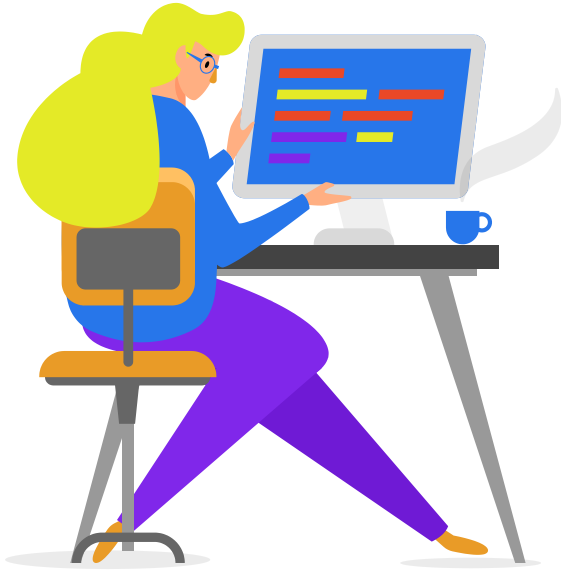


Object detection and Classification using Convolutional Neural Networks on CIFAR-10

Karim Triki
s5528602

Ines Haouala
s5483776

Table of Contents



01

Project Aims And Objectives

02

CNN Concepts

03

**Why the CIFAR-10 Dataset?
Data Preprocessing**

04

8 Models Training

05

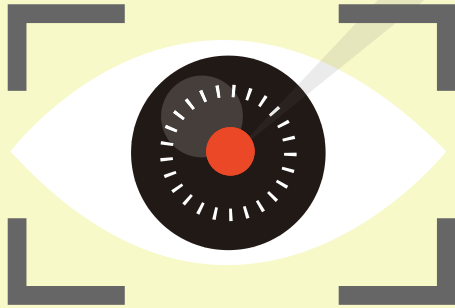
Results Analysis

06

Further Optimization Techniques

Project Aims And Objectives

**Some machine
learning applications**

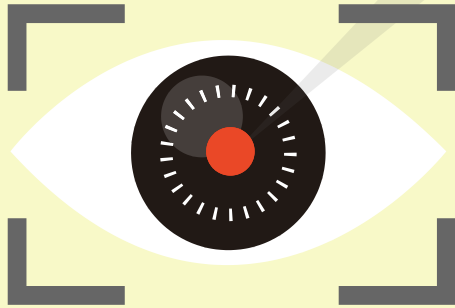


Aims

The main goal is studying the effect of different hyperparameters on the performance of Convolutional Neural Network (CNN) using CIFAR-10 dataset and improving the accuracy and efficiency of the CNN model by performing hyperparameter optimization and using advanced data preprocessing and enhancement techniques.

Project Aims And Objectives

**Some machine
learning applications**

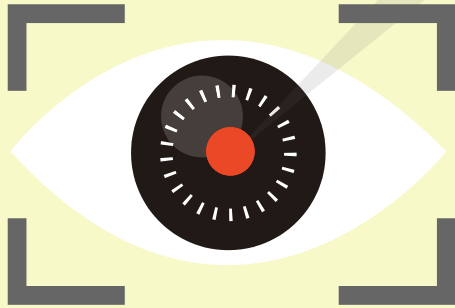


**DATA SET
COLLECTION**

By Using CIFAR-10 dataset of 60,000
32x32 color images divided into 10
classes, to train and evaluate the CNN
model.

Project Aims And Objectives

**Some machine
learning applications**

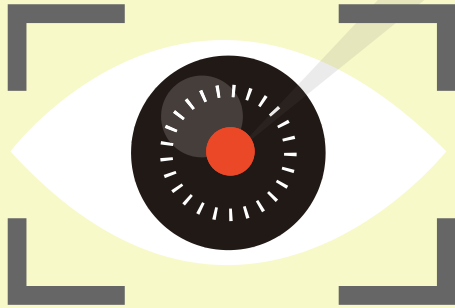


OUTCOME & APPLICATION

By Identify the optimal hyperparameter settings that increase the accuracy and generalizability of the model and applying the findings to improve CNN models for image classification tasks.

Project Aims And Objectives

**Some machine
learning applications**



**Image
Preprocessing**

normalizing pixel values in images from 0 to 1 In other part applying one-hot encoding to class labels in preparation for the CNN model.

Project Aims And Objectives

Some machine learning applications

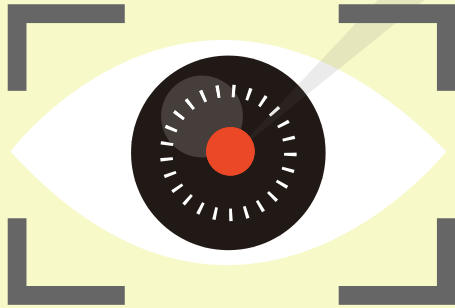


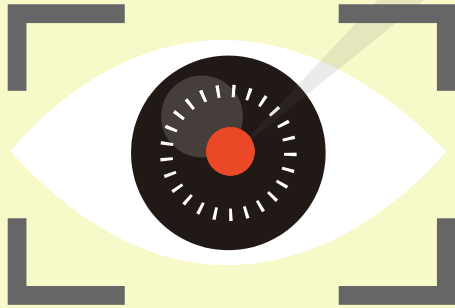
Image Augmentation

implementation of applications for image enhancement such as rotation, zoom, width shift and height shift to increase the variety of training data and improve the robustness of the model.

And Use ImageDataGenerator to manage training and validation data flows to implement this enhancement.

Project Aims And Objectives

**Some machine
learning applications**

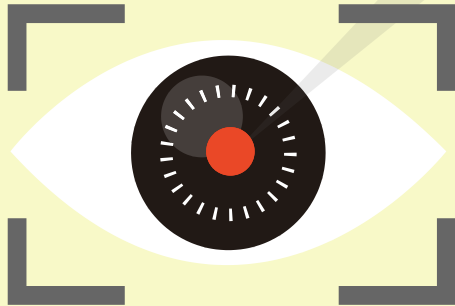


**Directory
Structure**

Organizing the dataset into appropriate directories for training, validation and testing to simplify the data loading and preprocessing steps.

Project Aims And Objectives

Some machine learning applications

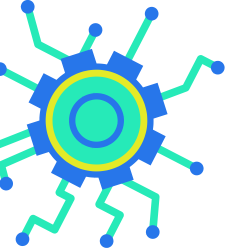


Training/Validation/ Testing Data Split

Monitoring system performance by dividing training data into training and validation subsets to prevent overfitting.

Ensure a balanced evaluation of the model during training by:

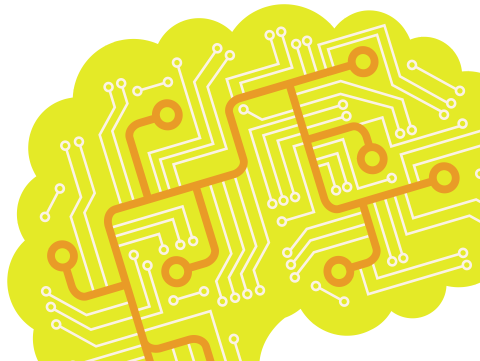
- Training Set: 80% of total dataset
- Validation Set: 20% of total dataset
 - Testing Set: 10,000 images

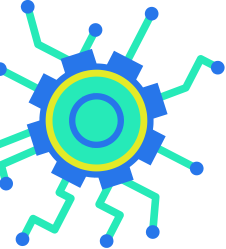


CNN Concepts

WHAT'S CNN

Convolutional neural networks (CNNs) are deep learning models specifically designed to process network data sets, such as pictures, they are particularly effective for image classification, object recognition, and other computer vision tasks because of their ability to capture spatial patterns in images

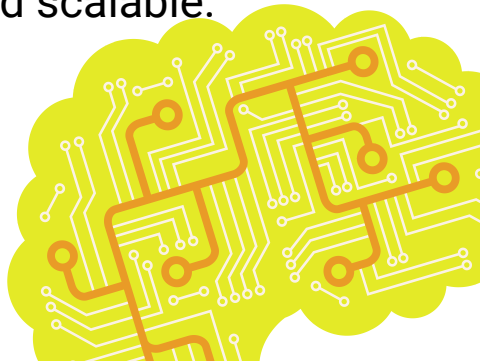


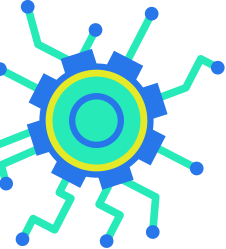


CNN Concepts

THE DIFFERENCE BETWEEN CNN & Fully Connected Neural Networks

A fully connected network (FNN) has neurons in one layer connected to all neurons in the next layer, which is computationally expensive and does not scale well for large models. In contrast, CNNs, use convolutional layers to process data in a local manner, reducing the number of parameters and making the model more efficient and scalable.





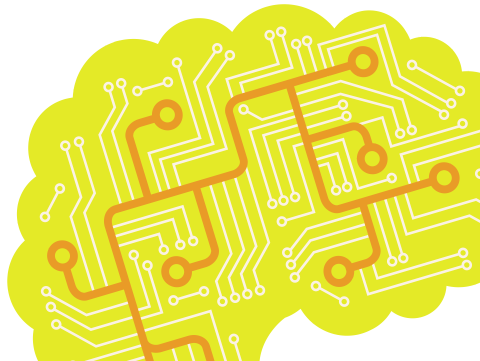
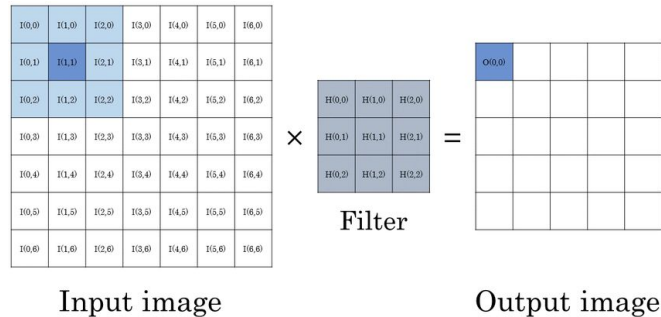
CNN Concepts

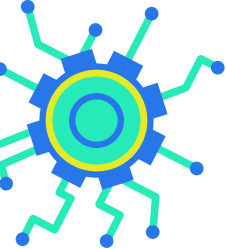
Convolution Filter or Kernel

A convolution filter (or kernel) is a small shape used to implement effects such as blurring, sharpening, and edge detection.

How it works:

The filter draws (convolves) across the input image, creating a feature map by calculating a dot product between the filter input and the receive location





CNN Concepts

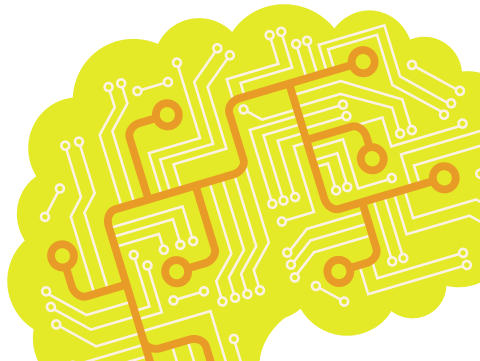
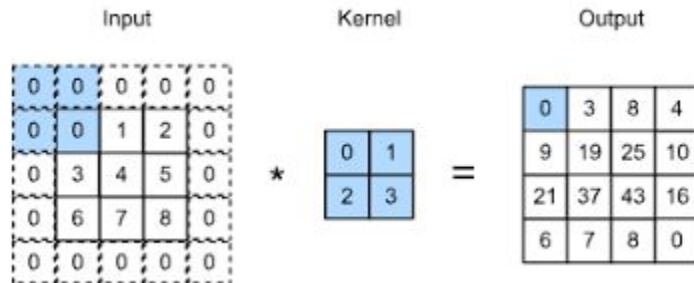
STRIDE AND PADDING

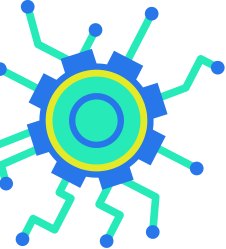
Stride:

The number of pixels by which the filter traverses the input image. Step 1 causes the filter to move one pixel at a time, while step 2 cuts every other pixel.

