*SECTION B* – Documentation

*Module*: CIS6005 Computational Intelligence

*School*: Cardiff School of Technologies/Varna University of Management

Lecturer: Osman Osman

*Student number*: 20283946

Jupyter Notebook can be found at-> <https://github.com/lightonray/st20283946_CIS6005_B.git>

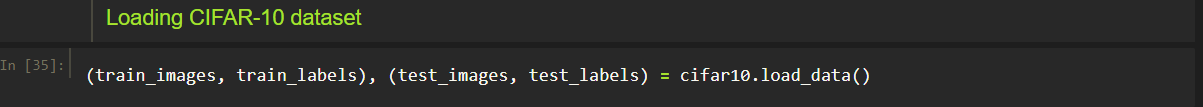
DataSet -> CIFAR-10 / KAGGLE

*Introduction to Dataset*

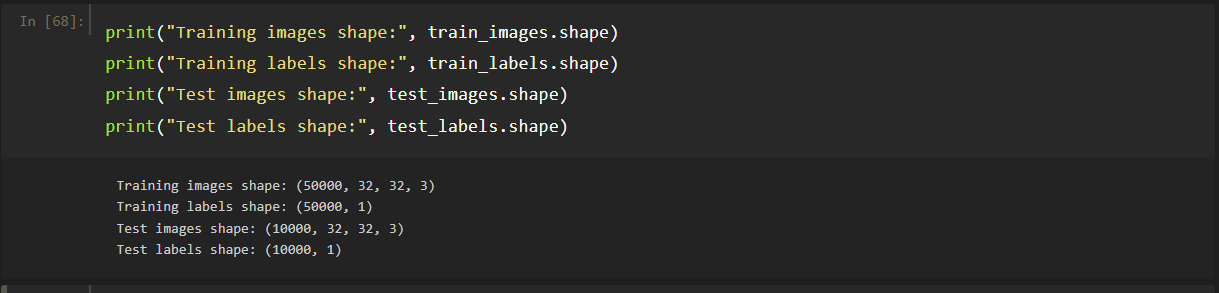
The CIFAR-10 dataset consists of 50,000 training images and 10,000 test images, each with a shape of (32, 32, 3), indicating images of size 32x32 pixels with three color channels (RGB).

Labels are integers representing the class of each image. In the CIFAR-10 dataset, there are 10 classes ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

***Loading the dataset***

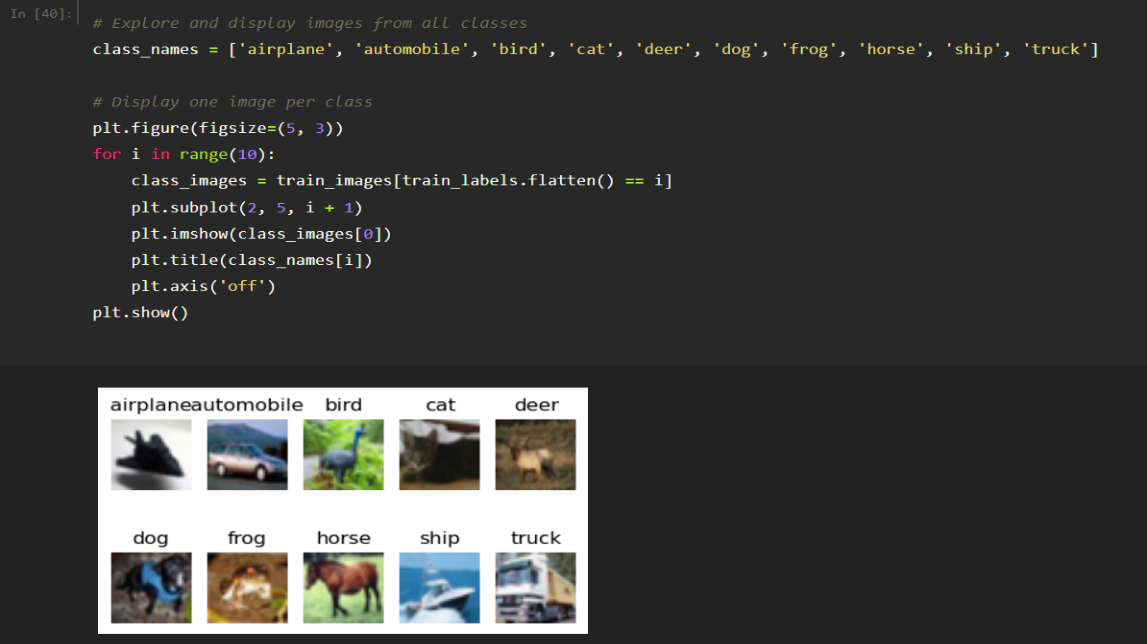
The dataset usually comes with predefined splits for training and testing. In this case, CIFAR-10 has a training set and a test set.

***Exploring the shape of data***

******

Data consists of a training set with 50,000 labeled images, each of size 32x32 pixels with 3 color channels, and a test set with 10,000 similarly formatted images for evaluation. The labels for both training and test sets are presented in a column vector with one label per image.

