Machine Learning for Engineers (ECSE 511)
Mini-Project #3

Group 18

Chonghui Zhang	260955721
Jiarui Xie	260961962
Siyuan Sun	260964824

Abstract

Image classification is the task that uses labelled examples to train a model to recognize the test data in order to assign one label from the fixed set of categories. It is time-consuming so that it triggers the investigation of machine learning techniques to categorize the image dataset. The objective of this project is to predict the associated label of each image and to output the total price of all articles presented in the image. Keras convolutional neural network (CNN) with 3 convolution layers and 3 dense layers was implemented to build the model, and it used the programming language Python to perform the image classification in the project. Three convolutional layers were constructed to extract features from the image step by step. To prevent overfitting, L2 regularization was applied to all of the dense layers. The performance of the model in terms of validation loss and validation accuracy was shown at each epoch. And the work was carried out by using Python on Google Colab.

1. Introduction

Image classification is one of the typical types of supervised learning model, it uses labelled examples to train a model to recognize the test data in order to assign one label from the fixed set of categories. It is exhaustive for a human-being to categorize a large collection of image datasets, which triggers the possibility of automation method of the image classification. Therefore, the computer is trained to learn the process of image classification.

From the deep learning perspective, the image classification problem can be solved by using transfer learning [1]. A modified version of the Fashion-MNIST dataset was constructed for this project, and the objective of this project is to predict the associated label of each image and to output the total price of all articles presented in the image. The implementation proposed in this project is to build a deep learning model by using Keras convolutional neural network (CNN), which uses the programming language Python to perform the image classification. One major challenge in the keras model is that the model overfits the training data easily and do not generalize well [2]. To improve the performance of the Keras model, implementations of regularization techniques are also used, including L2 regularization and dropout regularization. These different approaches are adopted and compared in terms of training accuracy and validation accuracy. A CNN model built for fashion image classification has been successfully constructed with accuracies of 99.18%, 99.6% and 99.2% for the training, validation and pre-test set, respectively. And this model has three convolutional layers and three dense layers.

The paper is organized as follows. Section 2 describes the preprocessing steps for preparing the input data. Section 3 describes the proposed approach implemented in the project. Section 4 summarizes the results of the keras model. Section 5 summarizes the key takeaways and main discussions of the project. Section 6 concludes the statement of contributions of each team member.

2. Datasets

In the modified version of the Fashion-MNIST dataset given in the project, each image contains three articles, and from the least to the most expensive, the articles are listed as: T-shirt/top, trouser, pullover, dress, coat. The project is to identify the associated label of the image and to output the total price of all articles shown in the image. And the expected output from the CNN model is an integer number between 5 and 13. In the training dataset, 60,000 training images and the associate class label of each image were provided. And 10,000 test images were used to evaluate the performance of the model built for this project.

The process of data preprocessing in this project started from shuffling the train and test sets in order to get rid of any form of implicit sorting, since it is better and easier to use randomly sorted data sets, and the initial dataset was shuffled and saved in the ‘idx.csv’. Afterwards, the images from the training set and test set were normalized to a range of -1 to 1, which speeded up convergence and improved predictive accuracy. The pictures from the training package were divided into three portions: 90% went to the training set, 5% went to the validation set, and 5% went to the pre-test set. Then the image dataset was reshaped into 3D tensors as the input to the keras model. After the model was built, to improve the performance of the dataset, L2 regularization and dropout regularization were implemented, which also prevents overfitting in the model.

3. Proposed Approach

The proposed approaches consist of training/validation dataset split, algorithm selection and regularization. Among all the image analysis techniques enabled by machine learning, CNN is the most popular approach because it accounts for the spatial relationship between pixels and reduces the number of parameters using shared weights. The base model used in this project is the keras CNN model. Sequential was used to build the keras model layer by layer and each layer has a weight that fits the model as it follows. The 'add' function was used to add convolutional and dense layers to the model. 'Activation' is the activation function for the layer as the activation function allows nonlinear interactions to be taken into account [3]. 'ReLU' and 'Leaky ReLU' were used for the activation function. All the layers added by the "add" function are indicated in Table 1. Two parameters were used to compile the model as the optimizer controls the learning rate [3]. 'Adam' was used as the optimizer and 'categorical_crossentropy' was used for the loss function.

