Image Classification

COMP2050 Report

Khau Lien Kiet

Assignment 3
Programming Project

30/05/2022

College of Engineering and Computer Science
VinUniversity

1. DESCRIPTION

Adapted from the Convolutional Neural Network (CNN) model, the objective of this programming project has involved the extraction of features from the image for observing the patterns in the dataset to classify one object in one image. Several contributed characteristics are initially established in the model: the input format is 3x32x32 (3 colors and 32x32 pixels); there are ten class categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck); the dataset used is *CIFAR-10* that is labeled subsets of 60,000 images dataset (each class contains 6,000 images). Our team aims to reach over 80% of the expected accuracy for the image classification task.

2. PROCEDURE

2.1 Pre-processing data

At the pre-processing state, our team performs three fundamental steps before building or training the model. Firstly, we import three crucial libraries - *Tensorflow*, *Numpy*, and *Keras*. While loading and normalizing the dataset from CIFAR-10, it is necessary for us to verify the data categorized in ten class names. After loading the data, we change the data type to 32-bit float and normalize it by dividing it by 255 (data is stored in the RGB tensor), this is to make the model converge faster. To provide the numerical value, we perform one-hot encoding to convert categorical data variables for the further classification task.

2.2 Building model

When it comes to the building model, we implement some *Keras functions* to support establishing the *CNN* within its sequential model. To attain our objective, the following built-in functions are used to operate:

- *Conv2D:* learning the pattern of the image by obtaining the image features with some parameters (filters, kernel_size, strides, activation, padding)
- *MaxPooling2D:* reducing the number of processing nodes, thus decreasing their params
- *Dropout:* lessening the number of contributed nodes from the previous layer towards the recent node at the latter layer => to avoid the overfitting issue
- *Flatten:* transforming all the matrices to the vector representation for the next process
- *Dense:* implement the activation function to convert the flattened vector to the output

Due to a large amount of data and parameters, for the utilization approach, our team uses *ADAM Optimizer* for higher efficiency of computation and faster convergence with less memory requirement.

In the *CNN functions*, to build the models, there are always two convolutional neural layers in each block and the number of filters keeps doubling each time, thus going deeper in the neural network. Since moving forward in the layers, the patterns would get more complex. Hence, there are larger combinations of patterns to capture. Therefore, increasing filter numbers as going deeper in the network helps capture many combinations as much as possible.

2.3 Training model

2.3.1 Train model

Given the dataset, *CIFAR-10* provides us with 50,000 data for the training set and 10,000 data for the testing set. To specify for the training dataset, the sequential model is used 80% of those data for training, whereas 20% of those are used for the validation method.

In terms of the training model phase, our team chose five blocks as a boundary for the model training since we manually observe that when it comes to the fourth or fifth block, the training accuracy could be reduced. The more layers are added to the model, the more severe the overfitting issue can occur.

2.3.2 Tune hyperparameters

Several main parameters might be used for tunning to increase both training and testing accuracy:

- Initial filter size of convolutional: 32
- Pooling size: 2x2
- Padding: True
- Activation function
- Number of layers
- Kernel size: 3x3

To result in an improvement in accuracy, the number of layers is the chosen strategy for our illustration. Within the function `cnn(num_layers)`, we could define the optimal number of layers that derives the highest accuracy in both training and testing, which are three convolutional blocks/ number of layers.

```
def cnn(num_layers):

  model = Sequential()

  if num_layers >= 1:
      model.add(Conv2D(32, (3,3), activation = 'relu', padding = 'same', input_shape = (32,32,3)))
      model.add(Conv2D(32, (3,3), activation = 'relu', padding = 'same'))
      model.add(MaxPooling2D(pool_size = (2,2), strides=(2,2)))
      model.add(Dropout(0.3))

      if num_layers >= 2:
          model.add(Conv2D(64, (3,3), activation = 'relu', padding = 'same', input_shape = (32,32,3)))
          model.add(Conv2D(64, (3,3), activation = 'relu', padding = 'same'))
          model.add(MaxPooling2D(pool_size = (2,2), strides=(2,2)))
          model.add(Dropout(0.3))

          if num_layers >= 3:
              model.add(Conv2D(128, (3,3), activation = 'relu', padding = 'same', input_shape = (32,32,3)))
              model.add(Conv2D(128, (3,3), activation = 'relu', padding = 'same'))
              model.add(MaxPooling2D(pool_size = (2,2), strides=(2,2)))
              model.add(Dropout(0.3))

              if num_layers >= 4:
                  model.add(Conv2D(256, (3,3), activation = 'relu', padding = 'same', input_shape = (32,32,3)))
                  model.add(Conv2D(256, (3,3), activation = 'relu', padding = 'same'))
                  model.add(MaxPooling2D(pool_size = (2,2), strides=(2,2)))
                  model.add(Dropout(0.3))

                  if num_layers >= 5:
                      model.add(Conv2D(512, (3,3), activation = 'relu', padding = 'same', input_shape = (32,32,3)))
                      model.add(Conv2D(512, (3,3), activation = 'relu', padding = 'same'))
                      model.add(MaxPooling2D(pool_size = (2,2), strides=(2,2)))
                      model.add(Dropout(0.3))
```

