Marco Galindo
ID: 310540013
2021, December 2nd
Deep Learning Fall 2021

Homework 2

1. Section 1 – MNIST Dataset

1.1. Network architecture and results

1.1.1. Model

In Table 1 the final configuration used for the convolutional network model implemented in the experiments can be seen. The inputs of the model are the 28x28 pictures provided by the MNIST dataset.

Table 1 Neural network specifications

Layer type	**Layer's details**	**Output Shape**
Convolutional input layer	4 filters Filter size: 5x5 Stride: 1x1 Padding: Same	28,28,4
Max pooling layer	--	14,14,4
Convolutional layer	8 filters Filter size: 5x5 Stride: 1x1 Padding: Same	14,14,8
Max pooling layer	--	7,7,8
Convolutional layer	16 filters Filter size: 5x5 Stride: 1x1 Padding: Same	7,7,16
Flatten layer	--	784
Fully connected layer	64 neurons	64
Fully connected output layer	10 neurons	10

The training hyperparameters are presented in the Table 2.

Table 2 Training hyperparameters

Optimizer	Adam
Loss	Sparse Categorical Cross Entropy Loss
Epochs	10
Activation of layers	ReLU

1.1.2. Stride and filter sizes

The output size of each convolutional layer is determined by the following equation:

$$Output\ size = \frac{Input\ size\ +\ padding\ -\ filter\ size}{stride} + 1$$

To test different filter sizes and strides, some padding had to be applied to counter the rapidly decreasing output size of the layers. A "same padding" was applied to the convolutional layers, which means that the padding's size was calculated so that the output size is equal to the input size.

In Table 3 we can see the results of different configurations of filter sizes and strides for the model. The rest of the hyperparameters remained the same as those described in Table 2.

Table 3 Different filter sizes and strides configurations, from best to worst performance

Filter size	Strides	Padding	Loss	Accuracy	ΔAccuracy
5x5	1,1	Same	0.0855	98.14%	---
3x3	1,1	None	0.0808	97.54%	-0.60%
7x7	1,1	Same	0.1078	97.49%	-0.65%
5x5	2,2	Same	0.0993	96.94%	-1.20%
7x7	3,3	Same	0.1412	95.59%	-2.55%
3x3	2,2	Same	0.1444	95.26%	-2.88%
3x3	3,3	Same	0.4314	85.43%	-12.71%

Initially, a 3x3 filter and a stride 1,1 was selected, but after executing these tests, the best combination turned out to be a filter of size 5x5 and a stride of 1,1. Increasing the stride was only hurtful to the model's performance, indicating that the model benefits from having more spatial information of the nearby pixels by having a slightly bigger filter size and small stride.

1.1.3. Learning curve plot and Accuracy of training and test sets

After defining the best configuration for stride and filter size, the following results and plots were obtained. The training loss and model accuracy for the test and training data can be observed in Figure 1.

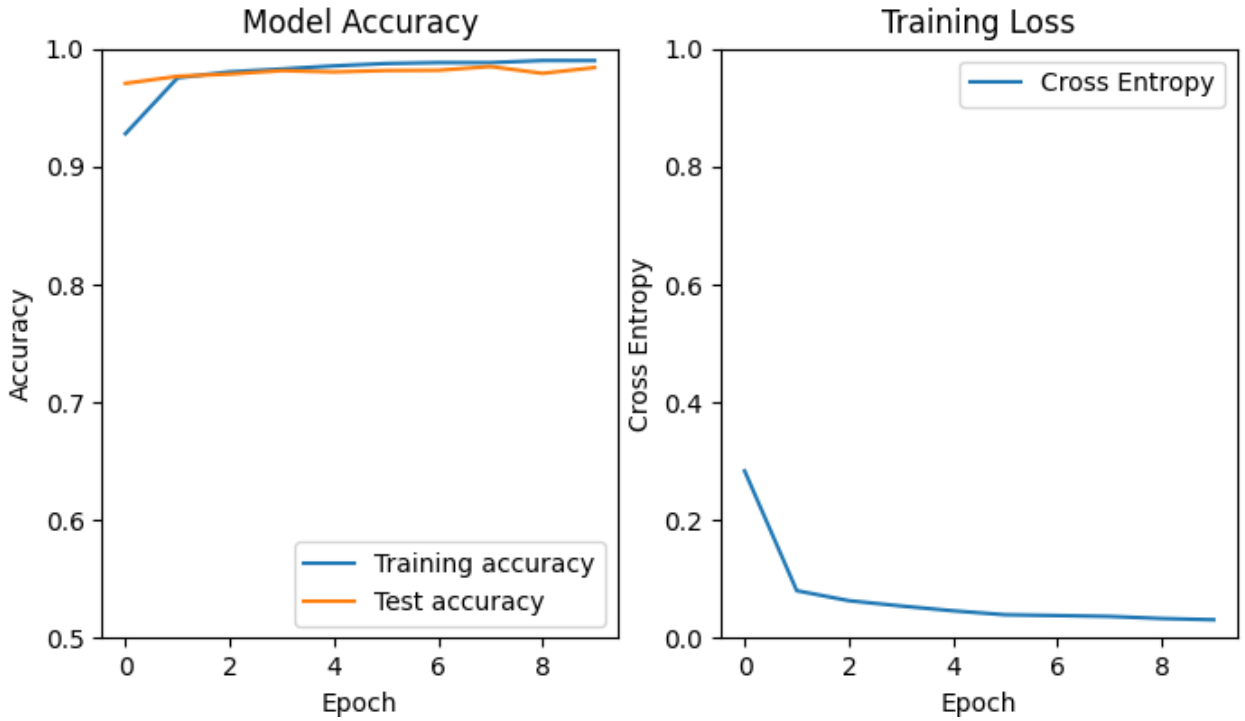


Figure 1 Model accuracy and training loss

The results for the test set were 0.0855 and 98.14% for the loss and accuracy respectively.

