Computer Vision - Homework 6

Ruchi Manikrao Dhore - W1652116

Monday 6th March, 2023

1 Introduction

We will create Conv2D and pooling layers from scratch for this challenge. The CIFAR-10 dataset will be used to train a neural network for image recognition utilizing these layers. We will also use the back-propagation method designed especially for Conv2D. We will track and plot the training loss curve over the course of our 20-epoch training process, as well as keep track of how the Conv2 filter bank weights are distributed across each epoch.

2 Input

As an input, 10 images were taken from each of the 10 classes from the CIFAR-10 dataset. The CIFAR-10 provides tiny images of the dimensions 32x32 and image classes like airplane, automobile, bird, cat etc. Figure 1 shows the original set of the images.

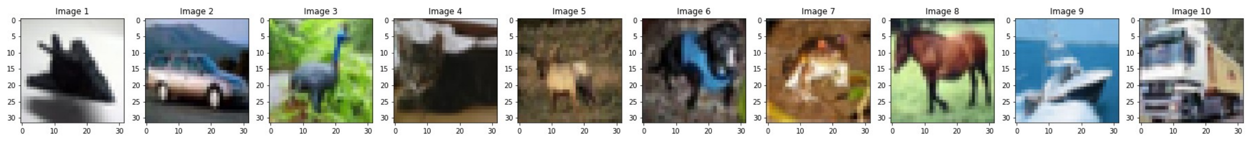


Figure 1: CIFAR-10 dataset original images

3 Conv2D layer

The conv2dCV class in Python uses numpy to define the Conv2D layer, which is frequently used in deep learning models for image processing. The init function of this class randomly initializes the weights of the filter banks while taking into account the number of input channels, output channels, kernel size, stride (which is set to 1), and padding (which is set to 0). The forward method applies convolution to an input tensor X by padding it with zeros, loops over the height and breadth dimensions of the output tensor using the filter bank weights, and returns the output tensor. The input tensor X, the gradient of the loss with respect to the output tensor, and the learning rate are input into the backward method, which computes the gradient of the loss with respect to the input tensor and the filter bank weights. The weights are updated, and the gradient of the loss with respect to the input tensor is returned.

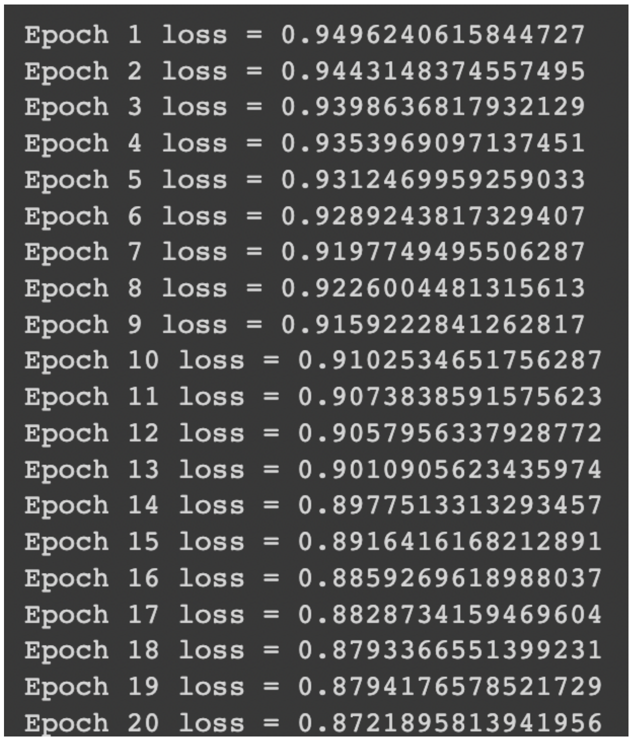
4 2D Max Pooling Layer

The 2d class maximum pool is used for the purpose of downsampling feature maps in convolutional neural networks, CV: defines a 2D max pooling layer. The pool size (also known as the pooling window size) and stride are set by the init method to initialize the pooling layer. The pooling layer's forward pass is carried out using the forward approach. It accepts an input tensor of shape X and, after performing the maximum pooling operation on each channel of the input tensor, returns an output tensor of shape. Based on the input dimensions, pool size, and stride parameters, the output height and width are calculated. The backward technique computes the gradient of the loss with respect to the input tensor while performing the backward pass of the pooling layer. The gradient of the loss relative to the output of the pooling layer and the initial input tensor X are inputs. It gives back the loss' gradient relative to the input tensor. By using the chain rule and propagating the gradient through the max pooling procedure, this is calculated. The gradient is only back-transmitted to the element that produced the highest value in the forward pass for each element in the input tensor.

5 Network Training

With the help of Keras, we design and train a neural network model to categorize images from the CIFAR 10 dataset. A last layer with no activation function follows the two fully connected layers with ReLU activation. The train step function specifies a single training step that uses the automated differentiation feature of Tensor Flow to compute the forward pass of the model and the gradient of the loss with respect to the trainable variables. After that, the optimizer is used to update the trainable variables. The train function defines the training loop, which calls train step for each batch of data and loops across the dataset for a specified number of epochs. Each epoch's loss history is kept track of and plotted at the conclusion. The data is loaded and normalized prior to training. For convenience of usage during training, the training data is subsequently transformed into a TensorFlow Dataset object. The model is then trained for a total of 20 epochs at a learning rate of 0.01.