Table 1: Introduction to the layers used in the proposed model and their functions.

Layer type	Module name	Function
Convolution	Conv2D ( )	Adding a 2D convolutional layer that specifies the number of channels, kernel size and padding or not.
Activation	Activation ( )	Adding an activation function to the convolutional or dense layers, such as 'tanh', 'softmax' and 'relu'.
Max pooling	MaxPooling2D ( )	Adding a max pooling function in a convolutional layer that aggregates adjacent cells and only keeps that maximum number, specifying the size of the kernel.
Leaky ReLU	LeakyReLU ( )	Adding a specific type of activation function called Leaky ReLU, with the learning rate specified.
Dropout	Dropout ( )	Adding a dropout function to the convolutional or dense layer that randomly discards weights between layers with the specified dropout ratio.

The pictures from the training package were divided into three portions: 90% went to the training set, 5% went to the validation set, and 5% went to the pre-test set. The majority of data belongs to the training set because the entire dataset is large enough for its 10% to be the validation and pre-test set. The validation set was used to select the optimum model. And the pre-test set was utilized to lower down the risk of overfitting.

Figure 1 illustrates the structure of the proposed CNN model. The input data is a 64*128 grayscale picture with one channel. Three convolutional layers are constructed to extract features from the picture step by step. Each layer consists of two convolution steps with Leaky Relu, one max pooling step that shrinks the size and one dropout step with a 20% dropout rate to reduce overfitting. Padding is added to retain the size at each convolution step. It should be noted that the first convolutional layer transforms a rectangular picture into a square shape to reduce irrelevant features. It is because there are many empty spots along the horizontal axis. The feature size is reduced to 8*8*128 at the end of the convolutional layers. Then, it is

flattened to a one-dimensional dense layer with 8192 neurons followed by three partially connected dense layers. Each dense layer has an activation function of ReLU and a dropout rate of 30%. Moreover, L2 regularization is applied to each dense layer with a regularization parameter of 0.006 to avoid overfitting. The last layer is an output layer with 9 neurons, which represent the 9 classes.

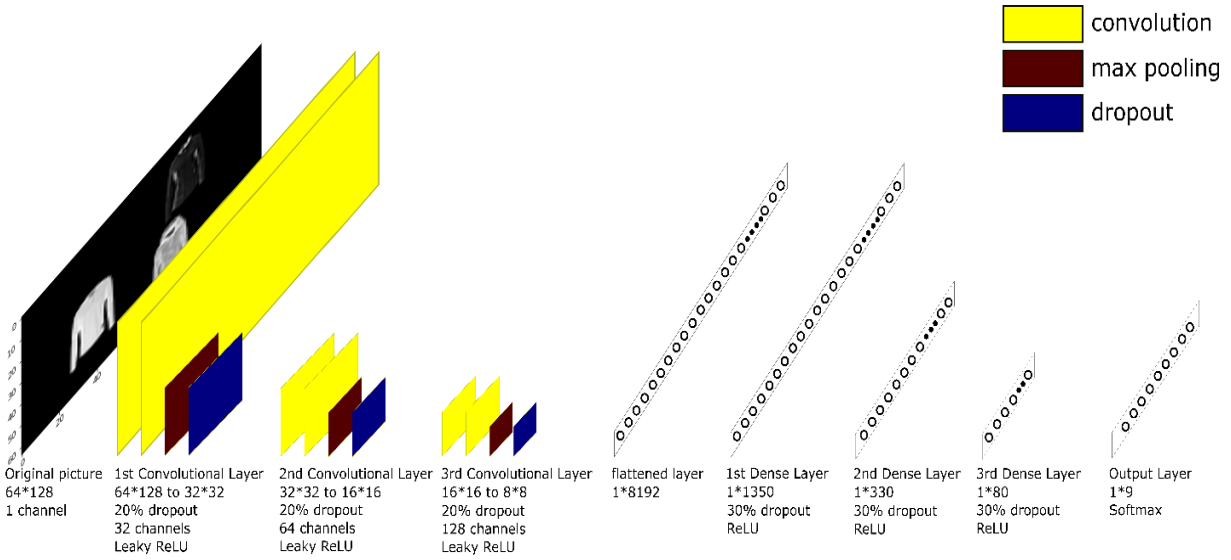


Figure 1: the structure of the CNN model.