1.1.4. Distribution of weights and biases

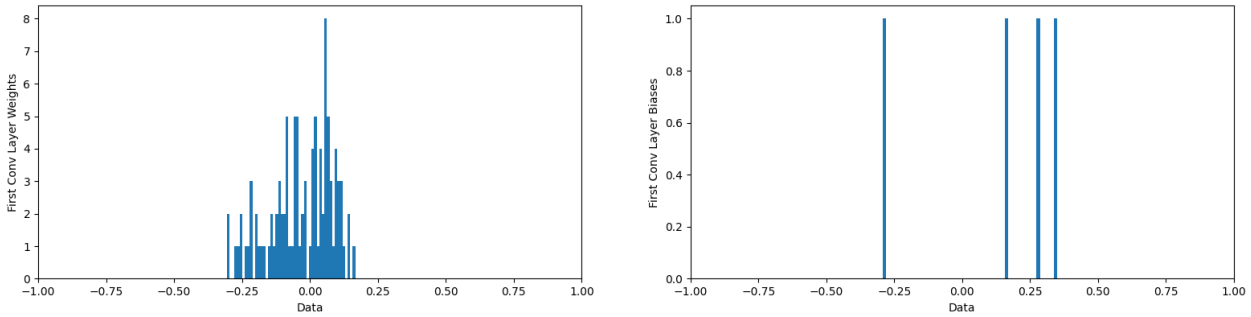


Figure 2 Weights and biases of the first convolutional layer

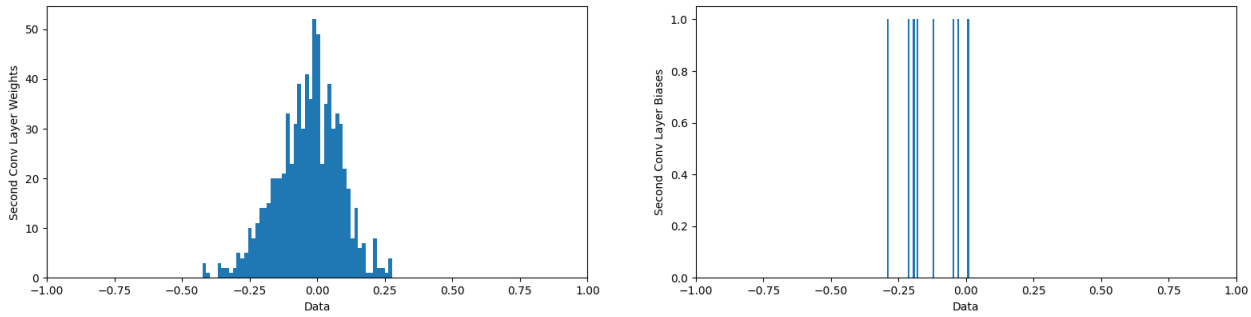


Figure 3 Weights and biases of the second convolutional layer

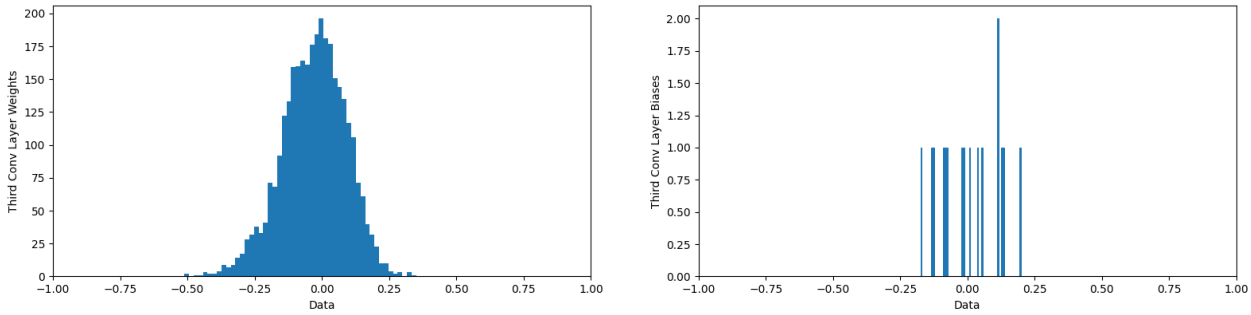


Figure 4 Weights and biases of the third convolutional layer

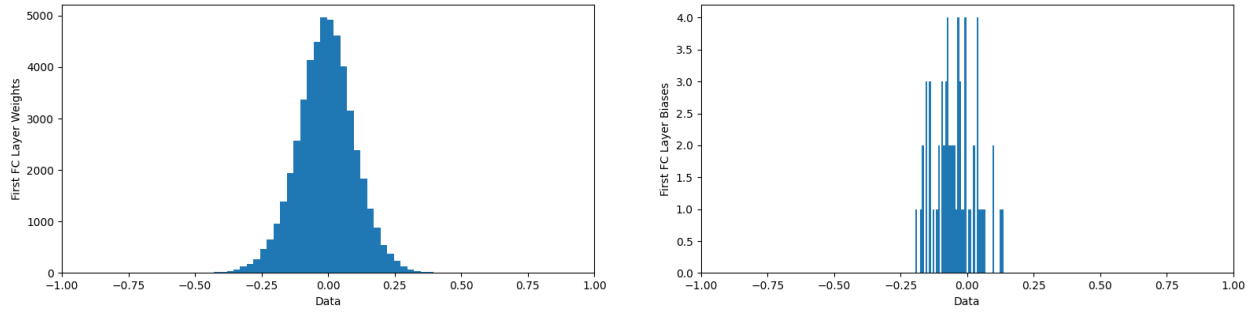


Figure 5 Weights and biases of the first fully connected layer

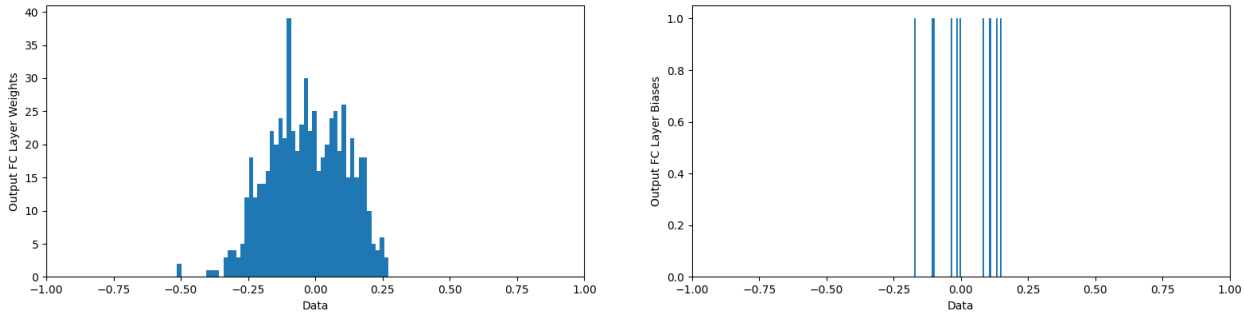


Figure 6 Weights and biases of the output layer

1.2. Examples

1.2.1. Correctly classified examples

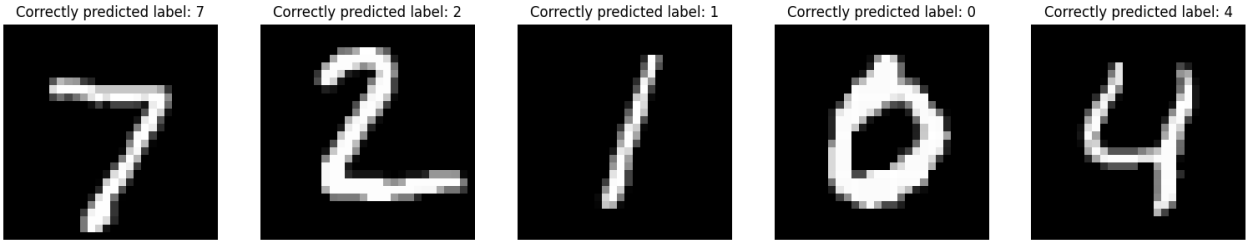


Figure 7 Five correctly predicted images from the test data

1.2.2. Incorrectly classified examples

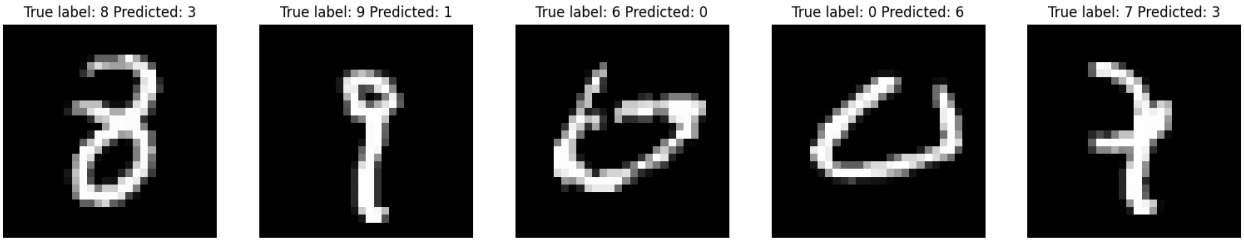


Figure 8 Five incorrectly predicted images from the test data

1.2.3. Discussion

As expected, the images that were incorrectly classified (seen in Figure 8) are images were the digits are not as well written as those correctly classified (seen in Figure 7), for example, the third digit from Figure 8 is meant to be a number 6 but it resembles a number 0 in a way that even humans may mistake it sometimes. The fourth image in the mislabeled numbers is a 0, mistakenly labeled as a number 6, which can be understandable because the digit in the image is not even fully closed, making it a lot more different from the rest of training data that represent a number 0. The first, second and fifth mislabeled numbers are not extremely hard images, but they are still examples of digits not clearly written.