Using plt.hist, a histogram of the filter weights is produced for each epoch (). Bins=50 sets the histogram's bin count to 50, and alpha=0.5 improves visualization by setting the histogram bars' transparencies to 0.5. Each histogram bar is labeled with the relevant epoch number using the label argument.



```
Epoch 1 loss = 0.9496240615844727
Epoch 2 loss = 0.9443148374557495
Epoch 3 loss = 0.9398636817932129
Epoch 4 loss = 0.9353969097137451
Epoch 5 loss = 0.9312469959259033
Epoch 6 loss = 0.9289243817329407
Epoch 7 loss = 0.9197749495506287
Epoch 8 loss = 0.9226004481315613
Epoch 9 loss = 0.9159222841262817
Epoch 10 loss = 0.9102534651756287
Epoch 11 loss = 0.9073838591575623
Epoch 12 loss = 0.9057956337928772
Epoch 13 loss = 0.9010905623435974
Epoch 14 loss = 0.8977513313293457
Epoch 15 loss = 0.8916416168212891
Epoch 16 loss = 0.8859269618988037
Epoch 17 loss = 0.8828734159469604
Epoch 18 loss = 0.8793366551399231
Epoch 19 loss = 0.8794176578521729
Epoch 20 loss = 0.8721895813941956
```

Figure 2: Epoch and Loss

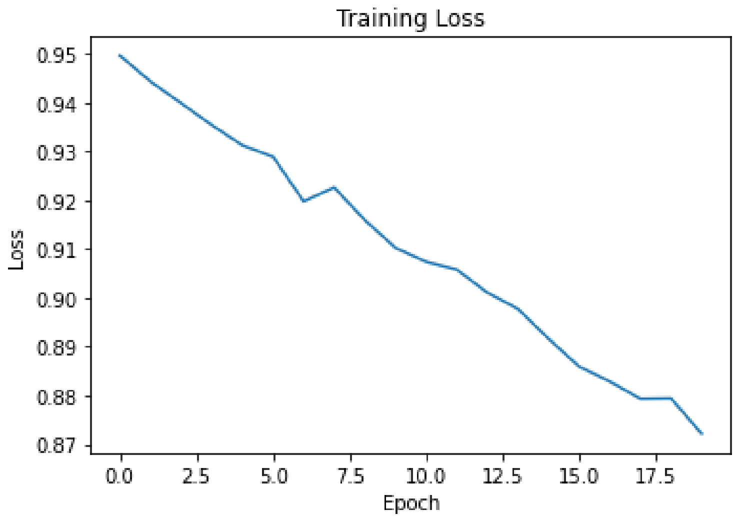


Figure 3: Training Loss

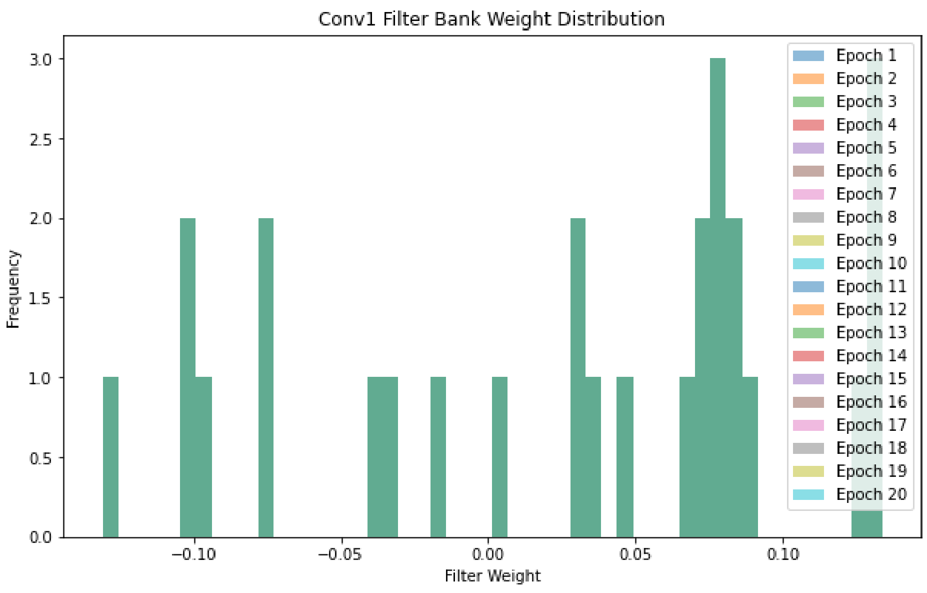


Figure 4: Filter Bank Weight Distribution

6 Conclusion

Using the CIFAR 10 dataset, we trained a convolutional neural network (CNN) to categorize pictures. Pre-processing the data and dividing it into training and validation sets was our first step. Then, using the Keras API, we created a CNN model with two convolutional layers and two fully connected layers. We kept track of the loss and accuracy metrics while training and recorded the second convolutional layer's filter bank weights to track how they changed over time. Also, we charted the distribution of the filter bank weights across epochs and the learning curves for the loss and accuracy during training.