Model_Scratch 02 - No data augmentation

Name: Alberto Pingo
Email: 2202145 @my.ipleiria.pt

Validation dataset: train5

Directories

This section sets up the directory paths used for training, validation, and test datasets based on the repository structure.

In [ ]:
```
import os

current_dir = os.getcwd()

# TWO FOLDERS UP
data_dir = os.path.abspath(os.path.join(current_dir, os.pardir, os.pardir
test_dir = os.path.join(data_dir, 'test')
train_dir = os.path.join(data_dir, 'train')

train_dirs = []
for i in range(1, 5):
    train_dirs.append(os.path.join(train_dir, 'train' + str(i)))


validation_dir = os.path.join(data_dir, 'train', 'train5')

print(current_dir)
print(data_dir)
print(test_dir)
print(train_dir)
print(validation_dir)
```

```
/home/pws/code/IA-image-classification/notebooks/models-S
/home/pws/code/IA-image-classification/data
/home/pws/code/IA-image-classification/data/test
/home/pws/code/IA-image-classification/data/train
/home/pws/code/IA-image-classification/data/train/train5
```

Preprocessing

Load the datasets and perform initial preprocessing. Images are resized to 32x32 pixels and batched.

In [ ]:
```
from keras.utils import image_dataset_from_directory
import tensorflow as tf

# Load training datasets from train1 to train4
train_datasets = []
```

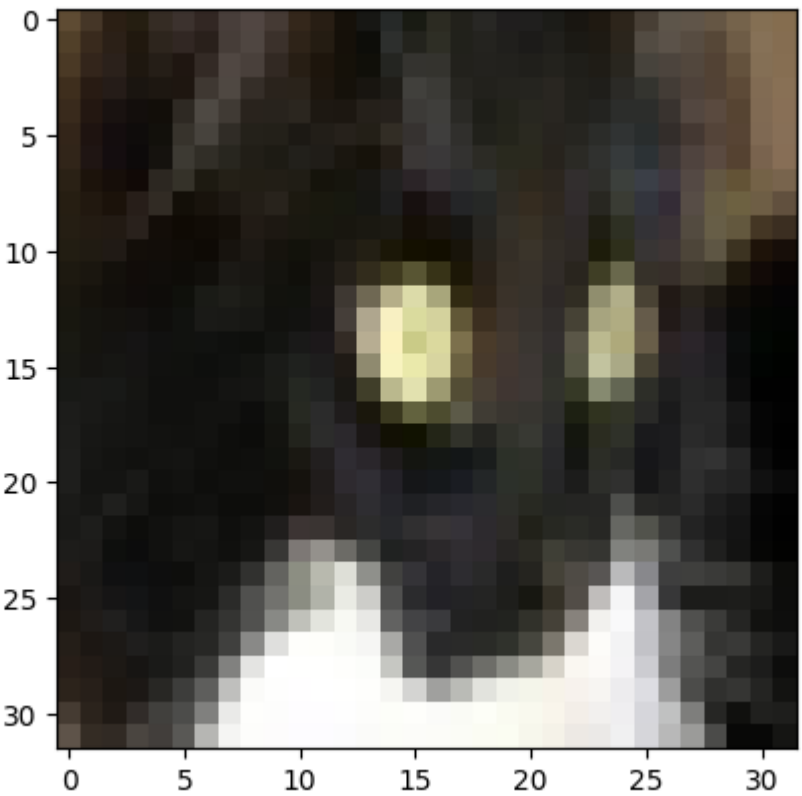
```
IMG_SIZE = 32
BATCH_SIZE = 64
for i in range(1, 5):
    dataset = image_dataset_from_directory(train_dirs[i-1], image_size=(I
    train_datasets.append(dataset)

train_dataset = train_datasets[0]
for dataset in train_datasets[1:]:
    train_dataset = train_dataset.concatenate(dataset)

validation_dataset = image_dataset_from_directory(validation_dir, image_s

test_dataset = image_dataset_from_directory(test_dir, image_size=(IMG_SIZ

class_names = validation_dataset.class_names
class_names = [class_name.split('_')[-1] for class_name in class_names]

print(class_names)
```

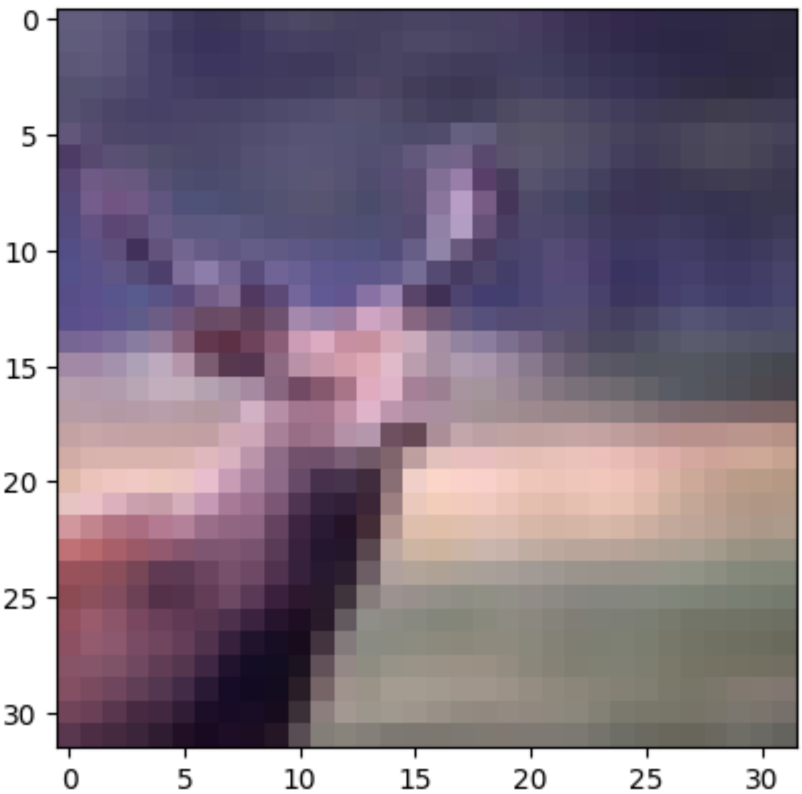
```
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
'ship', 'truck']
```

