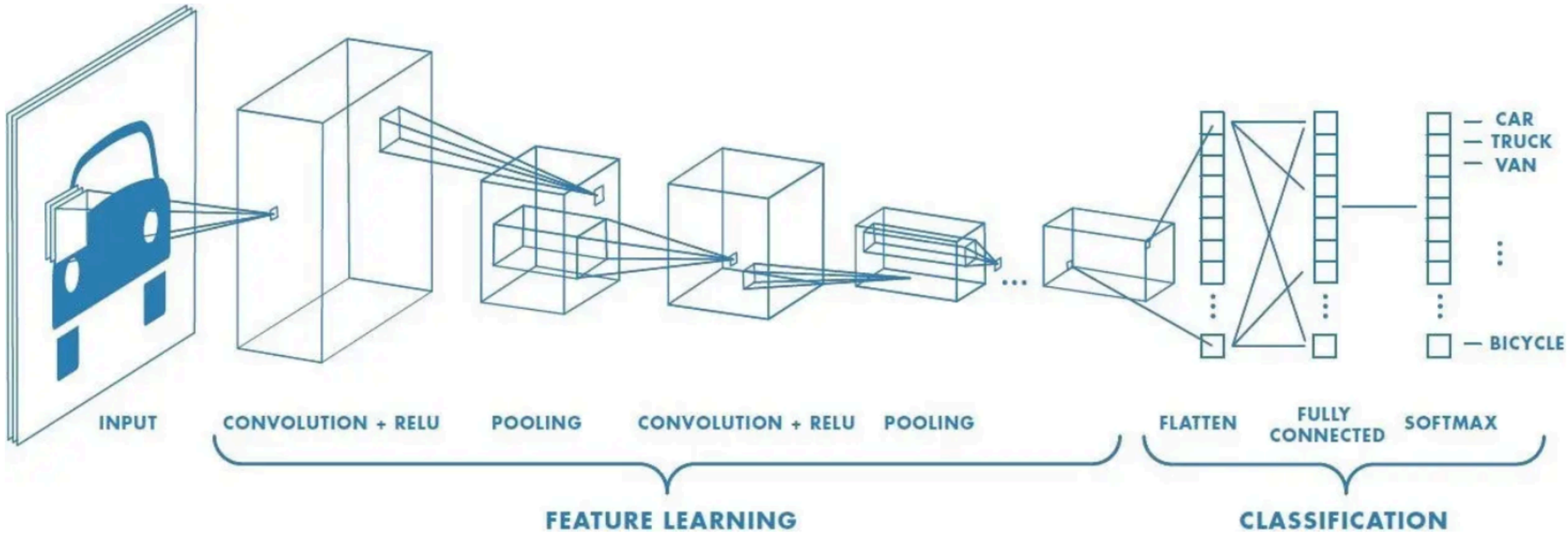
# CIFAR-10 IMAGE CLASSIFICATION

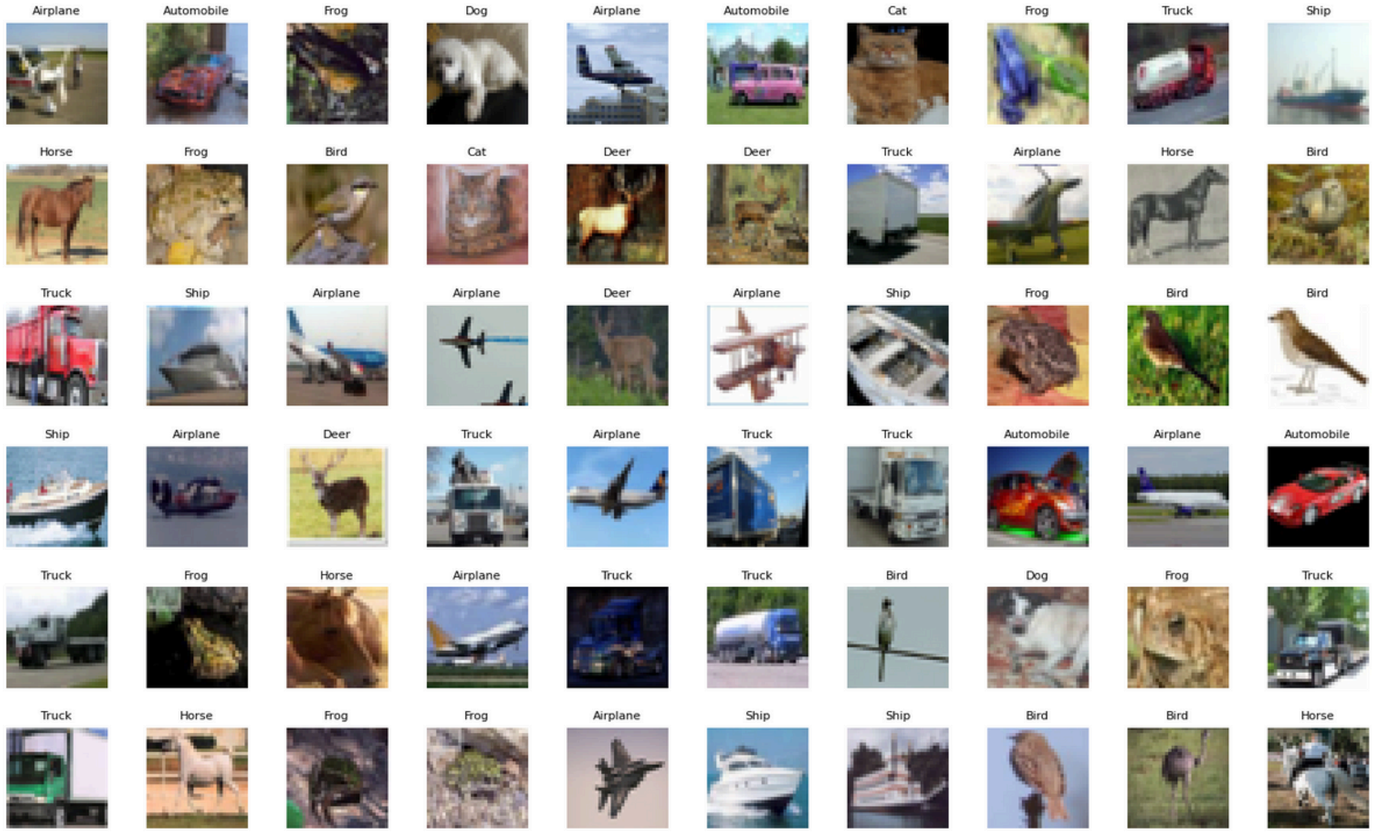
Adonis Kingsley Granita

# OBJECTIVE

Build a Convolutional Neural Network (CNN) to classify images into 10 distinct classes from the CIFAR-10 dataset.



# CIFAR-10




0 AIRPLANE

1 AUTOMOBILE

2 BIRD

3 CAT

4 DEER

5 DOG

6 FROG

7 HORSE

8 SHIP

9 TRUCK

# DATA PREPROCESSING

## NORMALIZATION

The pixel values where divided by 255, so that the scale ranges from 0 to 1

## HOT-ONE-ENCODING

I applied one-hot encoding to the labels converting them into binary matrix.