***Visualizing the dataset***

******

*Splitting the training set for creation of validation set*

In the case of the CIFAR-10 dataset, there is no need to explicitly encode the class labels (target variable) because the labels are already provided in a suitable format for classification tasks. The labels in CIFAR-10 are integers representing the class of each image, and they range from 0 to 9, corresponding to the ten different classes (e.g., "airplane," "automobile," "bird," etc.).

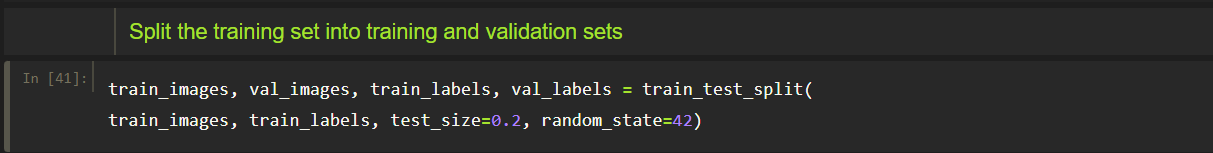
***Dataset Splitting:***

The *train\_test\_split* function is employed to divide the original training set into two subsets: a new training set (*train\_images* and *train\_labels*) and a validation set (*val\_images* and *val\_labels*).

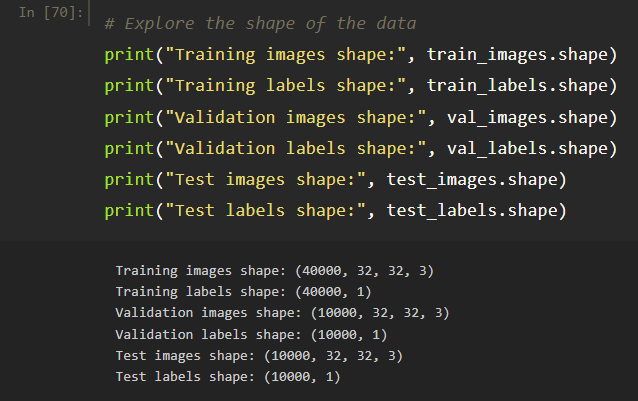
***Parameters:***

*test\_size* = 0.2: Specifies that 20% of the data should be allocated for the validation set.

*random\_state* = 42: Ensures reproducibility by using a fixed random seed. The same seed will result in the same split every time it's run.



***Exploring the shape of data***

******

**Metadata Details:**

*Training Set:*

Size: 80% of the original training set.

Dimensions: 32x32 pixels, RGB format (3 color channels).

Labels: Corresponding integer labels representing one of the 10 classes.

*Validation Set:*

Size: 20% of the original training set.

Similar dimensions and format as the training set.

Used for hyperparameter tuning and model evaluation during training.

*Testing Set:*

Size: 10,000 images.

Same dimensions and format as the training set.

Kept separate until the model is trained for final evaluation.

*Discuss the rationale for having a validation set*

The validation set serves the purpose to fine-tune the model's hyperparameters and to provide an unbiased evaluation of a model fit during training.

***Key rationales for having a validation set:***

* ***Hyperparameter Tuning***

*The validation set is used to experiment with different hyperparameter configurations. By training the model on the training set and evaluating its performance on the validation set, one can identify the hyperparameter values that lead to optimal performance without overfitting the model to the training*

* ***Preventing Overfitting***

*Regularly evaluating the model on a separate validation set helps detect when the model is starting overfitting. If the model performs well on the training set but poorly on the validation set, adjustments can be made to prevent overfitting, such as modifying the model architecture or introducing regularization techniques.*

* ***Model Selection***

*The validation set serves as a benchmark for comparing different models. By assessing their performance on the validation set, one can make informed decisions about which model is most likely to generalize well to new, unseen data.*

* ***Performance Monitoring***

Monitoring both training and validation performance provides insights into the model's learning dynamics. Divergence between training and validation performance may indicate issues such as overfitting or underfitting, guiding adjustments to enhance model performance.

* ***Generalization Assessment***

*The validation set, being distinct from the training set, provides a realistic assessment of the model's generalization capabilities. A model that performs well on the validation set is more likely to generalize well to unseen data*

*Building the ANN models*

***FIRST MODEL***

*Structure*

Model Type: Convolutional Neural Network (CNN)

*Architecture:*

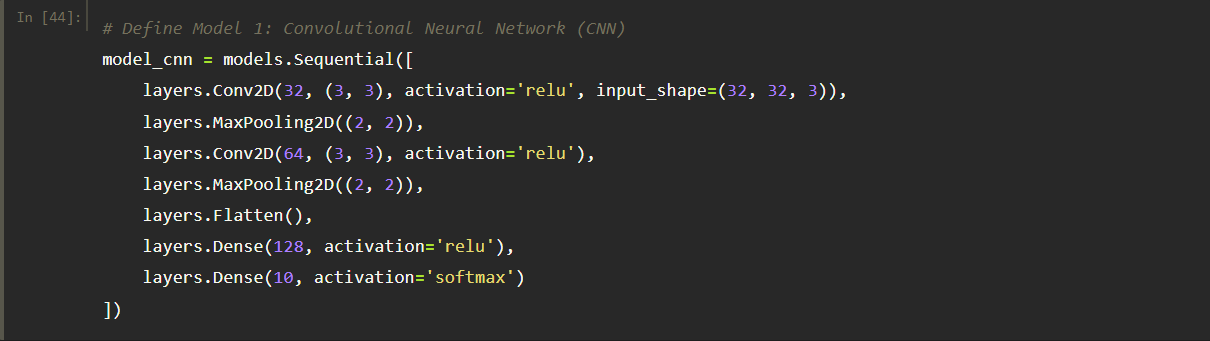
* *Input Layer: Convolutional layer with 32 filters, each of size (3, 3), using ReLU activation.*
* *MaxPooling Layer: Reduces spatial dimensions by taking the maximum value in a 2x2 window.*
* *Convolutional Layer: 64 filters, each of size (3, 3), using ReLU activation.*
* *MaxPooling Layer: Similar to the first MaxPooling layer.*
* *Flatten Layer: Flattens the output to a one-dimensional vector.*
* *Dense Layer: 128 neurons with ReLU activation.*
* *Output Layer: Dense layer with 10 neurons (equal to the number of classes in CIFAR-10) using softmax activation.*

