Model_Scratch 02 - With 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)
```

```
2024-06-22 21:13:29.856133: E external/local_xla/xla/stream_executor/cuda/
cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register
factory for plugin cuDNN when one has already been registered
2024-06-22 21:13:29.856173: E external/local_xla/xla/stream_executor/cuda/
cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register
factory for plugin cuFFT when one has already been registered
2024-06-22 21:13:29.862190: E external/local_xla/xla/stream_executor/cuda/
cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to regist
er factory for plugin cuBLAS when one has already been registered
2024-06-22 21:13:29.964286: I tensorflow/core/platform/cpu_feature_guard.c
c:182] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebui
ld TensorFlow with the appropriate compiler flags.
2024-06-22 21:13:31.281108: W tensorflow/compiler/tf2tensorrt/utils/py_uti
ls.cc:38] TF-TRT Warning: Could not find TensorRT
Found 10000 files belonging to 10 classes.
```

```
2024-06-22 21:13:33.348462: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.711339: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.711491: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.712127: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.712262: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.712378: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.794682: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.794860: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.795001: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:13:33.796277: I tensorflow/core/common_runtime/gpu/gpu_devic
e.cc:1929] Created device /job:localhost/replica:0/task:0/device:GPU:0 wit
h 257 MB memory:  -> device: 0, name: NVIDIA GeForce GTX 1060 6GB, pci bus
id: 0000:01:00.0, compute capability: 6.1
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)
```

Data Augmentation

Rendom change of flipping the image horizontally.
Random chance of moving the image horizontally and vertically [-10%, 10%].

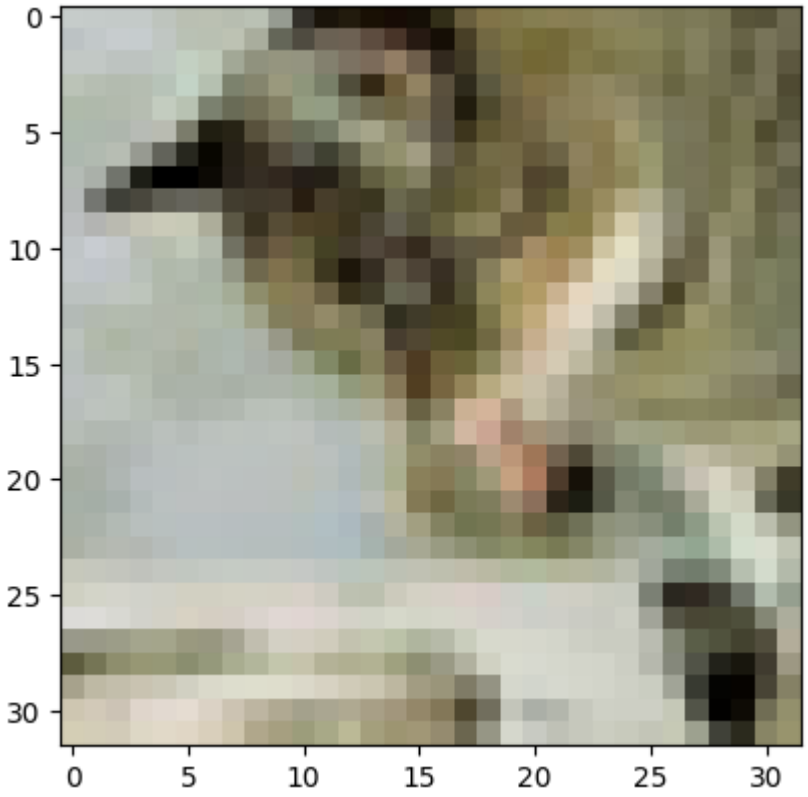
Tried with a more complex approach to data augmentation, but the results were worse because of the small size of the images.

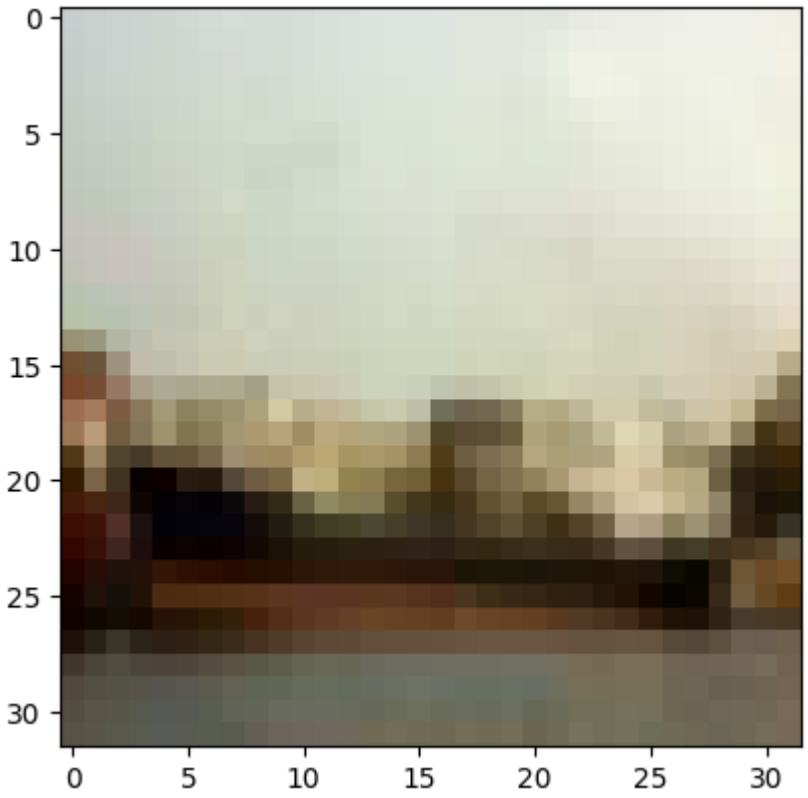
In [ ]:
```
from keras import layers

data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomTranslation(0.1, 0.1, fill_mode='nearest'),
])
```

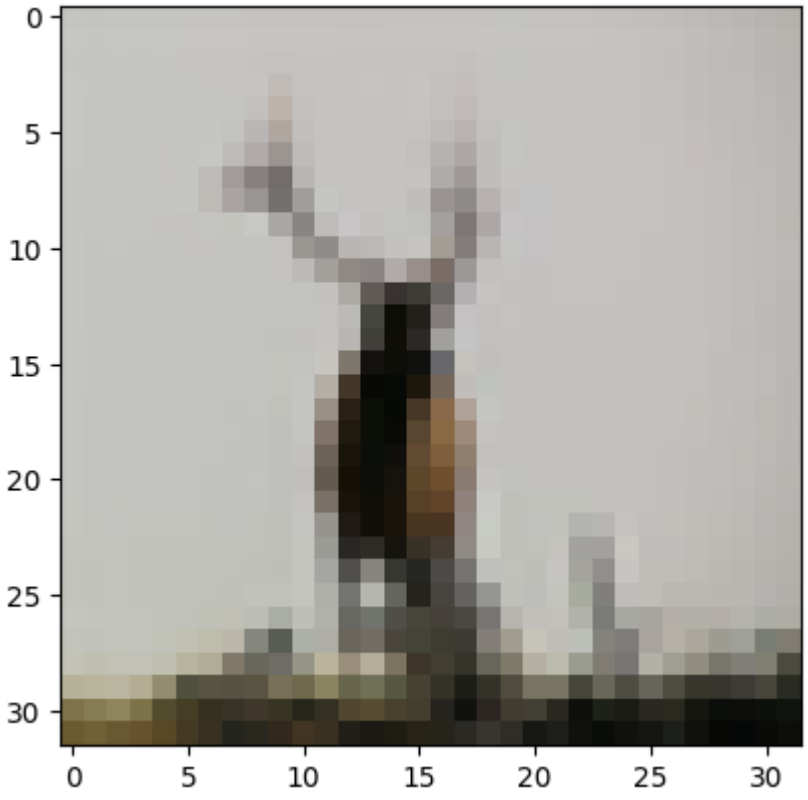
In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

#Plot some Augmented images
for images, labels in train_dataset.take(1):
    plt.figure(figsize=(10, 10))
    first_image = images[0]
    for i in range(4):
        ax = plt.subplot(2, 2, i + 1)
        augmented_image = data_augmentation(tf.expand_dims(first_image, 0
        plt.imshow(augmented_image[0] / 255)
        plt.axis('off')
```

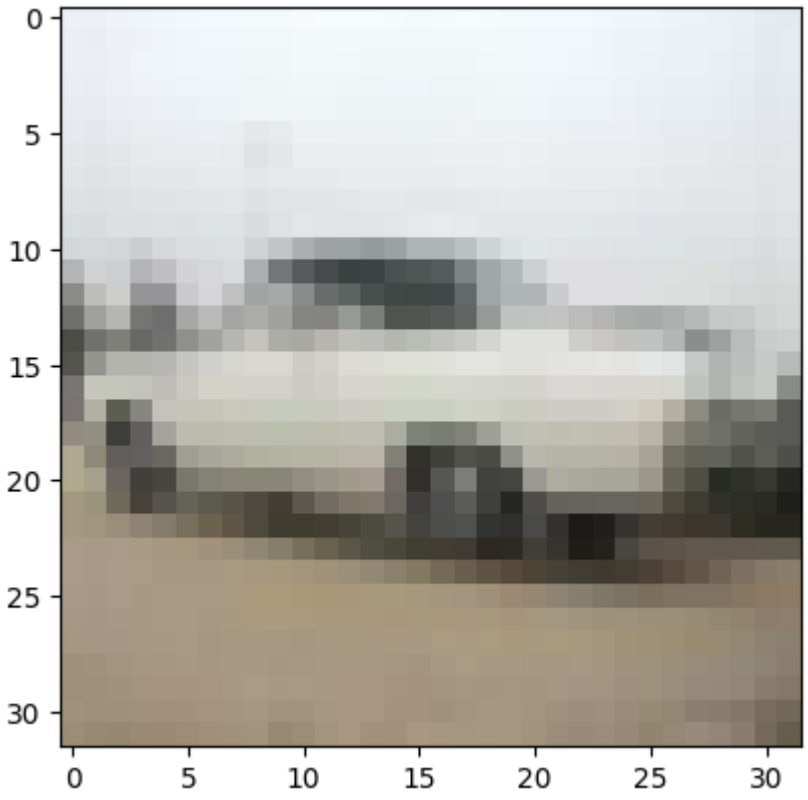


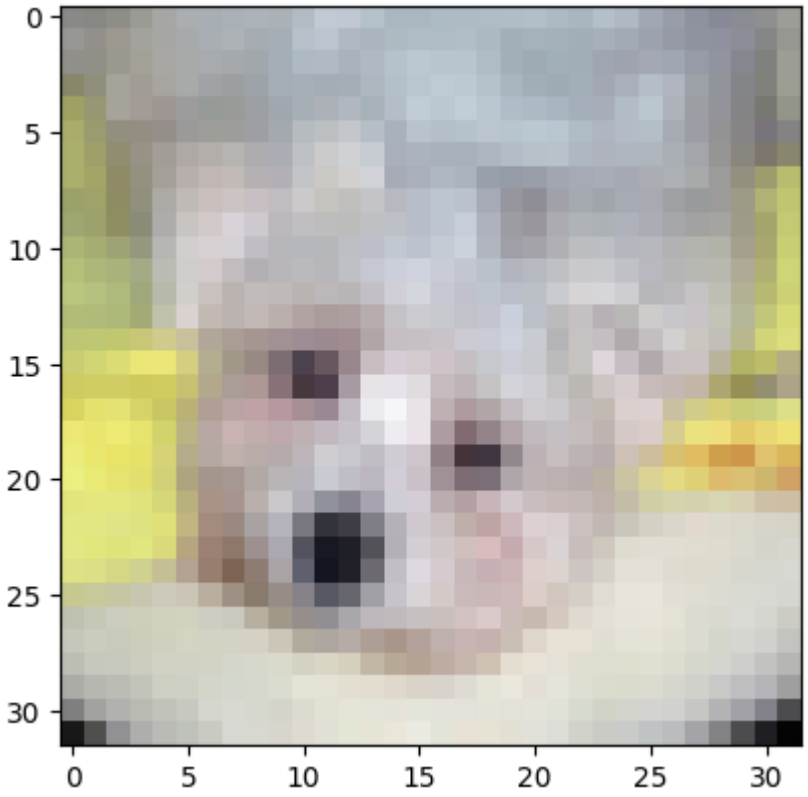

In [ ]:
```
import matplotlib.pyplot as plt