# DATA SPLIT

## TRAINING SET

42500 images

## VALIDATION SET

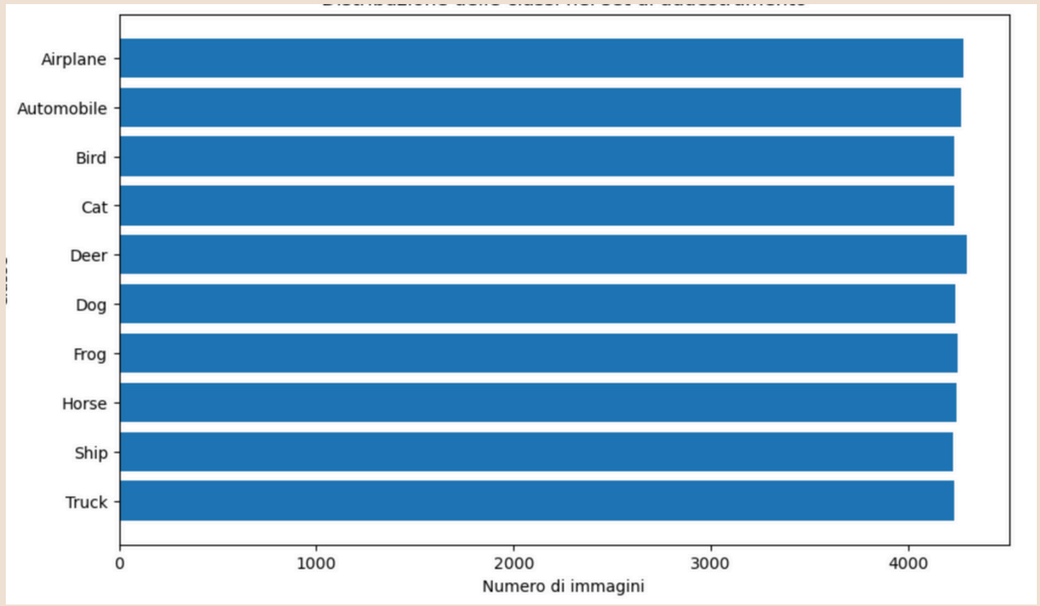
7500 images, 15% of the training set

## TEST SET

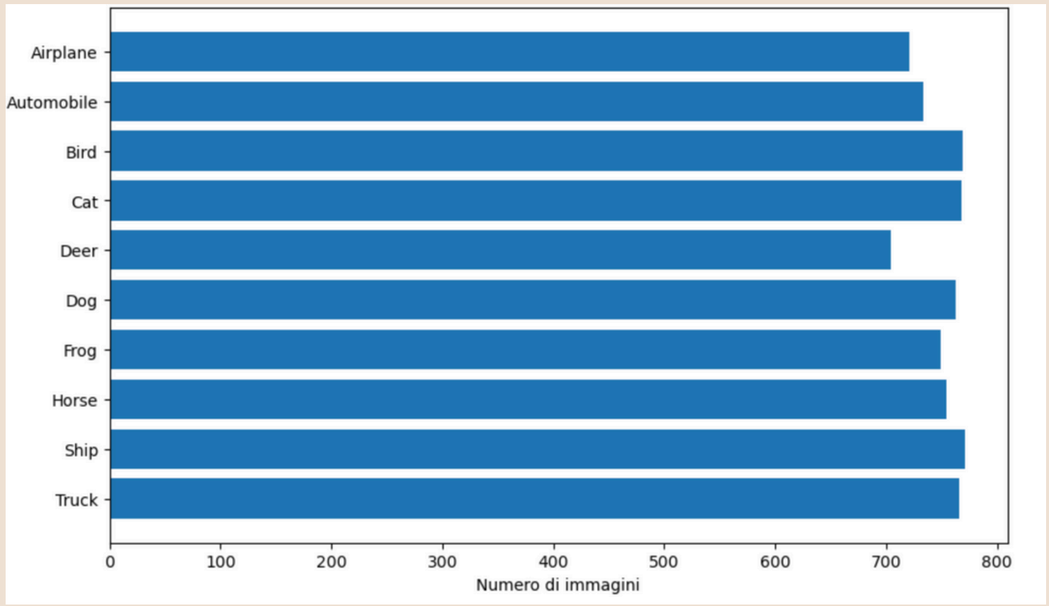
10 000 images

# CLASS DISTRIBUTION

## TRAINING SET




## VALIDATION SET



# EVALUATION METRICS

## ACCURACY

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

percentage of correct predictions made by the model out of all predictions.

## CATEGORICAL CROSSENTROPY LOSS

$$L_i = -\sum_{j=1}^{C} y_{ij} \log(\hat{y}_{ij})$$

how well the predicted probability distribution aligns with the true class, focusing on maximizing the probability of the correct class.