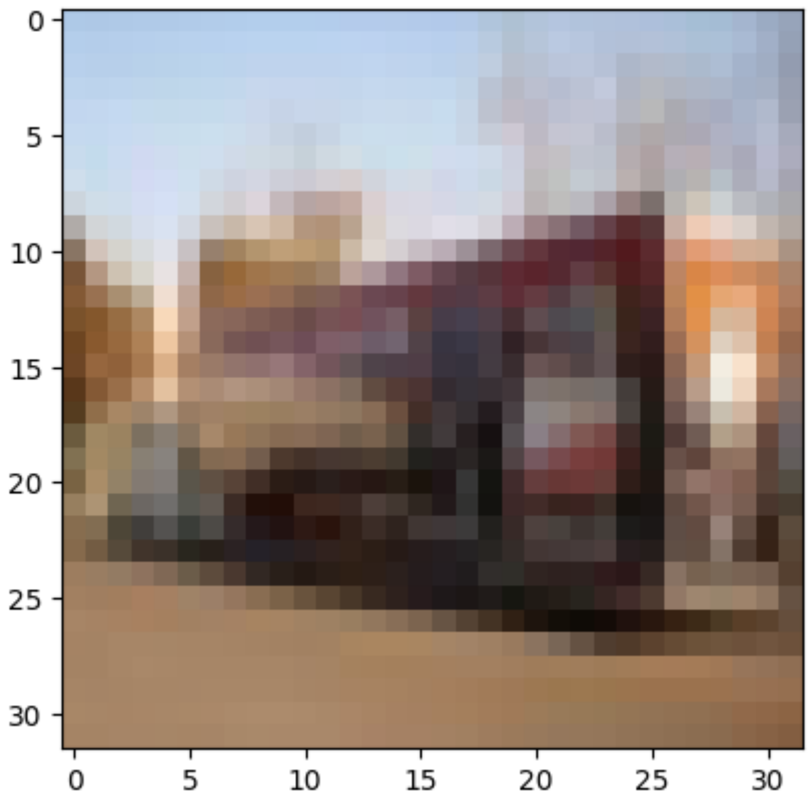
Configure the dataset for performance

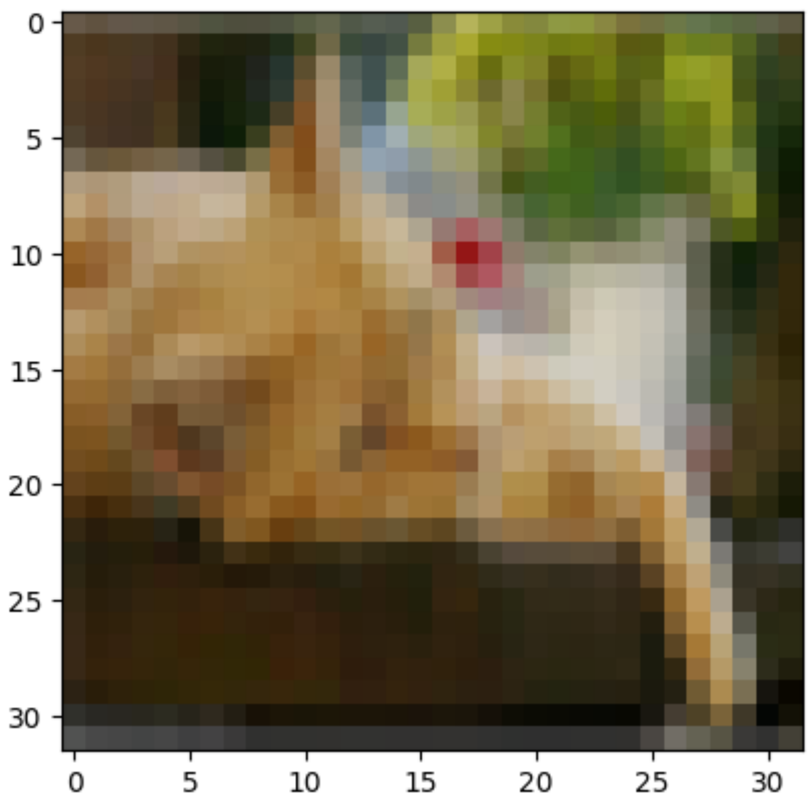
In [ ]:
```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.cache().shuffle(1000).prefetch(buffer_size=
validation_dataset = validation_dataset.cache().prefetch(buffer_size=AUTO
test_dataset = test_dataset.cache().prefetch(buffer_size=AUTOTUNE)
```