Padding:

Adding additional pixels around the boundaries of the input image to control the size of the output. The most common padding types are 'valid' (no padding) and 'same' (output size is the same as input).



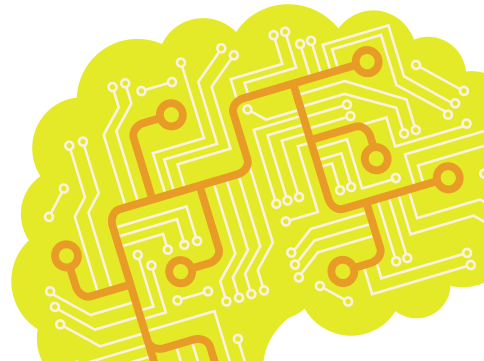


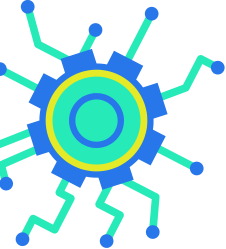
CNN Concepts

Activation Functions

It introduces nonlinearity into the model, enabling it to recognize complex patterns.

Common functions: ReLU (Rectified Linear Unit), Sigmoid, Tanh.



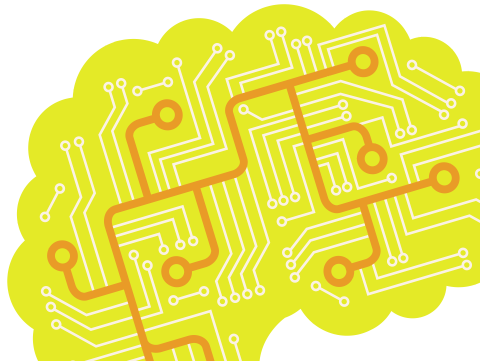


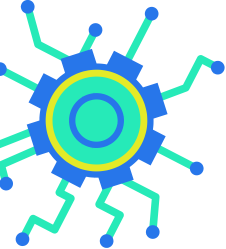
CNN Concepts

Fully Connected Layers and Model Evaluation Fully Connected Layer

Also known as the Dense Layer or Output Layer, it connects every node in the previous layer to every node in the current layer.

It aggregates the features extracted by the convolutional and pooling layers for the final classification. overfitting, optimum, inappropriate





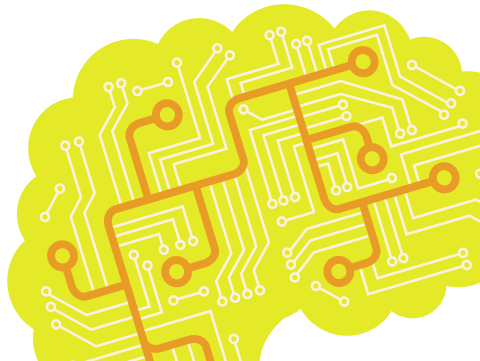
CNN Concepts

Fully Connected Layers and Model Evaluation Fully Connected Layer

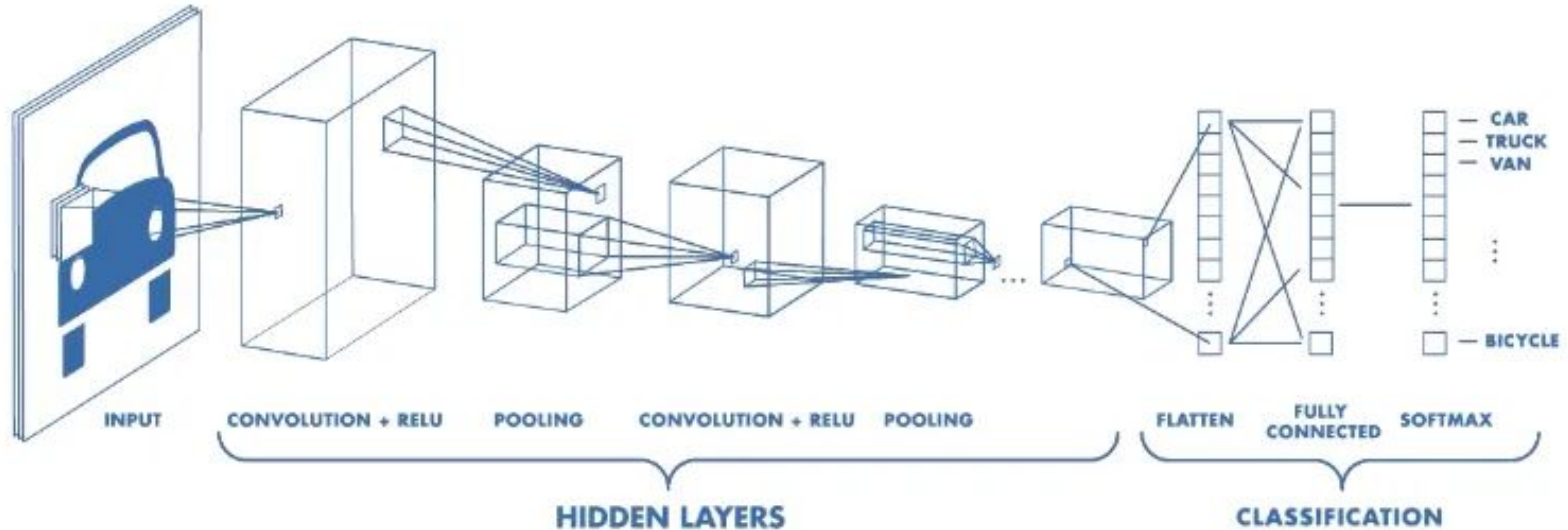
Overfitting: The model learns training data too well, including noise and detail, resulting in poor generalization to new data.

Underfitting: The model is too weak to capture underlying patterns in the training data, resulting in poor performance.

Optimum: The model achieves a good balance between bias and variance, captures the underlying pattern well and effectively generalizes to new data.



CNN General Structure



Source: <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

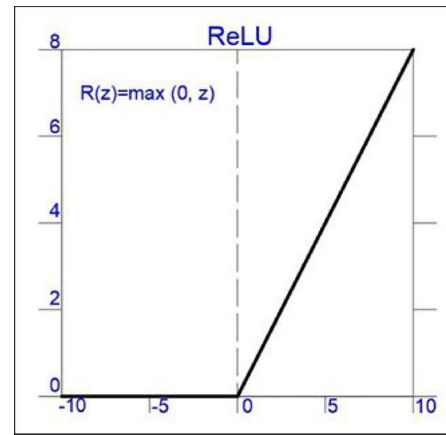
Max Pooling Layer

6. Max Pooling Layer

Max pooling is a sample-based discretization scheme. The aim is to simplify the input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing the consideration of features within bind subregions

Usage in Models:

reduce the spatial dimensions of the input volume and remove key features.



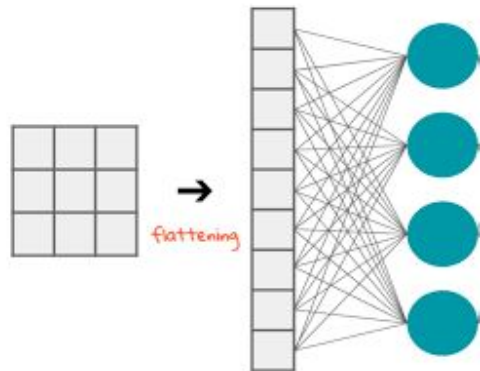
Flatten Layer

7. Flatten Layer

The Flatten layer in Keras is used to flatten the input. For example, if an input size is (batch_size, 28, 28, 1), then the output size will be (batch_size, 784), so the data is converted to a 1D array.

Usage in Models:

Transforming a 2D matrix into a vector to feed in fully connected layers.



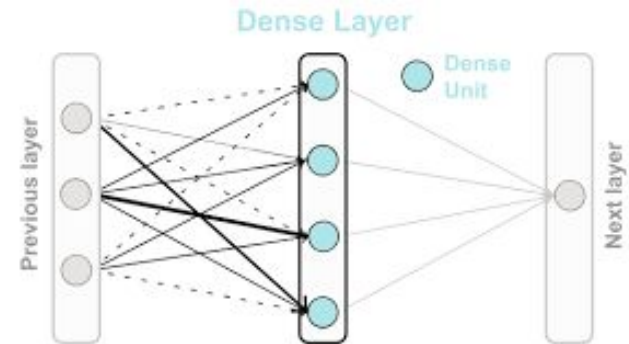
Dense (Fully Connected) Layer

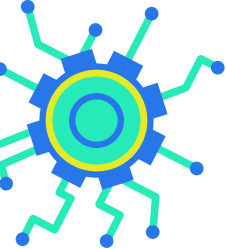
8. Dense Layer

A dense layer is a fully connected layer, meaning that each neuron receives input from all neurons in the previous layer. This layer is standard in most neural networks.

Usage in Models:

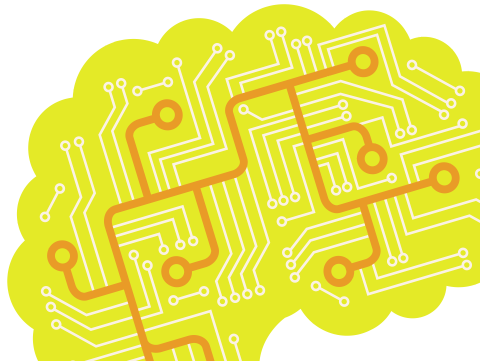
Classification or regression functions based on features extracted by previous layers.





General Architecture of the CNN Based on Our Code

- **Input layer:** Takes a 32x32 RGB image.
- **Convolutional Layers:** Apply filters to filter features such as edges, textures, and objects.
- **Pooling Layers:** Reduce the dimensionality of feature maps while retaining important information.
- **Fully connected layers:** Collect objects to make a final forecast.



CNN Training

Data Loading and Preprocessing:



- Ensure a balanced evaluation of the model during training by:
 - Training Set: 80% of total dataset
 - Validation Set: 20% of total dataset
 - Testing Set: 10,000 images
- The images are normalized by dividing the pixels by 255.0 to create a scale from 0 to 1
- Labels have been single-hot encoded using `tf.keras.utils.to_categorical()` to prepare

Data Augmentation:



- To increase model generalization and robustness, you implement the data enhancement technique using the `ImageDataGenerator` from Keras.
- Techniques for enlarging images include rotating, zooming, rotating, and rotating images.
- This helps to generate multiple training samples from the existing CIFAR-10 dataset.
-


Model Definition:




- You define several CNN models (from Model 1 to Model 8) that differ in architecture using the Keras Sequential API.
- Each model typically consists of convolutional layers followed by activation functions (e.g., ReLU), pooling layers, and fully connected (dense) layers.
- The output layer uses softmax activation for multiclass classification.