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

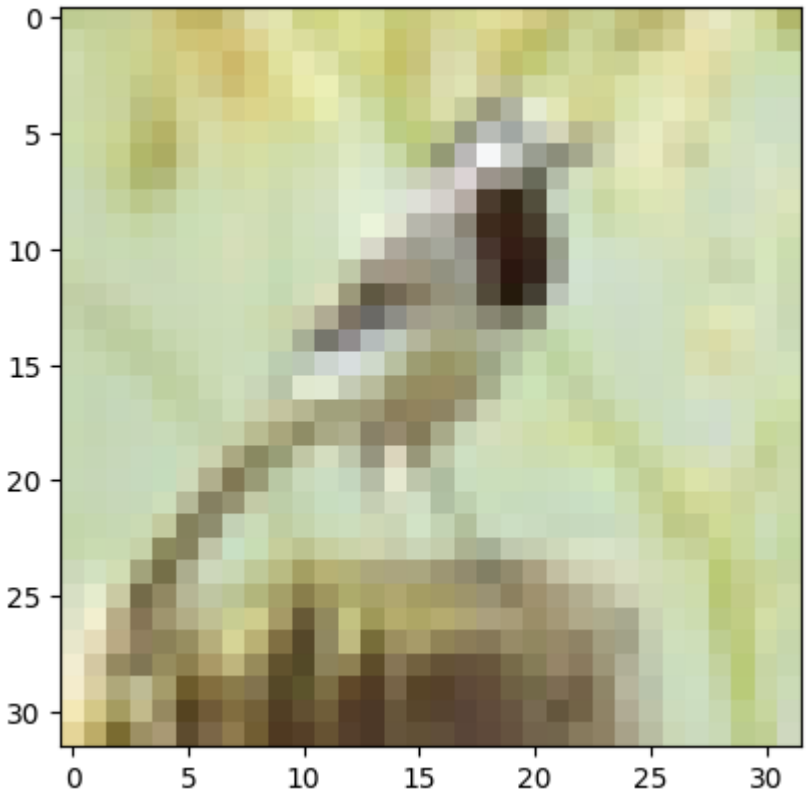

0
5
10
15
20
25
30
0
5
10
15
20
25
30

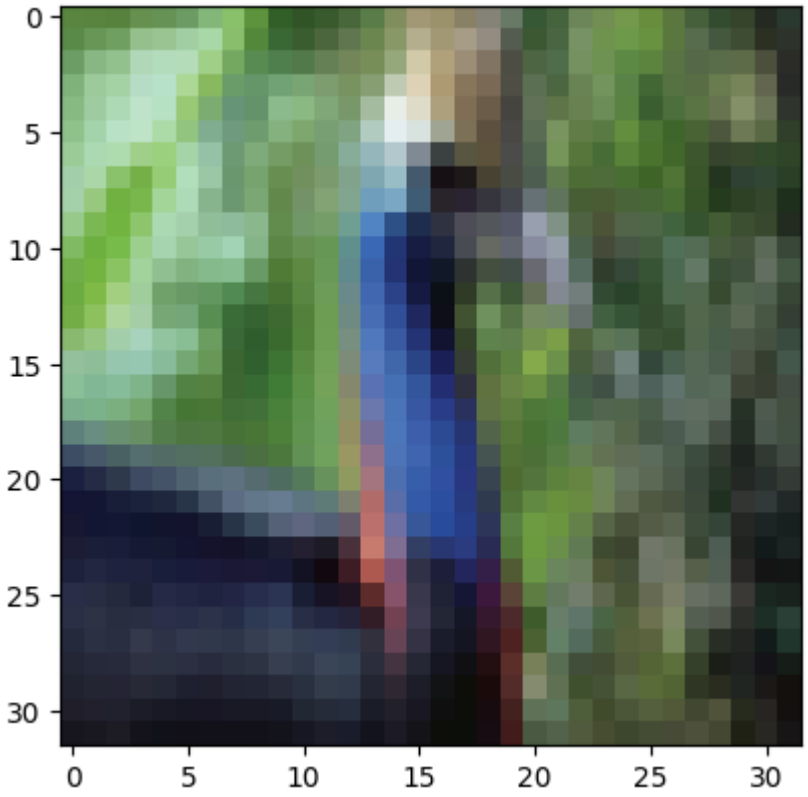

0
5
10
15
20
25
30
0
5
10
15
20
25
30

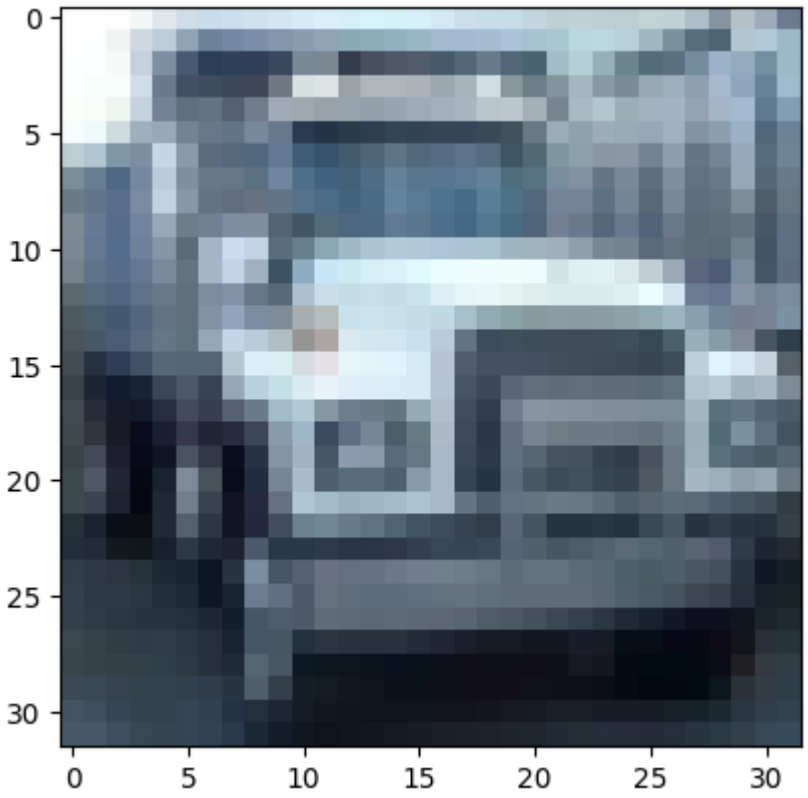

0
5
10
15
20
25
30
0
5
10
15
20
25
30

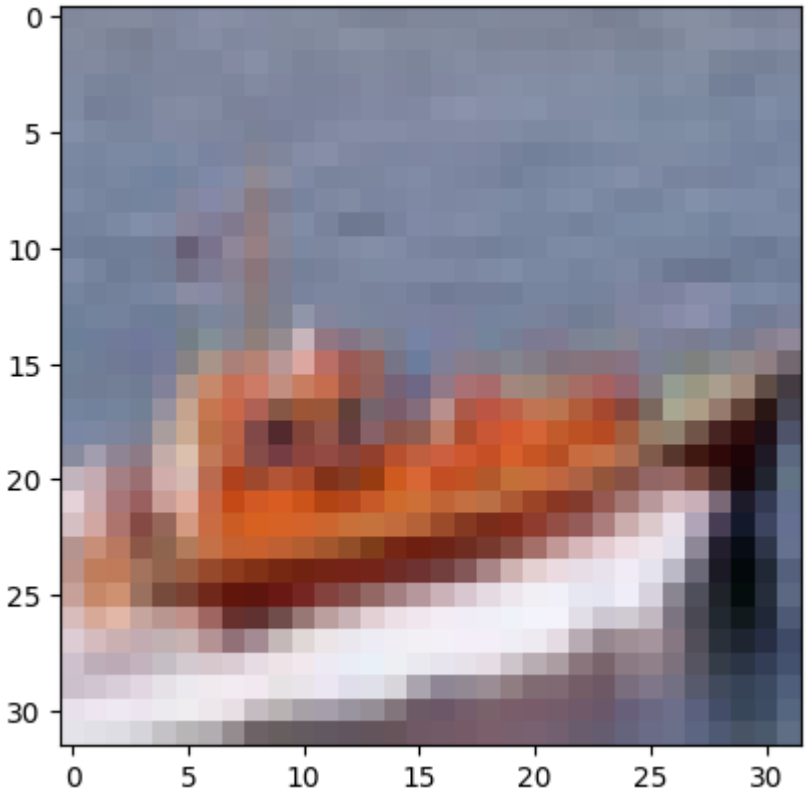

0
5
10
15
20
25
30
0
5
10
15
20
25
30


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


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, regularizers

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

x = data_augmentation(inputs)

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

## 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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 sequential (Sequential)     (None, 32, 32, 3)         0

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

 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (Batch  (None, 32, 32, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_1 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 128)         73856

 batch_normalization_2 (Bat  (None, 8, 8, 128)         512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 4, 4, 128)         0
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 sequential (Sequential)     (None, 32, 32, 3)         0

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

 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (Batch  (None, 32, 32, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_1 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 128)         73856