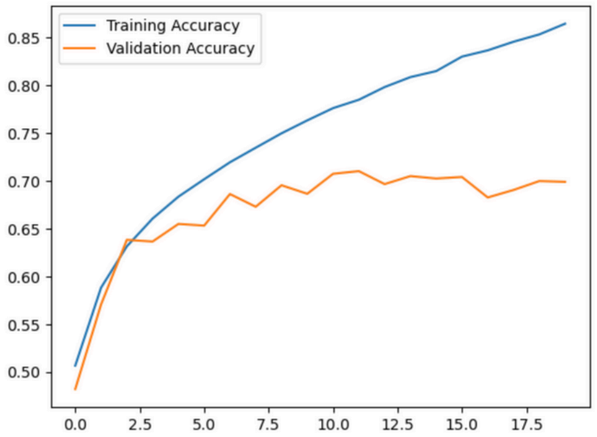
# MODEL 1

**PROBLEM**

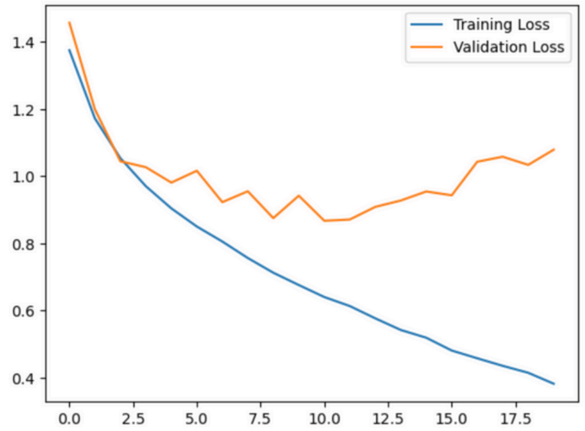
**OVERFITTING**

| Layer (type) | Output Shape |
|---|---|
| conv2d_6 (Conv2D) | (None, 30, 30, 32) |
| max_pooling2d_4 (MaxPooling2D) | (None, 15, 15, 32) |
| conv2d_7 (Conv2D) | (None, 13, 13, 64) |
| max_pooling2d_5 (MaxPooling2D) | (None, 6, 6, 64) |
| conv2d_8 (Conv2D) | (None, 4, 4, 64) |
| flatten_2 (Flatten) | (None, 1024) |
| dense_4 (Dense) | (None, 64) |
| dense_5 (Dense) | (None, 10) |

**ACCURACY**




**CATEGORICAL CROSSENTROPY LOSS**




**TOTAL PARAMS: 122,570**

**TEST ACCURACY: 0.6877**
**TEST LOSS: 1.1005**

# MODEL 2

## SOLUTIONS

- DROPOUT
- DATA AUGMENTATION

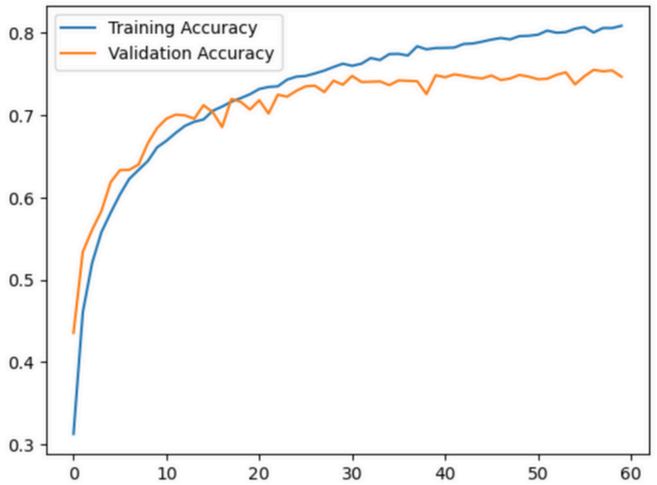
## PROBLEM

- OVERFITTING

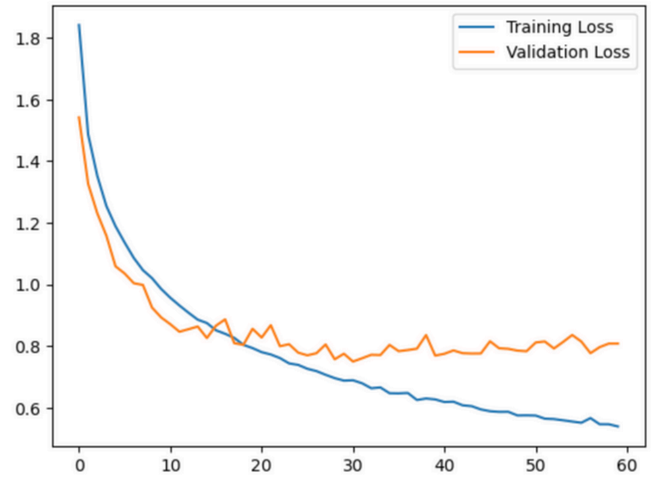
| Layer (type) | Output Shape |
|---|---|
| random_flip (RandomFlip) | (None, 32, 32, 3) |
| conv2d_3 (Conv2D) | (None, 30, 30, 32) |
| max_pooling2d_2 (MaxPooling2D) | (None, 15, 15, 32) |
| conv2d_4 (Conv2D) | (None, 13, 13, 64) |
| max_pooling2d_3 (MaxPooling2D) | (None, 6, 6, 64) |
| conv2d_5 (Conv2D) | (None, 4, 4, 64) |
| flatten_1 (Flatten) | (None, 1024) |
| dense_2 (Dense) | (None, 64) |
| dropout (Dropout) | (None, 64) |
| dense_3 (Dense) | (None, 10) |

