COMP-551
Applied Machine Learning
Kaggle Report
Team Weyavuong

Luya Ding	Eric Vuong	Weishi Wang
260627893	260630559	260540022

November 26, 2018

1 Abstract

In this Kaggle competition, the goal is classify Google free hand drawings. 10000 train data and their corresponding labels are given, and 10000 test data need to be classified into 31 classes. Three models are considered, support vector machine (SVM), feed forward neural network and convolutional neural network (CNN). SVM is used as baseline model, and has an accuracy of 0.089 after tuning parameters and preprocessing images. Feed forward neural network has an accuracy of 0.086 after tuning parameters and preprocessing. Convolutional neural network has a validation accuracy of 0.79 after batch normalization and tuning key parameters.

2 Introduction

Classification problem is one of the most crucial problem in modern machine learning. After CNN proved its high accuracy in recognizing hand written digits, it has been popularly used for image classification. The goal of this project is to implement CNN algorithm to recognize free hand sketches within 31 classes and compare with other methods such as linear SVM and feed forward neural network. In addition, the accuracy of CNN is improved adjusting the number of layers and parameters such as max pooling and batch numbers. The result of CNN after tuning is expected to perform better than the CNN baseline accuracy which is 0.46.

3 Feature Design

3.1 Data split

Train data are randomly split into train data and validation data with a proportion of 9 to 1. Thus, 9000 images are used as train set and the rest 1000 images are used as validation set.

3.2 Encoding

This data set consists of 31 classes, which are all composed of string. It is desirable to encode the classes with numbers so it can be easily fed to different functions without concerning about data type consistency. Labelbinarizer is used to encode the labels into increasing numerical values.

3.3 Noise Removal

In the given data, noises are intentionally added to images, making them difficult to classify. Linear SVM classifier was applied to unprocessed images, and the result on validation set is very poor, only 0.0396 before tuning, which is close to the random classifier. Thus, images need to be processed before training models. OpenCv package is used to identify all connected components and only keep the largest connected component. This algorithm successfully denoises all images (fig. 1-2).

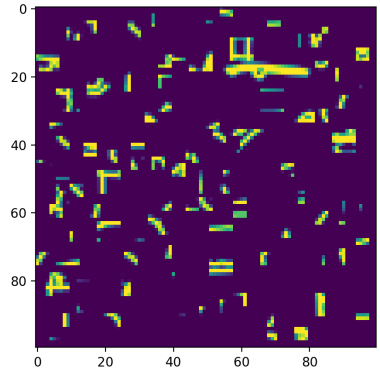

Figure 1: Image before denoise.

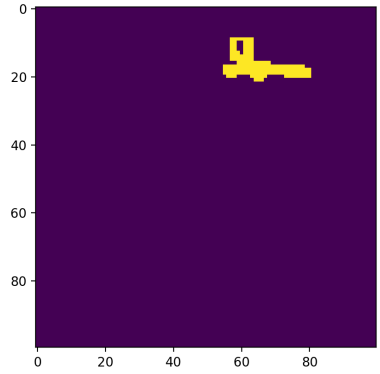

Figure 2: Image after denoise.

Moreover, there are more than 300 images in the train data set that are pure noise. The above algorithm will still choose the largest connected component, which is not desired since it should be empty. A threshold area of 50 pixels square is added to the algorithm, so that if the largest component is smaller than this threshold, the image will be interpreted as empty image (fig 3-4).

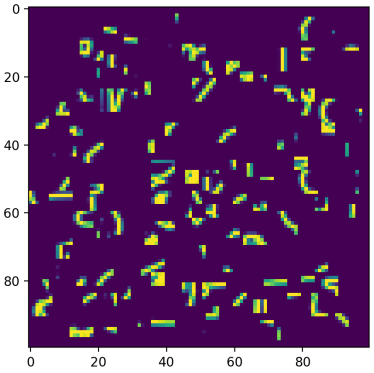

Figure 3: Pure noise image before processing.

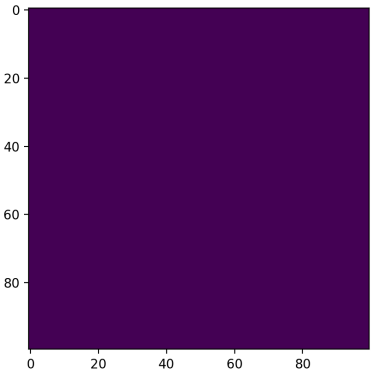

Figure 4: Pure noise image after processing.