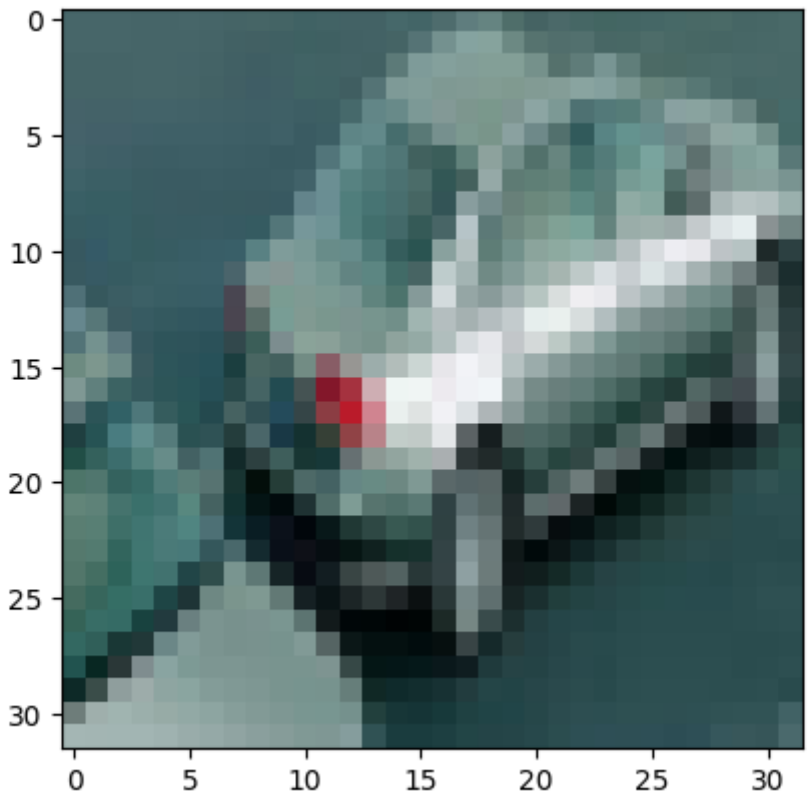
In [ ]:
```
import matplotlib.pyplot as plt

for data, _ in train_dataset.take(1):
    for i in range(9):
        plt.imshow(data[i].numpy().astype('uint8'))
        plt.show()
    break
```