**TOTAL PARAMS: 367,712**

### ACCURACY




### CATEGORICAL CROSSENTROPY LOSS




**TEST ACCURACY: 0.7556**
**TEST LOSS: 0.8183**

# MODEL 3

## SOLUTIONS

- L2 REGULARIZATION
- MORE COMPLEX MODEL

## PROBLEM

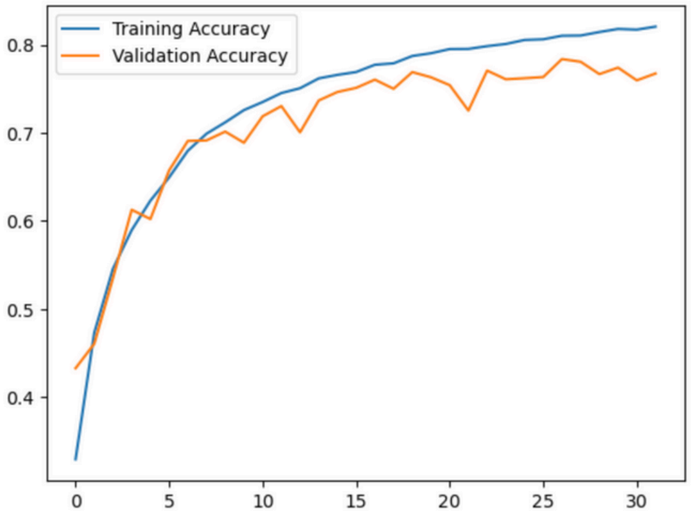
- PERFORMANCE COULD BE BETTER

| Layer (type) | Output Shape |
|---|---|
| random_flip_15 (RandomFlip) | (None, 32, 32, 3) |
| conv2d_59 (Conv2D) | (None, 32, 32, 64) |
| max_pooling2d_39 (MaxPooling2D) | (None, 16, 16, 64) |
| conv2d_60 (Conv2D) | (None, 16, 16, 128) |
| max_pooling2d_40 (MaxPooling2D) | (None, 8, 8, 128) |
| conv2d_61 (Conv2D) | (None, 8, 8, 256) |
| max_pooling2d_41 (MaxPooling2D) | (None, 4, 4, 256) |
| conv2d_62 (Conv2D) | (None, 4, 4, 256) |
| flatten_16 (Flatten) | (None, 4096) |
| dense_37 (Dense) | (None, 128) |
| dense_38 (Dense) | (None, 64) |
| dense_39 (Dense) | (None, 10) |

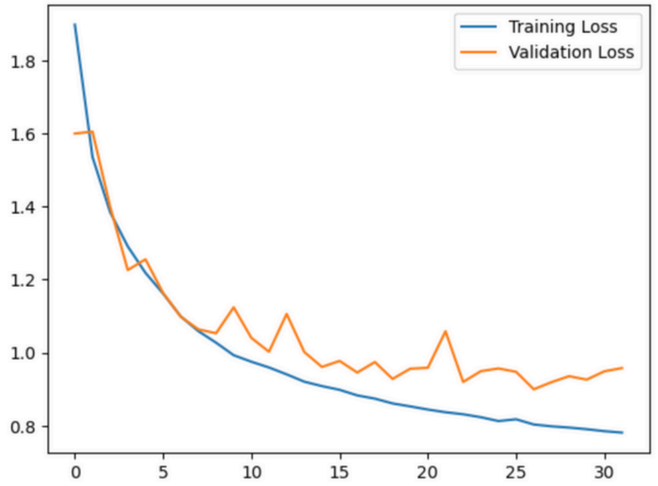
```
kernel_regularizer=regularizers.l2(0.001)
```

**TOTAL PARAMS: 1,494,218**

### ACCURACY




### CATEGORICAL CROSSENTROPY LOSS




**TEST ACCURACY: 0.8169**
**TEST LOSS: 0.8241**

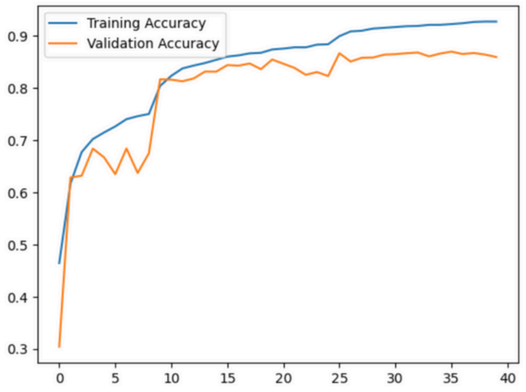
# FINAL MODEL

- **INCREASE DEPTH AND FILTERS**
  - STARTS AT 64, DOUBLES EACH BLOCK, UP TO 512
- **DROPOUT**
  - 0.2 ON THE FIRT BLOCKS
  - 0.5 ON THE LAST
- **L2 REGULARIZERS**
  - APPLIED TO ALL CONV LAYERS
- B**ATCH NORMALIZATION**
  - AFTER ALL CONV LAYERS
- TRAINING CALLBACKS
  - EARLY STOPPING
  - LEARNING RATE SCHEDULING