Figure 1: Function def for `cnn(num_layers)`

2.3.3 Test with test-set

Finally, we go through “comparing the result” by plotting for both training data and testing data. As mentioned earlier, within three convolutional blocks, it derives the highest accuracy, which is approximately 84%. Moreover, we observe that the accuracy line is eventually flattened out with the epoch that reaches 90. Indeed, since the model training is partially overfitted towards the training data, there is a slight difference between the accuracy of either the validation or testing dataset and the training one (averagely around 8%).

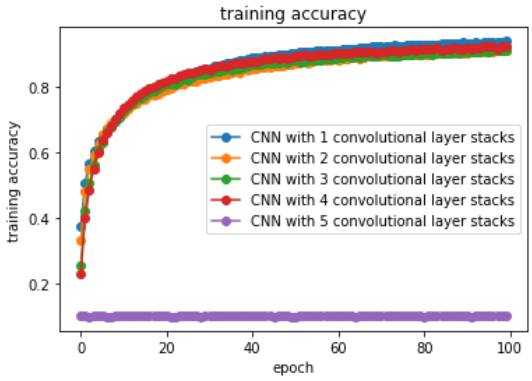

Figure 2: Training accuracy

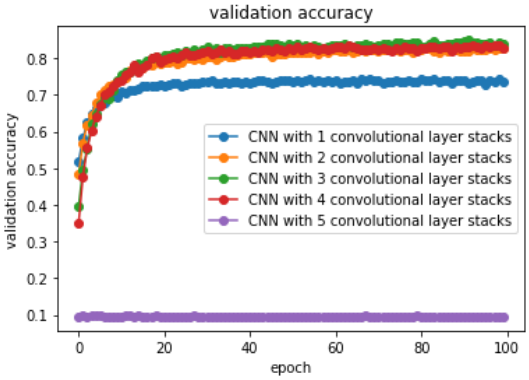

Figure 3: Testing accuracy

2.4 Experimenting with the privacy problem

This is a task that our team would further analyze in terms of the privacy tradeoff. With the existed library of Tensorflow, `tensorflow_privacy` is effective to demonstrate the implementation of differential privacy. Then, injecting the noise with the idea of *ADAM Optimizer* towards the gradient step is to maintain the privacy level for the data.

A noticeable point should be considered: the *noise_multiplier parameter* is to regulate the noise in the sample and thus add to the gradient before applying ADAM Optimizer.
After implementing the privacy aspect, we could observe the result is that there is a slight reduction in accuracy from approximately 84% to around 83%. This infers the tradeoff between utility and privacy.

2.5 Demo & Testing

In terms of this section, it is prepared for the demo day. Basically, we used the same dataset to re-evaluate and analyze how accurate our model is reached. Fortunately, every picture in that dataset is predicted correctly.

3. EVALUATION

On the programming project for an AI course, as a beginner, our team successfully classifies the image with three convolutional blocks at 84% accuracy. Indeed, we need to define a function `cnn(num_layers)`, thus determining the optimal number of layers, which creates a breakthrough in our assignment.

A further task that our team might challenge ourselves with is to deal with the privacy problem. One investigation about the correlation between ε and privacy is obtained that the smaller ε, the higher privacy. That idea would be explored in the future. However, it is obvious to analyze that there exists the tradeoff between the performance and privacy since the overall accuracy rate is decreased to 83.2%.