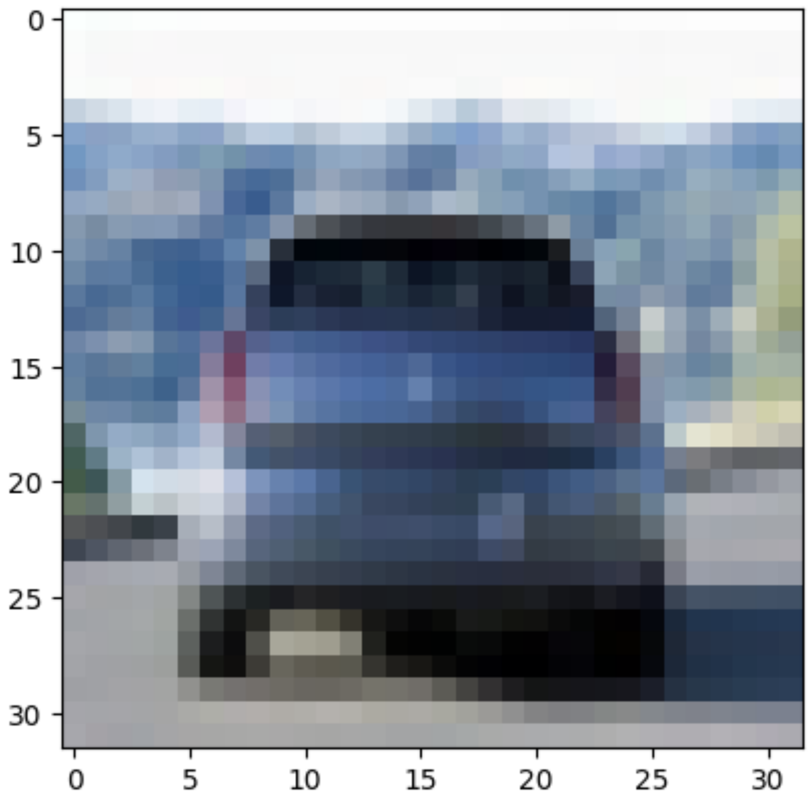


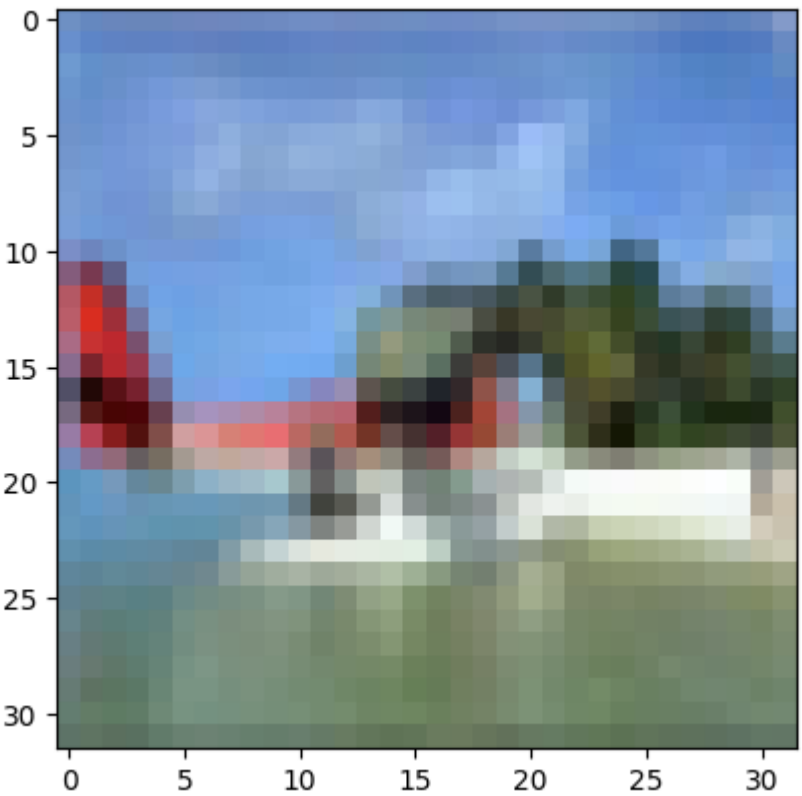


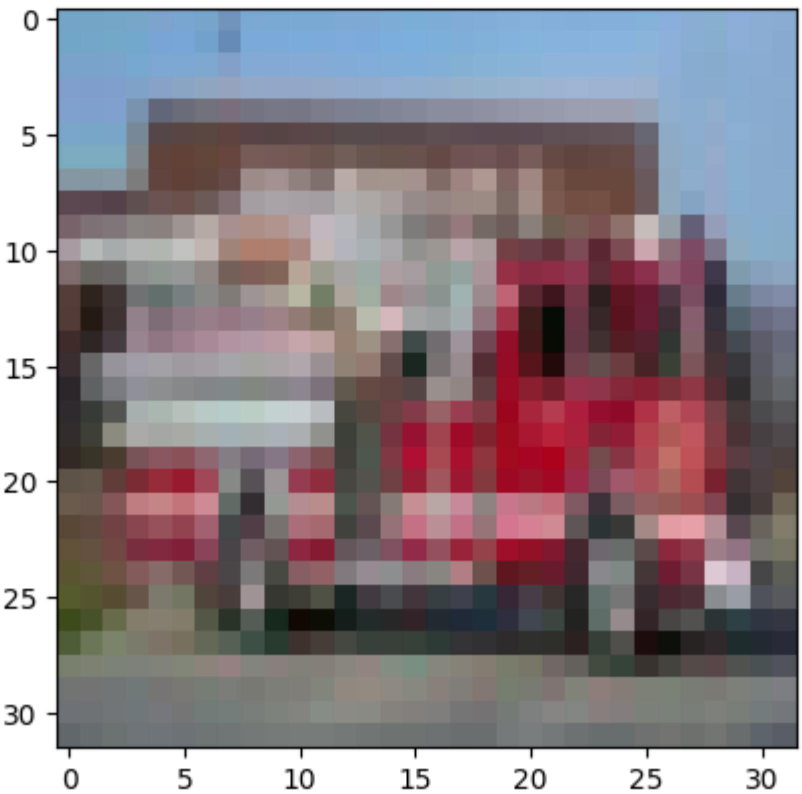

0
5
10
15
20
25
30
0 5 10 15 20 25 30

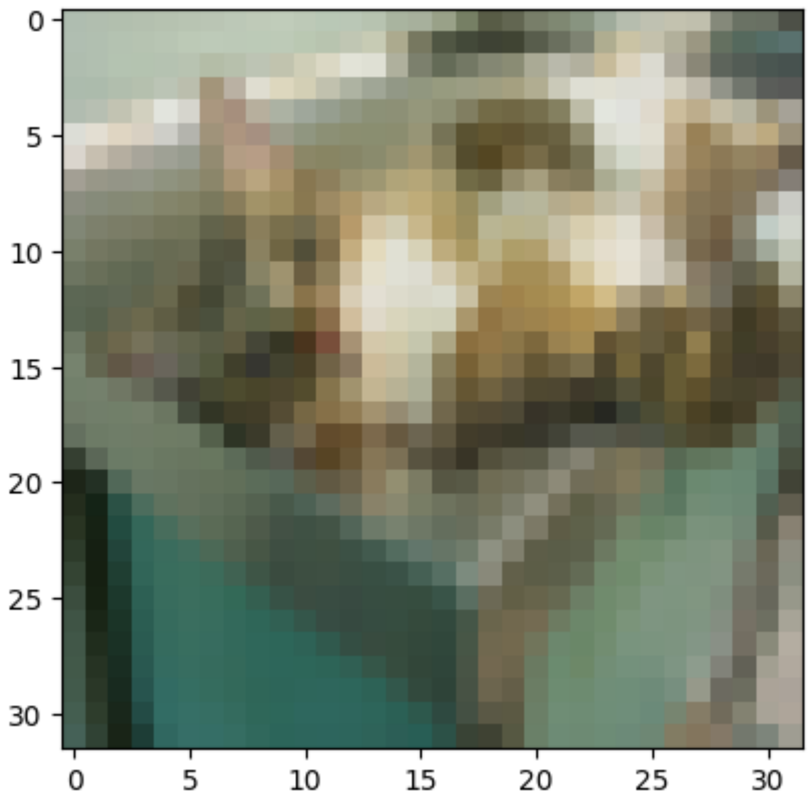

0
5
10
15
20
25
30
0 5 10 15 20 25 30


0
5
10
15
20
25
30
0 5 10 15 20 25 30


0
5
10
15
20
25
30
0 5 10 15 20 25 30


0
5
10
15
20
25
30
0 5 10 15 20 25 30


0
5
10
15
20
25
30
0 5 10 15 20 25 30



MODEL ARCHITECTURE

Build a Convolutional Neural Network (CNN) model.

Architecture:

Input -> Conv2D - BN -> MaxPooling2D -> Conv2D - BN-> MaxPooling2D -> Conv2D - BN-> MaxPooling2D -> Flatten -> Dense - BN -> Dropout -> Dense - BN -> Dropout -> Output

1. **Input Layer**
    - The input layer expects images of size 32x32 pixels with 3 color channels (RGB).
    - No data augmentation is applied to the input images.
    - The Rescaling layer, rescales the pixel values from the range [0, 255] to [0, 1].
2. **Convolutional Layers**
    - The model consists of 3 convolutional layers with 32, 64 and 128 filters respectively.
    - Use padding='same' to preserve the spatial dimensions of the feature maps.
3. **Max Pooling Layers**
    - Max pooling layers are used after each convolutional layer to reduce the spatial dimensions of the feature maps.
    - A pooling size of 2x2 is used.
4. **Fully connected layers**

- A dense layer with 512 units and ReLU activation function.
- A dense layer with 256 units and ReLU activation function.