4 Algorithms

4.1 Support Vector Machines

The baseline algorithms of this project is support vector machine(SVM). Support vector machine approaches the classification problem using a method called widest street approach, which maximize the distance of the hyperplane to all classes. The decision rule is to determine on which side of the street the data point is. This can be done by taking the projection a the target vector to a vector ω which is perpendicular to the hyperplane. A threshold of 1 and -1 can be used to differentiate classes. The goal is to maximize this distance, or, equivalantly, to minimize the norm $||\mathbf{w}||$. The problem can be formulated as:

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to}$$

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1,$$

$$\text{or } \ 1 - y_i(\mathbf{x}_i^T\mathbf{w} + b) \leq 0,$$

The quadratic program problem can be solved using Lagrange multiplier problem by minimizing:

$$L_p(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} \alpha_i(1 - y_i(\mathbf{x}_i^T\mathbf{w} + b))$$

with respect to $\mathbf{w}$, b and the Lagrange coefficients $\alpha_i \geq 0$

The optimal decision plan can be obtained after solving the optimization problem:

$$\frac{1}{||\mathbf{w}||} = \left(\sum_{i \in sv} \alpha_i\right)^{-1/2}$$

4.2 Feed Forward Neural Network

A fully connected feed forward neural network is implemented from scratch in this approach, with the use of numpy library only. Backpropagation is used to train the model by adjusting the weights and biases in order to find a minimum of the error (loss) function.

The architecture of this neural network is designed such that each layer has its corresponding weights, biases, cache and deltas arrays. Initial weights are randomized and initial biases are set to be zeros. In the feed forward process, the input of each layer passes the activation function, along with the weights and biases of the current layer. After the activation, the output of the current layer is saved into this layer's cache and passed to the subsequent layer as its input.

In this implementation, all layers except the last layer utilises sigmoid function as the activation function, while the softmax function is applied to the operation in the output layer.

After feeding forward, the output values are compared with the correct answer to compute the value of loss function and our goal is to minimize this value of loss. We calculate the gradient of this model's error with respect to each weight, which it denoted as deltas in the code. These deltas are computed backwards, starting from the output layer towards the input layer. With the deltas, we are able to update weights and biases via gradient descent, so as to reduce the error of the model. Cross entropy function is used to calculate the deltas of the output layer since the derivative function of softmax can be cancelled out by chain rule; we then apply the derivative function of sigmoid to compute the deltas for all other layers.

4.3 Convolutional Neural Network

Convolutional Neural Networks is a class of deep, feed-word artificial neural networks. As we are dealing with images, CNN are good for this type of task as they are good at extracting position-invariant features. There are various steps in a CNN.

4.3.1 Convolution Layer

The name itself revolves around convolution, which is the first step of convolutional neural network. Intuitively, given an input image, we may use a feature detector, or kernel (usually 3 x 3 matrix), which is applied to the original image to observe any matching pattern. The kernel performs bitwise multiplication a specific part of the original input image, such that it will detect certain features of the image and transform them into a new feature map, or convolved feature.

Using different filters, we can obtain various feature maps. Moreover, throughout the training, the convolutional neural network can apply various feature map on its own.

When applying feature detector, we may lose certain information, but the whole purpose of feature detector is to amplify and extract certain features from a given image.

4.3.2 ReLU Layer & Softmax

ReLU, a Rectified Linear Unit, is a rectifier function will be used after applying the convolutional layer. The reason is because we want to increase non-linearity into our network, because images themselves are usually not linear. Images usually do not have many linear elements as they are composed of borders, different colors, elements, etc. Thus, rectifier will allow us to break any linearity that the model is attempting to develop during training.

4.3.3 Pooling

Pooling is a crucial step in designing a convolutional neural network. Pooling allows the CNN to become more spatial invariant, where it doesn't care if features itself are distorted, the CNN will have a certain level of tolerance to image distortion. Thus, using the feature map applied by the kernels, max pooling was applied. By pooling, we are getting rid of all the important things that aren't using and extracting only all the highest numbers and mapping it into the pooled feature map. Thus, can preserve feature by taking into account of all the possible spatial distortion that can be present and also reducing the size of the features. This is also an important step, because it reduces the number of parameters, which decreases the chance of overfitting. Softmax has been explained in section 5.2.1.