The Adam optimizer is set up with a learning rate of 0.001 and a decay rate of 10^{-6} per epoch. The batch size is 1200. 1000 epochs will be executed. And the validation loss and validation accuracy are calculated at each epoch. The model with the minimum validation loss at the end of an epoch is stored and used as the best model for prediction. And the hyperparameters including learning rate, batch size, were chosen by test.

4. Results

Image classification in this project uses the deep learning model by applying keras CNN. One of the biggest issues with the deep learning model is that it easily overfits the training data, due to a large number of trainable parameters. To avoid the issue of overfitting, some regularization techniques are implemented. Two regularization techniques used in this project are weight regularization (L2 regularization) and dropout regularization. L1 regularization has also be tested, but the performance was not as good as L2 regularization. Following is the call that starts training the model for 50 epochs by using a batch size of 256. And 30000 training samples and 1000 validation samples were used in the model.

A common way to reduce overfitting is by putting constraints on network complexity by forcing its parameters (weights & biases) to take only small values and this process is called weight regularization. Table 2 shows the keras model performance with the base model and L2 Regularization with different parameters in terms of training accuracy and validation accuracy. According to Table 2, overfitting has been reduced compared to the base model. As the L2 loss_lambda increases, the training accuracy decreases as well as validation accuracy.

Table 2: Model Performance with L2 Regularization vs Base Model

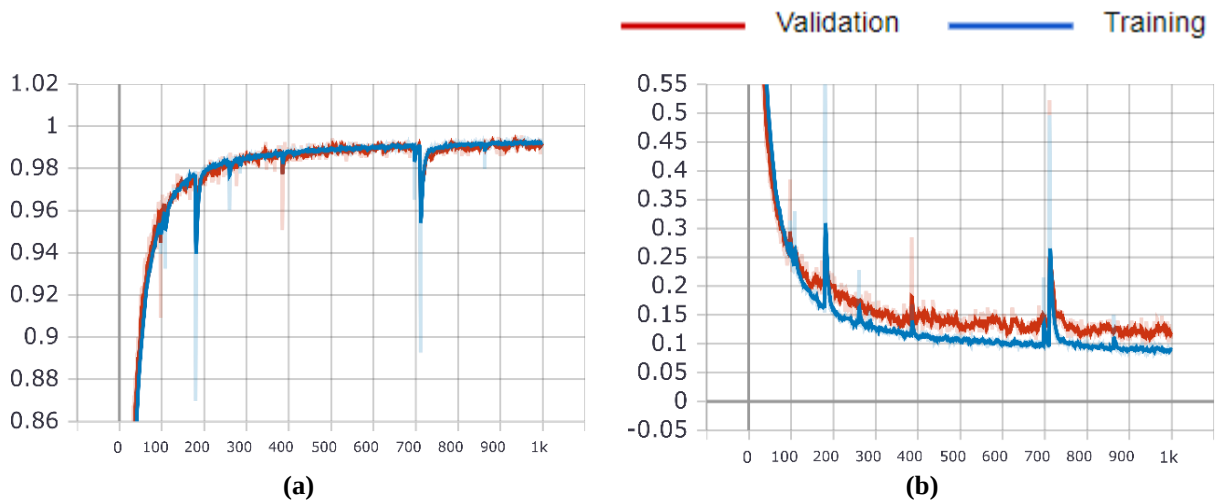
Configuration	Training Acc	Val Acc
Base Model	99%	78%
L2 loss_lambda 0.0002	93%	81%
L2 loss_lambda 0.0004	93%	87%
L2 loss_lambda 0.0006	92%	88%
L2 loss_lambda 0.002	90%	87%
L2 loss_lambda 0.005	88%	87%
L2 loss_lambda 0.01	77%	79%

