DEEP LEARNING

May 2024

TP3 Report:

Training a CNN From Scratch.

- HACHOUD Mohammed
- SEHILI Chaima

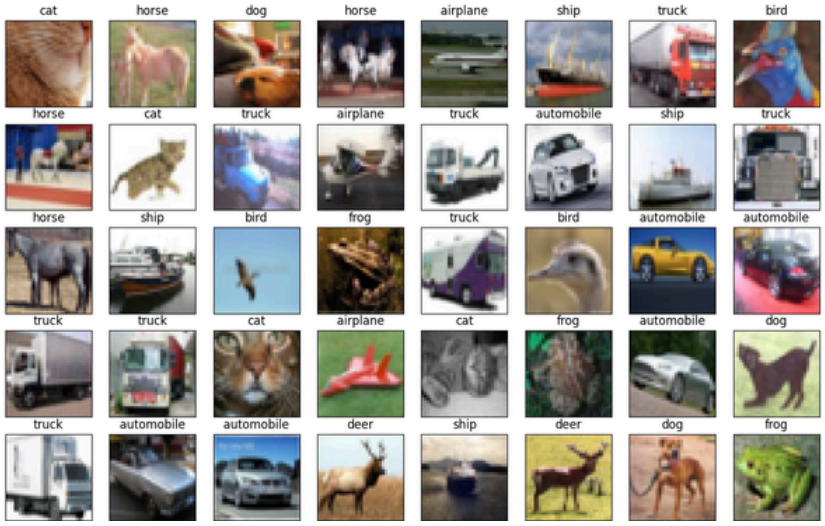
1. Load the **cifar-10** dataset and perform pre-processing: data normalization and data augmentation.

Cifar-10 dataset:

Comprising 60,000 color images, each of size 32x32 pixels, the dataset is segmented into 10 distinct classes, each representing a different object or creature.

The classes encompass the following:

- Airplane
- Automobile
- Bird
- Cat
- Deer
- Dog
- Frog
- Horse
- Ship
- Truck



Data Preprocessing:

Data Normalization:

Convert the pixel values data type to float32 type, and then normalizes them by subtracting the mean and dividing by the standard deviation of the training set.

```
x_train = x_train.astype('float32')
x_test  = x_test.astype('float32')
x_valid = x_valid.astype('float32')

mean = np.mean(x_train)
std  = np.std(x_train)

x_train = (x_train-mean)/(std+1e-7)
x_test  = (x_test-mean) /(std+1e-7)
x_valid = (x_valid-mean)/(std+1e-7)
```

One hot encoding labels:

convert the class labels to one-hot vectors

```
y_train = to_categorical(y_train, 10)
y_valid = to_categorical(y_valid, 10)
y_test  = to_categorical(y_test, 10)
```

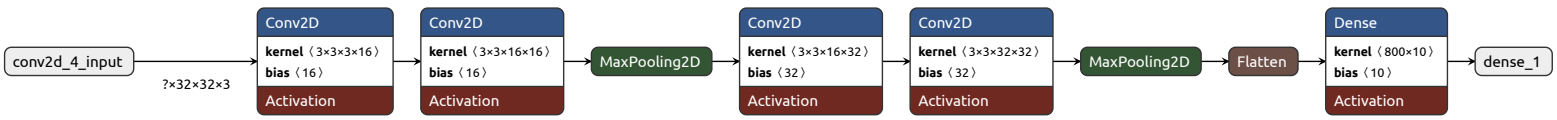
Data Augmentation:

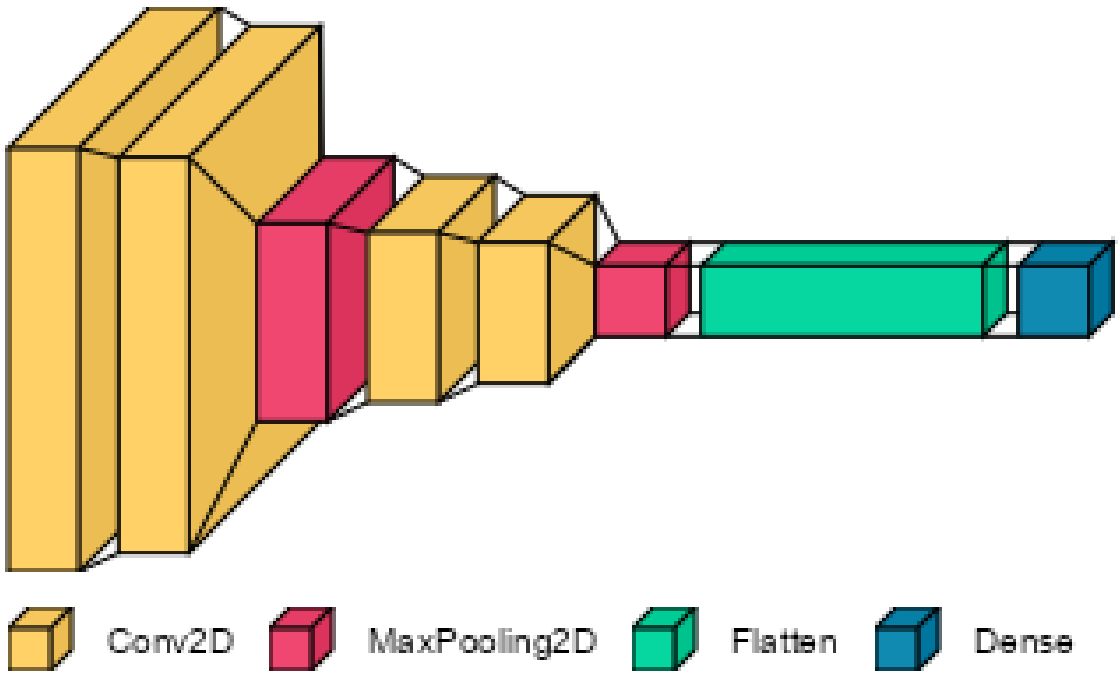
- Rotation
- Width and Height shift
- Horizontal Flip
- Zoom
- Brightness Range
- Shear Intensity: This transformation slants the shape of the image, helping the model to recognize objects in different perspectives.
- Channel Shift Intensity:the intensities of the RGB channels are randomly shifted

```
# Data augmentation
data_generator = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.12,
    height_shift_range=0.12,
    horizontal_flip=True,
    zoom_range=0.1,
    brightness_range=[0.9,1.1],
    shear_range=10,
    channel_shift_range=0.1,
)
```

2.Create a **convolutional neural network** via **Keras** with the following layers:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)            (None, 30, 30, 16)        448

 conv2d_5 (Conv2D)            (None, 28, 28, 16)        2320

 max_pooling2d_2 (MaxPoolin   (None, 14, 14, 16)        0
 g2D)

 conv2d_6 (Conv2D)            (None, 12, 12, 32)        4640

 conv2d_7 (Conv2D)            (None, 10, 10, 32)        9248

 max_pooling2d_3 (MaxPoolin   (None, 5, 5, 32)          0
 g2D)