Cnn training


Compilation and Optimization:

- 
- The models are compiled using an appropriate optimizer (e.g., learning rate of 0.001 with ADAM) and loss functions (e.g. categorical cross-entropy for multiclass classification).
 - Controls training progress through callbacks such as EarlyStopping to prevent overfitting by stopping training if it stops solving validation losses.

Training Execution:

- 
- Models are trained using the fit() function, where training data (train_images) and validation data (val_images) are fed into the model.
 - The batch size and number of epochs are configured to control the number of samples processed before updating the model weights and the number of iterations of the data set during training, respectively.

Evaluation and Analysis:

- 
- After training each instance, you test its performance in both training and validation sets.
 - Metrics such as accuracy and loss are calculated to measure how well the model learns and generalizes.
 - Training and validation accuracy/loss curves are analyzed to understand model behavior and determine if changes in hyper parameters are necessary.

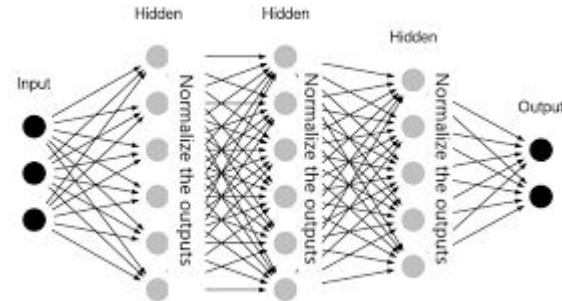
Batch Normalization

4. Batch Normalization

Batch normalization is a technique for improving the speed, performance, and robustness of artificial neural networks. This helps keep average production close to 0 and standard deviation close to 1, increasing input accuracy for each small batch.

Usage in Models:

Used between levels to improve training consistency and speed and to normalize activation.



Tools Used: Keras

1. Keras

Keras is a high-level neural network API, written in Python and running on top of TensorFlow, CNTK, or Theano. It is capable of simple and fast prototyping, supports convolutional networks and recursive networks, and runs seamlessly on both CPU and GPU.

Usage in Models:

Defining and training neural networks.
provides examples of layers, optimizers, and utilities for buildings.



Tools Used: TensorFlow

2. TensorFlow

TensorFlow is an open source library for statistics and machine learning. It provides a comprehensive, flexible ecosystem of tools, libraries, and community resources that enable researchers to push breakthroughs in ML, and developers to easily build and develop ML-enabled applications

Usage in Models:

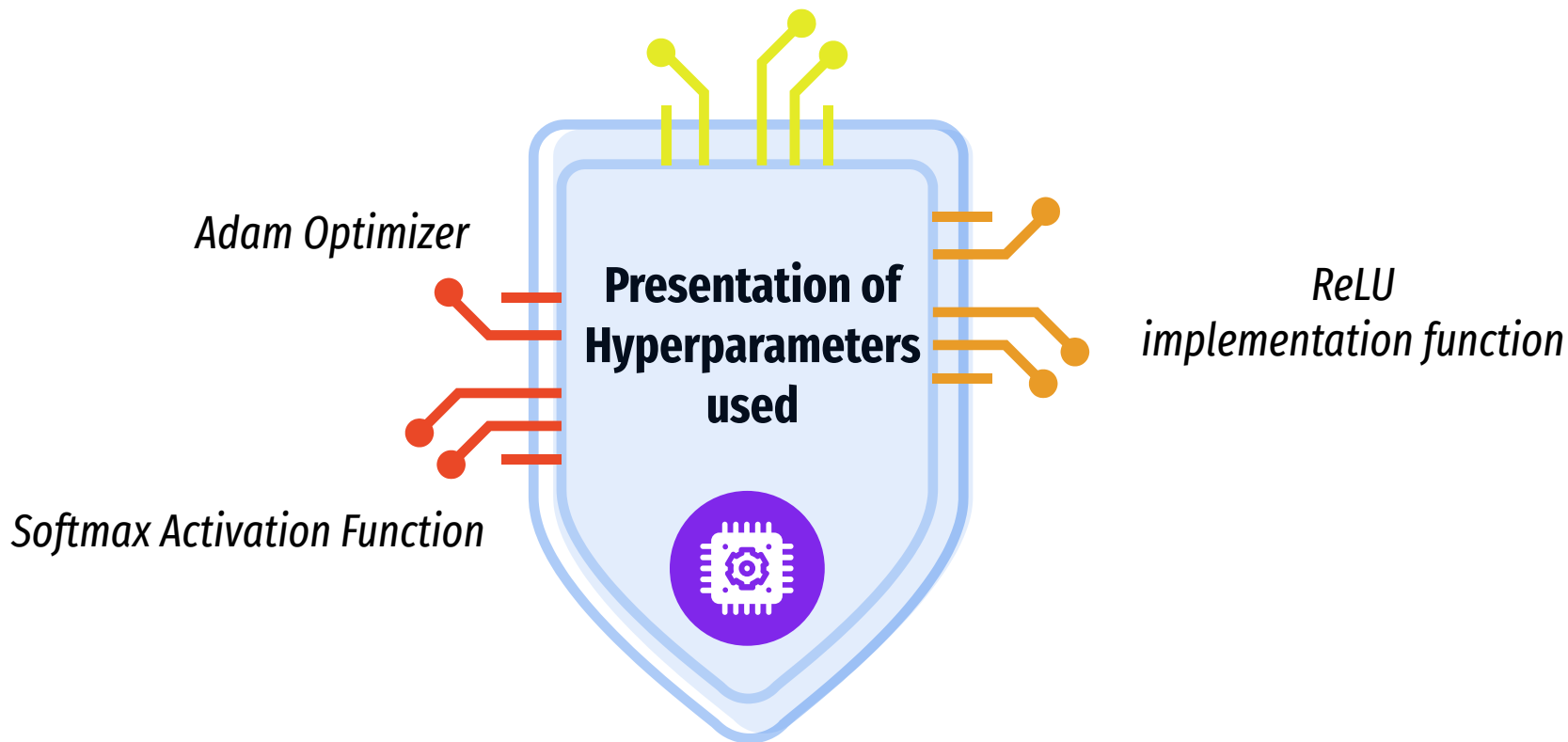
Backend engine for Keras.

Low performance with tensor manipulation.



TensorFlow

Important Concepts



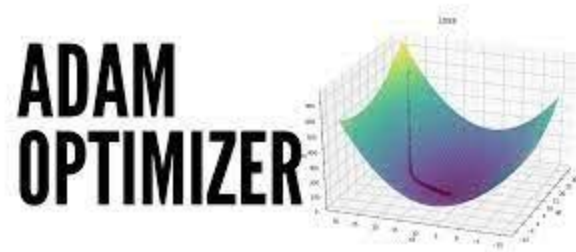
Adam Optimizer

3. Adam Optimizer

ADAM (Adaptive Moment Estimation) is an optimization algorithm that can be used to iteratively update network loads based on training data instead of classical stochastic gradient descent. ADAM combines the advantages of AdaGrad and RMSProp algorithms to provide an optimization algorithm It can handle sparse gradients in noisy problems.

Usage in Models:

Optimizing the weight of the model by reducing the loss function during training.



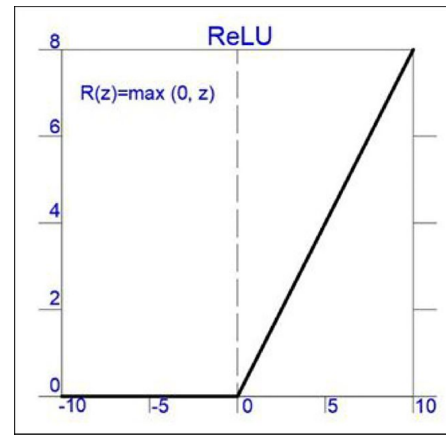
ReLU implementation function

5. ReLU implementation function

A Rectified Linear Unit (ReLU) is a functional function defined as the positive portion of its logic. For many neural networks, this has become the default activation function because the model it uses is easier to train and generally gives better performance.

Usage in Models:

introduce the nonlinearity for the model after each convolutional or fully connected layer.



Softmax Activation Function

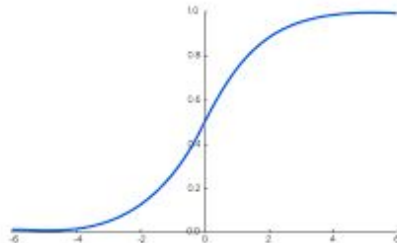
9. Softmax Activation Function

The Softmax function is a function function that converts a vector of values into a probability distribution. Each value in the output ranges from 0 to 1, and the sum of all output values is 1.

Usage in Models:

Generally, the output layer of a classification network is used to represent class probabilities.

Softmax Function



Why the CIFAR-10 Dataset?

01

Applied Research

It is mostly used for image classification comparison.
It allows equivalent evaluation of different algorithms.

02

Diverse but simple

Classes: 10 categories (airplane, car, bird, dog, cat, dog, frog, horse, ship, truck).
Images: 60,000 (training 50,000, testing 10,000).

03

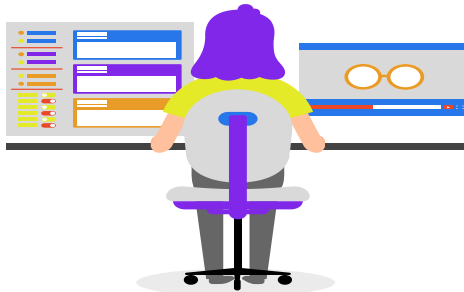
Moderate Size

Image Dimension: 32x32 pixels.
Balance: Manageable size for quick deployment and effective training.

04

Preprocessed and ready for use

Uniform images make preprocessing easier.
Popular libraries like TensorFlow and Keras can be easily installed.



05

Ideal for CNNs

It is designed to evaluate and train Convolutional Neural Networks (CNNs) suitable for image data.

06

Historical Interpretation

Produced by Alex Krizhevski, Vinod Nair and Jeffrey Hinton.
Important for enhancing deep learning research.

06

Community and resources

Full support, courses and resources are available.

07

Learning value

Great for machine learning and deep basics.
Demonstrate data development and preprocessing.

Data Preprocessing



**Cifar-10
preprocessing**

Batch Normalization

01

It is a way to improve the effectiveness of deep muscle training. This involves normalizing the inputs of each layer so that they have a mean of zero and a variance of one. This method is used in small data sets, hence the name "Batch Normalization".

One Hot Encoding

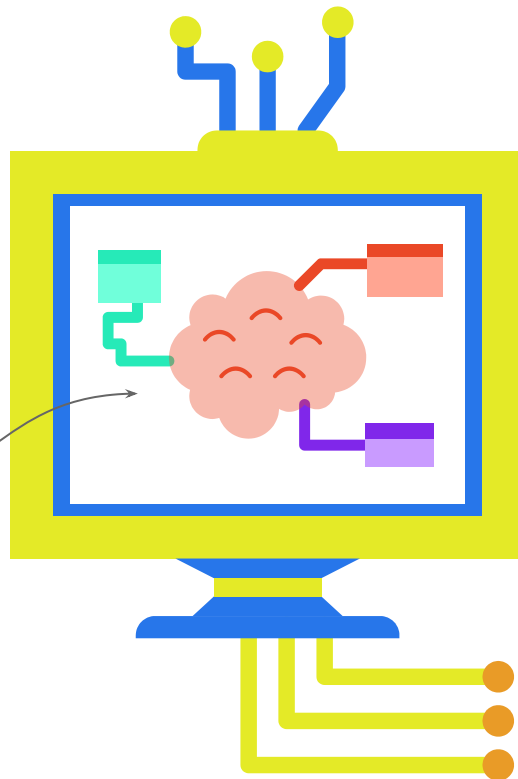
02

One Hot Encoding is a method of converting categorical labels into a Binary vectors that can be fed to machine-learning algorithms to improve their performance. In this encoding, each class is represented as a binary vector, with the only representation being class labeled corresponds to 1, all other positions are labeled 0s.