Dropout is one of the most effective and most commonly used regularization techniques for neural networks. It works by randomly dropping out (setting to zero) some output features (or nodes) of a layer during training. Table 3 shows the model performance with base model and the dropout regularization with different dropout rates. The dropout rate determines that how many nodes of the previous layer would be dropped during the training stage [2]. When 20% of the nodes of previous layer were randomly dropped, the training and validation accuracy achieved the highest number.

Table 3: Model Performance with Dropout vs other configurations

Configuration	Training Acc	Val Acc
Base Model	99%	78%
Dropout Rate 0.2	99%	81%
Dropout Rate 0.4	95%	82%
Dropout Rate 0.6	94%	85%

Figure 2 shows the validation accuracy and loss for the validation set and the training set, respectively. At the 936th epoch, the validation loss reaches its minimum at 0.0936, where the training and validation accuracies are 99.18% and 99.58%. Moreover, the accuracy of the pre-test set is 99.2%. A very high training accuracy indicates that there is minimal underfitting. The pre-test set has a substantially close accuracy to the training set, which means there is no overfitting to the training set. The accuracy of validation set is higher than the other sets. There is no sign of overfitting to the validation set, either.



(a) **(b)**

Figure 2: Smoothed learning curves with respect to number of epochs: (a) accuracy; and (b) loss.

5. Discussion and Conclusion

L2 regularization and dropout regularization are implemented in the project to avoid overfitting and to improve the performance of the keras base model. By using these regularization methods, overfitting has been reduced compared to the base model. And the different results were achieved by using different values for l2 loss_lambda and dropout rates. After comparing Table 2 and Table 3, we can see that the keras model gives the best performance and highest training and validation accuracy by using just dropout regularization. According to Figure 2, the validation loss reaches the minimum number after 936 epochs, and the high training accuracy and validation accuracy indicates that there is no overfitting to the training set and the validation set. As the number of epochs increases, the learning curve of accuracy and loss converges to a constant value indicating that kreas model built in the project gives the performance as expected.

In conclusion, a CNN model built for fashion image classification has been successfully constructed with accuracies of 99.18%, 99.6% and 99.2% for the training, validation and pre-test set, respectively. This model has three convolutional layers and three dense layers. The kernel size for all layers is 4*4. The numbers of channels are 32, 64 and 128 from the first to the last convolutional layer. The numbers of neurons from the first to the last dense layer are 1350, 330 and 80. And the last layer is an output layer with 9 classes. L2 regularization is applied to each dense layer with a regularization parameter of 0.006 to avoid overfitting. And the Adam optimizer is set up with a learning rate of 0.001 and a decay rate of 10^{-6} per epoch.

6. Statement of Contributions

Jiarui Xie was responsible for the CNN model construction.

Chonghui Zhang was responsible for the analysis of regularization and dropout.

Siyuan Sun was responsible for introduction, dataset and proposed approach.

References

[1] "Solve any Image Classification Problem Quickly and Easily," KD nuggets, [Online]. Available: https://www.kdnuggets.com/2018/12/solve-image-classification-problem-quickly-easily.html. [Accessed 2 December 2020].

[2] M. Bhobe, "Classifying Fashion with a Keras CNN (achieving 94% accuracy) — Part 2," 14 July 2019. [Online]. Available: https://medium.com/@mjbhobe/classifying-fashion-with-a-keras-cnn-achieving-94-accuracy-part-2-a5bd7a4e7e5a. [Accessed 2 December 2020].

[3] E. Allibhai, "Building A Deep Learning Model using Keras," towards data science, 17 September 2018. [Online]. Available: https://towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37#:~:text=Deep%20learning%20models%20are%20built,order%20to%20make%20better%20predictions.. [Accessed 3 December 2020].