5. **Output Layer**

- The output layer consists of 10 units (one for each class) with a softmax activation function.
- The softmax function outputs the probability distribution over the classes.

Overfitting measures

- Dropout layers are used after each Convolutional and Dense layer to prevent overfitting.

Batch Normalization

- Batch normalization is used after each Convolutional layer to normalize the activations of the previous layer at each batch.
- This helps to stabilize and speed up the training process.

In [ ]:
```
from tensorflow import keras
from keras import layers

inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

x = layers.Rescaling(1./255)(inputs)

## First Convolutional Block
x = layers.Conv2D(filters=32, kernel_size=3, padding='same', activation="
x = layers.BatchNormalization()(x) # Standardize the inputs to the next l
x = layers.MaxPooling2D(pool_size=2)(x)

# Second Convolutional Block

x = layers.Conv2D(filters=64, kernel_size=3, padding='same', activation="
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)

# Third Convolutional Block
x = layers.Conv2D(filters=128, kernel_size=3, padding='same', activation=
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)


x = layers.Flatten()(x)

x = layers.Dense(512, activation="relu")(x) # Fully connected layer
x = layers.BatchNormalization()(x)
x = layers.Dense(256, activation="relu")(x)
x = layers.BatchNormalization()(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(10, activation="softmax")(x)  # Softmax for multi-

model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 rescaling_1 (Rescaling)     (None, 32, 32, 3)         0

 conv2d_3 (Conv2D)           (None, 32, 32, 32)        896

 batch_normalization_5 (Bat  (None, 32, 32, 32)        128
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_6 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         73856

 batch_normalization_7 (Bat  (None, 8, 8, 128)         512
 chNormalization)

 max_pooling2d_5 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_3 (Dense)             (None, 512)               1049088

 batch_normalization_8 (Bat  (None, 512)               2048
 chNormalization)

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 rescaling_1 (Rescaling)     (None, 32, 32, 3)         0

 conv2d_3 (Conv2D)           (None, 32, 32, 32)        896

 batch_normalization_5 (Bat  (None, 32, 32, 32)        128
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_6 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)
```

```
 conv2d_5 (Conv2D)           (None, 8, 8, 128)         73856

 batch_normalization_7 (Bat  (None, 8, 8, 128)         512
 chNormalization)

 max_pooling2d_5 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_3 (Dense)             (None, 512)               1049088

 batch_normalization_8 (Bat  (None, 512)               2048
 chNormalization)

 dense_4 (Dense)             (None, 256)               131328

 batch_normalization_9 (Bat  (None, 256)               1024
 chNormalization)

 dropout_1 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 10)                2570

=================================================================
Total params: 1280202 (4.88 MB)
Trainable params: 1278218 (4.88 MB)
Non-trainable params: 1984 (7.75 KB)
_________________________________________________________________
```

Compile Model

Loss function:
Use the *Sparse Categorical Crossentropy* loss function because it is a `multi-class classification` problem.

Optimizer: RMSprop
Exploring the *RMSprop* optimizer.

In [ ]:
```
from keras import optimizers

model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=optimizers.RMSprop(learning_rate=0.001),
    metrics=['acc'])
```

Train Model

Train the model with Early stopping, Model checkpoint, and Learning rate reduction callbacks.

In [ ]:
```
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPla

learning_rate_reduction = ReduceLROnPlateau(
```

```
    monitor='val_acc',
    patience=3,
    verbose=1,
    factor=0.5,
    min_lr=1e-5)

early_stop = EarlyStopping(monitor='val_acc',
                           patience=3,
                           restore_best_weights=True)
model_checkpoint = ModelCheckpoint('models/S02/checkpoints/S02-cp.h5', sa

history = model.fit(
    train_dataset,
    epochs=100,
    validation_data=validation_dataset,
    callbacks=[early_stop, model_checkpoint, learning_rate_reduction])
```

```
Epoch 1/100
628/628 [==============================] - 8s 9ms/step - loss: 1.4976 - ac
c: 0.5026 - val_loss: 1.3383 - val_acc: 0.5427 - lr: 0.0010
Epoch 2/100
  6/628 [..............................] - ETA: 6s - loss: 1.1421 - acc: 0
.6250
/home/pws/miniconda3/envs/tensorflow/lib/python3.11/site-packages/keras/sr
c/engine/training.py:3103: UserWarning: You are saving your model as an HD
F5 file via `model.save()`. This file format is considered legacy. We reco
mmend using instead the native Keras format, e.g. `model.save('my_model.ke
ras')`.
  saving_api.save_model(
628/628 [==============================] - 5s 8ms/step - loss: 0.9245 - ac
c: 0.6764 - val_loss: 0.9298 - val_acc: 0.6768 - lr: 0.0010
Epoch 3/100
628/628 [==============================] - 5s 8ms/step - loss: 0.7158 - ac
c: 0.7510 - val_loss: 0.9715 - val_acc: 0.6770 - lr: 0.0010
Epoch 4/100
628/628 [==============================] - 5s 8ms/step - loss: 0.5592 - ac
c: 0.8069 - val_loss: 0.8793 - val_acc: 0.7212 - lr: 0.0010
Epoch 5/100
628/628 [==============================] - 5s 7ms/step - loss: 0.4246 - ac
c: 0.8552 - val_loss: 1.1924 - val_acc: 0.6894 - lr: 0.0010
Epoch 6/100
628/628 [==============================] - 5s 8ms/step - loss: 0.3057 - ac
c: 0.8959 - val_loss: 0.8939 - val_acc: 0.7559 - lr: 0.0010
Epoch 7/100
628/628 [==============================] - 5s 7ms/step - loss: 0.2220 - ac
c: 0.9242 - val_loss: 1.0023 - val_acc: 0.7341 - lr: 0.0010
Epoch 8/100
628/628 [==============================] - 5s 8ms/step - loss: 0.1749 - ac
c: 0.9402 - val_loss: 1.5092 - val_acc: 0.6651 - lr: 0.0010
Epoch 9/100
622/628 [============================>.] - ETA: 0s - loss: 0.1389 - acc:
0.9530
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.000500000023748725
7.
628/628 [==============================] - 5s 8ms/step - loss: 0.1388 - ac
c: 0.9530 - val_loss: 1.2263 - val_acc: 0.7216 - lr: 0.0010
```