Data Augmentation

Data Augmentation is a technique for artificially increasing the size and variety of a training data set by modifying existing data. This is accomplished through transformations that do not alter the underlying properties but rather change the shape.

In the CIFAR-10 data set, we can apply different enhancements to each image. When we have an image of a cat, we can create several variations of this image using various enhancements, such as a rotating cat, an enlarged cat, or an inverted cat



- **Enhances Model Generalization**
- **Reduces Overfitting**
- **Increases Dataset Size**

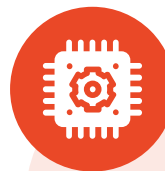
**Why we should
Use it?**

- **Rotation**
- **Zooming**
- **Horizontal Flipping**

**Common
methods**

Presentation of the 8 Trained Models

Model 1: Baseline Model



Convolutional Layer

- Filters: 32
- Filter Size: 3x3

Max Pooling Layer

- Pool Size: 2x2

Convolutional Layer

- Filters: 64
- Filter Size: 3x3

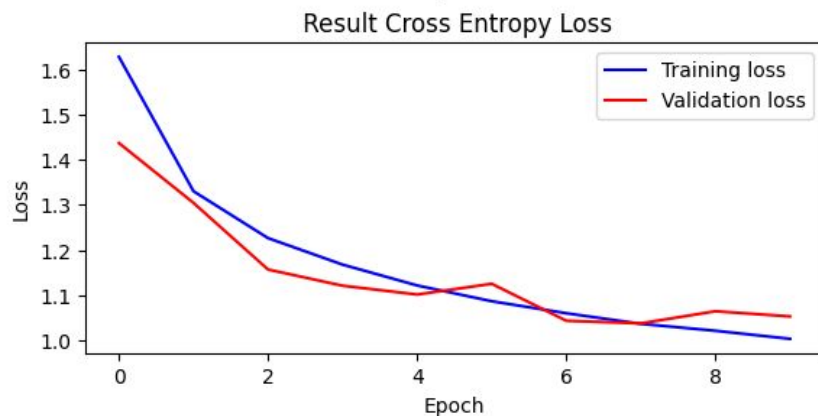
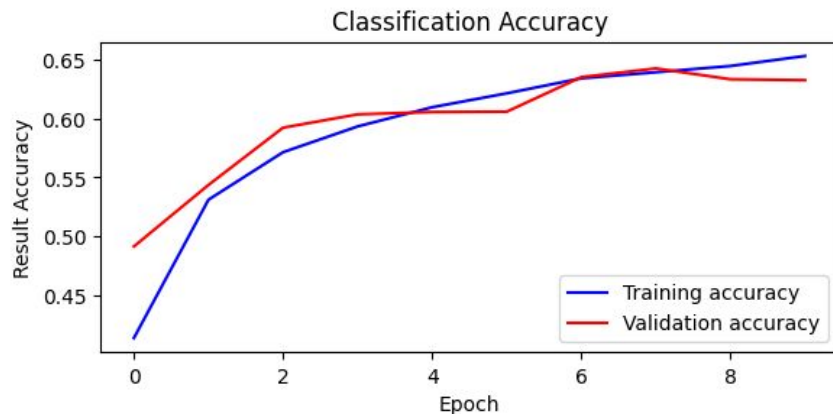
Max Pooling Layer

- Pool Size: 2x2

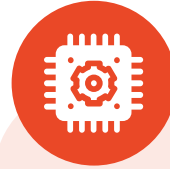
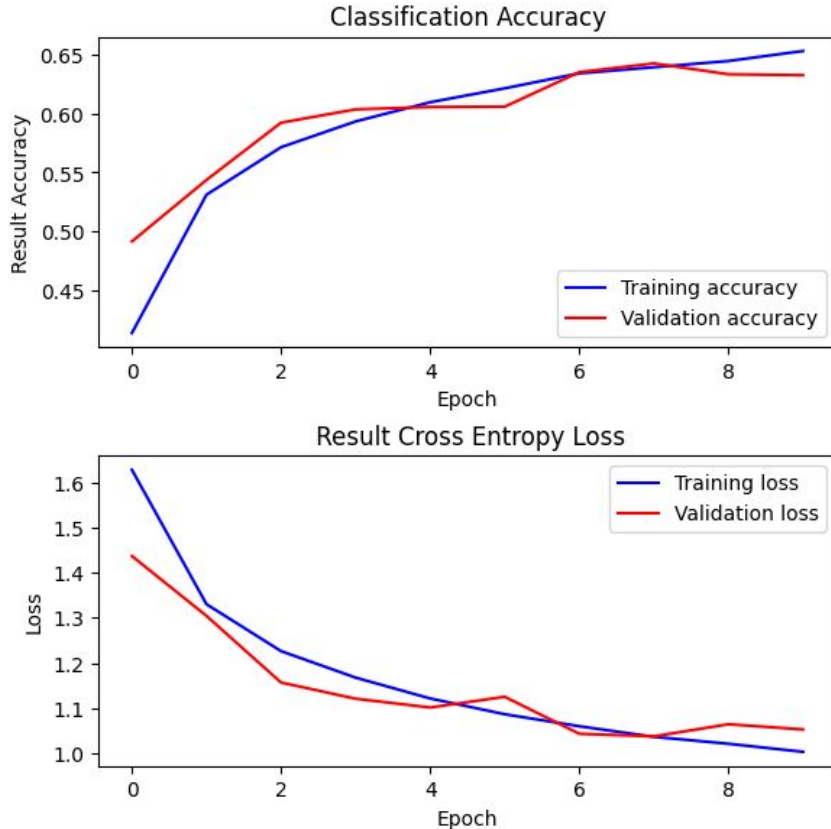
Flatten Layer

Output Layer

- Neurons: 10
- Activation: Softmax



Model 1: Baseline Model



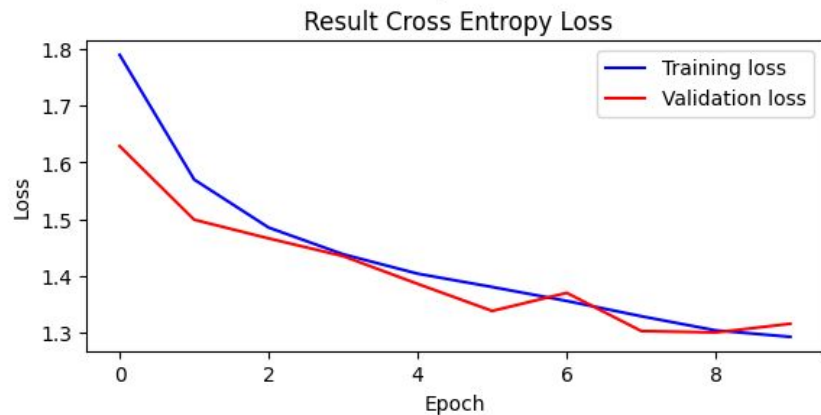
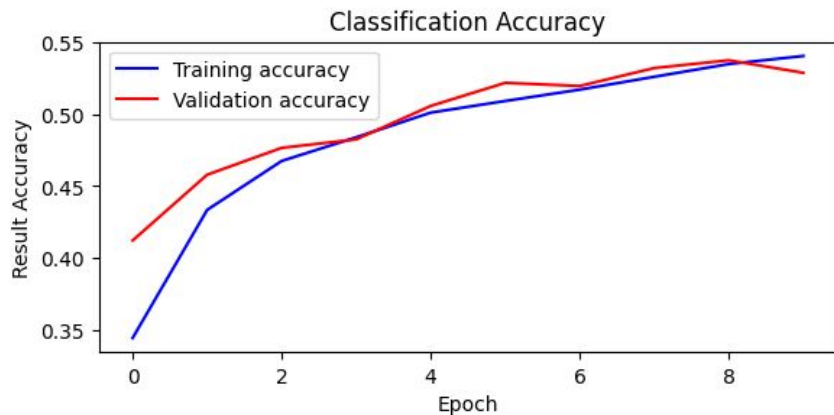
Model 1: Evaluation & Performance

Performance: The model performs reasonably well but lacks sophistication "refinement"

Overfitting/Underfitting: Underfitting is possible, as indicated by the relatively high loss value.

Improvements are needed: Consider adding regularization techniques or increasing model complexity for better performance.

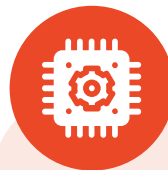
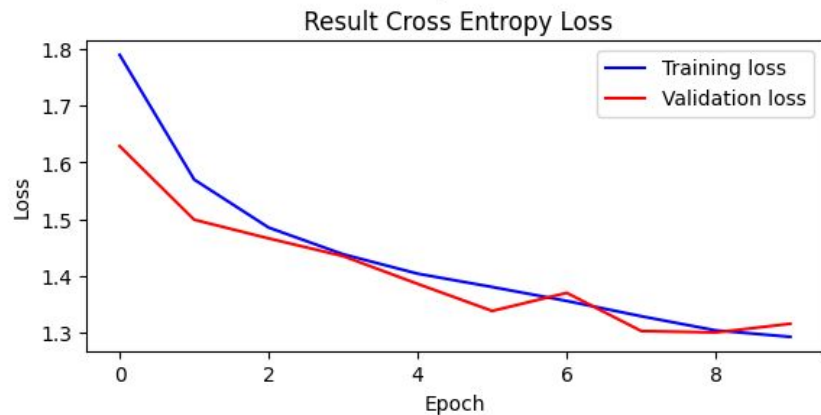
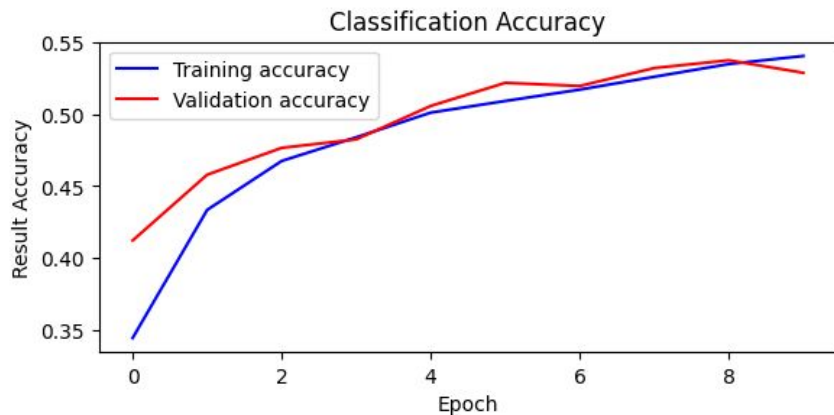
Model 2: Filter Size Hyperparameter



Model 2: Evaluation & Performance

In Model 2, the kernel or filter size was increased from 3x3 in the baseline model to 7x7