4.3.4 Flattening

Flattening is a simple and crucial step. Once the pooled layer is computed, all the pooled feature maps will then be flattened into the fully connected neural network further processing.

5 Methodology

5.1 Support Vector Machine

5.1.1 Kernel

As SVM is considered as a baseline method in this project, simple kernel should be implemented. Thus, the kernel of SVM is linear throughout the parameters tuning process.

5.1.2 Intercept Scaling

Intercept scaling is a synthetic feature weight, which is subject to L1 or L2 regularization as all other features. Regularization effect on synthetic feature weight can be increased by decreasing Intercept scaling. This parameter is tuning using brute force by iterating different values and fix other parameters.

5.1.3 Max Iteration

Max iteration number can affect the prediction due to overfitting and underfitting problem. There is not really a way to know when overfitting will occur, thus, the only way to tune this is to iterate within a range and record the value that gives the highest score. Tuning results are showed in fig 5-6.

5.2 Feed Forward Neural Network

5.2.1 Activation function

Softmax function is used for the output layer so that the output is a probability distribution (sums up to 1), while simple sigmoid function is applied to all other layers as it squashes the neuron's preactivation between 0 and 1, and it's always bounded, positive and strictly increasing. Although tanh and relu are considered possible choices for the activation function in many scenarios, however sigmoid function generates the best performance result in our case and is hence preserved.

5.2.2 Hyperparameters

Hyperparameters, including the learning rate, number of nodes (Except for number of nodes in the input layer, which is 100*100, i.e. number of pixels in an image, and in the output layer, which is 31, i.e. number of output classes) and number of layers are determined by cross-validation. Learning rates from 1e-4 to 1, number of hidden layers from 1 to 3, number of neurons in each hidden layer from 31 to 10000 were tested. The reasoning of choosing the number of nodes in each layer is based on Jeff Heaton's theory that 'the optimal size of the hidden layer is usually between the size of the input and size of the output layers'.

5.2.3 Updating weights and biases

Gradient Descent is used to update weights and biases in the implementation of backpropagation, as this is simple and powerful enough for this model. More sophisticated approaches are discussed in section 7.1 for future improvement.

5.2.4 Preprocessing the image

Similar preprocessing as discussed above is applied to the input data before fitting this model.

5.3 Convolutional Neural Network

Batch Normizalition, Dropout, Activation, Flatten, Max Pool, Conv2D filters, Kernel size, Strides, Loss (Categorical Crossentropy), Optimizer (adam).

5.3.1 Batch Normalization

Usually, normalization is used to normalize the input layer by adjusting and scaling the activation functions. This can increase the learning speed. Thus, the same principle can be applied for the hidden layers within the neural network. This reduces overfitting because of its regularizing nature, adding some noise to each of the hidden layers' activation.

5.3.2 Dropout

Similar to the batch normalization, the dropouts also adds noise to the hidden layers. As a result, when a convolutional neural network uses batch normilization, we will use less dropouts in order to overcome the limitations of their combination by avoiding the variance shift risks.[1]. One of the main feature in using dropout for neural networks is to reduce the risk of over fitting. In short, dropout is used to prevent over-fitting by attempting at reducing interdependent learning amongst the weights by turning off some of the weights randomly during training.

5.3.3 Activation Function

ReLU has been used after applying the convolutional layer. This is because we would want to promote non-linearity in image recognition as aforementioned in section 4.3.2. Softmax has been applied at the output as discussed in section 5.2.1

5.3.4 Max Pooling

Max pooling's goal is to down-sample an input image to reduce computational cost by reducing the number of parameters that are required for learning and also help over-fitting by providing an abstracted form of the representation of an image. More specifically, max pooling is used and is done by applying a filter that extracts the maximum value within an non-overlapping subregion of the convoluted feature maps.

5.3.5 Optimization with Adam

Adam is an optimization algorithm that is often used instead of stochastic gradient descent to update network weights.