1.3. Feature maps

The following feature maps represent the activations of the different filters in the convolutional layers of the model. We can observe that as we plot deeper layers, the filters capture bigger areas of the image, not focusing so much on minor details but on larger features. This resembles the effect of neural networks that only have fully connected layers, where we know that the first layers capture simple details such as lines and deeper layers recognize more complicated patterns as faces or body parts.

The feature maps presented in Figure 10, Figure 11 and Figure 12 are the representations of the input image shown on Figure 9.

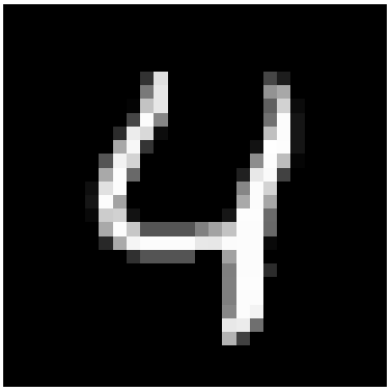


Figure 9 Image used in the feature maps plotting

1.3.1. First convolutional layer

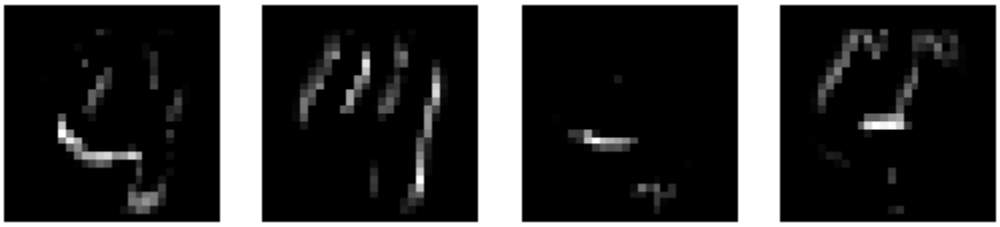


Figure 10 Feature map of the first convolutional layer

1.3.2. Second convolutional layer

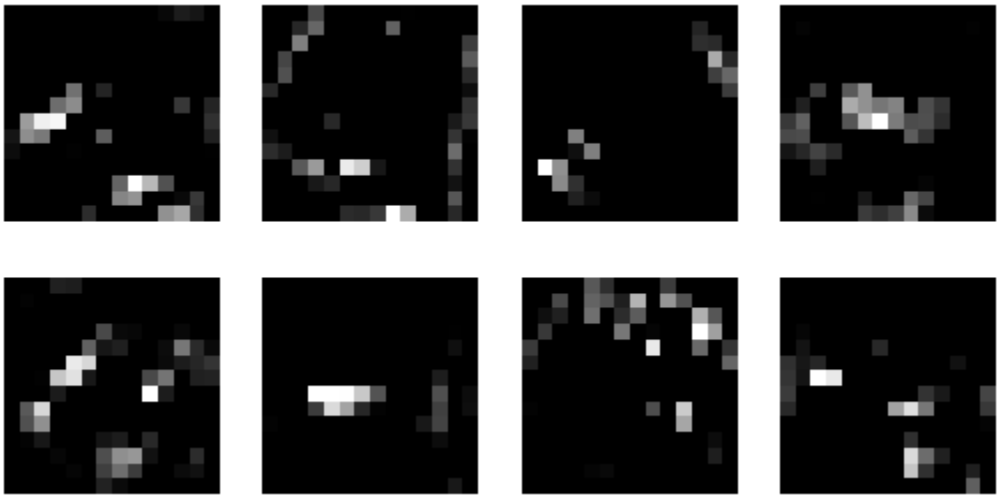


Figure 11 Feature map of the second convolutional layer

1.3.3. Third convolutional layer

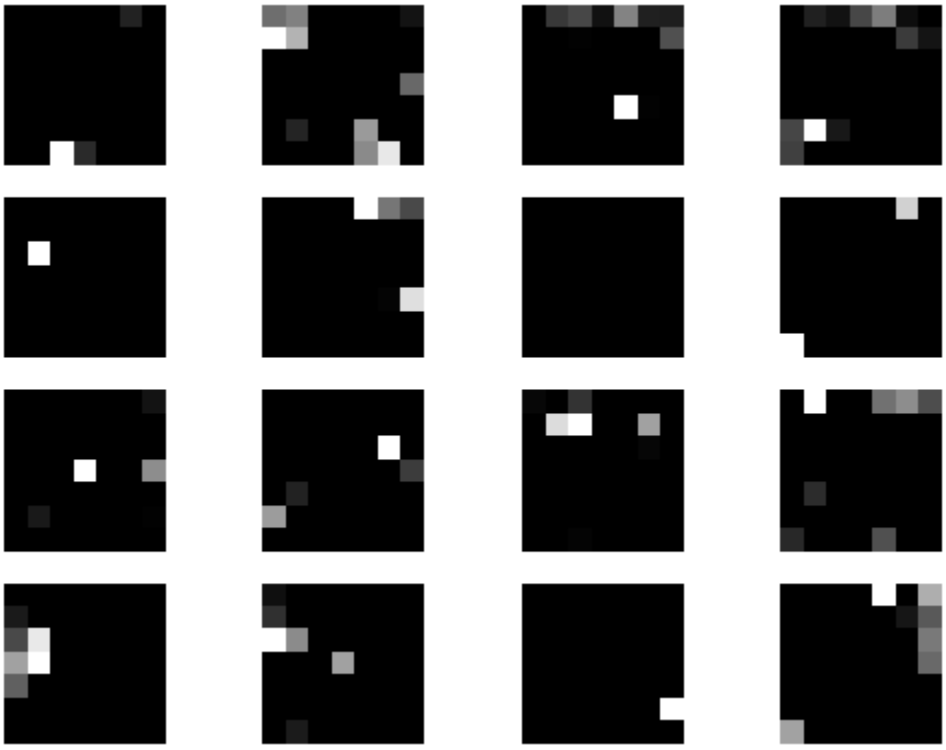


Figure 12 Feature map of the third convolutional layer

1.4. Regularization

After adding L2 regularization to all the layers, the model was able to improve its generalization to unseen data, by slightly improving the great result it had already achieved. The new model has a final loss and accuracy of 0.1263 and 98.44% respectively and it can be seen in Figure 13 that the model accuracy for the test data was always above the accuracy for the training data, due to the regularization applied to the layers. Obviously, at this level of accuracy, improving the model is already a difficult task, as it approaches 100% of accuracy, so this level of improvement is also a good thing.