Model 2: Filter Size Hyperparameter



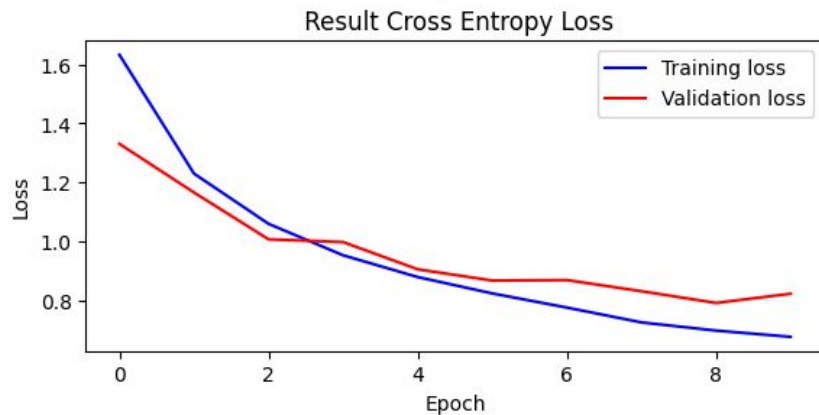
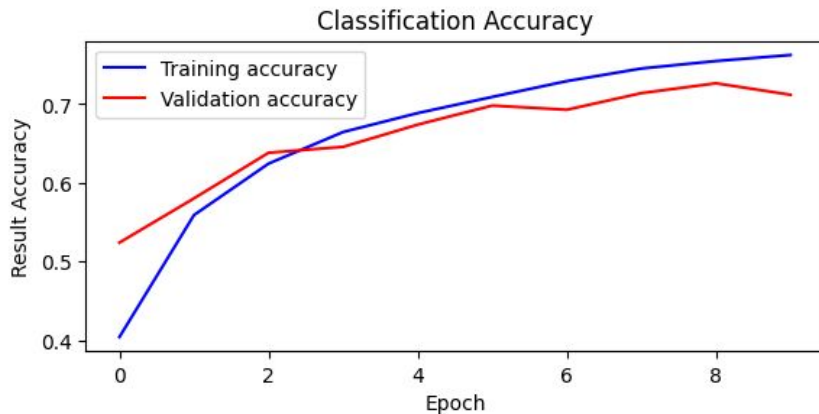
Model 2: Evaluation & Performance

Performance: Batch normalization shows promise to stabilize learning and speed up convergence.

Overfitting/Underfitting: Stability reduces the risk of overfitting, but additional adjustments may be necessary.

Improvements Needed: Try batch normalization settings or on parameters to improve performance.

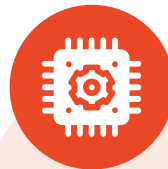
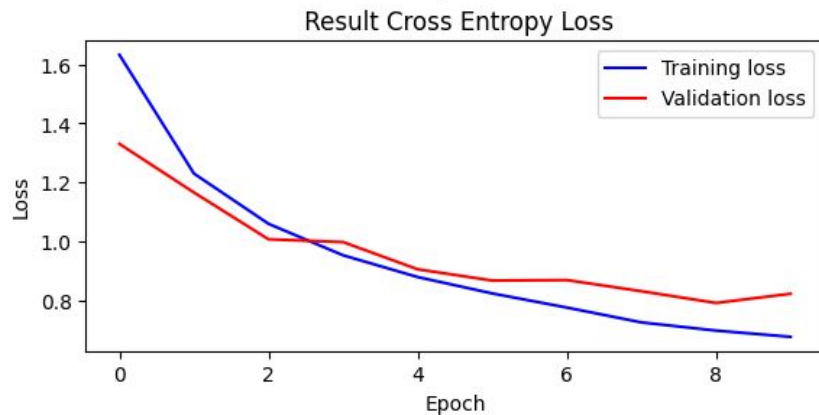
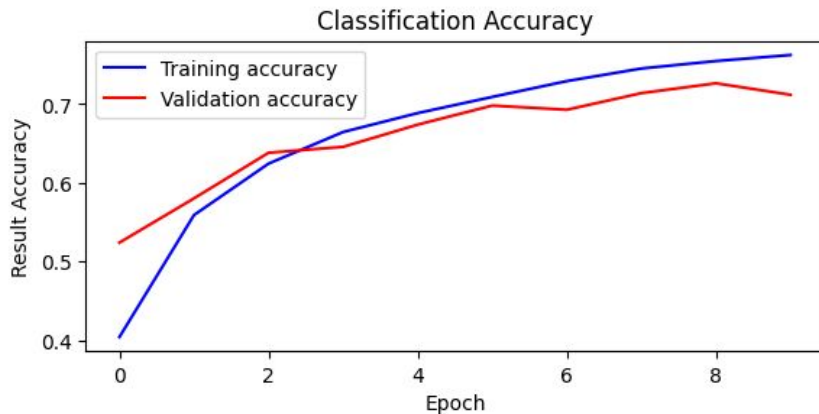
Model 3: Number of Convolutional Layer



Model 3: Evaluation & Performance

In Model 3, two additional convolutional layers were added to the baseline model, thus in total four convolutional layers

Model 3: Number of Convolutional Layer



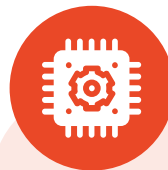
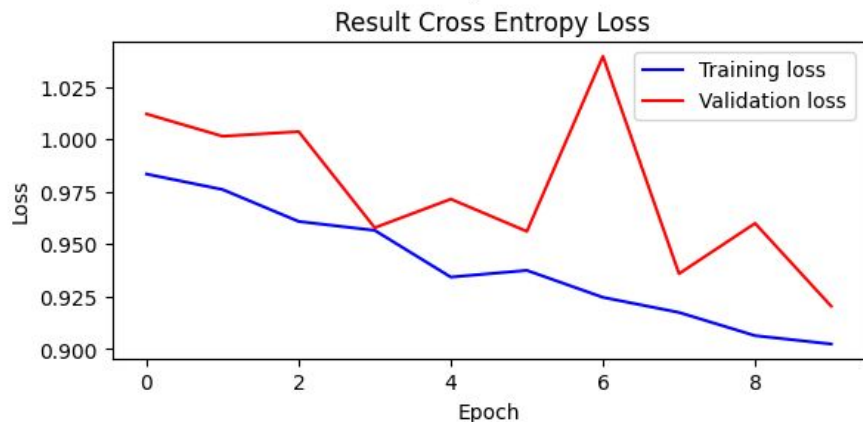
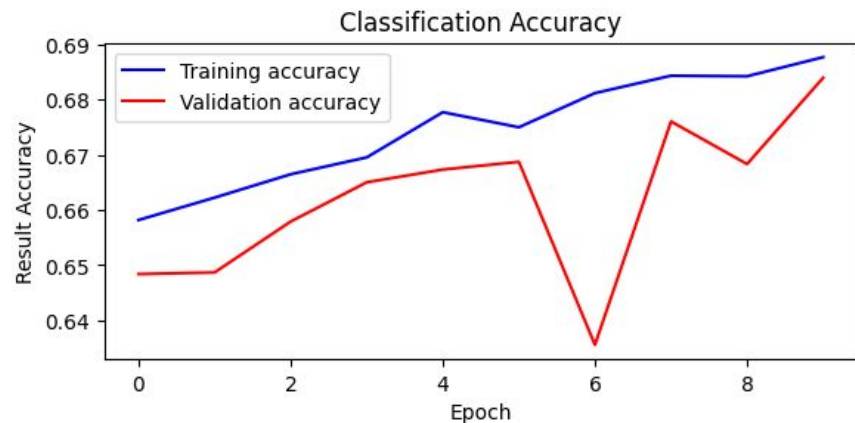
Model 3: Evaluation & Performance

Performance: A learning rate of 0.0001 prevents training, slowing learning.

Overfitting/Underfitting: Underfitting was found due to insufficient number of studies.

Improvements Needed: Increase the number of classes or change the adapter settings for faster matching.

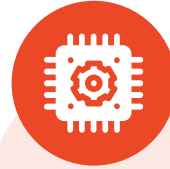
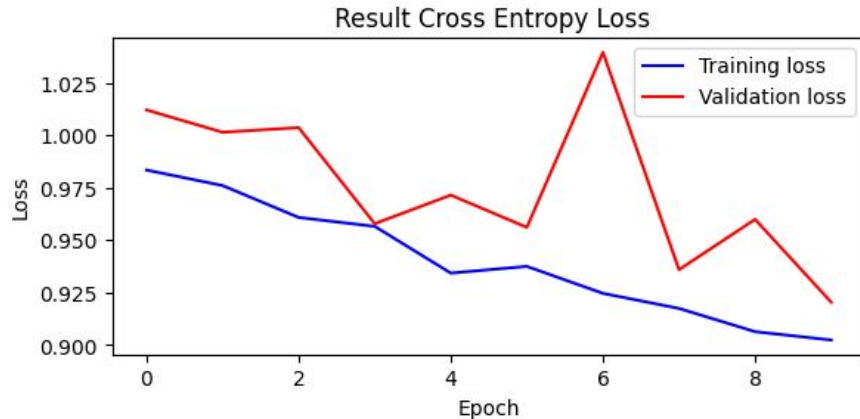
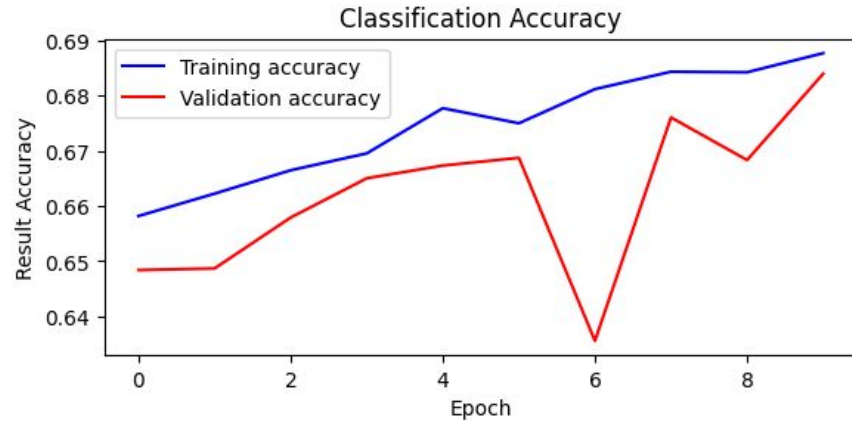
Model 4: Batch size Hyperparameter



Model 4: Evaluation & Performance

In Model 4, the batch size is increased from 128 from baseline model to 512

Model 4: Batch size Hyperparameter



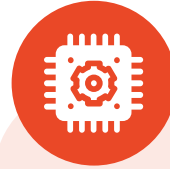
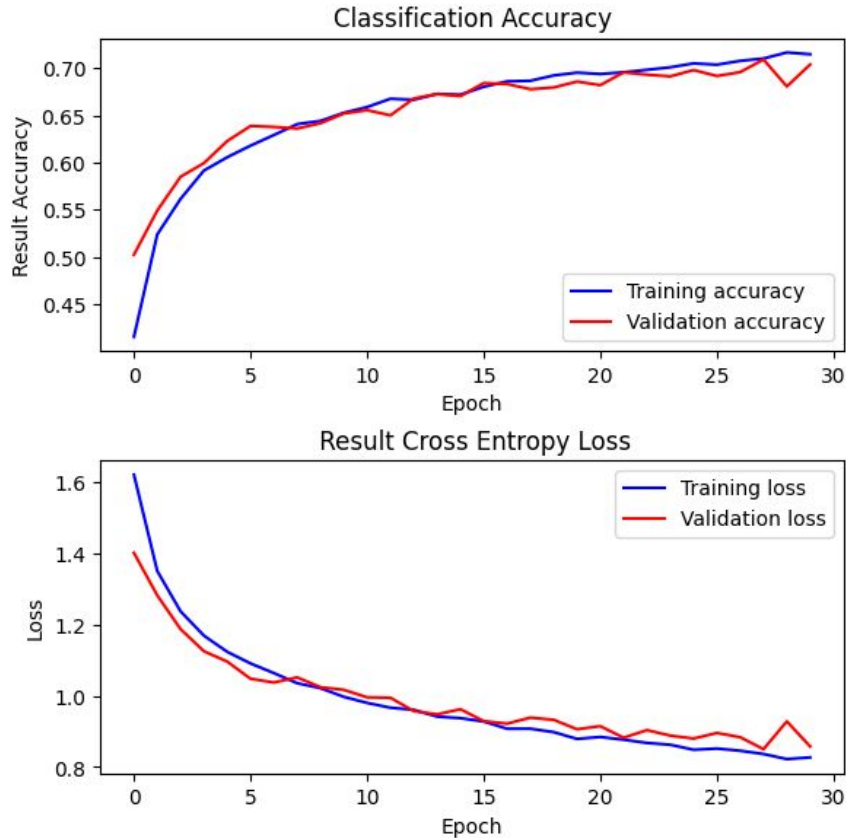
Model 4: Evaluation & Performance

Performance: Data enhancement effectively improves model generalization.