 batch_normalization_2 (Bat  (None, 8, 8, 128)         512
```

```
chNormalization)

max_pooling2d_2 (MaxPoolin  (None, 4, 4, 128)         0
g2D)

flatten (Flatten)           (None, 2048)              0

dense (Dense)               (None, 512)               1049088

batch_normalization_3 (Bat  (None, 512)               2048
chNormalization)

dense_1 (Dense)             (None, 256)               131328

batch_normalization_4 (Bat  (None, 256)               1024
chNormalization)

dropout (Dropout)           (None, 256)               0

dense_2 (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=6,
                           restore_best_weights=True)
model_checkpoint = ModelCheckpoint('models/S02/checkpoints/S02-DA-cp.h5',

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

```
Epoch 1/100
2024-06-22 21:13:42.893017: I external/local_xla/xla/stream_executor/cuda/
cuda_dnn.cc:454] Loaded cuDNN version 8904
2024-06-22 21:13:44.811580: I external/local_xla/xla/service/service.cc:16
8] XLA service 0x7c7cec577f40 initialized for platform CUDA (this does not
guarantee that XLA will be used). Devices:
2024-06-22 21:13:44.811657: I external/local_xla/xla/service/service.cc:17
6]   StreamExecutor device (0): NVIDIA GeForce GTX 1060 6GB, Compute Capab
ility 6.1
2024-06-22 21:13:44.817436: I tensorflow/compiler/mlir/tensorflow/utils/du
mp_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CR
ASH_REPRODUCER_DIRECTORY` to enable.