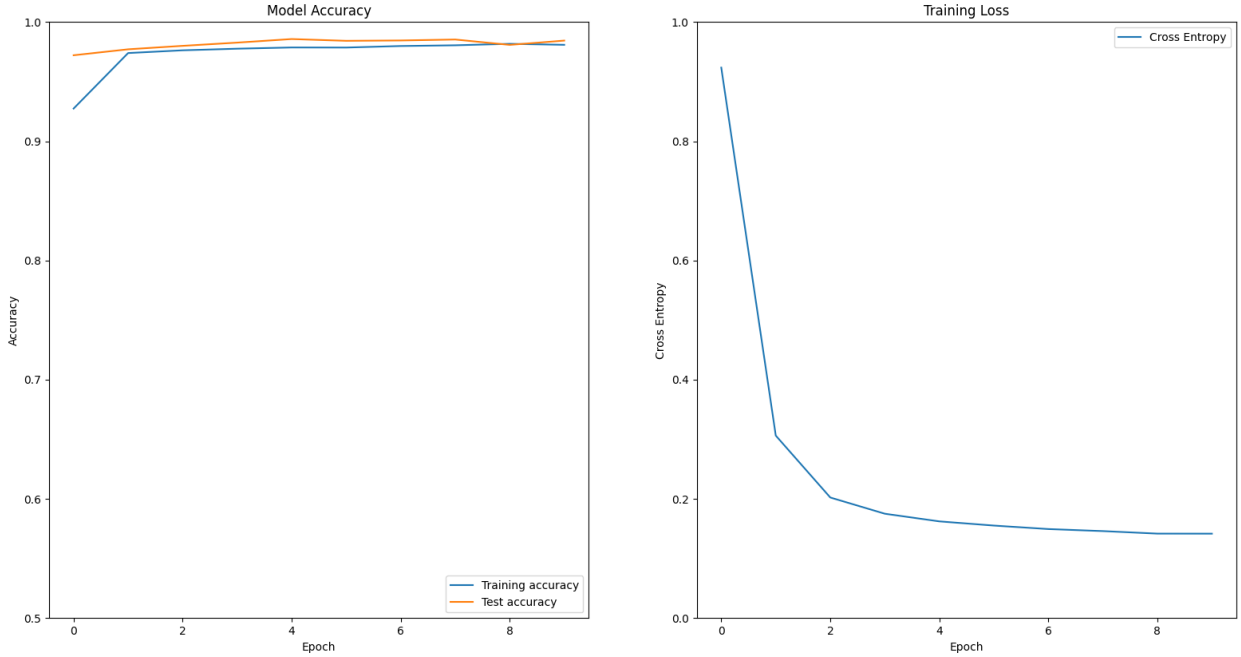


Figure 13 Accuracy and loss with regularization

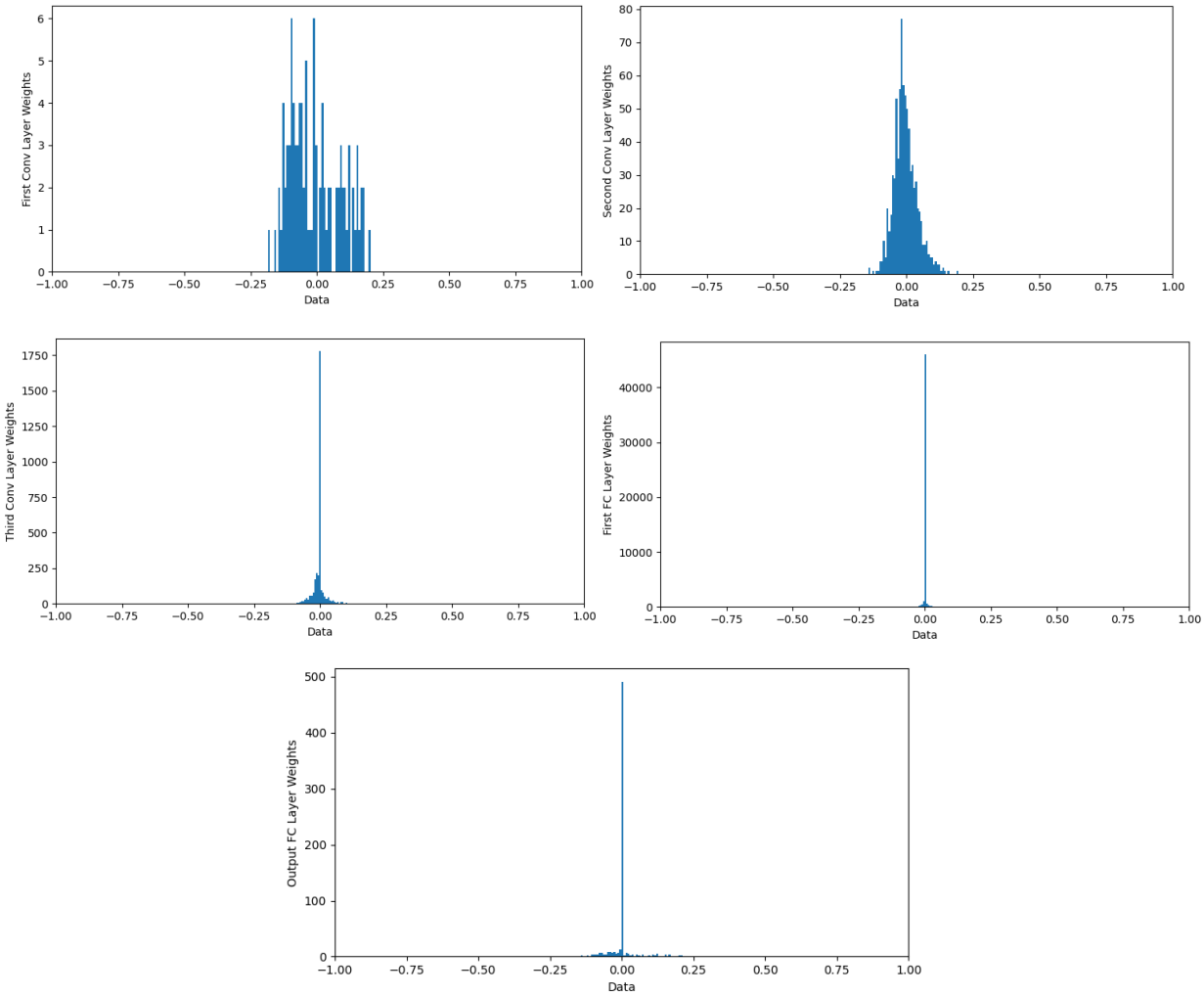


Figure 14 Weights' histogram of all the layers after adding L2 regularization

2. Section 2 – CIFAR-10 Dataset

2.1. Network architecture and results

2.1.1. Model

In Table 4 the final configuration used for the convolutional network model implemented in the experiments can be seen. The inputs of the model are the pre-processed pictures provided by the CIFAR-10 dataset.

Table 4 Configuration of the neural network for the CIFAR-10 Dataset

Layer type	**Layer's details**	**Output Shape**
Convolutional input layer	16 filters Filter size: 3x3 Stride: 1,1 Padding: Same	32,32,16
Max pooling layer	--	16,16,16
Convolutional layer	32 filters Filter size: 3x3 Stride: 1x1 Padding: Same	16,16,32
Max pooling layer	--	8,8,32
Convolutional layer	64 filters Filter size: 3x3 Stride: 1x1 Padding: Same	8,8,64
Flatten layer	--	4096
Fully connected layer	64 neurons	64
Fully connected layer	128 neurons	128
Fully connected output layer	10 neurons	10

The training hyperparameters for this section are presented in the Table 5.

Table 5 Hyperparameters for the CIFAR-10 problem

Optimizer	Adam
Loss	Sparse Categorical Cross Entropy Loss
Epochs	15
Activation of layers	ReLU

2.1.2. Stride and filter sizes

Similar to section 1, different stride and filter sizes were tested for this particular dataset and the results can be seen in Table 6.

Table 6 Different stride and filter sizes for the CIFAR-10 problem

Filter size	Strides	Padding	Loss	Accuracy	ΔAccuracy
3x3	1,1	Same	1.1179	71.20%	------
3x3	2,2	Same	1.4857	69.08%	-2.12%
5,5	2,2	Same	1.1143	62.54%	-8.66%
5,5	1,1	Same	1.0841	61.33%	-9.87%

The configuration that provided the best results was the first one tested, with a filter size of 3x3 and a stride of 1,1.

2.1.3. Learning curve plot and Accuracy of training and test sets

In Figure 15 one can observe the training loss and model accuracy for the test and training data. In the model accuracy for the test and training data, it is noticeable that approximately after epoch 3, even though the training accuracy increases, the test accuracy remains similar, indicating that the model is incapable of improving its generalization to new unseen data and is overfitting the training set.

Some type of regularization is known to mitigate the overfitting problem, as we can see in the section 2.4, where the training and test accuracy are more aligned after implementing an L2 regularization to the model.