Overfitting/Underfitting: Reduces the risk of overfitting due to various training issues.

Improvements Needed: Continue to use data for improvement; Consider looking for new ways to improve.

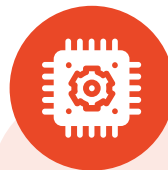
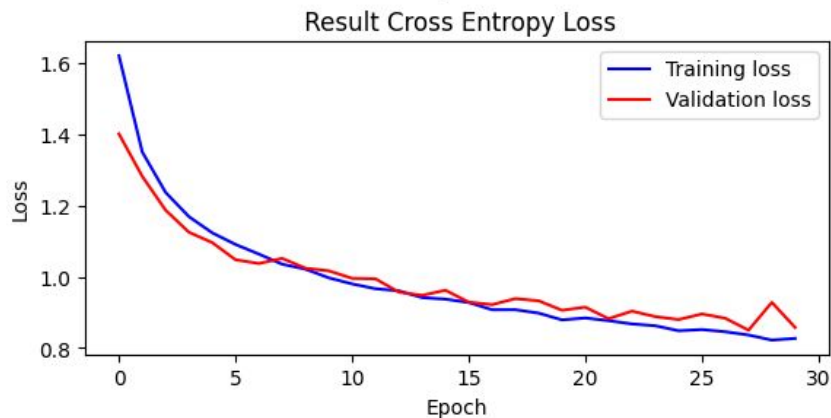
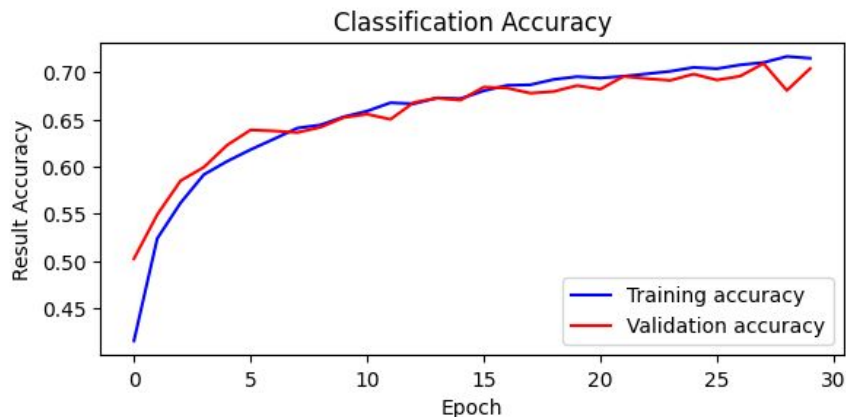
Model 5: Epoch Hyperparameter



Model 5: Evaluation & Performance

In model 5, the Epoch is increased from 10 in base model to 30

Model 5: Epoch Hyperparameter



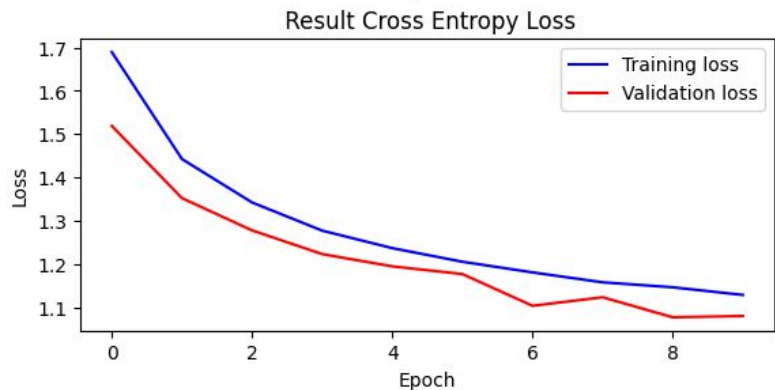
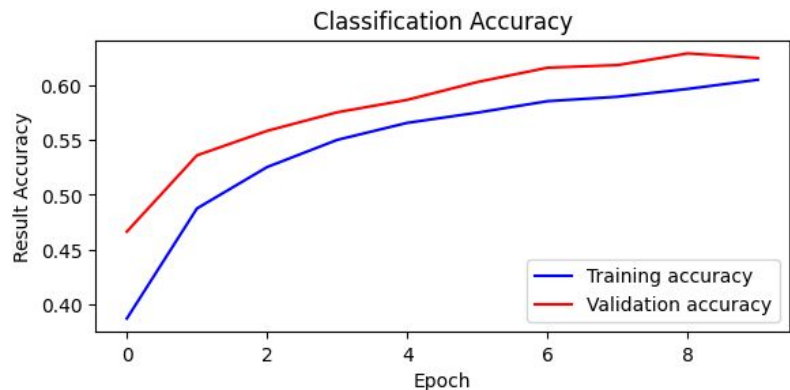
Model 5: Evaluation & Performance

Performance: Early stops help avoid overfitting and improve validation performance.

Overfitting/Underfitting: A balanced model with minimal risk of overfitting.

Improvements Needed: IFine tune the initial position parameters for the best performance.

Model 6: Dropout Rate Hyperparameter (Dropout Regularization)



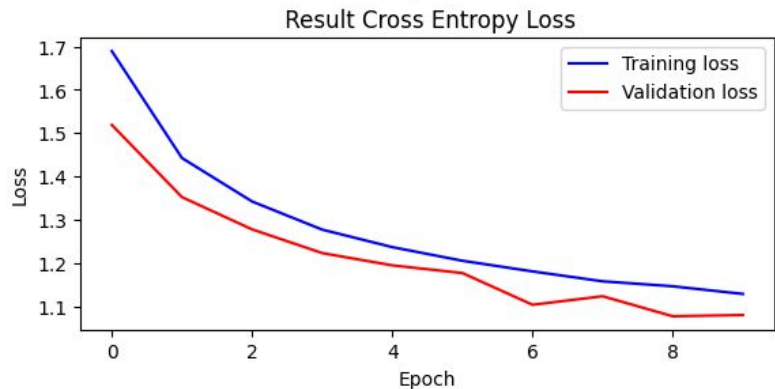
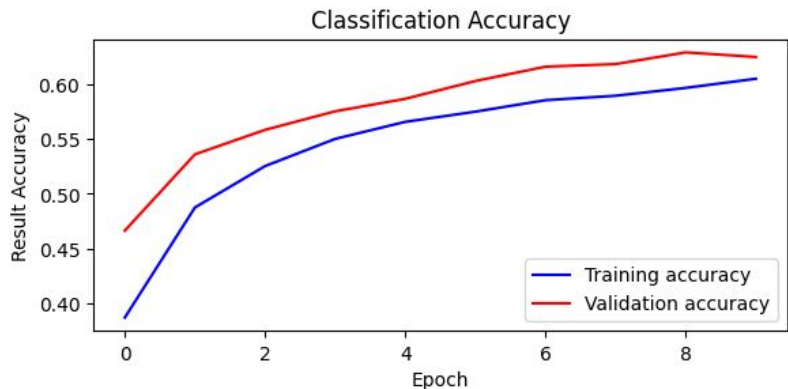
Model 6: Evaluation & Performance

In model 6, the following changes were added:

- Added Dropout rate of $p=0.2$ (keep probability=0.8) at input layer
- Added Dropout rate of $p=0.4$ (keep probability=0.6) after the last pooling layer

As a general rule the dropout should be added after the last pooling layer as well as the input layer.

Model 6: Dropout Rate Hyperparameter (Dropout Regularization)



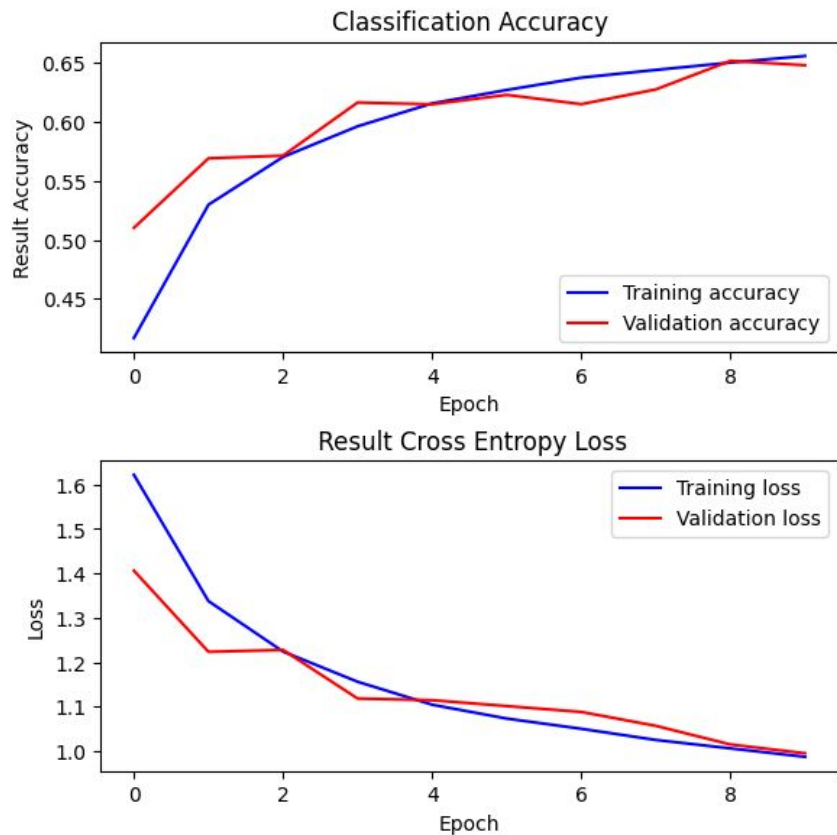
Model 6: Evaluation & Performance

Performance: Dropout effectively reduces overfitting and increases validation performance.

Overfitting/Underfitting: Dropout helps prevent overfitting, improves normalization.

Improvements Needed: use different dropout rates / additional regularization techniques.

