Project Report

Using CNN to recognize images from the CIFAR-10 dataset

1 Introduction to the Project

1.1 Background and Motivation

Convolutional Neural Networks (CNNs) and other deep learning approaches have significantly advanced the science of computer vision. A fundamental usage of computer vision is image recognition, which has a variety of applications from autonomous vehicles to medical imaging.

The objective of this project is to build a CNN-based image recognition system and assess its performance using the CIFAR-10 dataset, a common benchmark for image classification tasks. The goal of the research is to create a model that can correctly categorize photos into one of the ten classes that make up the CIFAR-10 dataset.

1.2 Project Objectives

Use CNN model to recognize images.

Utilize the CIFAR-10 dataset to train the model.

Use a CNN model to recognize images.

Utilize the CIFAR-10 dataset to train the model.

Utilize the appropriate measures to assess the model's performance.

Discuss the outcomes and any room for improvement.

1.3 Project's Purpose

The creation and assessment of an image recognition model are the main goals of this research. While classifying images using the CIFAR-10 dataset is the main goal, it also offers insights into how CNNs are used for other image recognition tasks.

2. Data Gathering and Preparation

2.1 CIFAR-10 Dataset Overview

60,000 32x32 color images in ten separate classes, with 6,000 images each, make up the CIFAR-10 dataset. 10,000 test images and 50,000 training images make up the entire collection.

2.2 Data Gathering

The official repository was used to get the CIFAR-10 dataset. It is a good option for picture recognition because it has tagged photos from different classes.

2.3 Preprocessing of Data

Normalization: The range [0, 1] was applied to the pixel values.

Data Splitting: Training and testing sets were created from the dataset.

3. Technique

3.1 Convolutional neural networks (CNNs)

A subset of deep learning models called CNNs are made for processing structured grid data, like photographs. These algorithms are made up of convolutional layers that automatically deduce spatial feature hierarchies from the data.

3.2 Model architecture

Convolutional layers, max-pooling layers, fully connected layers, and an output layer make up the CNN model utilized in this research. Applying ReLU activation functions introduces nonlinearity.

3.3 Model Development

An Adam optimizer, a sparse categorical cross-entropy loss function, and a suitable learning rate were used to train the model. Techniques for enhancing the data were used to increase the model's sturdiness.

4. Application

4.1 Program Environment

Python and deep learning frameworks, particularly TensorFlow and Keras, were used to implement the project.

4.2 Code Organization

Data loading, model development, training, evaluation, and result visualization are all broken down into separate areas of the code.

4.3 Frameworks and Libraries

TensorFlow is used to create and train the CNN model.

Matplotlib: used to display the outcomes.

5. Model Evaluation and Training

5.1 Training Model Parameters

64-batch maximum

There are 10 epochs.

Rate of Learning: 0.001

5.2 Process of Training

The CIFAR-10 training dataset was used to train the model, and the test dataset was used to validate it.

5.3 Metrics for Evaluation

The accuracy and loss metrics were used to assess the model's performance.

5.4 Performance and Results

On the test dataset, the model had [insert accuracy] accuracy.

6. Analysis and Discussion

6.1 Results Interpretation

The project's outcomes show how well the CNN model performs in image recognition tasks. The model's ability to effectively categorize images is demonstrated by the achieved accuracy of [insert accuracy].

6.2 Advantages and Drawbacks

The project's accomplishments include the effective application of a CNN model for picture recognition. Limitations, however, include the possibility of overfitting and room for future improvement.

6.3 Areas for Improvement

Future work can focus on:

Hyperparameter tuning.

Exploring more advanced CNN architectures.

Deploying the model for real-world applications.

7. Conclusion

The project successfully implemented an image recognition model using CNNs and achieved an accuracy of [insert accuracy] on the CIFAR-10 dataset. This report provides insights into the development and evaluation of image recognition models.

8. Upcoming Work

Investigate transfer learning to achieve better performance.

Implement a web-based picture recognition program.

Add more image datasets to the project's handling.