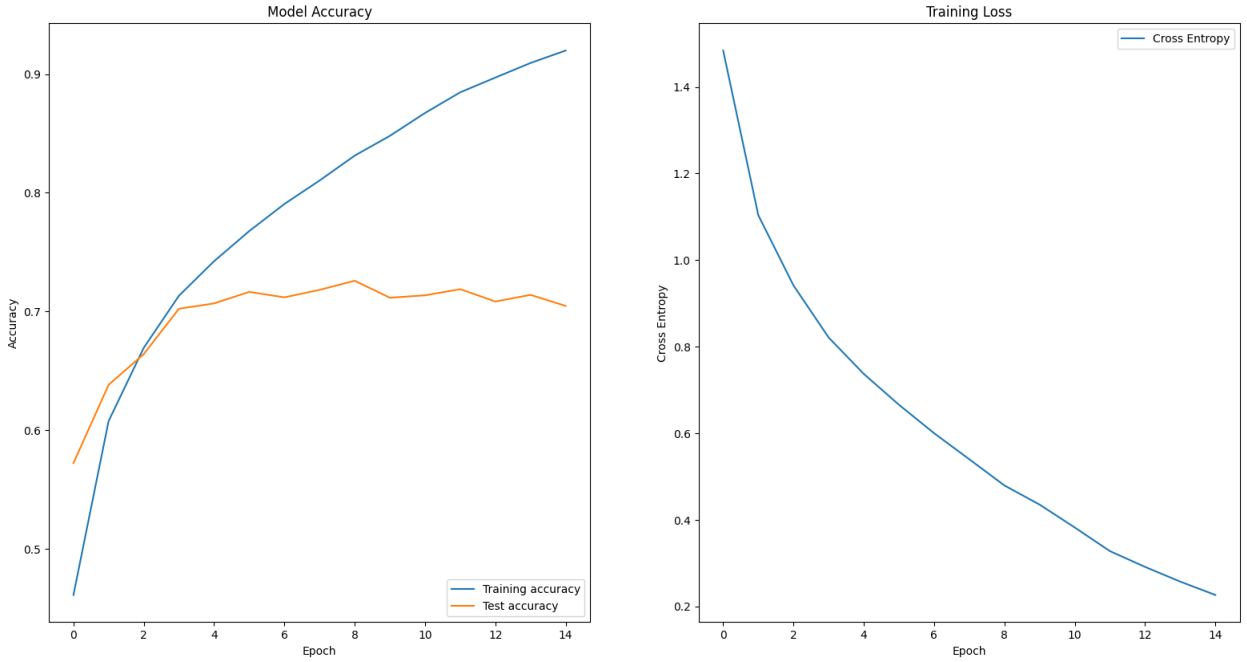


Figure 15 Model accuracy and training loss for the CIFAR-10 problem

The results for the test set were 1.1179 and 71.20% for the loss and accuracy respectively.