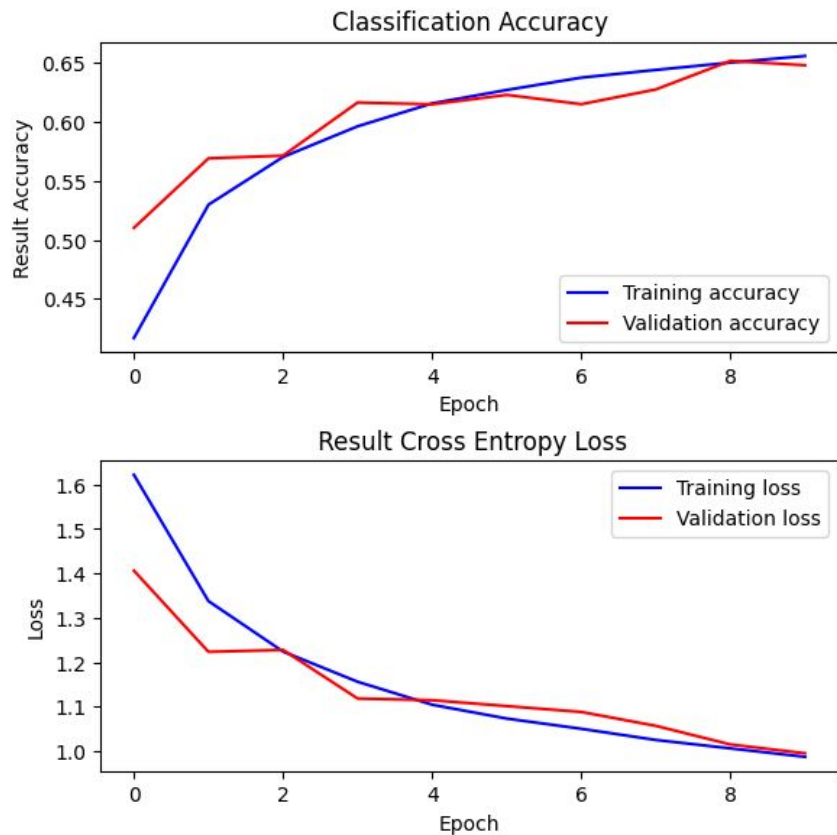
Model 7: Optimizer Hyperparameter



Model 7: Evaluation & Performance

In model 7, we overwrite the optimizer from Adam to SGD but you can test with other algorithm optimizers as shown in the commented code below

Model 7: Optimizer Hyperparameter



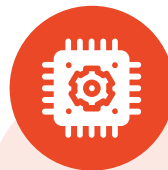
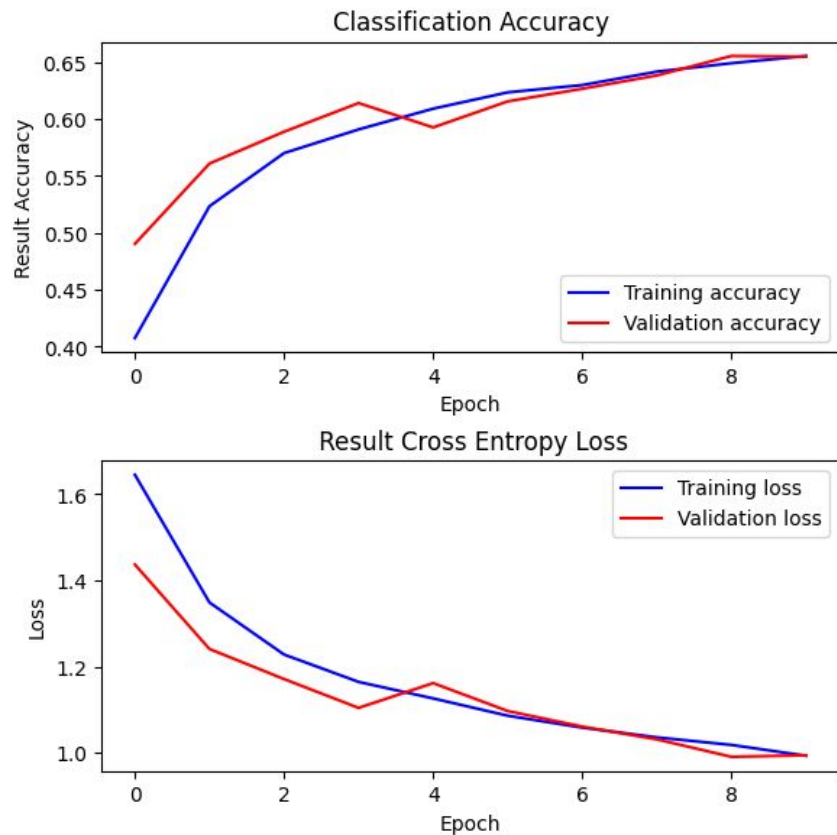
Model 7: Evaluation & Performance

Performance: Switching to SGD improves performance but requires further optimization.

Overfitting/Underfitting: Overfitting is slightly reduced, but the speed of convergence should be improved.

Improvements Needed: Try other with other activation functions like ELUs or GELUs for better compatibility.

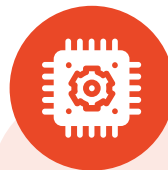
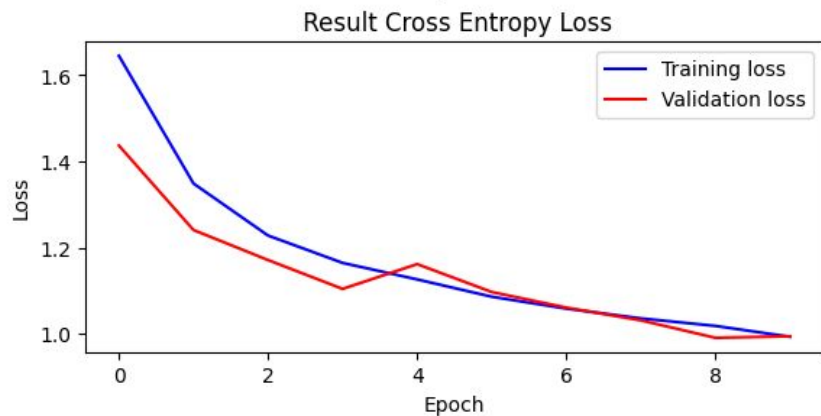
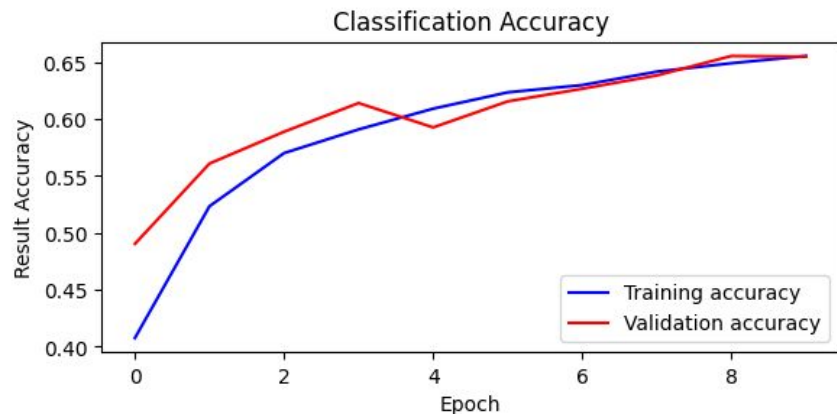
Model 8: Activation functions



Model 8: Evaluation & Performance

In model 8, the ReLu activation functions is replaced with LeakyRelu for the first convolutional layers: It is possible to test with other activation functions such as: Exponential Linear Units (ELUs), Swish, Sigmoid, Gaussian Error Linear Unit (GELU), Hyperbolic Tangent (Tanh), etc

Model 8: Activation functions



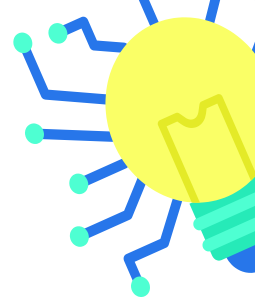
Model 8: Evaluation & Performance

Performance: LeakyReLU implementation improves performance by handling sensitive missing information.

Overfitting/Underfitting: Balanced performance with reduced risk of overfitting.

Improvements Needed: Try other enabling functions such as ELU or GELU to improve performance.

Model Performance Metrics



01

Image Classification vs. Object Detection

02

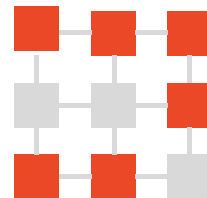
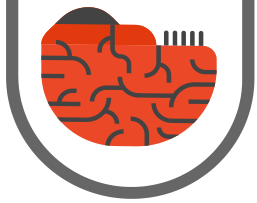


Image Classification Metrics

- Accuracy
- Precision
- Recall (Sensitivity)
- F1 Score
- Area Under the ROC Curve (AUC)

Object Detection Metrics

- Mean Average Precision (mAP)
- Intersection over Union (IoU)



Model Performance Metrics

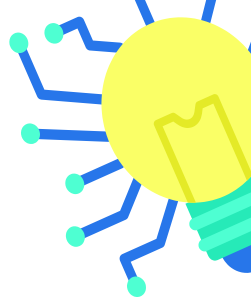


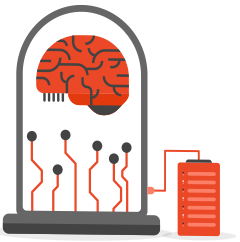
Image Classification Metrics:

- **Accuracy:**

Accuracy is The ratio of a well-segmented image to the total number of images. It measures how often the model predicts correctly.

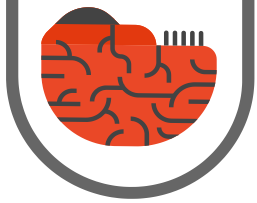
Pros:

- Provides easy consideration of model performance.



Cons:

- Can be misleading for unbalanced data sets in which some categories are overrepresented, causing the model to perform better on the majority category while ignoring the minority categories



Model Performance Metrics

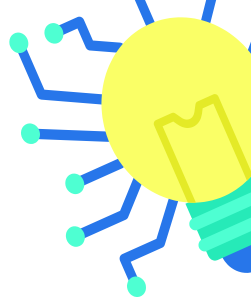


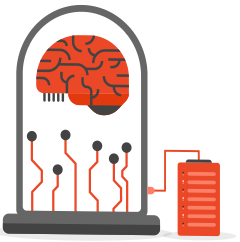
Image Classification Metrics:

- **Precision:**

Specificity is the proportion of images that are classified as part of a particular class that actually belongs to that class. as calculated : ***Precision = True Positives / True Positives + False Positives***

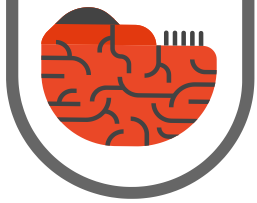
Pros:

→ Shows how well the model avoids false positives.



Cons:

→ High precision is possible at the expense of some good accuracy (low recall).



Model Performance Metrics

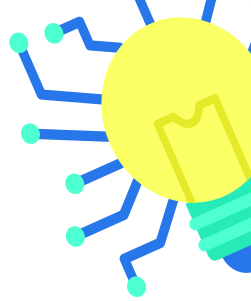


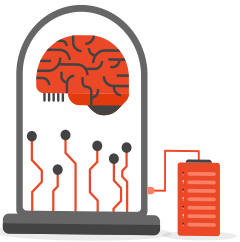
Image Classification Metrics:

- **Precision:**

Specificity is the proportion of images that are classified as part of a particular class that actually belongs to that class. as calculated : ***Precision = True Positives / True Positives + False Positives***

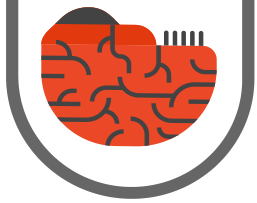
Pros:

→ Shows how well the model avoids false positives.



Cons:

→ High precision is possible at the expense of some good accuracy (low recall).