WARNING: All log messages before absl::InitializeLog() is called are writt
en to STDERR
I0000 00:00:1719087224.860796  135320 device_compiler.h:186] Compiled clus
ter using XLA!  This line is logged at most once for the lifetime of the p
rocess.
628/628 [==============================] - 11s 11ms/step - loss: 1.6528 -
acc: 0.4524 - val_loss: 1.1419 - val_acc: 0.5954 - lr: 0.0010
Epoch 2/100
 15/628 [..............................] - ETA: 4s - loss: 1.1153 - acc:
0.5958
/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 9ms/step - loss: 1.1086 - ac
c: 0.6110 - val_loss: 1.0601 - val_acc: 0.6335 - lr: 0.0010
Epoch 3/100
628/628 [==============================] - 6s 9ms/step - loss: 0.9522 - ac
c: 0.6705 - val_loss: 0.9207 - val_acc: 0.6807 - lr: 0.0010
Epoch 4/100
628/628 [==============================] - 6s 9ms/step - loss: 0.8459 - ac
c: 0.7068 - val_loss: 0.7693 - val_acc: 0.7300 - lr: 0.0010
Epoch 5/100
628/628 [==============================] - 6s 9ms/step - loss: 0.7720 - ac
c: 0.7316 - val_loss: 0.7561 - val_acc: 0.7390 - lr: 0.0010
Epoch 6/100
628/628 [==============================] - 5s 9ms/step - loss: 0.7269 - ac
c: 0.7481 - val_loss: 0.7567 - val_acc: 0.7417 - lr: 0.0010
Epoch 7/100
628/628 [==============================] - 6s 9ms/step - loss: 0.6827 - ac
c: 0.7661 - val_loss: 0.9545 - val_acc: 0.6982 - lr: 0.0010
Epoch 8/100
628/628 [==============================] - 5s 9ms/step - loss: 0.6442 - ac
c: 0.7783 - val_loss: 0.7213 - val_acc: 0.7539 - lr: 0.0010
Epoch 9/100
628/628 [==============================] - 5s 8ms/step - loss: 0.6100 - ac
c: 0.7898 - val_loss: 0.6309 - val_acc: 0.7871 - lr: 0.0010
Epoch 10/100
628/628 [==============================] - 5s 8ms/step - loss: 0.5848 - ac
c: 0.8000 - val_loss: 0.6869 - val_acc: 0.7647 - lr: 0.0010
Epoch 11/100
628/628 [==============================] - 5s 8ms/step - loss: 0.5602 - ac
c: 0.8085 - val_loss: 0.6550 - val_acc: 0.7870 - lr: 0.0010
Epoch 12/100
625/628 [============================>.] - ETA: 0s - loss: 0.5334 - acc:
0.8179
Epoch 12: ReduceLROnPlateau reducing learning rate to 0.000500000023748725
7.
628/628 [==============================] - 5s 8ms/step - loss: 0.5335 - ac
c: 0.8178 - val_loss: 1.2359 - val_acc: 0.6080 - lr: 0.0010
Epoch 13/100
628/628 [==============================] - 5s 8ms/step - loss: 0.4624 - ac
c: 0.8400 - val_loss: 0.5660 - val_acc: 0.8116 - lr: 5.0000e-04
Epoch 14/100
628/628 [==============================] - 5s 8ms/step - loss: 0.4299 - ac
c: 0.8510 - val_loss: 0.5473 - val_acc: 0.8191 - lr: 5.0000e-04
Epoch 15/100
628/628 [==============================] - 5s 8ms/step - loss: 0.4028 - ac
c: 0.8624 - val_loss: 0.5733 - val_acc: 0.8139 - lr: 5.0000e-04
Epoch 16/100
628/628 [==============================] - 5s 9ms/step - loss: 0.3973 - ac
c: 0.8625 - val_loss: 0.5535 - val_acc: 0.8193 - lr: 5.0000e-04
Epoch 17/100
628/628 [==============================] - 6s 9ms/step - loss: 0.3810 - ac
c: 0.8683 - val_loss: 0.5368 - val_acc: 0.8242 - lr: 5.0000e-04
Epoch 18/100
628/628 [==============================] - 5s 8ms/step - loss: 0.3684 - ac
c: 0.8739 - val_loss: 0.6329 - val_acc: 0.7967 - lr: 5.0000e-04
Epoch 19/100
628/628 [==============================] - 5s 9ms/step - loss: 0.3588 - ac
c: 0.8759 - val_loss: 0.5325 - val_acc: 0.8278 - lr: 5.0000e-04
Epoch 20/100
628/628 [==============================] - 5s 9ms/step - loss: 0.3448 - ac
c: 0.8820 - val_loss: 0.6181 - val_acc: 0.8127 - lr: 5.0000e-04
```

```
Epoch 21/100
628/628 [==============================] - 5s 9ms/step - loss: 0.3348 - ac
c: 0.8856 - val_loss: 0.5393 - val_acc: 0.8245 - lr: 5.0000e-04
Epoch 22/100
628/628 [==============================] - ETA: 0s - loss: 0.3229 - acc:
0.8892
Epoch 22: ReduceLROnPlateau reducing learning rate to 0.000250000011874362
8.