2.1.4. Distribution of weights and biases

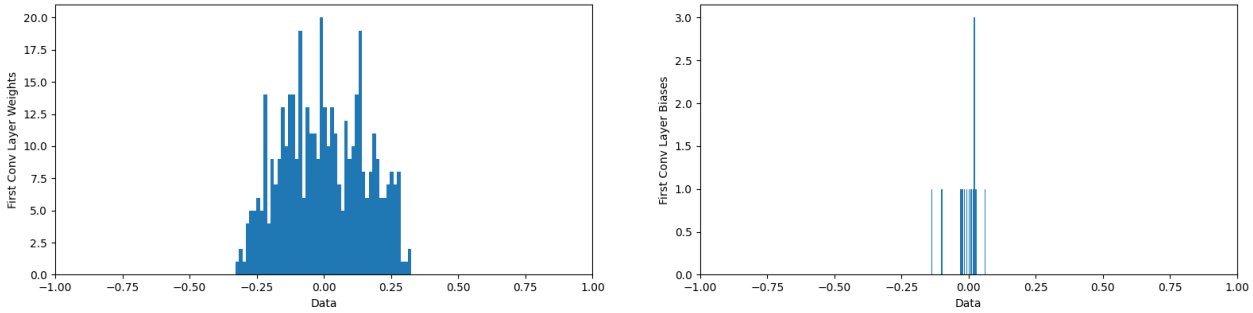


Figure 16 Weights and biases of the first convolutional layer

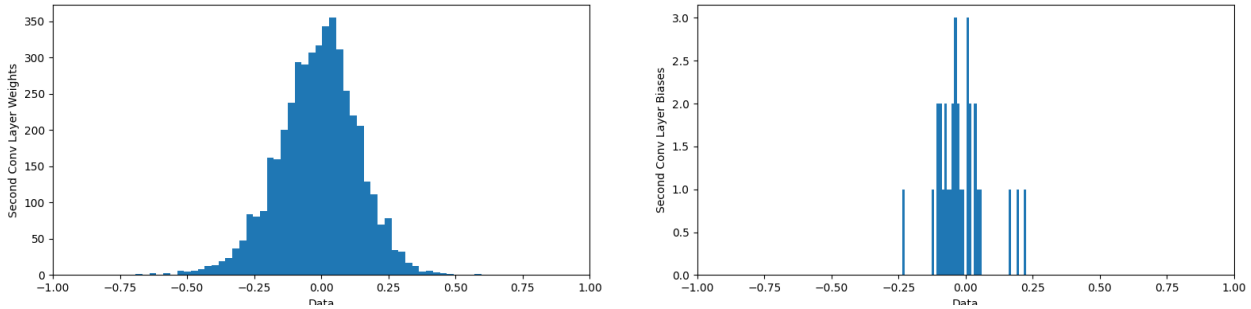


Figure 17 Weights and biases of the second convolutional layer

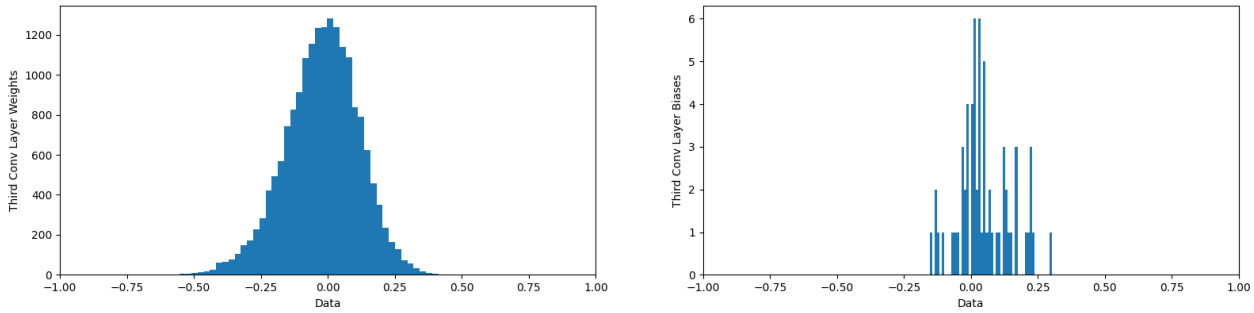


Figure 18 Weights and biases of the third convolutional layer

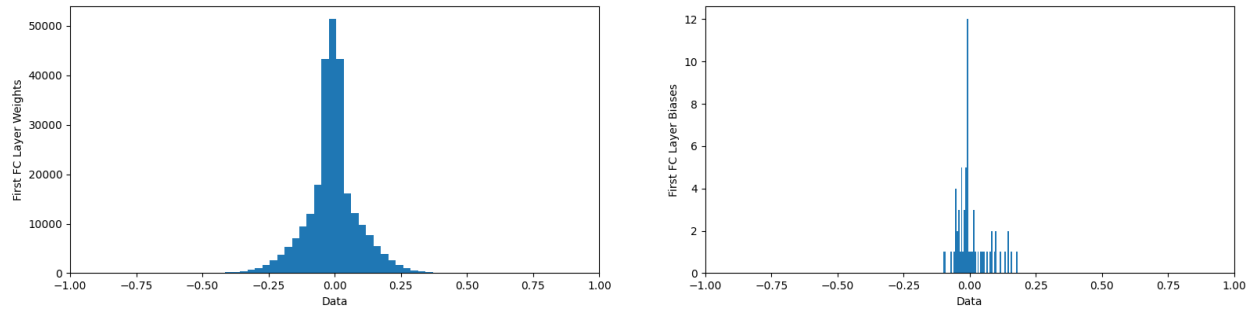


Figure 19 Weights and biases of the first Fully Connected layer

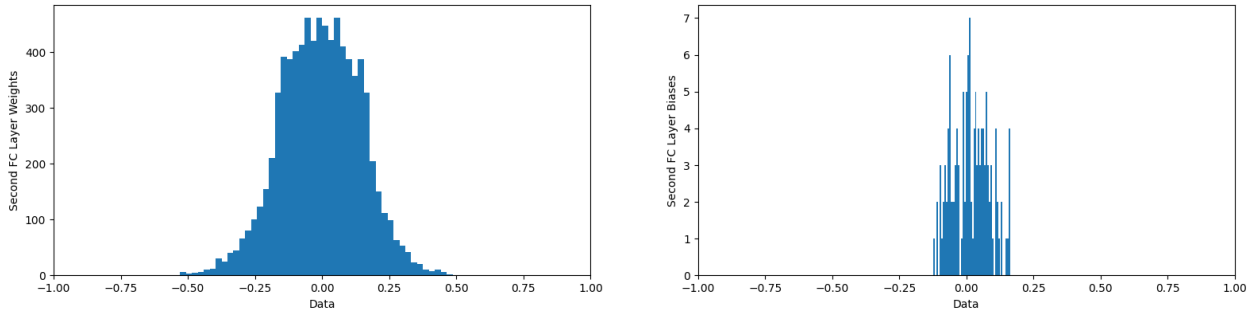


Figure 20 Weights and biases of the second Fully Connected layer

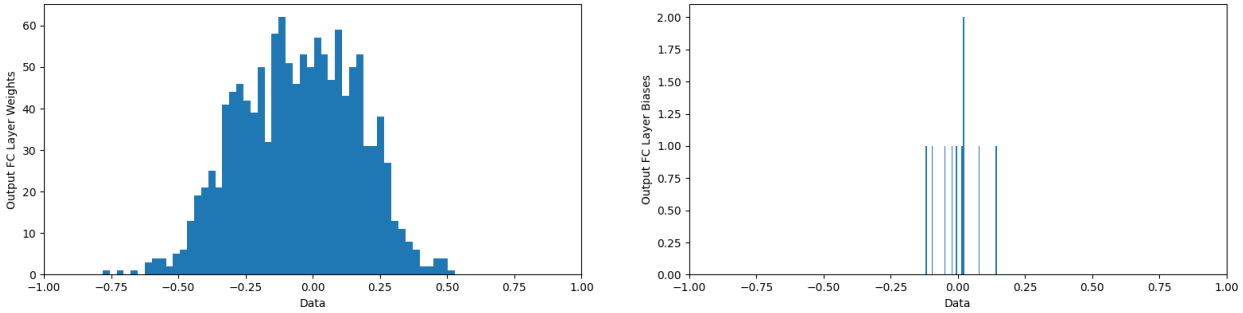


Figure 21 Weights and biases of the third Fully Connected layer

2.2. Examples

2.2.1. Correctly classified examples

Figure 22 Five correctly classified examples of the CIFAR-10 test dataset

2.2.2. Incorrectly classified examples

Figure 23 Five incorrectly classified examples of the CIFAR-10 test dataset

2.2.3. Discussion

The conclusions that can be drawn by analyzing the results from Figure 22 and Figure 23 are similar to those ones in the previous section, as most of the incorrectly classified examples from the test dataset are images where even a human can mislabel the object inside them. For example, the fourth picture in Figure 23 can easily be a truck instead of an automobile, so it makes sense for the neural network to predict this label. Of course, the images' size also impacts the performance of the neural network, as we cannot expect many details of these kind of complex labels in a 32x32 color image. It is worth noticing that some of the correctly predicted images are also somewhat tricky to the human eye, as is the case of the second image, where the picture also has some of the characteristics found on a bird.

As always with this kind of datasets, it is obvious that high-quality images improve the maximum performance an object recognition neural network can have, by allowing the neural network to focus on smaller details that are solely connected to a specific label and not shared between different ones.

2.3. Feature maps

The feature maps presented in Figure 25, Figure 26 and Figure 27 are the representations of the journey taken by the image from Figure 24 in the neural network, going through the deeper layers' filters. We can notice that earlier layers focus on small details, such as straight and curved lines, and deeper layers focus on more general regions of the image.