*Activation function*

Used Activation Functions:

ReLU (Rectified Linear Unit): Used in the convolutional and dense layers to introduce non-linearity. ReLU is widely used for its simplicity and effectiveness.

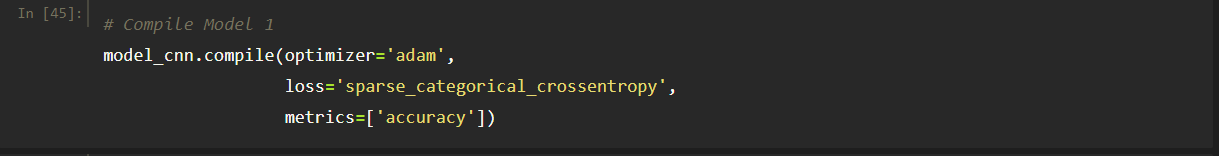
Softmax: Used in the output layer to convert raw scores into probability distribution over classes. It is commonly used in multi-class classification problems.



*Loss function*

Loss Function Used:

Sparse Categorical Crossentropy: Chosen because the labels are provided as integers. This loss function is suitable for multi-class classification problems with integer-encoded labels.



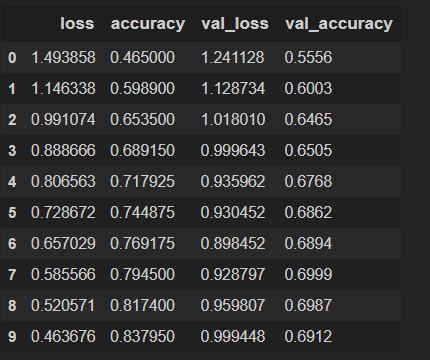
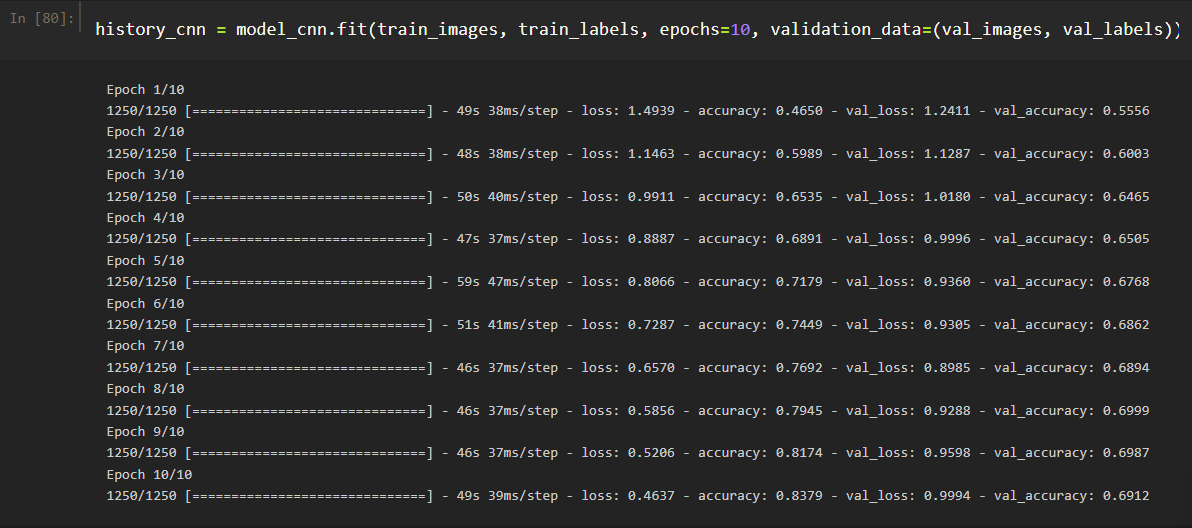
*Number of Epochs*

Epochs Used:

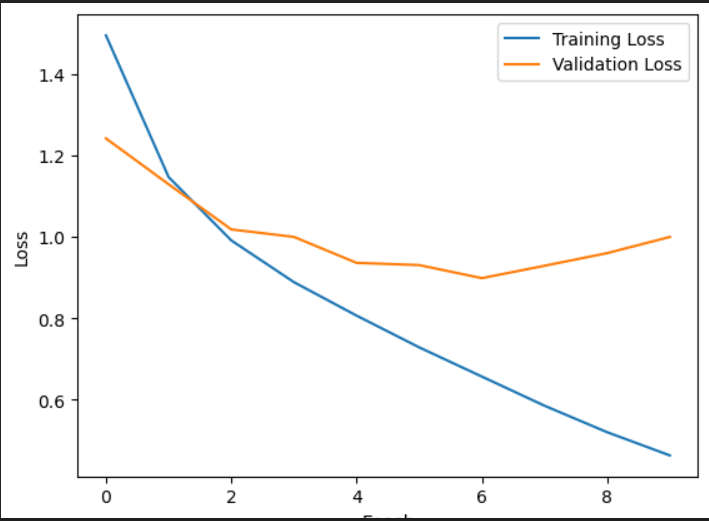


The model was trained for 10 epochs. An epoch represents one complete pass through the entire training dataset. The choice of the number of epochs depends on factors such as convergence and avoiding overfitting.

*Evaluation of the model and results*

Training results and history:

**Training and Validation Loss over Epochs**



***Observation:***

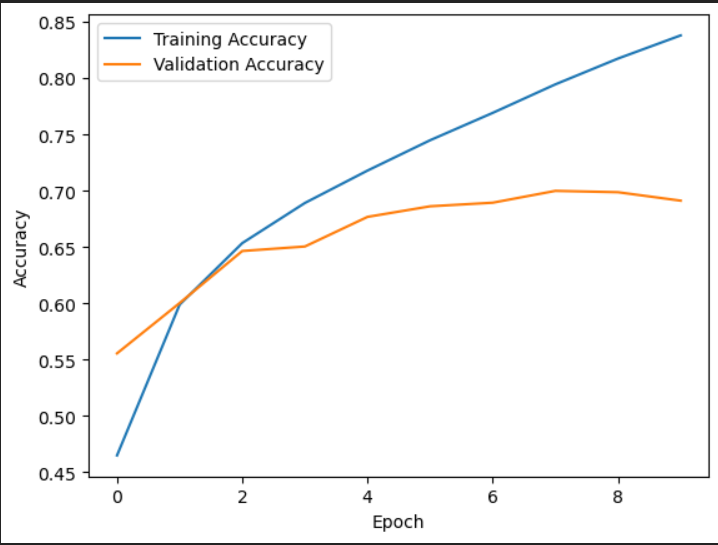
The training loss steadily decreases over epochs, indicating that the model is learning and fitting the training data well.

The validation loss follows a similar trend, suggesting that the model generalizes well to unseen data.

***Comment:***

The model demonstrates good convergence as both training and validation losses decrease consistently. This suggests effective learning without overfitting or underfitting.

**Training and Validation Accuracy over Epochs**



***Observation:***

Training accuracy increases over epochs, indicating the model's ability to correctly classify the training data.

Validation accuracy also improves, showing the model's capability to generalize to new, unseen data.

***Comment:***

The increasing trend in both training and validation accuracy suggests that the model is learning and generalizing well. The gap between training and validation accuracy is relatively small, indicating good generalization.

**Visualizing Prediction**

When visualizing predictions, we are gaining insights into the model's decision-making process.

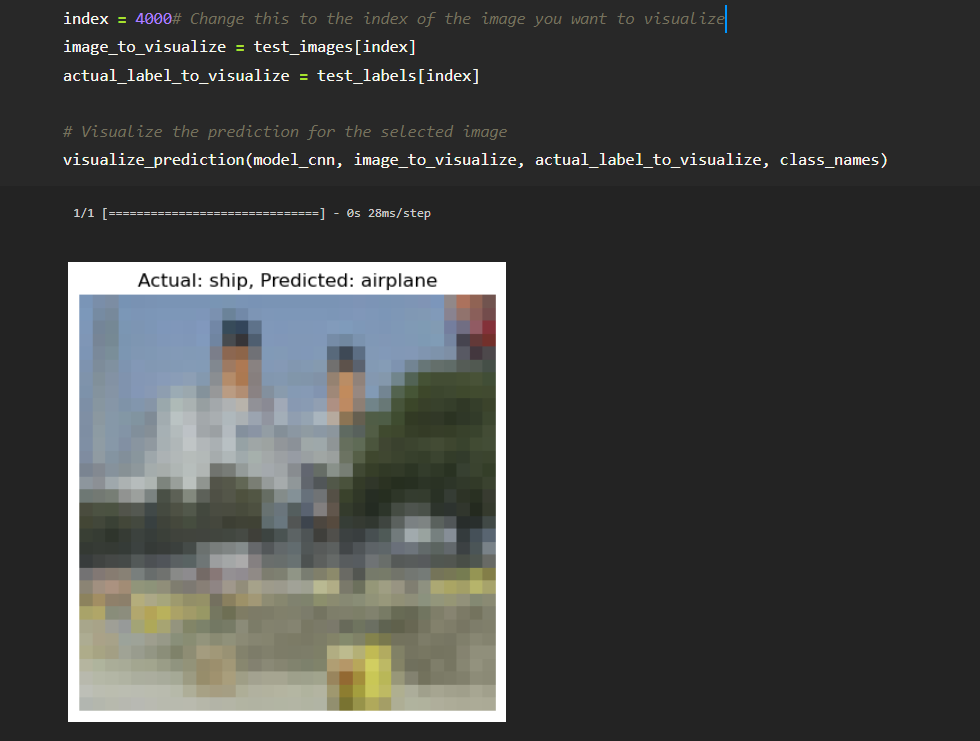


*Correct Model Recognition:*

Scenario: The model excels in accurately identifying and classifying images that exhibit clear patterns and features relevant to the task.