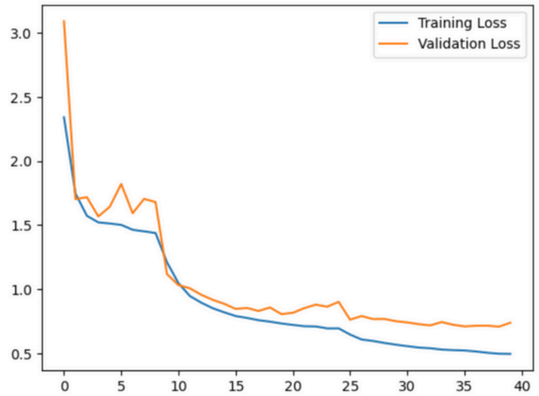
| Layer (type) | Output Shape |
|---|---|
| random_flip_4 (RandomFlip) | (None, 32, 32, 3) |
| conv2d_15 (Conv2D) | (None, 32, 32, 64) |
| batch_normalization (BatchNormalization) | (None, 32, 32, 64) |
| max_pooling2d_10 (MaxPooling2D) | (None, 16, 16, 64) |
| dropout_3 (Dropout) | (None, 16, 16, 64) |
| conv2d_16 (Conv2D) | (None, 16, 16, 128) |
| batch_normalization_1 (BatchNormalization) | (None, 16, 16, 128) |
| max_pooling2d_11 (MaxPooling2D) | (None, 8, 8, 128) |
| dropout_4 (Dropout) | (None, 8, 8, 128) |
| conv2d_17 (Conv2D) | (None, 8, 8, 256) |
| batch_normalization_2 (BatchNormalization) | (None, 8, 8, 256) |
| max_pooling2d_12 (MaxPooling2D) | (None, 4, 4, 256) |
| dropout_5 (Dropout) | (None, 4, 4, 256) |
| conv2d_18 (Conv2D) | (None, 4, 4, 512) |
| batch_normalization_3 (BatchNormalization) | (None, 4, 4, 512) |
| max_pooling2d_13 (MaxPooling2D) | (None, 2, 2, 512) |
| dropout_6 (Dropout) | (None, 2, 2, 512) |
| flatten_5 (Flatten) | (None, 2048) |
| dense_10 (Dense) | (None, 128) |
| dense_11 (Dense) | (None, 64) |
| dense_12 (Dense) | (None, 10) |

# FINAL MODEL EVALUATION

### ACCURACY




### CATEGORICAL CROSSENTROPY LOSS




### CLASSIFICATION REPORT

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.87 | 0.87 | 1000 |
| 1 | 0.94 | 0.91 | 0.93 | 1000 |
| 2 | 0.85 | 0.81 | 0.83 | 1000 |
| 3 | 0.79 | 0.69 | 0.74 | 1000 |
| 4 | 0.83 | 0.87 | 0.85 | 1000 |
| 5 | 0.83 | 0.79 | 0.81 | 1000 |
| 6 | 0.84 | 0.94 | 0.88 | 1000 |
| 7 | 0.90 | 0.90 | 0.90 | 1000 |
| 8 | 0.86 | 0.94 | 0.90 | 1000 |
| 9 | 0.92 | 0.90 | 0.91 | 1000 |
| accuracy | | | 0.86 | 10000 |
| macro avg | 0.86 | 0.86 | 0.86 | 10000 |
| weighted avg | 0.86 | 0.86 | 0.86 | 10000 |

**TEST ACCURACY: 0.8637**
**TEST LOSS:0.7141**

# FUTURE DEVELOPMENTS

## 01 PETRAINED MODELS

Use transfer learning could had lead to a better performing model

## 02 TRAINING TIME

increasing the number of epoch of the final model could have increase its performance

## 03 MODEL SEARCH

using for instance Neural Architecture Search - NAS to find the right complexity of the model

# THANKS

**LINK TO THE MODELS**