Figure 24 Image used in the plotting of feature maps in the CIFAR-10 dataset

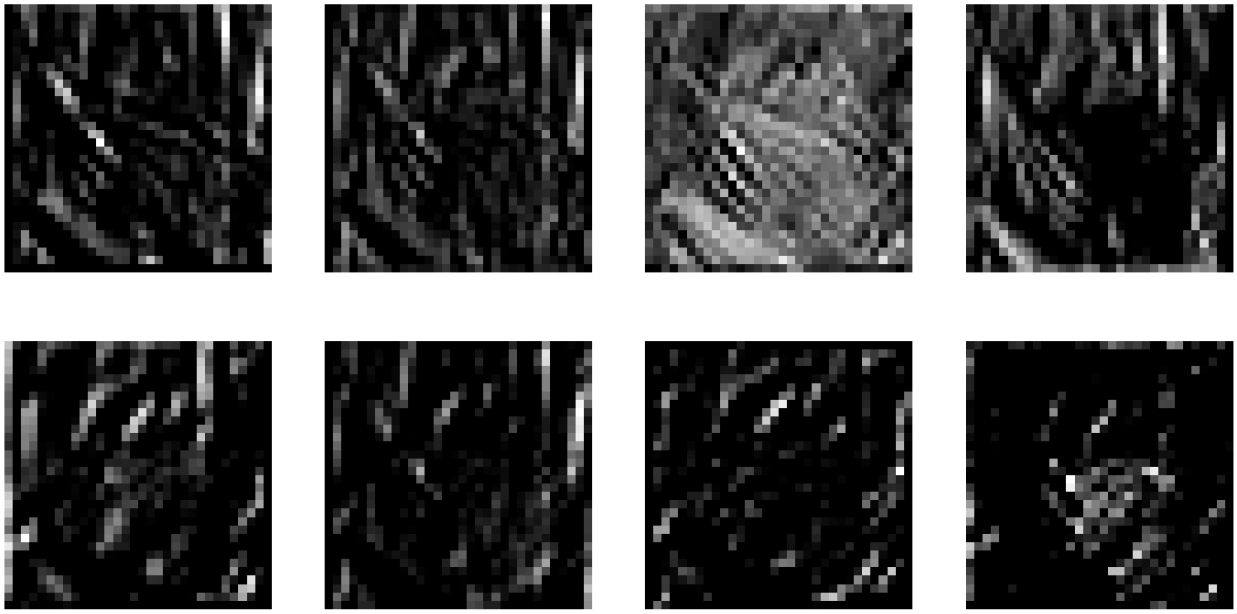


Figure 25 Feature map from the first convolutional network

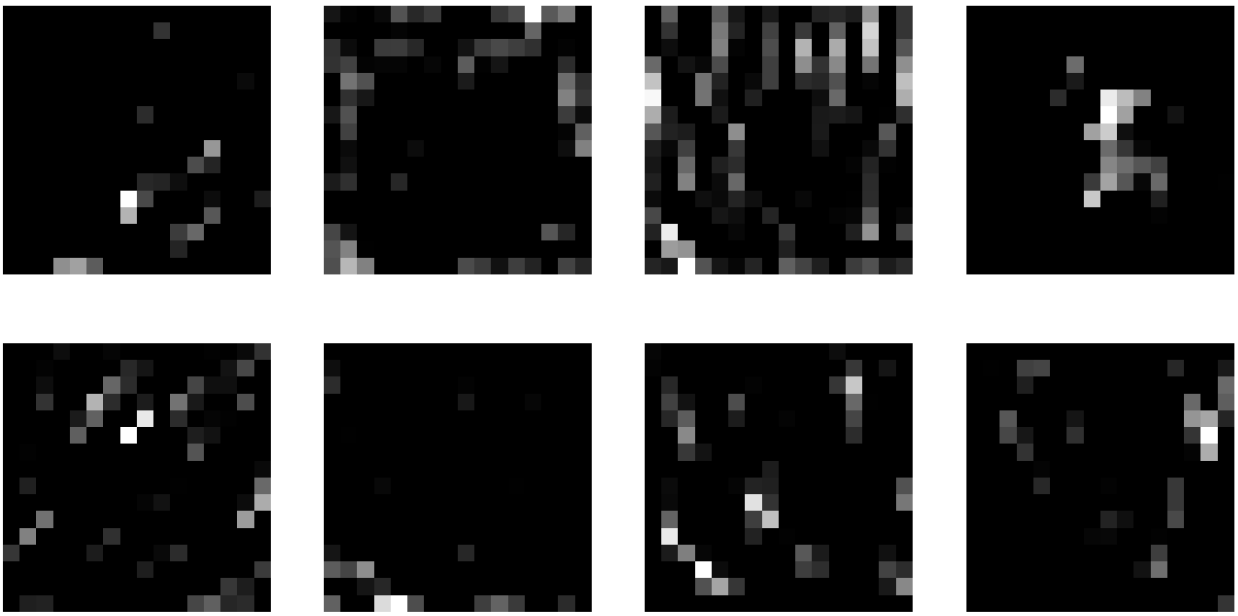


Figure 26 Feature map from the second convolutional network

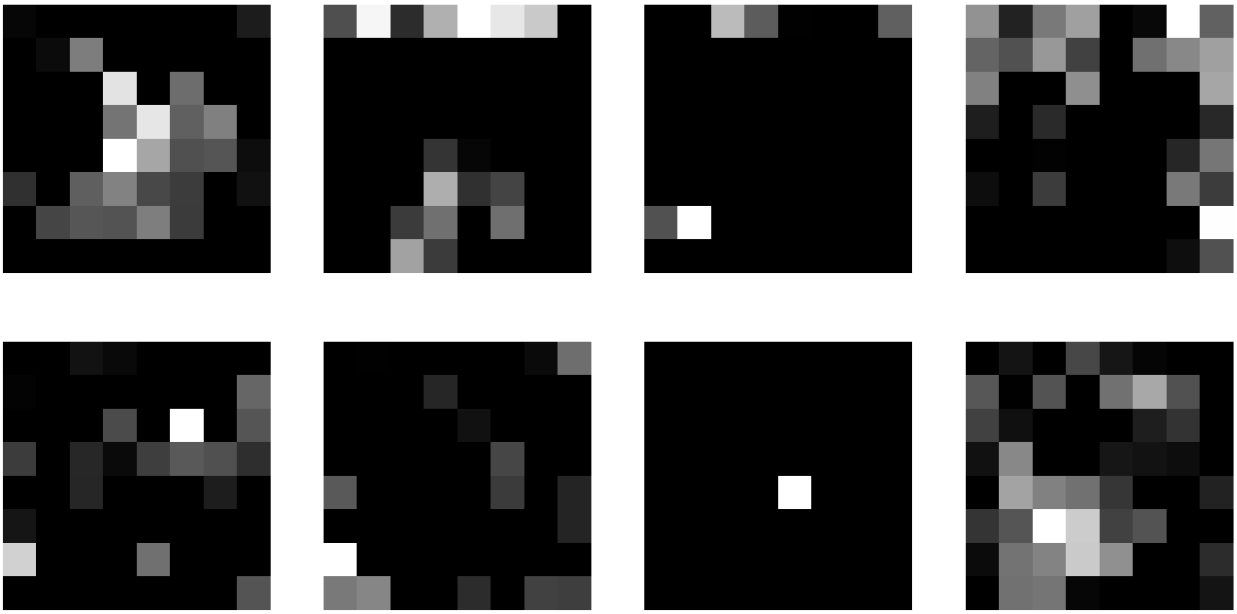


Figure 27 Feature map from the third convolutional network

2.4. Regularization

To solve the overfitting problem showed in Figure 15 an L2 regularization was applied to all layers. This experiment maintained all other settings and hyperparameters of the previous model. The model accuracy and training loss plot can be seen in Figure 28, and the final loss and accuracy for test data were 1.7382 and 48.77% respectively. It can be noted that even though there is no more overfitting, the result is a lot worse than the previous, unregularized model.

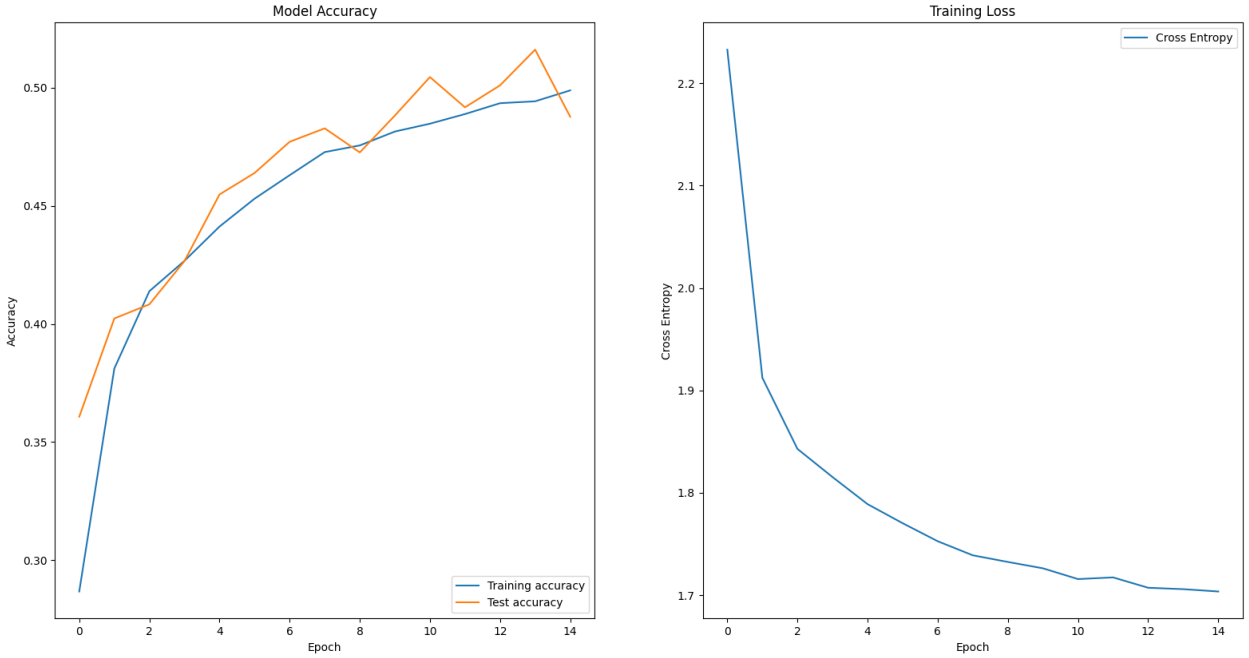
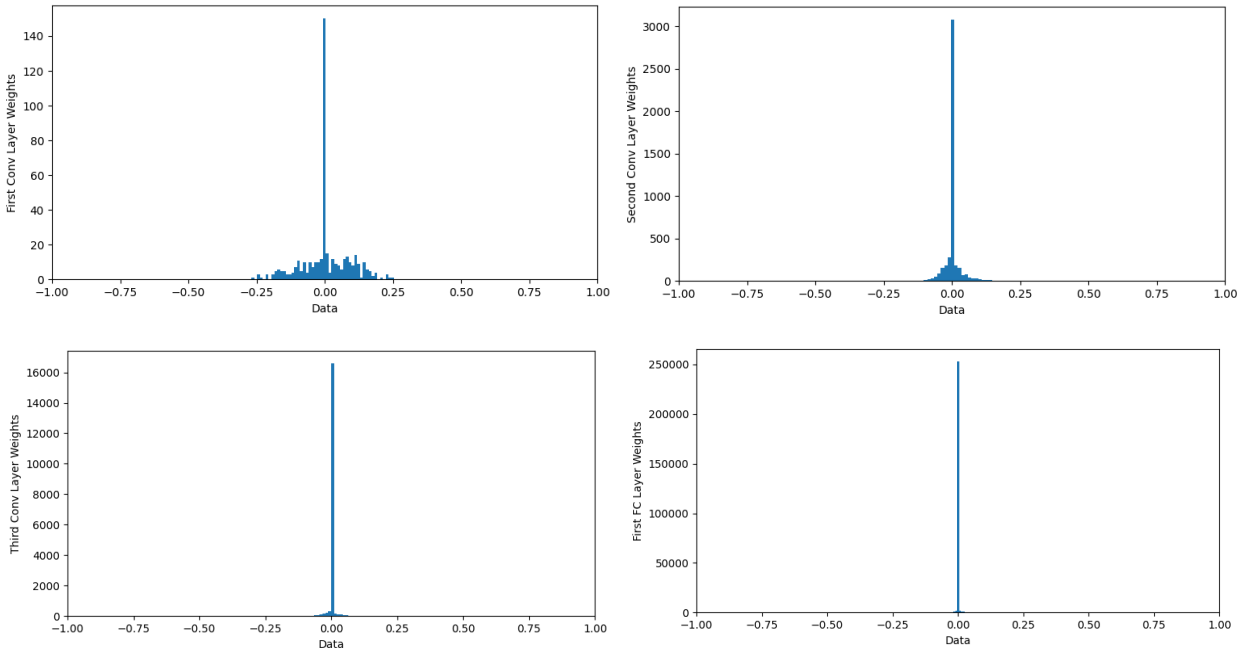