5.3.6 Preprocessing the Image

The advantage of using convolutional neural network is that the network will learn to filter out the noise by itself when applying feature detection and max-pooling. Thus, the act of preprocessing the image to remove noise is not necessary. To show this hypothesis, the same configurations of a neural network has been run twice: one with preprocessing and one without. In fact, the validation accuracy decreases slightly if the image preprocessing has been applied, with a drop from 0.78 to 0.73 for the validation accuracy. This can be explained by the maxpooling effect, where it can extract the important features of the feature maps, thus making it space invariant and make the network ignore the noises generated.

6 Results

6.1 Support vector machine

The accuracy of prediction results using SVM on raw train data before preprocessing images is 0.0396 on validation set, only slightly better than

the random classifier, which is 0.0366. Tuning parameters is unnecessary due to extremely low accuracy.

The best prediction accuracy obtained after preprocessing images and tuning parameters on validation set is 0.089. Two parameters of SVM model are tuned, intercept scaling and max iteration (fig. 5 & fig. 6).

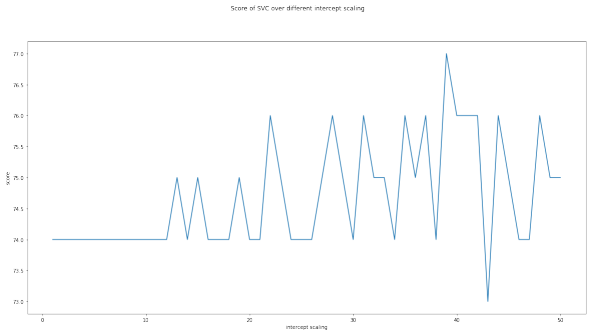

Figure 5: Score of SVM model with different intercept scaling

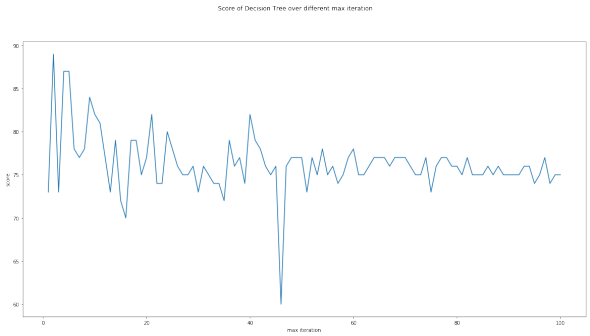

Figure 6: Score of SVM model with different max iteration

6.2 Feed Forward Neural Network

From the graph, we can observe that the loss continuously declines with the increase of number of epoch, although the rate of decrease becomes slow after around 30 iterations. This is in accordance with what we expect, as the goal of this model in general is to minimize the value of the loss function.

After tuning the hyperparameters with cross validation and image preprocessing, the best result achieved by using this simple feed forward neural network is:

Training accuracy: 0.08644444444444445

Validation accuracy: 0.065

Compared with random classifier (which has an accuracy of 0.03666), this model generates a better result after learning. And compared with the SVM classifier, the accuracy does not have a large difference. However, the performance is obviously not as good if compared with the fine-tuned convolutional neural network.

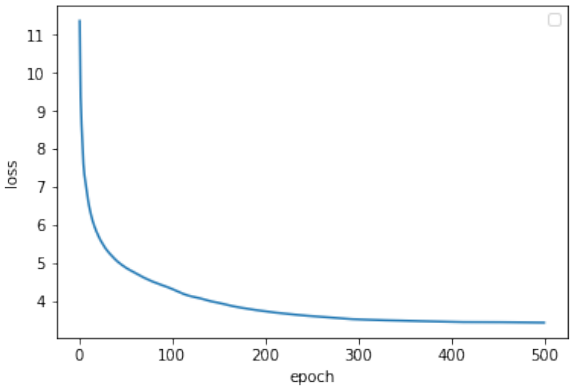

Figure 7: loss decreases with more iterations of feed forward and back propagation

6.3 Convolutional Neural Network

To accelerate the training, an Nvidia GTX 1080 GPU has been used, wuth cudnn version 7.1.4 & CUDA 9.0. With the aforementioned methodologies, our best convolutional ceural network result is as shown below fig. 8:

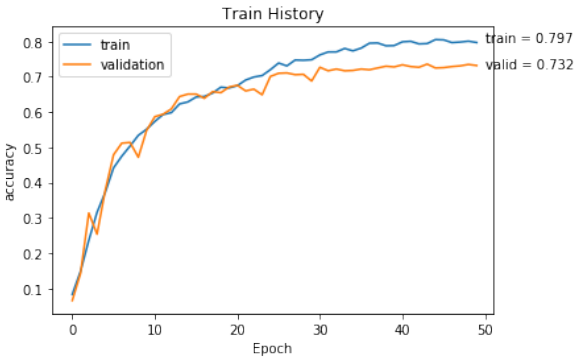

Figure 8: CNN with configurations dropouts set at 0.25.