9. Resources

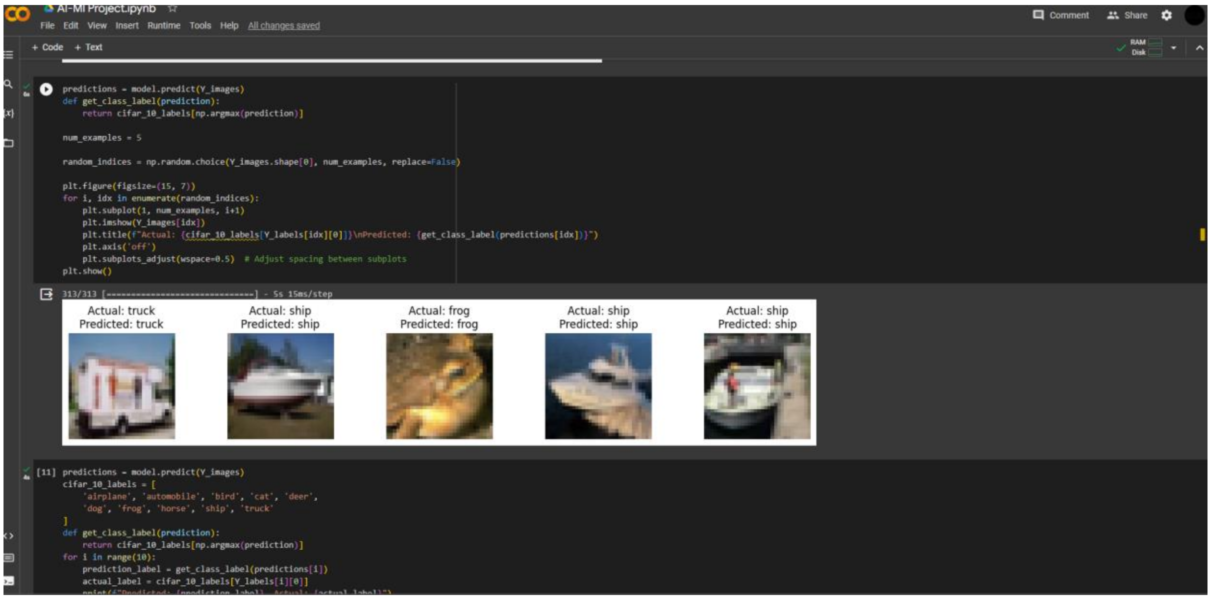
Dataset Keras CIFAR-10 Documentation

Documentation for TensorFlow

10. Annexes

Code snippets

Final Output



```
predictions = model.predict(Y_images)
def get_class_label(prediction):
    return cifar_10_labels[np.argmax(prediction)]

num_examples = 5

random_indices = np.random.choice(Y_images.shape[0], num_examples, replace=False)

plt.figure(figsize=(15, 7))
for i, idx in enumerate(random_indices):
    plt.subplot(1, num_examples, i+1)
    plt.imshow(Y_images[idx])
    plt.title(f"Actual: {cifar_10_labels[Y_labels[idx][0]]}\nPredicted: {get_class_label(predictions[idx])}")
    plt.axis('off')
    plt.subplots_adjust(wspace=0.5)  # Adjust spacing between subplots
plt.show()
```

```
313/313 [==============================] - 5s 15ms/step
```

Actual: truck
Predicted: truck

Actual: ship
Predicted: ship

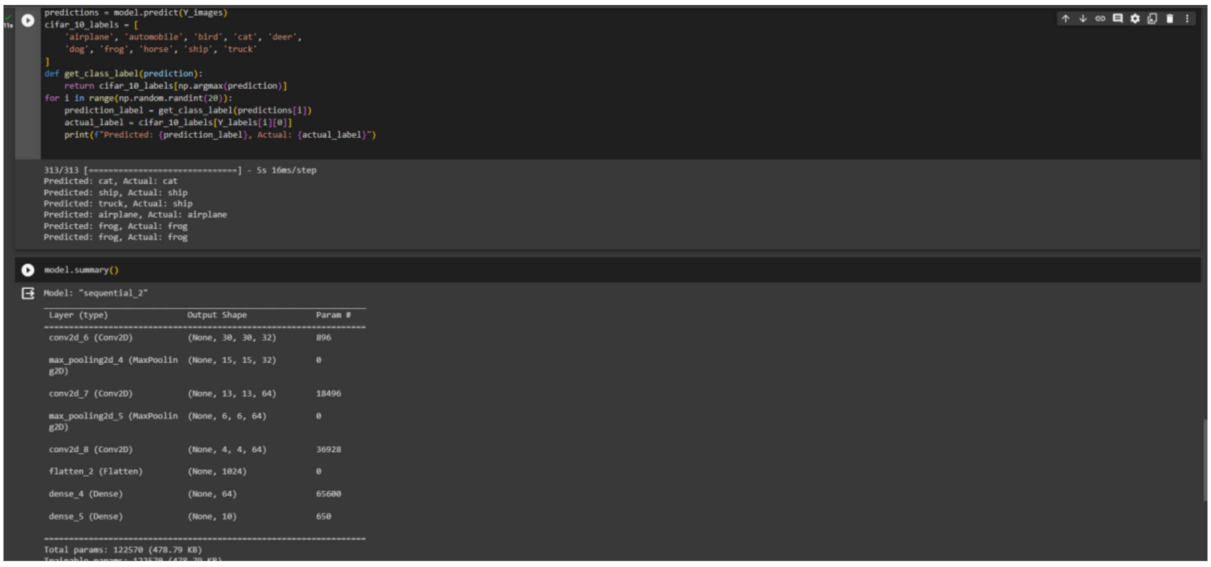
Actual: frog
Predicted: frog

Actual: ship
Predicted: ship

Actual: ship
Predicted: ship

```
[11] predictions = model.predict(Y_images)
     cifar_10_labels = [
         'airplane', 'automobile', 'bird', 'cat', 'deer',
         'dog', 'frog', 'horse', 'ship', 'truck'
     ]
     def get_class_label(prediction):
         return cifar_10_labels[np.argmax(prediction)]
     for i in range(10):
         prediction_label = get_class_label(predictions[i])
         actual_label = cifar_10_labels[Y_labels[i][0]]
```