Save Model

In [ ]:
```
keras.models.save_model(model, 'models/S02/S02-model.h5')
```

```
/tmp/ipykernel_127701/2816346251.py:1: UserWarning: You are saving your mo
del as an HDF5 file via `model.save()`. This file format is considered leg
acy. We recommend using instead the native Keras format, e.g. `model.save
('my_model.keras')`.
  keras.models.save_model(model, 'models/S02/S02-model.h5')
```

Load Model

In [ ]:
```
keras.models.load_model('models/S02/S02-model.h5')
```

Out[ ]:
```
<keras.src.engine.functional.Functional at 0x7082600e6690>
```

EVALUATION

Evaluate the model on the validation dataset.

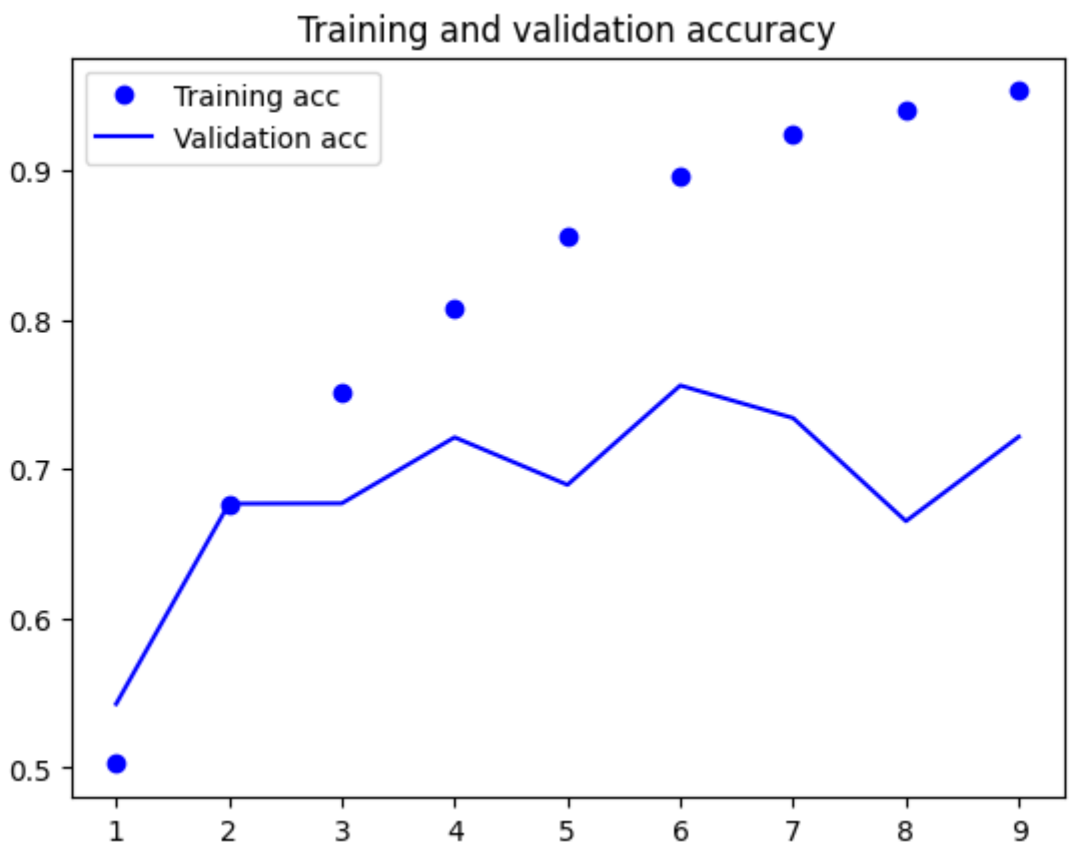
In [ ]:
```
val_loss, val_acc = model.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

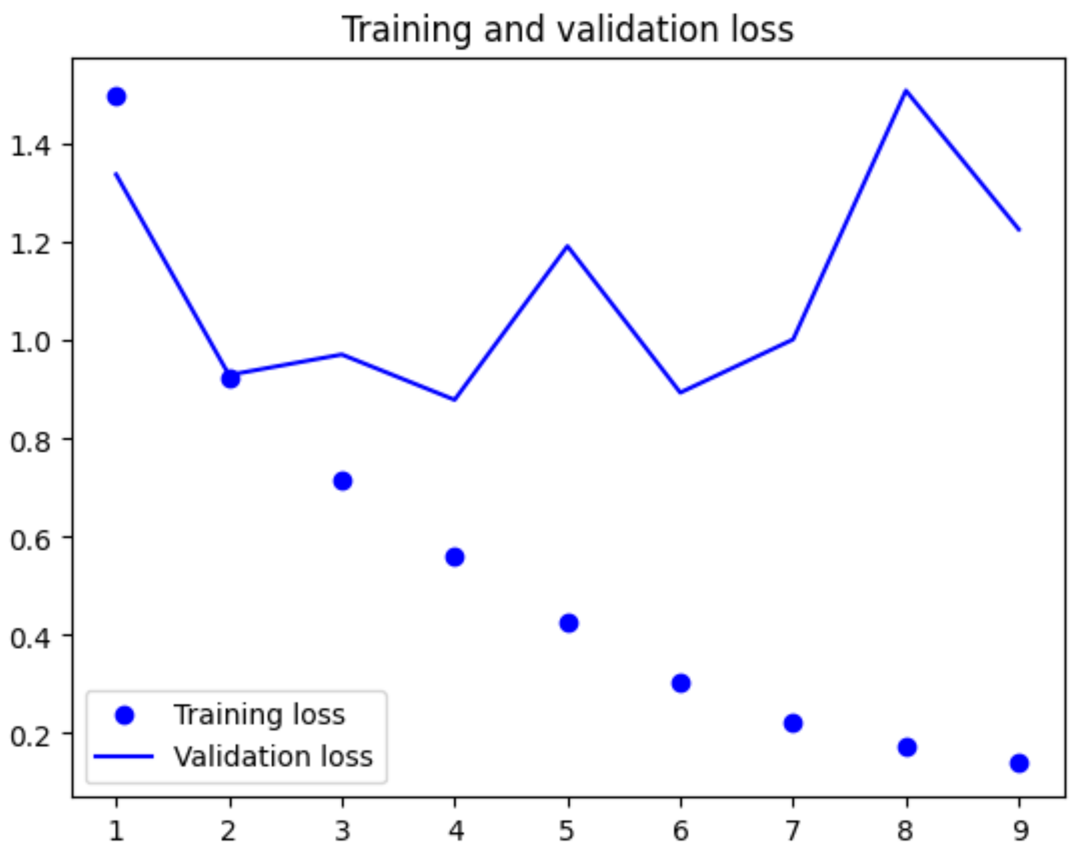
```
157/157 [==============================] - 0s 3ms/step - loss: 0.8939 - ac
c: 0.7559
val_acc: 0.7559000253677368
```

Training and Validation Curves

Plot the training and validation accuracy and loss curves.

In [ ]:
```
import matplotlib.pyplot as plt