 flatten_1 (Flatten)          (None, 800)               0

 dense_1 (Dense)              (None, 10)                8010
```



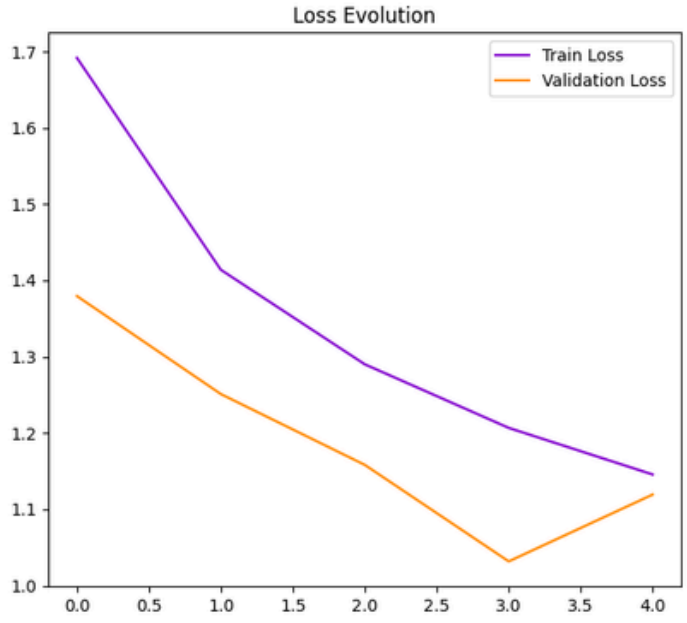


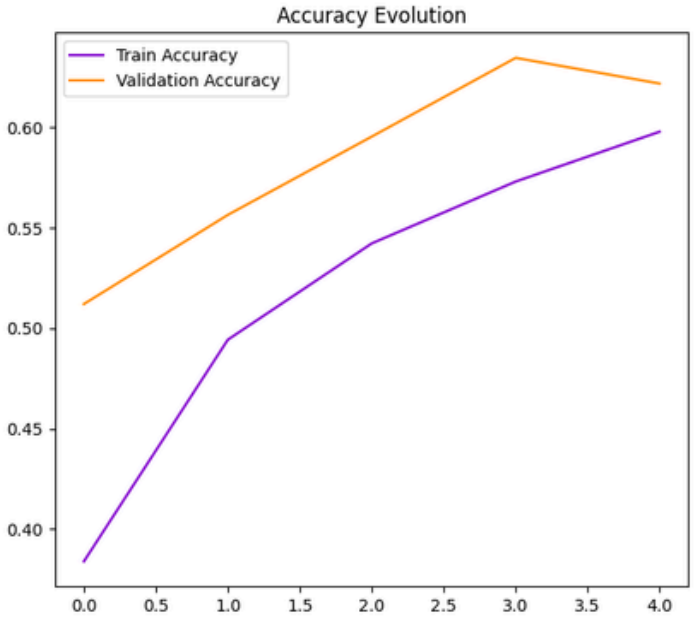
3.**Fit** the neural network for the training data.

- Adam optimizer with its default settings
- batch size of 64
- accuracy as a metric
- categorical_crossentropy loss
- print the metric after each epoch for both the train and the test set!
- train the neural network for 5 epochs