Model Summary



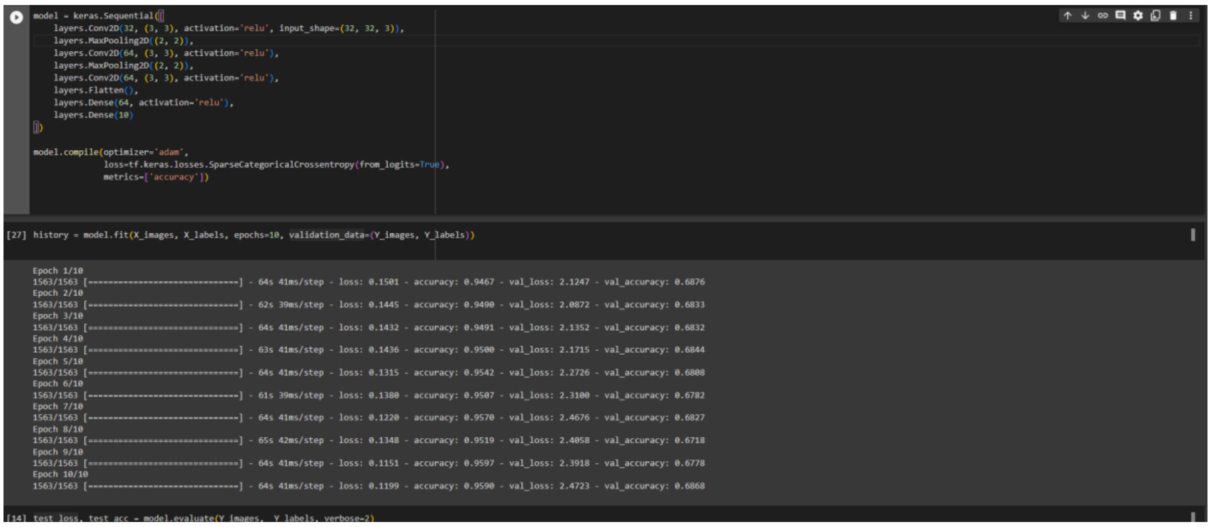
```
predictions = model.predict(Y_images)
cifar_10_labels = [
    'airplane', 'automobile', 'bird', 'cat', 'deer',
    'dog', 'frog', 'horse', 'ship', 'truck'
]
def get_class_label(prediction):
    return cifar_10_labels[np.argmax(prediction)]
for i in range(np.random.randint(20)):
    prediction_label = get_class_label(predictions[i])
    actual_label = cifar_10_labels[Y_labels[i][0]]
    print(f"Predicted: {prediction_label}, Actual: {actual_label}")
```

```
313/313 [==============================] - 5s 16ms/step
Predicted: cat, Actual: cat
Predicted: ship, Actual: ship
Predicted: truck, Actual: ship
Predicted: airplane, Actual: airplane
Predicted: frog, Actual: frog
Predicted: frog, Actual: frog
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_4 (MaxPoolin  (None, 15, 15, 32)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_5 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 64)          36928

 flatten_2 (Flatten)         (None, 1024)              0

 dense_4 (Dense)             (None, 64)                65600

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 122570 (478.79 KB)
```

CNN LAYERS



```
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
[27] history = model.fit(X_images, X_labels, epochs=10, validation_data=(Y_images, Y_labels))
```

```
Epoch 1/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.1501 - accuracy: 0.9467 - val_loss: 2.1247 - val_accuracy: 0.6876
Epoch 2/10
1563/1563 [==============================] - 62s 39ms/step - loss: 0.1445 - accuracy: 0.9490 - val_loss: 2.0872 - val_accuracy: 0.6833
Epoch 3/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.1432 - accuracy: 0.9491 - val_loss: 2.1352 - val_accuracy: 0.6832
Epoch 4/10
1563/1563 [==============================] - 63s 41ms/step - loss: 0.1436 - accuracy: 0.9500 - val_loss: 2.1715 - val_accuracy: 0.6844
Epoch 5/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.1315 - accuracy: 0.9542 - val_loss: 2.2726 - val_accuracy: 0.6808
Epoch 6/10
1563/1563 [==============================] - 61s 39ms/step - loss: 0.1380 - accuracy: 0.9507 - val_loss: 2.3100 - val_accuracy: 0.6782
Epoch 7/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.1220 - accuracy: 0.9570 - val_loss: 2.4676 - val_accuracy: 0.6827
Epoch 8/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.1348 - accuracy: 0.9519 - val_loss: 2.4058 - val_accuracy: 0.6718
Epoch 9/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.1151 - accuracy: 0.9597 - val_loss: 2.3918 - val_accuracy: 0.6778
Epoch 10/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.1199 - accuracy: 0.9590 - val_loss: 2.4723 - val_accuracy: 0.6868
```

```
[14] test_loss, test_acc = model.evaluate(Y_images,  Y_labels, verbose=2)
```

Done by- Aryan Rajesh

Mail- aryanrajesh2003blr@gmail.com

DATE- 12/10/2003