Image Classification Model

Shufeng Zhang, Xuehan Wang, Luxuan Zhang, Ruonan Zhao

April 29, 2024

Abstract

CIFAR-10 is a dataset that is used in many image recognition applications. Our current project is to build a CNN based data classification network to achieve classification on CIFAR-10 dataset.

1 Introduction

1.1 CIFAR-10

CIFAR-10 is a foundational dataset in the field of computer vision and machine learning. It was created by researchers at the Canadian Institute for Advanced Research (CIFAR) and is widely used for developing and benchmarking image classification algorithms.

The dataset consists of 60,000 images, each with dimensions of 32x32 pixels, making them relatively small compared to many other image datasets. These images are divided into 10 classes, with each class representing a different object category. Here are the classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck. Each class contains 6,000 images, split evenly between a training set and a test set. This balanced distribution ensures that models trained on CIFAR-10 are evaluated fairly across different classes.

One of the reasons CIFAR-10 is so popular is because it presents several challenges for classification algorithms. The images are relatively low resolution, making it difficult to discern fine details. Additionally, the objects in the images may appear in various poses, orientations, and lighting conditions, further complicating the classification task.

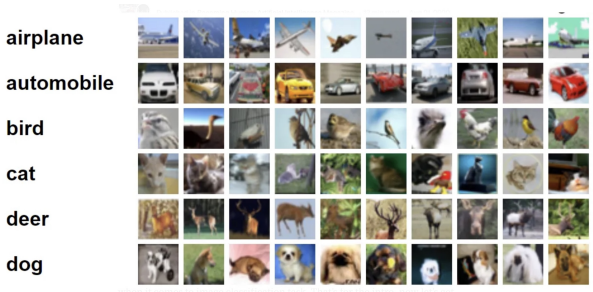

Figure 1: CIFAR-10

1.2 Implications

Since each image in the CIFAR-10 data set contains only 3072 pixels, our research can be applied to compressed images. This is especially important for surveillance data, which is subject to lossy compression algorithms in order to store several hours of footage. Since we can classify compressed images in the CIFAR-10 data set, we can generalize our findings to classify objects in compressed surveillance footage.

2 Method

2.1 Data Processing

Once the CIFAR-10 data-set is loaded, it must be normalized to a scale from 0 to 1 before being used to train the model.1 Each pixel in the data-set falls between the values 0 and 255. To accomplish normalization, we simply divide each pixel by 255.0. After normalization, an RGB value of (0,0,0) represents a black pixel and an RGB value of (1,1,1) represents a white pixel. We trained the model on Google Colab-

oratory using a free GPU hardware accelerator. Our setup offered much less computational power than required for training state-of-theart models to classify CIFAR-10 data. Therefore, our main challenge was to build and train an accurate model under the constraint of low computational power.

2.2 Training processing

We loaded the data-set using: 'from keras.data-sets import cifar10'. Keras contains a handful of data-sets including CIFAR-10.Then we did one hot encode target value. Implement a function that can compute the accuracy of our model on either the validation set or test set when necessary.Normalize training data and testing data to fit in the range, [0,1].Initialize and construct a convolutional neural network (CNN) using the Tensorflow Keras Sequential Model API: 'model = Sequential()' Compile the model with the Adam optimizer and the categorical cross entropy loss. Fit the model with a reasonable set of hyper-parameters such as batch size, learning rate, and number of epochs.Evaluate the model on the validation set and graph the cross-entropy loss and prediction accuracy.Improve upon the baseline model by repeating Steps above with dropout, data augmentation, and batch normalization. Identify the model that achieved the highest prediction accuracy on the validation set. Train the model on the unpartitioned training set and evaluate its performance on the test set.Evaluate the final model and graph the loss and accuracy.

2.3 Convolutional Neural Network

A Convolutional Neural Network (CNN) [1] is a type of deep neural network that is particularly powerful for tasks involving image recognition, processing, and classification. CNNs are inspired by the biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex.

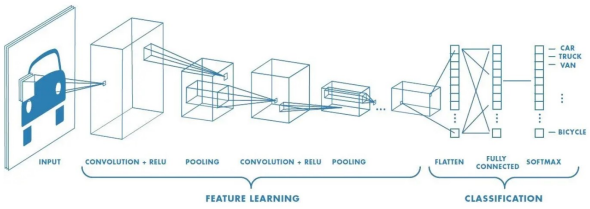

Figure 2: convolutioal neural network

2.3.1 Main parts of the model

CNN model usually contains five parts.

Convolutional Layers: These layers perform a convolution operation that involves a filter (or kernel) which is slid over the input image (or previous layer's output), calculating the dot product at each position. In our model [2], we constructed 6 layers to deal with the image classification task. Given an input image or feature map X and a filter K, the convolution operation (denoted by $*$) is defined as:

$$(X * K)(i, j) = \sum_m \sum_n X(m, n)K(i - m, j - n)$$

where $X(m, n)$ is the pixel value at position (m, n) of the image X, and $K(i - m, j - n)$ is the value at position $(i - m, j - n)$ in the kernel K. The coordinates (i, j) refer to positions in the output feature map.

Activation Functions: Typically, a non-linear activation function like ReLU (Rectified Linear Unit) [3] is applied to introduce non-linear properties to the system, helping the network learn complex patterns.

ReLu:

$$ReLu(z) = max(0, z)$$

Pooling Layers: Pooling (often max pooling) reduces the spatial dimensions (width and height,

not depth) of the input volume for the next convolutional layer. It is used to reduce the computational complexity and to achieve translation invariance in feature detection.

Fully Connected Layers: After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have full connections to all activations in the previous layer.