```
- loss: 1.7163 - accuracy: 0.3773 - val_loss: 1.4266 - val_accuracy: 0.4860

- loss: 1.4255 - accuracy: 0.4877 - val_loss: 1.2515 - val_accuracy: 0.5533

- loss: 1.3079 - accuracy: 0.5365 - val_loss: 1.1670 - val_accuracy: 0.5922

- loss: 1.2373 - accuracy: 0.5617 - val_loss: 1.0996 - val_accuracy: 0.6158

- loss: 1.1795 - accuracy: 0.5835 - val_loss: 1.0453 - val_accuracy: 0.6345
```

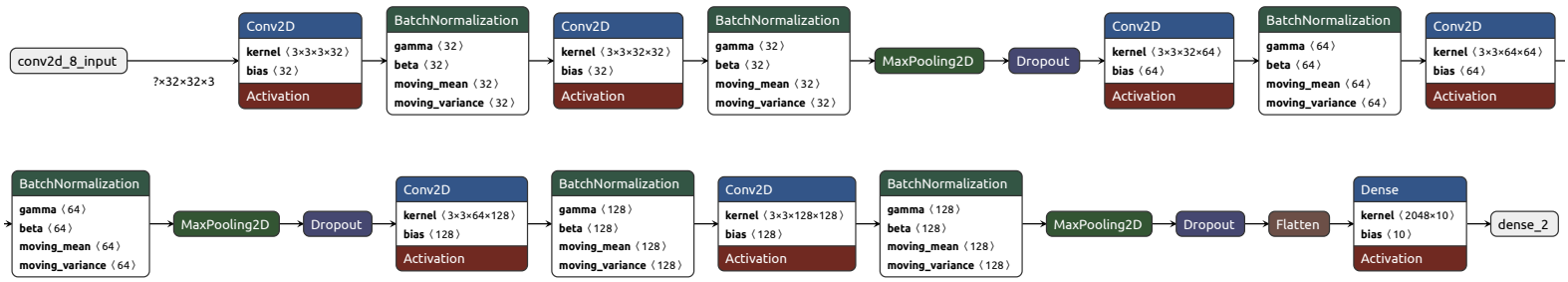
Learning curves:

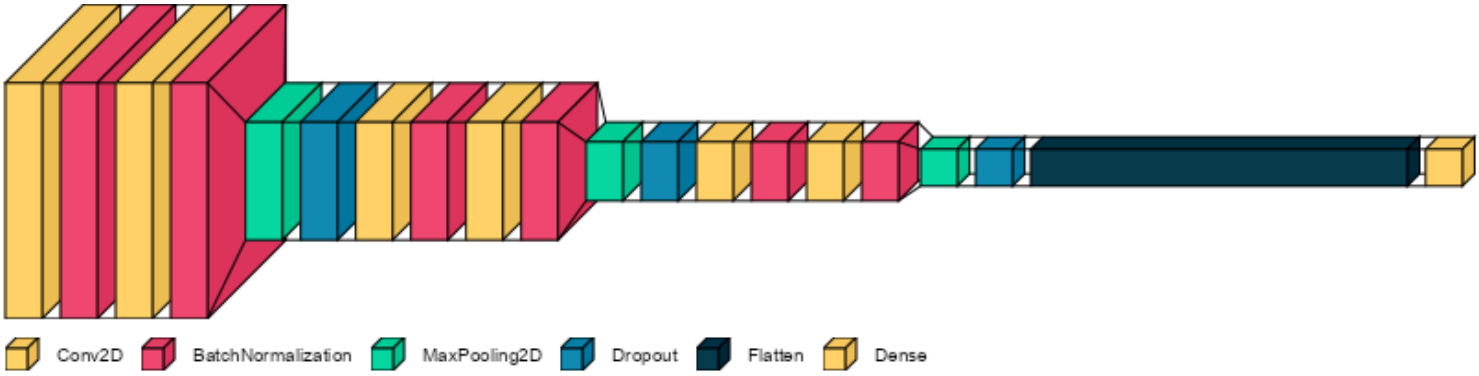




Test Accuracy after 5 epochs:
0.6352999806404114

4. **Improving CNN:** try to fit another convolutional neural network that can achieve 70% accuracy on the test set (with only 5 epochs).



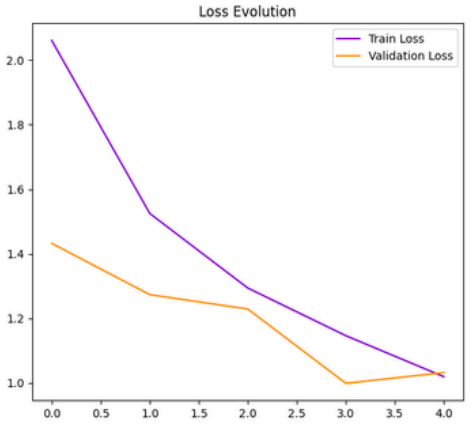


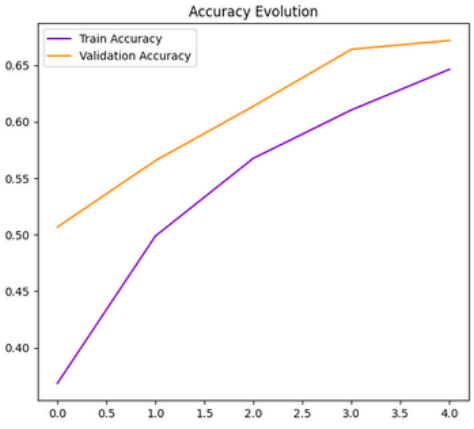
The improvements done over the first model:

- **Batch Normalization:**Batch normalization layers are added after each convolutional layer. Batch normalization helps in stabilizing and accelerating the training process by normalizing the input of each layer, reducing internal covariate shift, and acting as a regularizer.