Figure 28 Model accuracy and training loss for the CIFAR-10 problem after L2 regularization



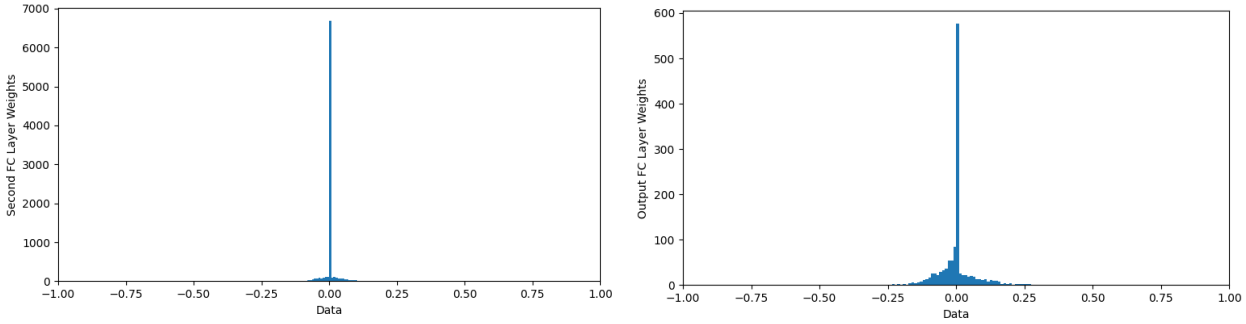


Figure 29 Weights' histogram of all the layers after adding L2 regularization for the CIFAR-10 dataset

By further tweaking the configuration of the neural network and applying the changes shown in Table 7, the result of the model is the one shown in Figure 30, with a loss and accuracy values of 1.0726 and 67.95% respectively. As seen, the model accuracy plot shows that the overfitting problem is resolved, but the accuracy is still worse than the original model. In these situations, we can assume that regularization is not enough to improve the model and other techniques should be used in further experiments, such as data augmentation, other regularization techniques or other neural network architectures (like the already established YOLO algorithm).

Table 7 Changes in parameters of the model after regularization

Parameter adjusted	**Original setting**	**Changed setting**
Adam learning rate	0.001	0.002
Epochs	15	20
L2 regularization	Applied to all layers	Applied to the first and third convolutional layer and to the first fully connected layer.

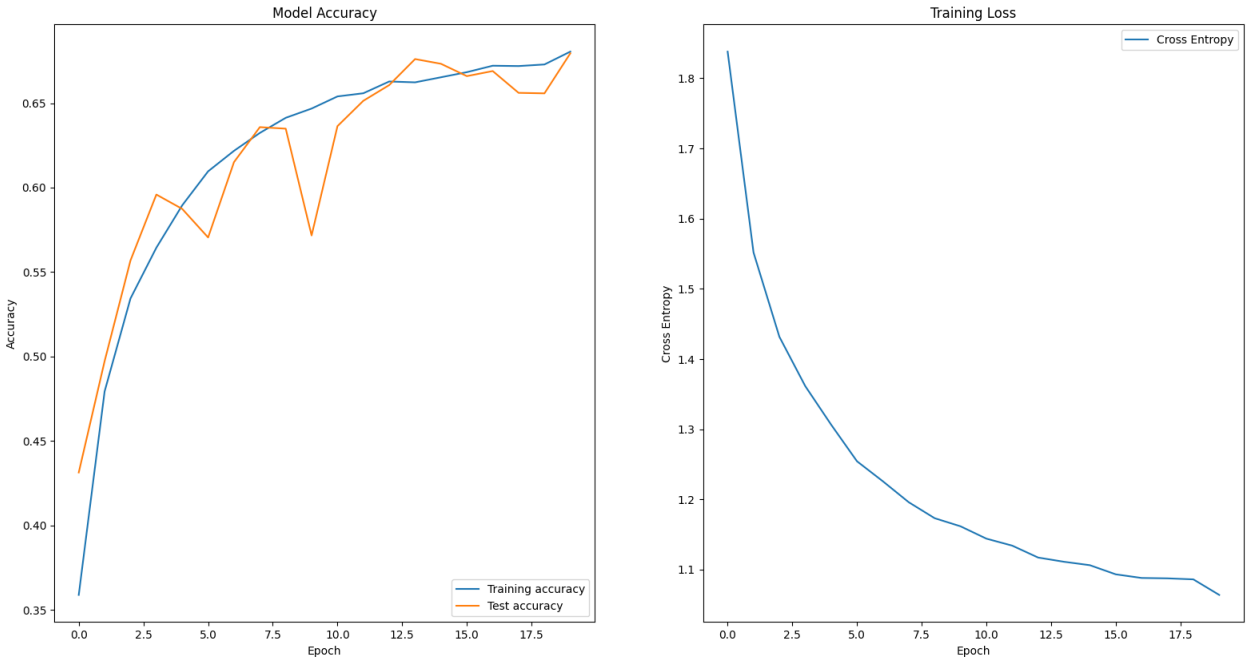


Figure 30 Model accuracy and training loss for the CIFAR-10 problem after L2 regularization and parameter changes

2.5. Pre-processing of images

When working with the CIFAR-10 dataset, some pre-processing work had to be applied to the images before using them as input for the neural network model. The images went to a normalization process that converted each value of the image's array of pixels into a value between 0 and 1, on its three-color channels. This was done by dividing each value into 255, which is the maximum value a pixel can have.

Main References

- Andrew Ng, *Neural Networks and Deep Learning* [MOOC]. Coursera. https://www.coursera.org/learn/convolutional-neural-networks
- https://keras.io/api/layers/regularizers/
- https://www.tensorflow.org/tutorials/images/cnn
- http://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html
- https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5
- https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam
- https://www.cs.toronto.edu/~kriz/cifar.html