# Extract the history from the training process
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

# Plot the training and validation accuracy
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

# Plot the training and validation loss
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





Confusion Matrix

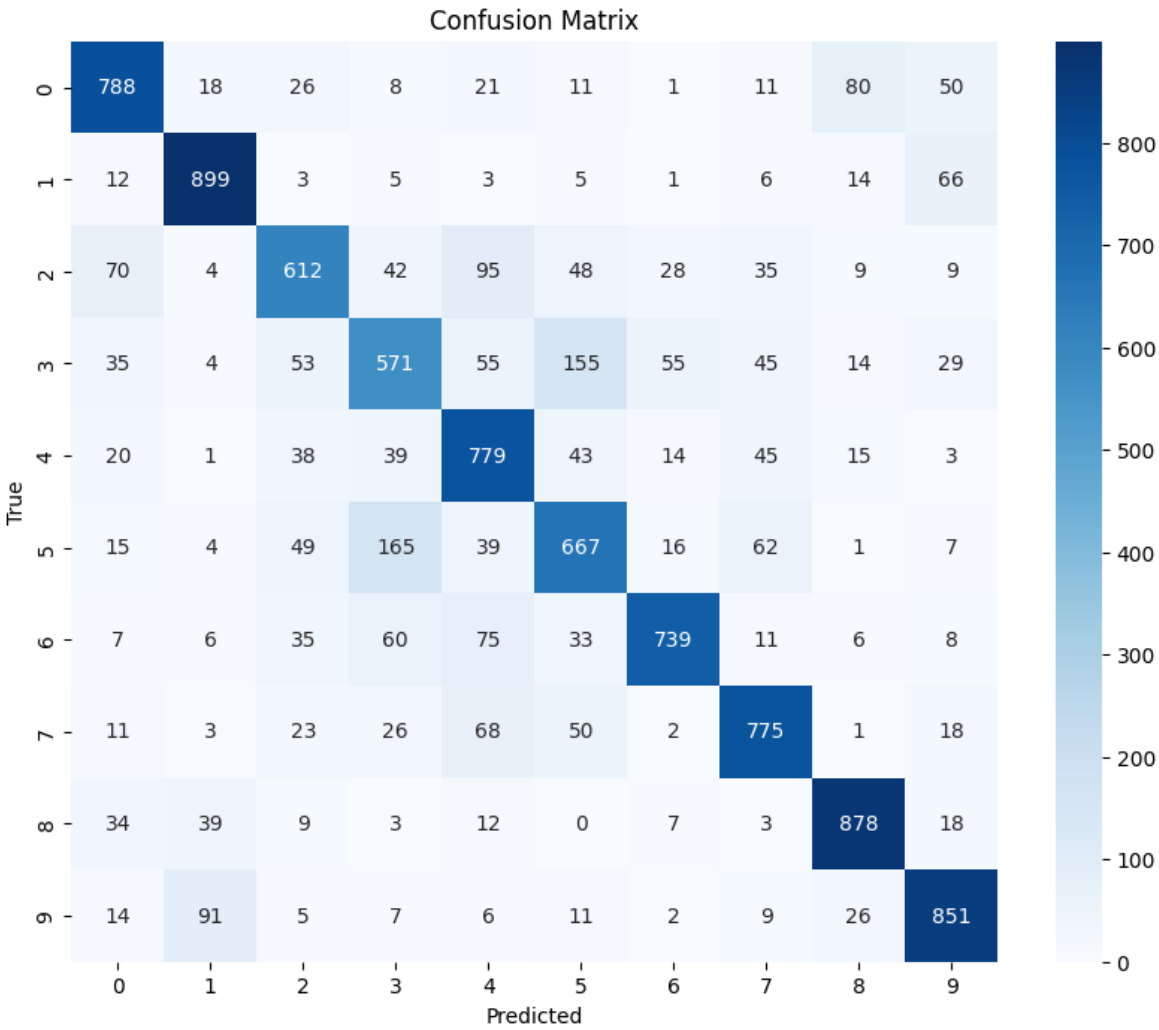
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

```
y_true = []
y_pred = []

for features, labels in validation_dataset:
    predictions = model.predict(features)
    y_true.extend(labels.numpy())
    y_pred.extend(np.argmax(predictions, axis=1))

y_true = np.array(y_true)
y_pred = np.array(y_pred)

cm = confusion_matrix(y_true, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
2/2 [==============================] - 0s 26ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 4ms/step
```

```
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
```

```
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
1/1 [==============================] - 0s 101ms/step
```