- **Dropout:** Dropout layers are included after each max-pooling layer. Dropout is a regularization technique that randomly sets a fraction of input units to zero during training, which helps prevent overfitting by reducing the interdependencies between neurons.
- **Increased Depth and Width:** The model has more convolutional layers (3 sets of Conv2D-BatchNorm pairs) compared to the previous model. Also, the **number of filters in each convolutional layer is increased.**
- **Padding:** The **'same'** padding is used in all convolutional layers. This **ensures that the spatial dimensions** of the feature maps remain the same after convolution, which can be advantageous in preserving spatial information.

Overall, these improvements make the second model more robust, better regularized, and potentially more effective in capturing and learning features from the input data compared to the first model.

Learning curves:

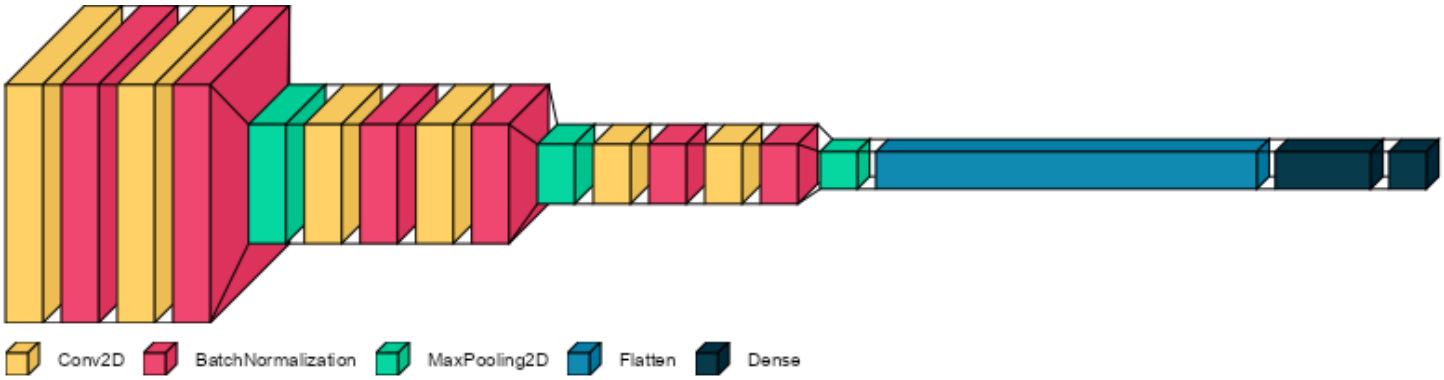


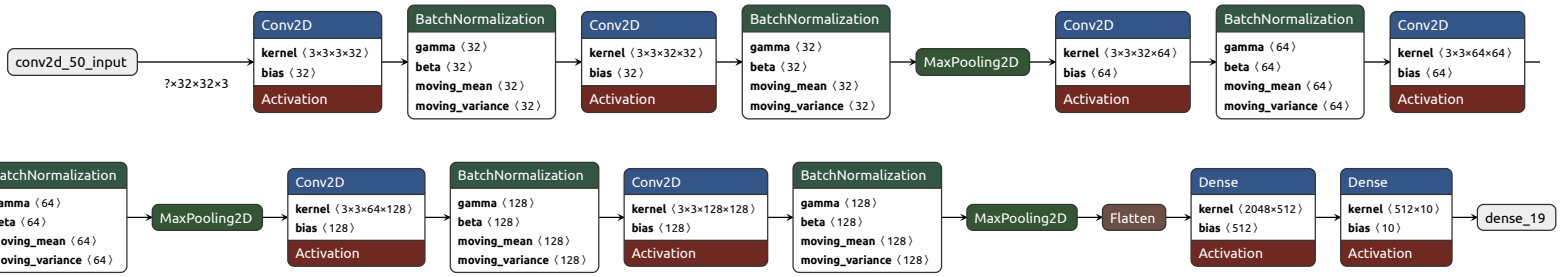