628/628 [==============================] - 5s 9ms/step - loss: 0.3229 - ac
c: 0.8892 - val_loss: 0.5697 - val_acc: 0.8147 - lr: 5.0000e-04
Epoch 23/100
628/628 [==============================] - 5s 9ms/step - loss: 0.2913 - ac
c: 0.9007 - val_loss: 0.5387 - val_acc: 0.8336 - lr: 2.5000e-04
Epoch 24/100
628/628 [==============================] - 5s 9ms/step - loss: 0.2719 - ac
c: 0.9065 - val_loss: 0.5348 - val_acc: 0.8349 - lr: 2.5000e-04
Epoch 25/100
628/628 [==============================] - 5s 8ms/step - loss: 0.2682 - ac
c: 0.9084 - val_loss: 0.5980 - val_acc: 0.8245 - lr: 2.5000e-04
Epoch 26/100
628/628 [==============================] - 5s 9ms/step - loss: 0.2607 - ac
c: 0.9104 - val_loss: 0.5219 - val_acc: 0.8394 - lr: 2.5000e-04
Epoch 27/100
628/628 [==============================] - 5s 8ms/step - loss: 0.2533 - ac
c: 0.9132 - val_loss: 0.5482 - val_acc: 0.8357 - lr: 2.5000e-04
Epoch 28/100
628/628 [==============================] - 5s 8ms/step - loss: 0.2429 - ac
c: 0.9180 - val_loss: 0.5823 - val_acc: 0.8289 - lr: 2.5000e-04
Epoch 29/100
626/628 [============================>.] - ETA: 0s - loss: 0.2358 - acc:
0.9195
Epoch 29: ReduceLROnPlateau reducing learning rate to 0.000125000005937181
4.
628/628 [==============================] - 5s 9ms/step - loss: 0.2360 - ac
c: 0.9195 - val_loss: 0.6094 - val_acc: 0.8245 - lr: 2.5000e-04
Epoch 30/100
628/628 [==============================] - 5s 9ms/step - loss: 0.2194 - ac
c: 0.9263 - val_loss: 0.5396 - val_acc: 0.8443 - lr: 1.2500e-04
Epoch 31/100
628/628 [==============================] - 5s 9ms/step - loss: 0.2069 - ac
c: 0.9305 - val_loss: 0.5457 - val_acc: 0.8381 - lr: 1.2500e-04
Epoch 32/100
628/628 [==============================] - 5s 8ms/step - loss: 0.2073 - ac
c: 0.9310 - val_loss: 0.5386 - val_acc: 0.8425 - lr: 1.2500e-04
Epoch 33/100
625/628 [============================>.] - ETA: 0s - loss: 0.2030 - acc:
0.9293
Epoch 33: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-0
5.
628/628 [==============================] - 5s 9ms/step - loss: 0.2034 - ac
c: 0.9291 - val_loss: 0.5567 - val_acc: 0.8411 - lr: 1.2500e-04
Epoch 34/100
628/628 [==============================] - 5s 9ms/step - loss: 0.1945 - ac
c: 0.9342 - val_loss: 0.5416 - val_acc: 0.8417 - lr: 6.2500e-05
Epoch 35/100
628/628 [==============================] - 5s 9ms/step - loss: 0.1919 - ac
c: 0.9337 - val_loss: 0.5352 - val_acc: 0.8439 - lr: 6.2500e-05
Epoch 36/100
628/628 [==============================] - 5s 9ms/step - loss: 0.1847 - ac
c: 0.9379 - val_loss: 0.5245 - val_acc: 0.8490 - lr: 6.2500e-05
```

```
Epoch 37/100
628/628 [==============================] - 5s 9ms/step - loss: 0.1839 - ac
c: 0.9366 - val_loss: 0.5393 - val_acc: 0.8451 - lr: 6.2500e-05
Epoch 38/100
628/628 [==============================] - 5s 8ms/step - loss: 0.1829 - ac
c: 0.9397 - val_loss: 0.5397 - val_acc: 0.8464 - lr: 6.2500e-05
Epoch 39/100
622/628 [============================>.] - ETA: 0s - loss: 0.1811 - acc:
0.9395
Epoch 39: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-0
5.
628/628 [==============================] - 5s 9ms/step - loss: 0.1809 - ac
c: 0.9395 - val_loss: 0.5375 - val_acc: 0.8455 - lr: 6.2500e-05
Epoch 40/100
628/628 [==============================] - 5s 9ms/step - loss: 0.1742 - ac
c: 0.9399 - val_loss: 0.5363 - val_acc: 0.8473 - lr: 3.1250e-05
Epoch 41/100
628/628 [==============================] - 5s 8ms/step - loss: 0.1744 - ac
c: 0.9416 - val_loss: 0.5467 - val_acc: 0.8439 - lr: 3.1250e-05
Epoch 42/100
625/628 [============================>.] - ETA: 0s - loss: 0.1709 - acc:
0.9423
Epoch 42: ReduceLROnPlateau reducing learning rate to 1.5625000742147677
e-05.
628/628 [==============================] - 5s 9ms/step - loss: 0.1706 - ac
c: 0.9424 - val_loss: 0.5391 - val_acc: 0.8462 - lr: 3.1250e-05
```

Save Model

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

```
/tmp/ipykernel_135234/967095280.py:1: UserWarning: You are saving your mod
el as an HDF5 file via `model.save()`. This file format is considered lega
cy. We recommend using instead the native Keras format, e.g. `model.save
('my_model.keras')`.
  keras.models.save_model(model, 'models/S02/S02-DA-model.h5')
```

Load Model

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

Out[ ]:
```
<keras.src.engine.functional.Functional at 0x7c7d8c562990>
```

EVALUATION

Evaluate the model on the validation dataset.