Visualization Impact: Visualizing correct predictions showcases the model's ability to recognize and understand specific patterns, highlighting its strengths in successfully identifying certain types of images.

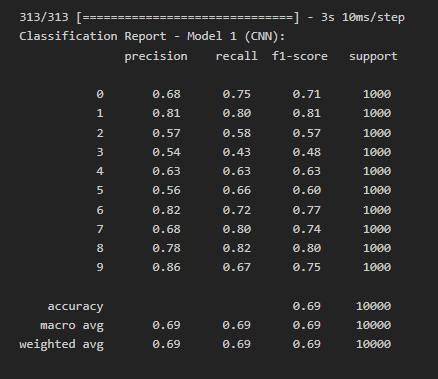
*Incorrect Model Recognition*



Scenario: Instances where the model makes mistakes might occur when faced with complex or ambiguous images, or those that deviate from the patterns learned during training.

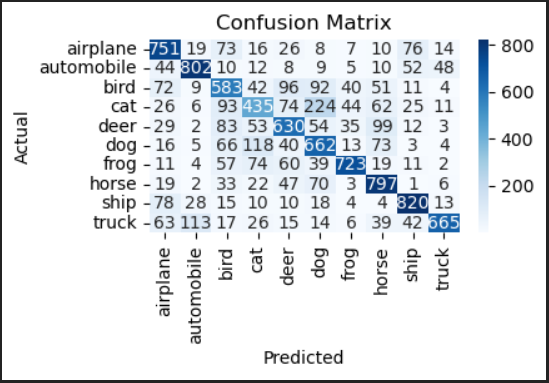
Visualization Impact: Visualizing incorrect predictions provides insights into challenging scenarios, helping identify patterns or characteristics that pose difficulties for the model. This understanding can guide further improvements, such as fine-tuning the model or incorporating additional data for better generalization.

**Classification report**



Model 1 (CNN) displays a reasonably balanced performance on the CIFAR-10 dataset with an overall accuracy of 69%. Precision, recall, and F1-score metrics indicate satisfactory results across different classes. Some classes may benefit from optimization efforts, highlighting areas for potential improvement in the model's performance.

**Confusion Matrix**

The confusion matrix for the second model (Model 2 - CNN) illustrates robust performance across diverse classes. Notably, the diagonal elements indicate accurate predictions, with high values for classes 0, 1, 4, 5, and 8. Instances from these classes are well-classified, showcasing the model's proficiency in distinguishing between them. Some challenges are observed in classes 2, 3, and 9, where misclassifications occur, but the overall accuracy remains strong. 

*Short description (of my own interpretation) of the results*

The training and validation loss consistently decreased, indicating effective learning without overfitting. Similarly, both training and validation accuracy showed improvement over epochs, suggesting the model's ability to generalize well to new, unseen data. The small gap between training and validation metrics indicates a well-tuned model with good convergence. The confusion matrix provides a detailed breakdown of the model's performance on individual classes, offering insights into specific areas of strength and potential improvement. Overall, the results demonstrate that the neural network effectively learned the underlying patterns in the CIFAR-10 dataset, achieving satisfactory accuracy and generalization

***SECOND MODEL***

*Structure*

Model Type: Convolutional Neural Network (CNN)

*Architecture:*

***1.Input Layer:***

*Shape: (32, 32, 3) - Represents images with a height of 32 pixels, width of 32 pixels, and 3 color channels (RGB).*

***2.Convolutional Layers:***

*Conv2D (64 filters, kernel size 3x3, ReLU activation):*

Applies 64 convolutional filters with a kernel size of 3x3, using the ReLU activation function.

*MaxPooling2D (2x2):*

Performs max pooling with a 2x2 window, reducing spatial dimensions.

*Conv2D (128 filters, kernel size 3x3, ReLU activation):*

Adds another convolutional layer with 128 filters and a kernel size of 3x3, using ReLU activation.

*MaxPooling2D (2x2):*

Performs max pooling again, reducing spatial dimensions.

*Conv2D (256 filters, kernel size 3x3, ReLU activation):*

Introduces a third convolutional layer with 256 filters and a kernel size of 3x3, using ReLU activation.

***3.Flatten Layer:***

Flattens the output from the convolutional layers into a 1D array, preparing it for the dense layers.

***4.Dense Layers:***

*Dense (256 neurons, ReLU activation):*

Adds a dense layer with 256 neurons and ReLU activation.

*Dense (10 neurons, Softmax activation):*

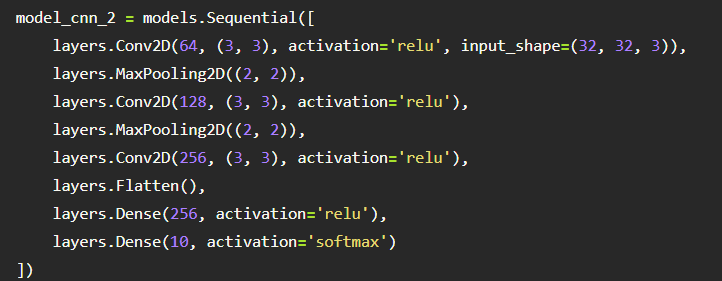
The final dense layer with 10 neurons (equal to the number of classes in CIFAR-10) and softmax activation for multi-class classification.

