Convolutional Neural Network on CIFAR-10

Mihai Bojescu
Master in Artificial Intelligence and Optimisation
Faculty of Computer Science of University "Alexandru Ioan Cuza" of Iași
Iași, Romania
bojescu.mihai@gmail.com

***Abstract*—This document contains details on a Convolutional Neural Network such as its structure, how each layer contributes to the output, how the forward function works and how the gradient flow works. The document also details the experiments performed using the given neural network.**

***Index Terms*—neural network, convolution, convolutional neural network, nn, cnn, gradient flow, forward function**

I. Introduction

This document details the structure of the Convolutional Neural Network used in this project, layer initialisation, the steps that are performed in the $forward$ function, the gradient flow of the project, results of the project and a comparison between the current project and the previous one. This project is built on top of the previous one, which would mean this project is a continuation of the work and refinement of the previous project.

II. Layers

The Convolutional Neural Network model in this project contains 8 layers:

1) 3 convolutional layers
2) 3 max-pool 2d layers
3) 2 dropout layers
4) 2 fully-connected layers

In the following subsections, each layer will be sumarised.

A. Convolutional layers

The convolutional layers play the most important role in improving the accuracy of the model. They account to an increase in the accuracy (compared to the previous project) of 34.5%. Due to the way they work, they are quintesential to building accurate models that can be used on image-datasets such as CIFAR-10 or MNIST.

The network includes 3 convolutional layers - $self.conv1$, $self.conv2$, $self.conv3$ - in the hopes that this would drive the accuracy higher. After a convolutional layer is applied, the results go through a MaxPool2d layer. The number of layers was chosen as a compromise between speed and accuracy.

B. MaxPool2d layers

The results of the convolutions are passed through MaxPool2d layers in order to scale-down the data. This makes the data faster to process.

The project's model contains 3 MaxPool2d layers, one for each convolutional layer.

C. Dropout layers

Dropout layers contribute to regularization by introducing randomisation during training, preventing the model from relying too heavily on specific activations. The stochastic nature of dropout contributes to a more robust and generalizable model.

In this project, 2 dropout layers - $self.dropout1$, $self.dropout2$ - were added for this reason.

D. Fully-connected layers

The fully-connected layers in this project's model take the flattened results of the convolutions, and multiplies them with each the matrix of each layer. The results are passed through a ReLU activation function, and then are passed to the dropout layers.

In this project, 3 fully-connected layers - $self.fc1$, $self.fc2$, $self.fc3$ - are used.

III. Initialisation

The model can be initialised in two ways:

1) randomly, when in training mode
2) using pre-existing weights and biases, when running in inference mode

A. Random initialisation

When the model is run in training mode, the weights and biases of the layers are selected at random. Since the fully-connected layers are passed through a ReLU activation function, they are initialised using the Kaiming Uniform initialisation function.

B. Initialisation using pre-existing weights and biases

The model was trained multiple times. In order to be evaluated, a Python inference script - $inference.py$ - ran by a bash script - $run\_inference.sh$ - was added. The scripts preloads weights and biases available from previous training sessions, thus, the weights and biases are not random.

IV. The forward function

The forward function of the model runs in multiple steps:

1) ensure the data is on the specified device
2) pass the data through the 3 max-pooled convolutional layers
3) flatten the data in order to pass it to the fully-connected layers

4) pass the data through the 3 ReLU-activated-fully-connected layers, while running dropout between each layer

The steps were explained in the *Layers* section above.

V. Gradients flow

A. Forward pass

The input data is fed through the convolutional layers, activation functions, and pooling layers during the forward pass.

Dropout layers, such as self.dropout1 and self.dropout2, are applied after the first and second fully connected layers, respectively.

The dropout layers randomly set a fraction of the activations to zero during training. This helps prevent overfitting by introducing stochasticity.

B. Backward pass

During the backward pass, the gradients are computed with respect to the loss.

The dropout layers operate differently during training and testing. During training, the dropout layers randomly zero out activations, so during backpropagation, only the non-zeroed activations contribute to the gradient.

During testing, dropout layers are usually turned off or scaled to account for the dropout probability.

VI. Results

In tabular form, the results would be the following:

Run	Epochs	Training accuracy	Validation accuracy
1	25	91.77%	41.98%
2	25	94.15%	77.45%
3	25	96.60%	75.66%
4	25	97.45%	78.24%
5	25	97.54%	77.82%
6	25	94.84%	77.21%
7	25	96.87%	74.31%
8	25	97.60%	77.84%
9	25	97.99%	77.69%
10	25	95.92%	77.36%
11	25	96.81%	77.12%
12	25	97.50%	74.66%
13	25	97.63%	76.83%
14	25	97.50%	74.66%
15	25	93.25%	76.58%

Fig. 1: Results

While the timings are the following:

Process	Epochs	Runs	Best time	Worst time
Base model (CPU)	25	2	2324s	2413s
Base model (CUDA)	25	3	150.3s	151.1s
Compiled model (CUDA)	25	3	160.3s	165.9s
Traced model (CUDA)	25	3	144.5s	148.8s
Scripted model (CUDA)	25	3	145.1s	149.8s

Fig. 2: Timings

VII. Result comparison between the current project and the previous one

This section will perform a comparison between the two projects. The current project is built on top of the previous one, thus it can be said this is an iteration over the previous project. As a comparison, this would be the results:

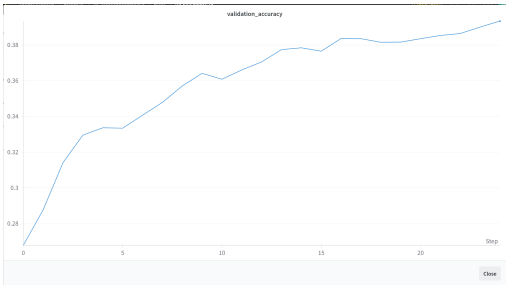

Fig. 3: Validation accuracy of the previous project

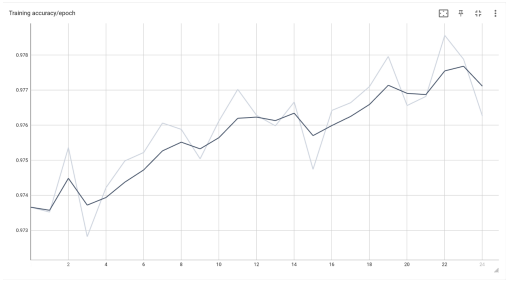

Fig. 4: Validation accuracy of the current project

The charts present the best-achieved results for each project. The difference is a staggering 38.5%, which should prove how important convolutional and dropout layers are.

VIII. Conclusions

Convolutional layers are quintesential to building accurate neural network models that will be used on image classification. They are a rather time-expensive solution especially when training, but their usage pays-off compared to projects that don't use them.