Model Performance Metrics

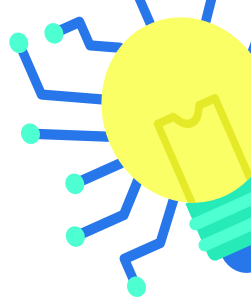


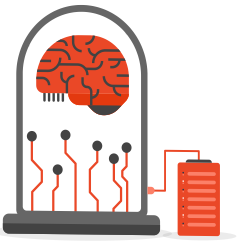
Image Classification Metrics:

- **Recall (Sensitivity):**

Recall measures the proportion of positive positives correctly identified by the model. as calculated :
Recall = True Positives / True Positives + False Negatives.

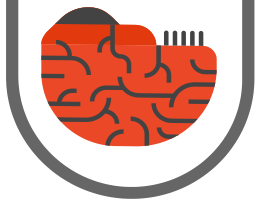
Pros:

- Demonstrates the ability of the model to capture all relevant information.



Cons:

- too much memory can lead to too many false positives, reducing accuracy.



Model Performance Metrics

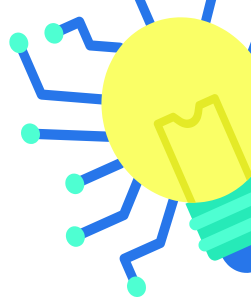


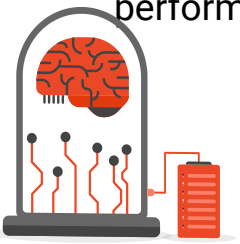
Image Classification Metrics:

- **F1 Score:**

F1 Score is the harmonic mean of precision and recall, combining the two metrics into a single measure. as calculated : $F1 = 2 \times (Precision \times Recall / Precision + Recall)$.

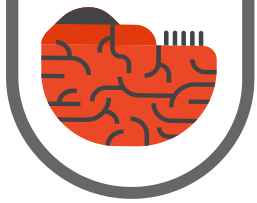
Pros:

- Strikes the right balance between accuracy and recall, providing comprehensive performance measurement.



Cons:

- It doesn't provide insight into specific types of errors (false positives vs. false negatives).



Model Performance Metrics

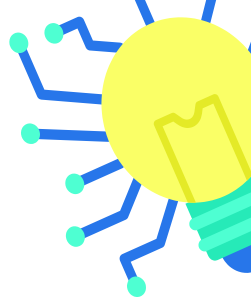


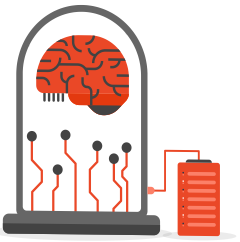
Image Classification Metrics:

- **Area Under the ROC Curve (AUC):**

The ROC curve plots the true positive rate (TPR) compared to the false positive rate (FPR) at different classification thresholds. The AUC summarizes the performance of the model in all constraints.

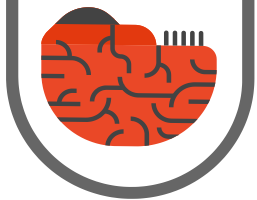
Pros:

- Provides a metric of model performance on threshold settings.

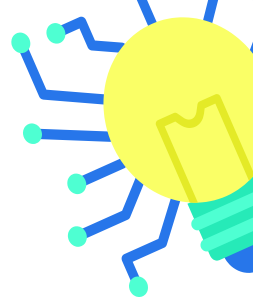


Cons:

- Can be difficult to interpret without context.



Model Performance Metrics



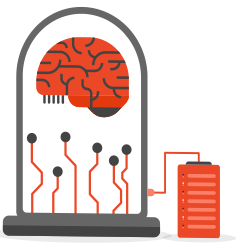
Object Detection Metrics:

- **Mean Average Precision (mAP):**

The mAP is the average of the Average Precisions (AP) for all subjects. The AP in each subject is calculated by using precision recall values for various inter-uniform association (IoU) thresholds.

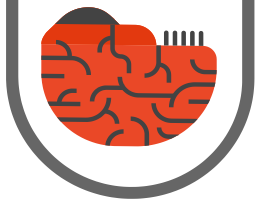
Pros:

- Combines analysis quality (precision) with local accuracy.

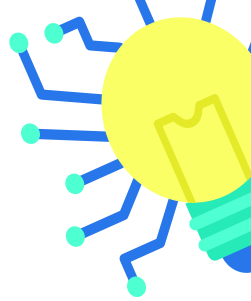


Cons:

- Requires calculation of AP scores and averages for each course, which can be a complicated calculation.



Model Performance Metrics



Object Detection Metrics:

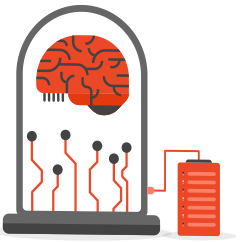
- **Intersection over Union (IoU):**

IoU is measured between the predicted bounding box and the ground truth bounding box. as calculated:

Precision = Area of Overlap / Area of Union

Pros:

- Provides accurate positioning of the model.



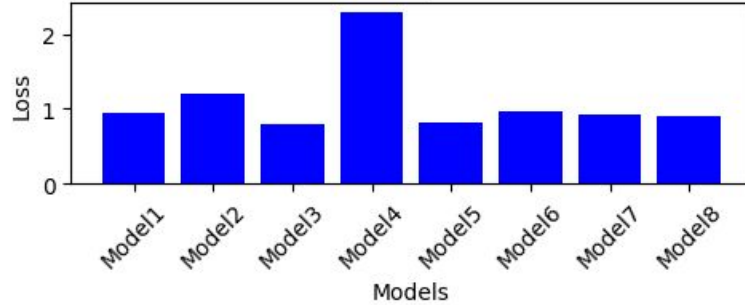
Cons:

- The IoU threshold chosen to consider an assay correct can significantly affect the results

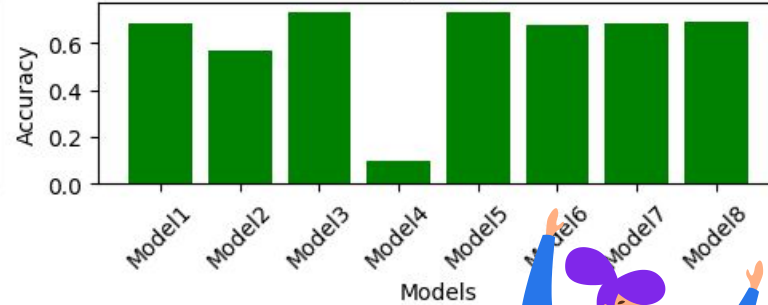
Results & Analysis

Results on Testing Data

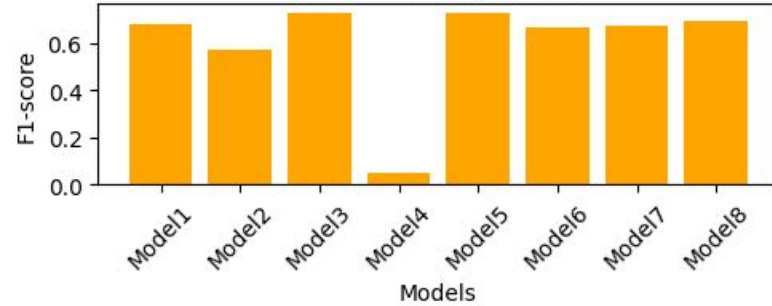
Loss Comparison



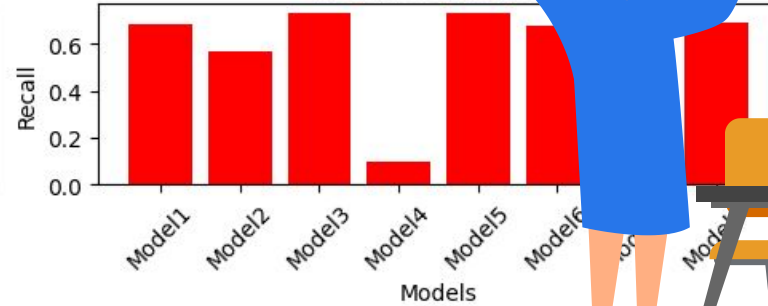
Accuracy Comparison



F1-score Comparison



Recall Comparison



Results & Analysis



Overview of Model Performance

01

Overview of Model Performance

- **Evaluation metrics:** loss, accuracy, F1 score, recall
- **Models evaluated:** different 8 models

General Observations

02

General Observations

- Consistent Performance
- Improvement Over Baseline
- Degradation in Performance
- Overall Low Scores

Insights from Metrics

03

Insights from Metrics

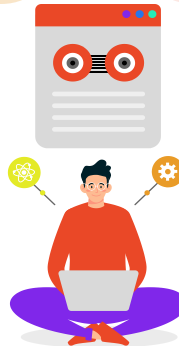
- Loss
- Accuracy
- F1 Score
- Recall

Final Deductions

04

Final Deductions

- Increasing Epochs and Convolutional Layers
- Hyperparameter Tuning



Further Optimization Techniques

1

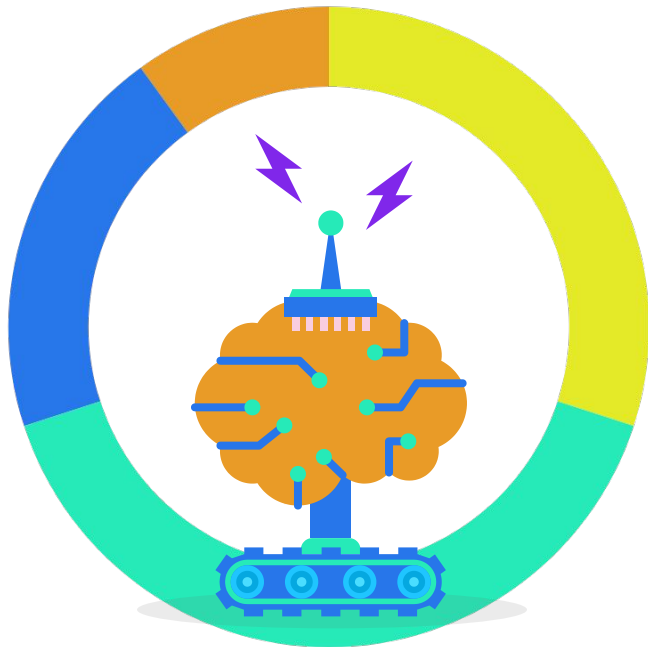
Routine Methods

The L2 constant scheme deducts a penalty for the square of the weighting magnitude in the loss function, and prevents the model from becoming too complex and overfitting the training data

2

Transfer Learning

Using pre-trained images and optimizes them on the current data set, using known features for better matching.



3

Learning Rate Scheduling

Adjusts study rates during training to find optimal estimates, avoid minimal excesses and ensure smooth learning.

4

Adding Dense Layers

Adding additional fully connected layers increases the model's ability to recognize complex patterns and integrate features from the convolutional layer.

Conclusion

In this project, we researched and optimized Convolutional Neural Network (CNN) models to improve image classification performance using CIFAR-10 data set. We aimed to increase the accuracy and generalization of the model by leveraging techniques such as data enhancement, batch normalization, single-hot encoding, advanced optimization techniques, etc. In our method, use pretrained models , fine-tune hyperparameters, regularize and prevent overfitting. Through repeated testing and adaptation along with applying various techniques, we sought to build robust and efficient CNN algorithms that can accurately classify images while ensuring real-world adaptation challenges



THE TEAM



Karim Triki
Student ID: 5528602



Ines Haouala
Student ID: 5483776

THANK YOU