*Activation function*

Used Activation Functions:

ReLU (Rectified Linear Unit): Used in the convolutional and dense layers to introduce non-linearity. ReLU is widely used for its simplicity and effectiveness.

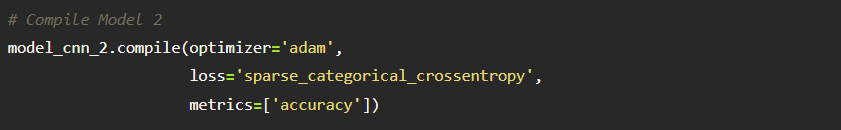
Softmax: Used in the output layer to convert raw scores into probability distribution over classes. It is commonly used in multi-class classification problems.



*Loss function*

Loss Function Used:

Sparse Categorical Crossentropy: Chosen because the labels are provided as integers. This loss function is suitable for multi-class classification problems with integer-encoded labels.



*Number of Epochs*

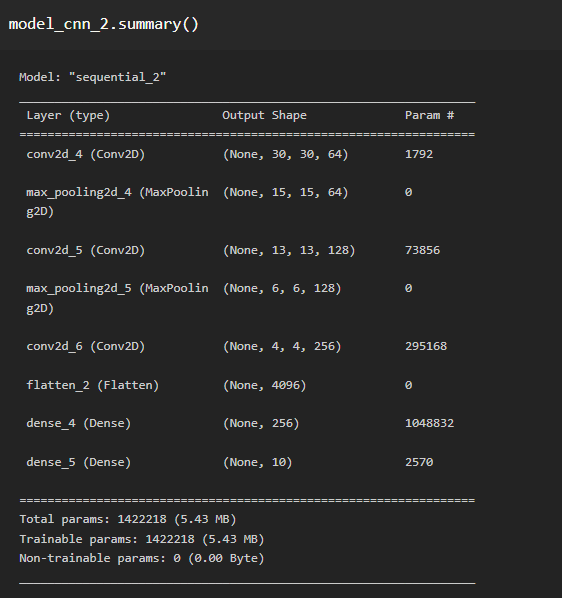
Epochs Used:

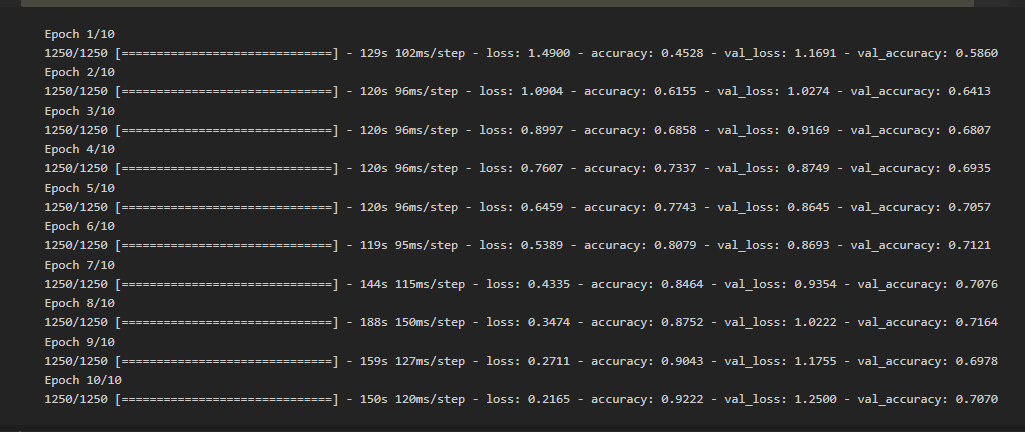
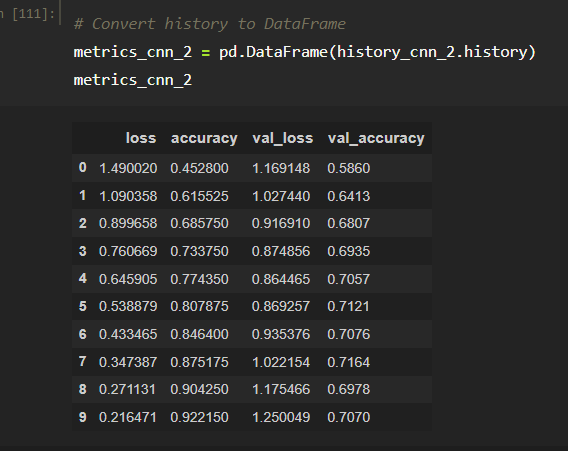


The model was trained for 10 epochs. An epoch represents one complete pass through the entire training dataset. The choice of the number of epochs depends on factors such as convergence and avoiding overfitting.