Softmax: It is used to classify the outputs into various classes based on the training dataset.

3 Process of Our Project

3.1 General Training Processing

We loaded the data-set using: 'from keras.data-sets import cifar10'. Keras contains a handful of datasets including CIFAR-10.Then we did one hot encode target value. Implement a function that can compute the accuracy of our model on either the validation set or test set when necessary.Normalize training data and testing data to fit in the range, [0,1] [4].Initialize and construct a convolutional neural network (CNN) using the Tensorflow Keras Sequential Model API: 'model = Sequential()' Compile the model with the Adam optimizer and the categorical cross entropy loss. Fit the model with a reasonable set of hyper-parameters such as batch size, learning rate, and number of epochs.Evaluate the model on the validation set and graph the cross-entropy loss and prediction accuracy.Improve upon the baseline model by repeating Steps above with dropout, data augmentation, and batch normalization. Identify the model that achieved the highest prediction accuracy on the validation set. Train the model on the unpartitioned training set and evaluate its performance on the test set.Evaluate the final model and graph the loss and accuracy.

3.2 Building model

We began constructing our model with three VGG (Visual Geometry Group) blocks. The architecture of a VGG is similar to that of a regular CNN. However, a VGG block implements convolutions with 3x3 kernels and a 2x2 maximum pooling. After training for our model for 100 epochs, it achieved an accuracy of 75.64% on the validation set. While this is much better than a model that randomly assigns images to classes, our model is still quite inaccurate. The model's unusually high accuracy on the training set leads us to believe that our model is over-fit.

3.3 Improvements to the model

After the situation that the model is over-fit, we decided to use regularization and data augmentation to improve our model. So we implemented dropout, a regularization technique that randomly selects certain neurons and prevents their activations from being used in the forward pass. We added a dropout layer after each VGG block and dense layer. We followed the standard convention of increasing the dropout probability by 0.1 after each block. After implementing dropout, our model's accuracy increased to 80.93% on the validation set, which is a 5.3% improvement from the baseline model. The model's accuracy on the training set is also closer to its accuracy on the validation set, leading us to believe that our model is less over fit.

But adding dropout layers enhanced our model's performance, but it still doesn't eliminate the risk of over-fitting. To further decrease the risk of over-fitting, we can augment the original data. Data augmentation involves randomly shifting, rotating, adding noise, or applying random filters to the images. This adds more variation to the data set, forcing the model to learn more sophisticated techniques for properly classifying an image.

We planned on implementing multiple image transformations to our data, but due to the computational limitations of our Google Colaboratory setup, we were forced to use only translations and

reflections. Using additional filters would have required the model to be trained over at least 200 epochs. However, with our current setup, our model could only train on the GPU accelerator for approximately 150 epochs.

After implementing data augmentation, the prediction accuracy of our model increased to 82.99% on the validation set. This is roughly a 2% improvement upon the regularized model.

As a neural network gets deeper and more complex, it can be highly sensitive to the random initialization of weights during training. To improve upon the accuracy from the regularized model with data augmentation, we implemented a batch normalization layer after each convolutional and dense layer. Batch normalization is a method for standardizing the activation of a single layer across a batch. The model performs batch normalization a total of seven times in a single forward pass. After implementing batch normalization, our model achieved a prediction accuracy of 88.58validation set. This is an improvement of approximately 5.5Due to the technical limitations of our setup, we were only able train the model over 150 epochs.

4 Results

In our final modified model, we trained 150 epochs and the results of cross-entropy loss and accuracy are shown in Figure 3. We trained the regularized model

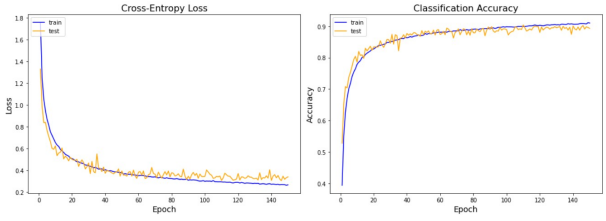

Figure 3: training process

with data augmentation and batch normalization on the complete, unpartitioned training set and evaluated its performance on the test set. The model was still trained over 150 epochs using the Adam optimizer. It achieved an accuracy of 89.31While our model achieved close to a 90setup, it could have achieved an accuracy well over 90Our free subscription of Google Colaboratory only offered as single GPU as hardware accelerator. Furthermore, we were allotted a relatively small amount of RAM and time on the hardware accelerator to train our model. This resulted in us training a relatively simple model with minimal data preprocessing over a small number of epochs.

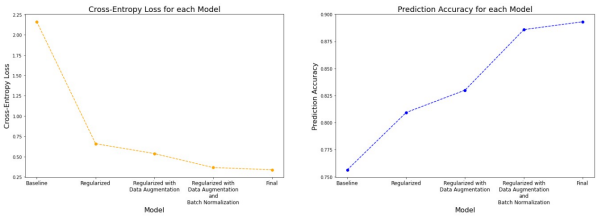

Figure 4: Comparison of results of improved models at each step

5 Conclusion

The CNN model can achieve good results on the CIFAR-10 dataset, but if the original model is used as the basis for direct training, there will be problems such as overfitting, so we need to improve the data, the model and the training process on this basis to achieve the best results. There have been many improvements based on cnn, our improvement is just the tip of the iceberg, in the future, we will learn more scholars' ideas so that the model can achieve better results.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[2] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural

networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[3] Evgeny A Smirnov, Denis M Timoshenko, and Serge N Andrianov. Comparison of regularization methods for imagenet classification with deep convolutional neural networks. *Aasri Procedia*, 6:89–94, 2014.

[4] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.