```
Test Accuracy after 5 epochs:
      0.7019000053405762
```

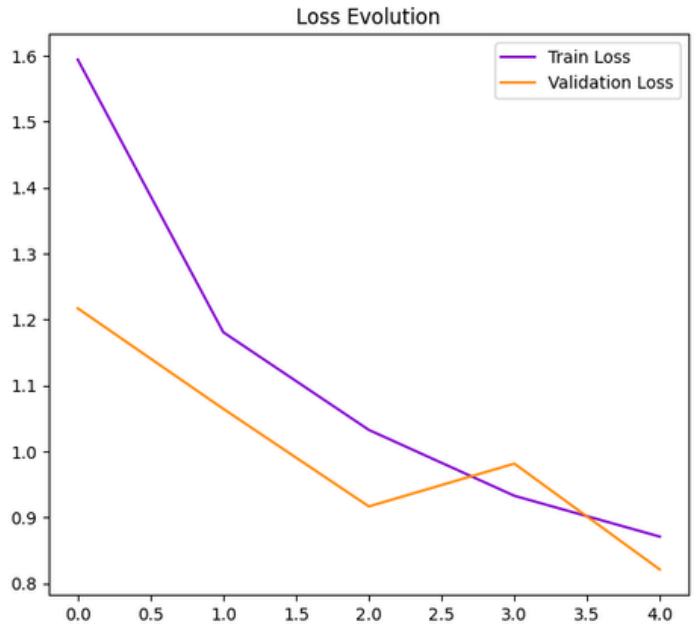
Another enhancements have been made to the second model that achieved a **76% test accuracy after 5 epochs**:

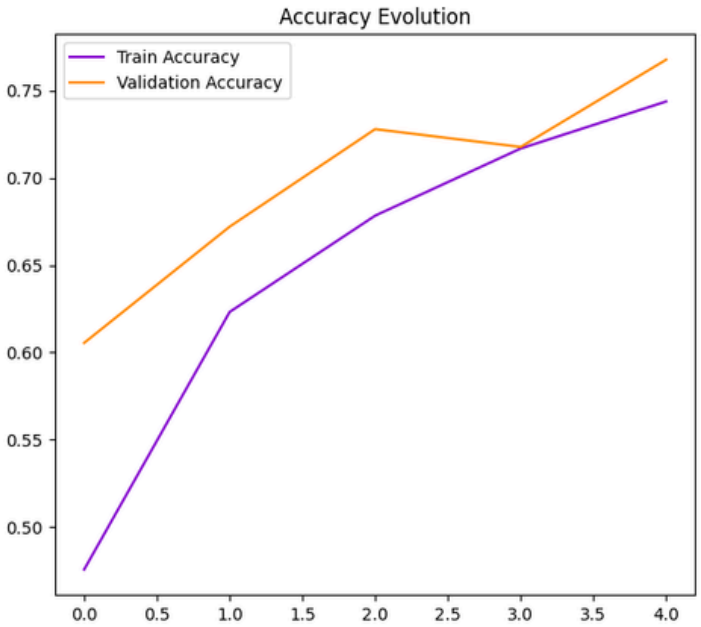
- Introducing **kernel regularization** techniques: L2 with weight decay 0.0001.
- the inclusion of an **extra dense layer** with 512 units, coupled with ReLU activation.





Learning curves:






```
Test Accuracy: 0.7692999839782715
```