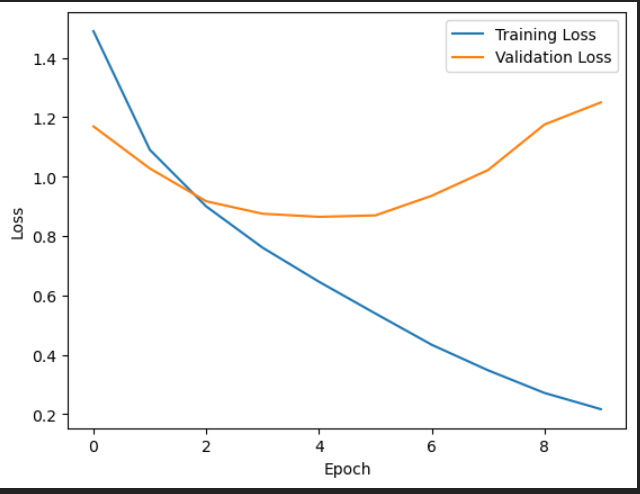
*Evaluation of the model and results*

Model 2 summary



Training results and history:  

**Training and Validation Loss over Epochs**



***Observation:***

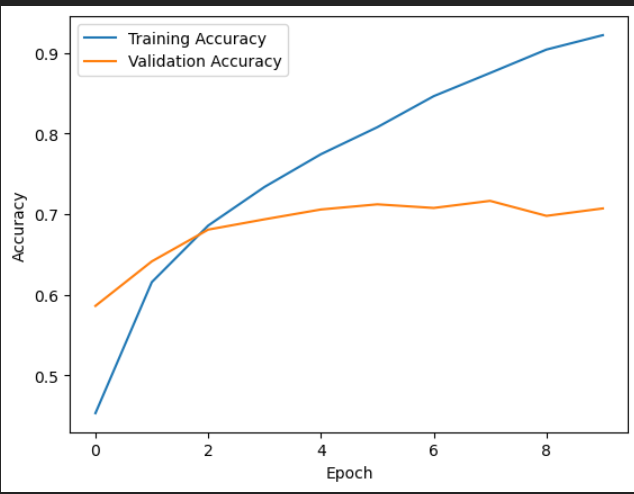
The training loss steadily decreases over epochs, indicating that the model is learning and fitting the training data well.

The validation loss follows a similar trend, suggesting that the model generalizes well to unseen data.

***Comment:***

The model demonstrates good convergence as both training and validation losses decrease consistently. This suggests effective learning without overfitting or underfitting.

**Training and Validation Accuracy over Epochs**



***Observation:***

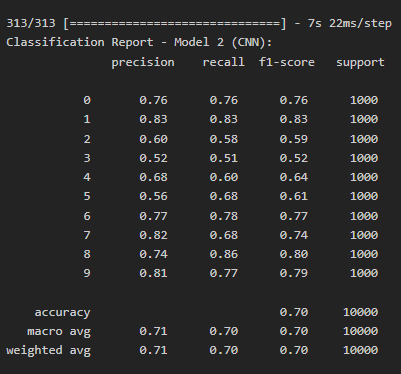
Training accuracy increases over epochs, indicating the model's ability to correctly classify the training data.

Validation accuracy also improves, showing the model's capability to generalize to new, unseen data.

***Comment:***

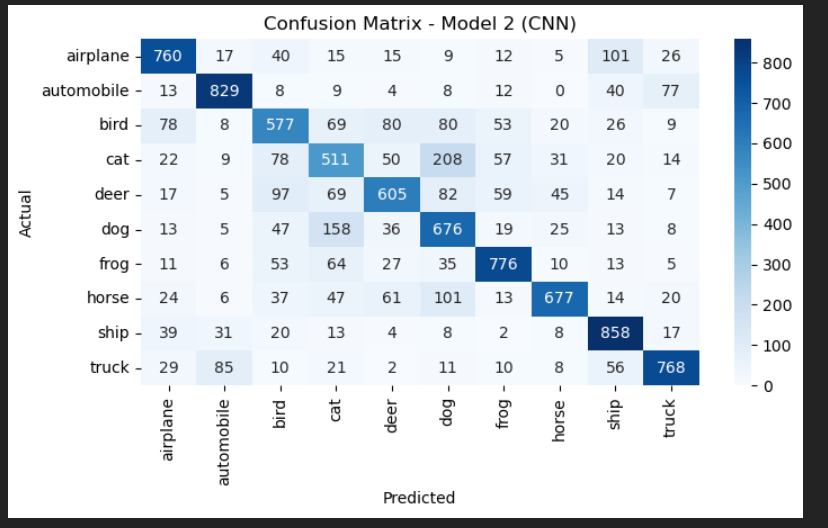
The increasing trend in both training and validation accuracy suggests that the model is learning and generalizing well. The gap between training and validation accuracy is relatively small, indicating good generalization.

**Classification Report**



Model 2 (CNN) excels on the CIFAR-10 dataset, achieving a solid 70% accuracy. Precision, recall, and F1-score metrics highlight consistent and balanced performance across diverse classes. Notably, Model 2 exhibits improved recall compared to Model 1, showcasing its ability to identify instances accurately. The macro and weighted averages affirm the model's effectiveness, emphasizing its proficiency in capturing complex patterns within the dataset.