In [ ]:
```python
# Display the confusion matrix
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[788  18  26   8  21  11   1  11  80  50]
 [ 12 899   3   5   3   5   1   6  14  66]
 [ 70   4 612  42  95  48  28  35   9   9]
 [ 35   4  53 571  55 155  55  45  14  29]
 [ 20   1  38  39 779  43  14  45  15   3]
 [ 15   4  49 165  39 667  16  62   1   7]
 [  7   6  35  60  75  33 739  11   6   8]
 [ 11   3  23  26  68  50   2 775   1  18]
 [ 34  39   9   3  12   0   7   3 878  18]
 [ 14  91   5   7   6  11   2   9  26 851]]
```

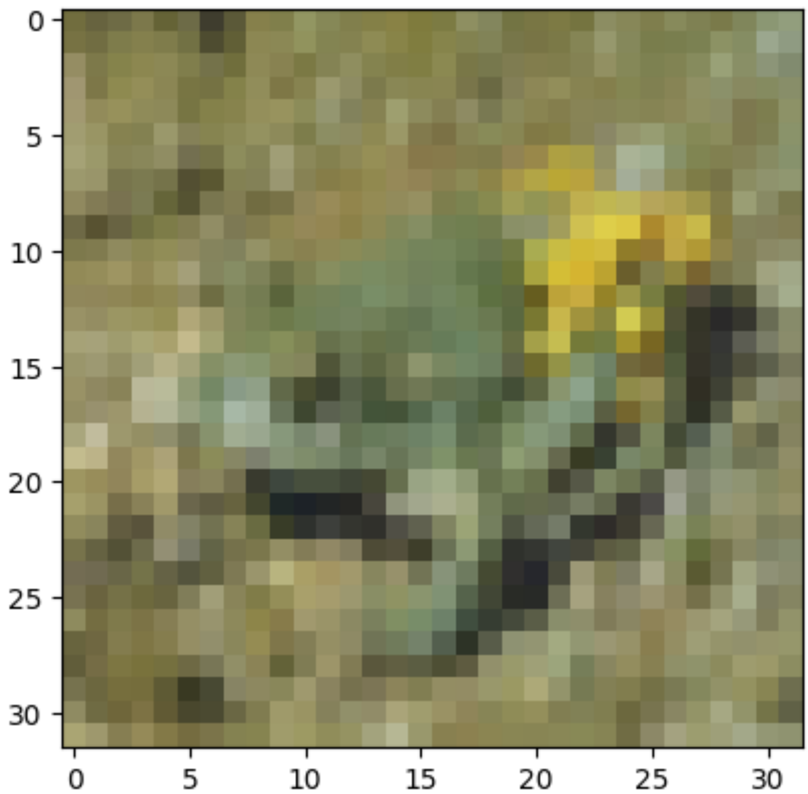
In [ ]:
```python
from sklearn.metrics import classification_report

report = classification_report(y_true, y_pred, target_names=class_names)
print(report)
```

```
              precision    recall  f1-score   support

    airplane       0.78      0.78      0.78      1014
  automobile       0.84      0.89      0.86      1014
        bird       0.72      0.64      0.68       952
         cat       0.62      0.56      0.59      1016
        deer       0.68      0.78      0.72       997
         dog       0.65      0.65      0.65      1025
        frog       0.85      0.75      0.80       980
       horse       0.77      0.79      0.78       977
        ship       0.84      0.88      0.86      1003
       truck       0.80      0.83      0.82      1022

    accuracy                           0.76     10000
   macro avg       0.76      0.76      0.75     10000
weighted avg       0.76      0.76      0.75     10000
```

Predictions

Predict and visualize the results for a sample image.

In [ ]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.preprocessing import image

# Load an image
img = tf.keras.preprocessing.image.load_img(train_dirs[0] + '/006_frog/al
# img = tf.keras.preprocessing.image.load_img(train_dirs[0] + '/000_airpl

# Preprocess the image
img_array = image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

plt.imshow(img)
plt.show()

print(img_array.shape)
result = model.predict(img_array)
print("Result: ", result.round())
```




```
(1, 32, 32, 3)
1/1 [==============================] - 0s 27ms/step
Result:  [[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]]
```