data set better and finally we obtained a 92.4% accuracy, which was pretty good. Furthermore, we noticed that in our case utilizing only the first 10 layers performs better in digit detection, probably because the full 16-layer version pretrained on ImageNet focuses more on object detection. Lastly we tried the ResNet model, this one achieved the highest accuracy of 95% despite its high loss. But as the loss trends for both training and testing sets were declining, we may increase the number of epochs until the early stopping is triggered.