It seems that overfitting starts to happen at around 25 epochs, where the validation accurate starts to diverge from the training accuracy. Convergences happens at around 0.73, which yielded an accuracy of 0.75 on the Kaggle Competition leaderboard. Then, reducing the dropouts of weights to 0.1 helped increase the validation accuracy as explained in section 5.3.2. Figure 9 shows the effect of reducing dropouts. It can be seen that the validation accuracy has increased ever so slightly as

the dropout level decreased. This is because of the fact that batch normalization has been applied at each layer.

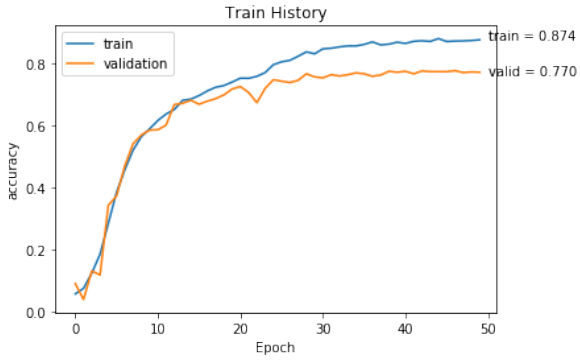

Figure 9: CNN with configurations dropouts set to 0.10.

The most important parameters in the convolutional neural network is the convolutional and pooling part, where extracting the feature maps and applying max pooling methodology played an important role in the validation accuracy.

It was noticed that applying batch normalization at the fully connected layers and adding dropouts of 0.5 each also helped increased the final results. With regularization contributed by the combination of batch normalization and dropouts, this reduced the overfitting. The final model design uses 6 layers of convolutional layers with the first 2 and last 2 having max pooling applied. Then, a dropout value of 0.1 and 0.25 has been applied to the 2 hidden layers with units 512 and 256. Moreover, batch normalization and flattening been applied to both hidden layers as well.

7 Discussion

7.1 Support Vector Machine

The linear support vector machine is used as a baseline model for this project. As a baseline, only linear kernel is considered, and only two parameters, intercept scaling and max iteration, are tuned. Linear SVM method can be desirable when data are linear separable and relatively even distributed. Thus, linear SVM does not give accurate prediction for image classification and especially 31 classes.

7.2 Feed Forward Neural Network

This approach is implemented from scratch, so it's rather simple and easy to understand and document. The disadvantage is, however, compared with more complex convolutional neural network which can be fine-tuned with simple changes, because we are not allowed to use any library except numpy, the tuning process is much more time consuming, a smallest change leads to large modification of the code, therefore the hyperparameters/parameters we use might be far from the ideal ones.

Another potential area to work on in the future is to consider using Stochastic Gradient Descent with momentum in updating weights instead of simple gradient descent, which may help accelerate gradients vectors in the right directions, leading to faster converging.

7.3 Convolutional Neural Network

In the convolutional neural network, it was observed that dropouts, image rotation, batch normalization, adding more convoluted layers and max-pooling all contributed to building a more performant model. The model can be improved by making more data augmentation, such as rotating the image, shifting width, creating more distortions and performing vertical flips. When presented with limited dataset, finding ways to augment the data is important. Also, more images may have been downloaded from the Google Quickdraw dataset as well to enhance the training.

8 Statement of contributions

All of the team members has participated in the discussed of all sections. More focused attention was given to the SVM & Image Processing by Weishi, and the feed forward neural network by Luya and CNN and training the network by Eric Vuong.

We hereby state that all the work presented in this report is that of the authors.

References

[1] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," *CoRR*, vol. abs/1801.05134, 2018. [Online]. Available: http://arxiv.org/abs/1801.05134