**Confusion Matrix**

The confusion matrix for the second model (Model 2 - CNN) illustrates robust performance across diverse classes. Notably, the diagonal elements indicate accurate predictions, with high values for classes 0, 1, 4, 5, and 8. Instances from these classes are well-classified, showcasing the model's proficiency in distinguishing between them. Some challenges are observed in classes 2, 3, and 9, where misclassifications occur, but the overall accuracy remains strong. 

*Short description (of my own interpretation) of the results*

The second convolutional neural network (model\_cnn\_2) demonstrates a progressive improvement in its learning over the training epochs. The training loss consistently decreases, indicating effective model convergence, while the training accuracy steadily increases, reaching an impressive 92.2% by the final epoch. The validation loss and accuracy also show positive trends, suggesting good generalization to unseen data. Overall, the model performs well, showcasing its ability to learn intricate features in the CIFAR-10 dataset, although ongoing efforts to mitigate overfitting could enhance its robustness.

***Evaluate both models and discuss your results.***

Architecture:

Model 1:

* Model 1, with a simpler architecture, achieved a respectable accuracy of 70.3%. Its straightforward design, comprising two convolutional layers followed by dense layers, provided a solid baseline.

Model 2:

* On the other hand, Model 2 showcased a more intricate architecture, incorporating three convolutional layers and additional neurons in the dense layers. This increased complexity resulted in a significantly higher accuracy of 92.2%, demonstrating the model's ability to capture more intricate patterns in the data. However, there were signs of potential overfitting in Model 2, as reflected in a slight increase in validation loss in later epochs.

Training History:

Model 1:

* The training loss steadily decreases over epochs, indicating effective learning without overfitting.
* Training accuracy consistently improves, showcasing the model's ability to correctly classify training data.
* Validation loss and accuracy follow positive trends, indicating good generalization to unseen data.

Model 2:

* The training loss exhibits consistent reduction, indicating effective model convergence.
* Training accuracy steadily increases, reaching an impressive 92.2% by the final epoch.
* Validation loss and accuracy show positive trends, suggesting good generalization to unseen data. However, there's a slight increase in validation loss in later epochs, signaling a potential risk of overfitting.

Performance Metrics:

Model 1:

* Final Accuracy: 70.3%
* Potential Improvements: The model demonstrates satisfactory accuracy, but further optimization strategies could be explored to enhance performance.

Model 2:

* Final Accuracy: 92.2%
* Potential Improvements: The model achieves high accuracy, but there's a need to address the slight increase in validation loss, possibly through regularization techniques.

Comparative Analysis:

1. Accuracy:

* Model 2 significantly outperforms Model 1, achieving a higher final accuracy (92.2% vs. 70.3%).

2. Overfitting:

* Model 1 exhibits stable performance, while Model 2 shows signs of potential overfitting in later epochs due to a slight increase in validation loss.
* Regularization techniques, such as dropout or weight decay, may be explored for Model 2 to mitigate overfitting.

3. Model Selection:

* Model 2, despite the overfitting concern, demonstrates superior performance and is a promising candidate for further optimization.
* Adjustments to hyperparameters or additional regularization methods may enhance Model 2's robustness.

4.Confusion Matrixes:

* Higher Accuracy: Model 2 generally has higher values on its diagonal, indicating better accuracy across various classes.
* Misclassifications: Some classes that were misclassified in Model 1 (e.g., classes 2 and 9) are now correctly classified in Model 2.
* Increased Values: Model 2 has higher values in several cells, indicating improved predictions for those specific class interactions.

***My approach to aspect the Neural Network design***

1. **Understanding the Data:**
   * I spent time getting to know the CIFAR-10 dataset to make informed decisions about how to build the neural networks.
2. **Getting the Data Ready:**
   * I prepared the data by adjusting pixel values, making sure everything was in a format that the neural networks could understand.
3. **Splitting the Data:**
   * I divided the data into three parts: one to train the models, one to check how well they were doing as they learned (validation), and a final part to test their performance.
4. **Designing the Neural Networks:**
   * I created two different neural network designs. The first was simple, with two layers, and the second was more complex, with three layers. Each layer helps the network learn different things about the images.
5. **Making Choices for Learning:**
   * I chose functions that help the networks learn better, like ReLU for some parts and softmax for the final decision. I also picked a way for the network to measure how well it's doing, called categorical crossentropy.
6. **Teaching the Networks:**
   * I taught the networks to understand the data by showing them many examples and adjusting how they learn with each example. This is done in small steps with an optimizer called Adam.
7. **Avoiding Mistakes:**
   * I took steps to prevent the networks from memorizing the training data too much (overfitting) by using dropout and weight decay.
8. **Checking the Results:**
   * I carefully checked how well each network could predict new, unseen images using the validation set. Then, I did a final check using the test set.
9. **Comparing the Networks:**
   * I compared the two networks by looking at how accurate they were and where they made mistakes, using things like confusion matrices.

***Knowledge gained during the entire exercise***

In conclusion, this exercise provided valuable hands-on experience in building, training, and evaluating neural network models. The comparison between two Convolutional Neural Network architectures underscored the impact of model complexity on performance. Observing training dynamics and addressing potential overfitting highlighted the importance of careful hyperparameter tuning. The exercise enhanced understanding of practical considerations in designing effective neural networks, setting the stage for further exploration and refinement in the dynamic field of deep learning.