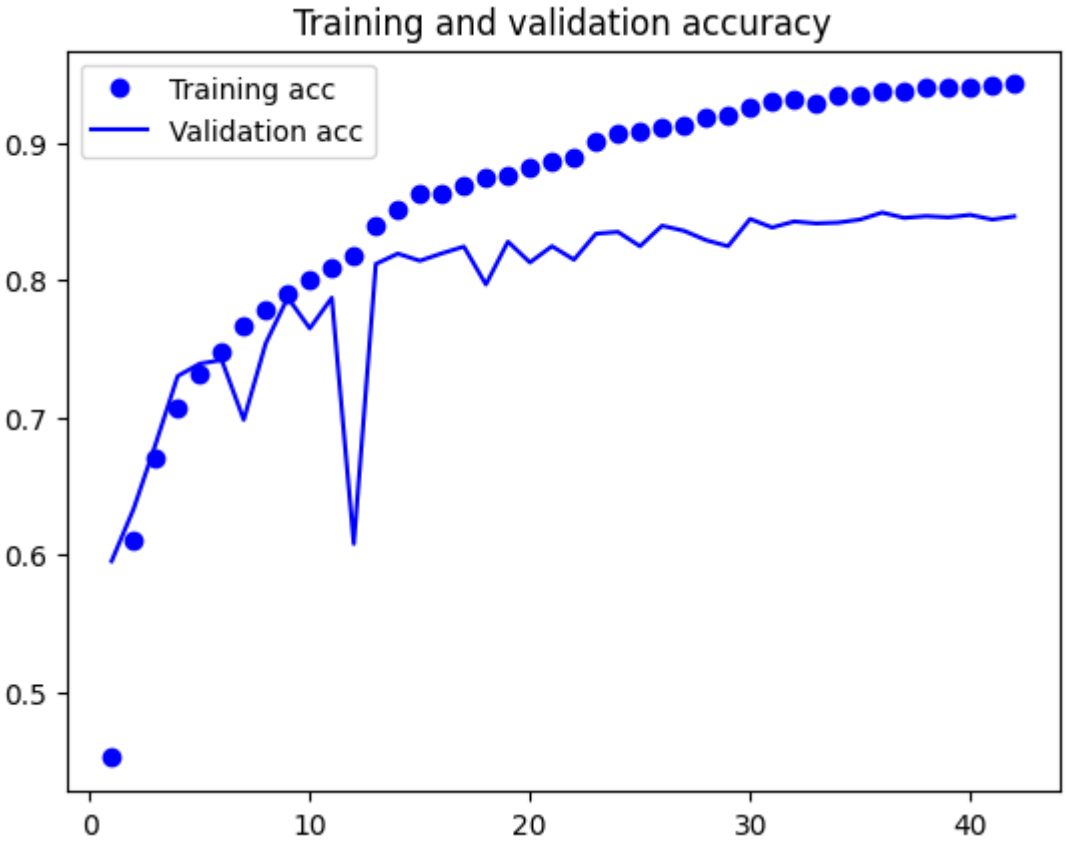
In [ ]:
```
val_loss, val_acc = model.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

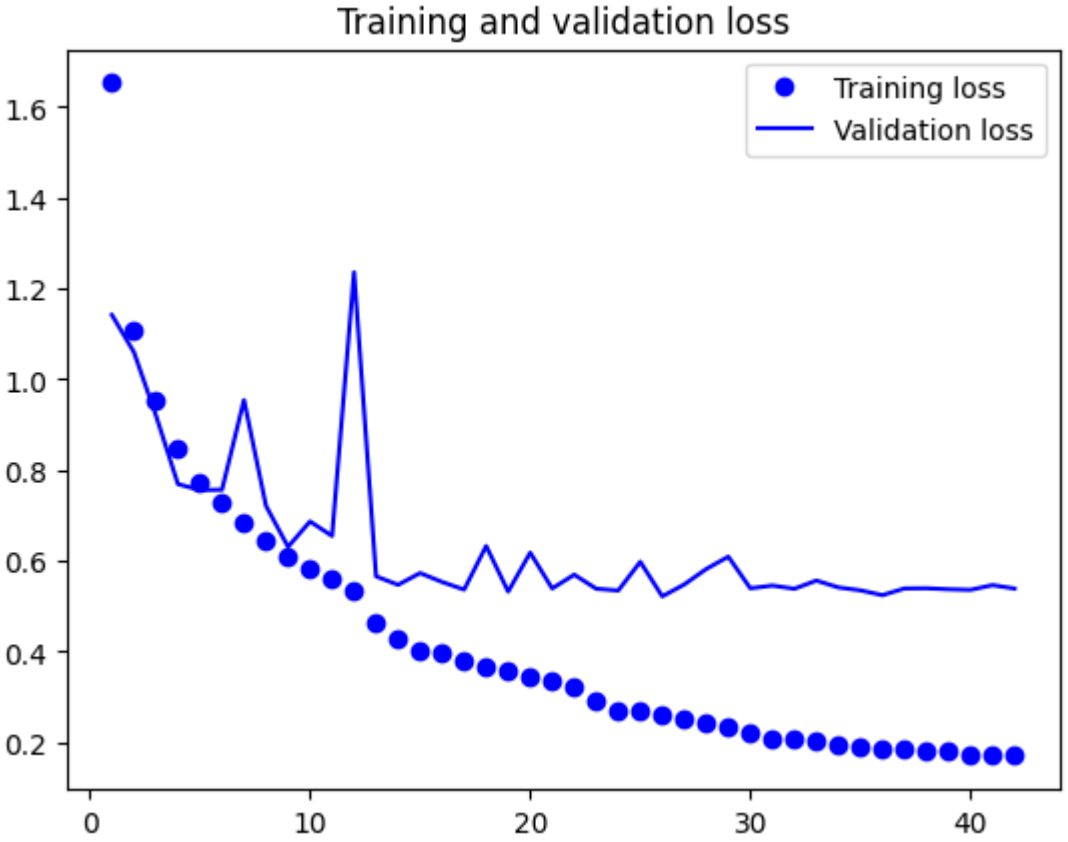
```
157/157 [==============================] - 0s 3ms/step - loss: 0.5245 - ac
c: 0.8490
val_acc: 0.8489999771118164
```

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()
```





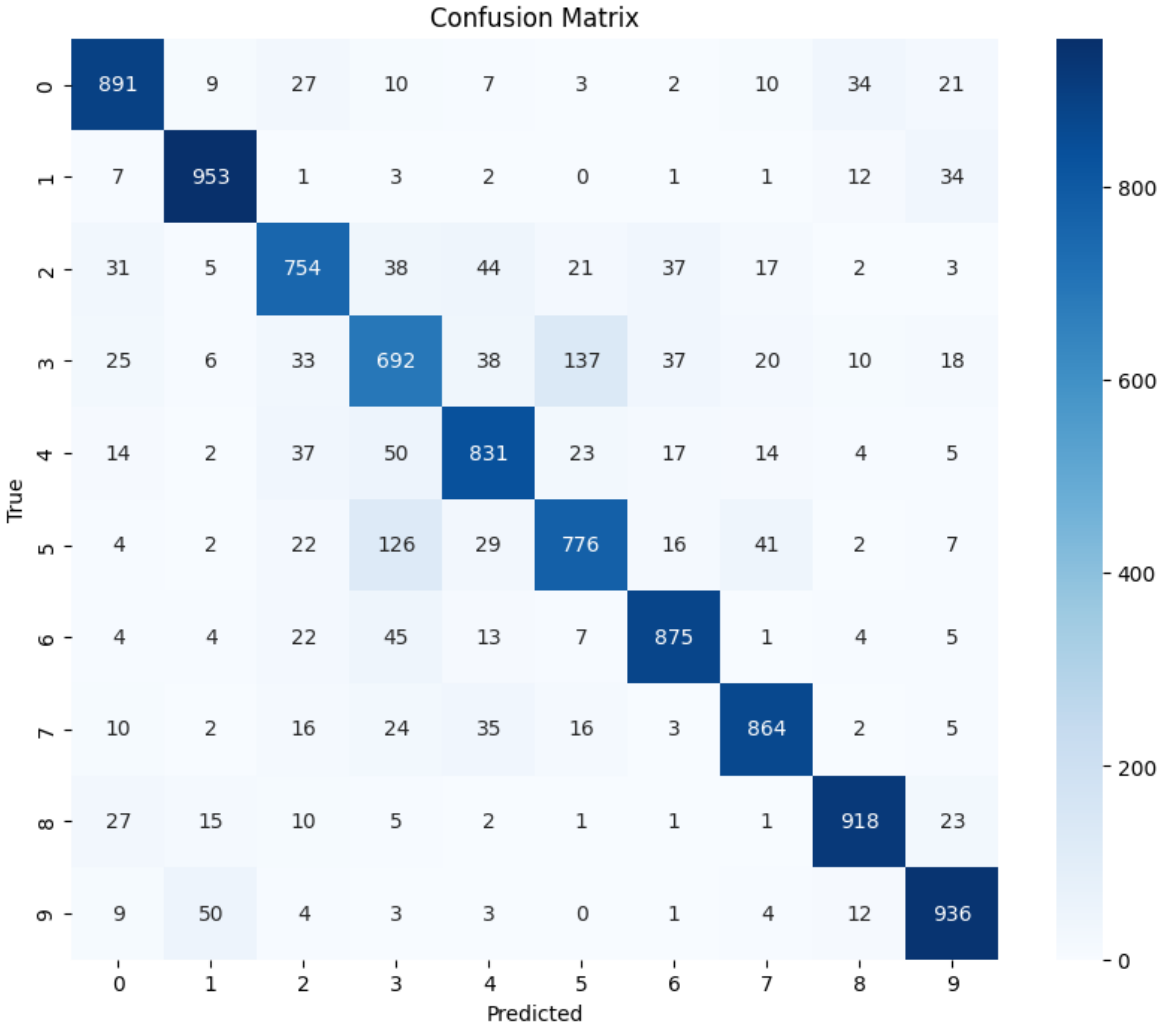
Confusion Matrix

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 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 6ms/step
2/2 [==============================] - 0s 7ms/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 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 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/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 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 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 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 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 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 7ms/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 3ms/step
2/2 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 8ms/step
2/2 [==============================] - 0s 3ms/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 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 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 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 4ms/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 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 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 2ms/step
2/2 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 3ms/step
1/1 [==============================] - 0s 109ms/step
```



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

```
Confusion Matrix:
[[891   9  27  10   7   3   2  10  34  21]
 [  7 953   1   3   2   0   1   1  12  34]
 [ 31   5 754  38  44  21  37  17   2   3]
 [ 25   6  33 692  38 137  37  20  10  18]
 [ 14   2  37  50 831  23  17  14   4   5]
 [  4   2  22 126  29 776  16  41   2   7]
 [  4   4  22  45  13   7 875   1   4   5]
 [ 10   2  16  24  35  16   3 864   2   5]
 [ 27  15  10   5   2   1   1   1 918  23]
 [  9  50   4   3   3   0   1   4  12 936]]
```

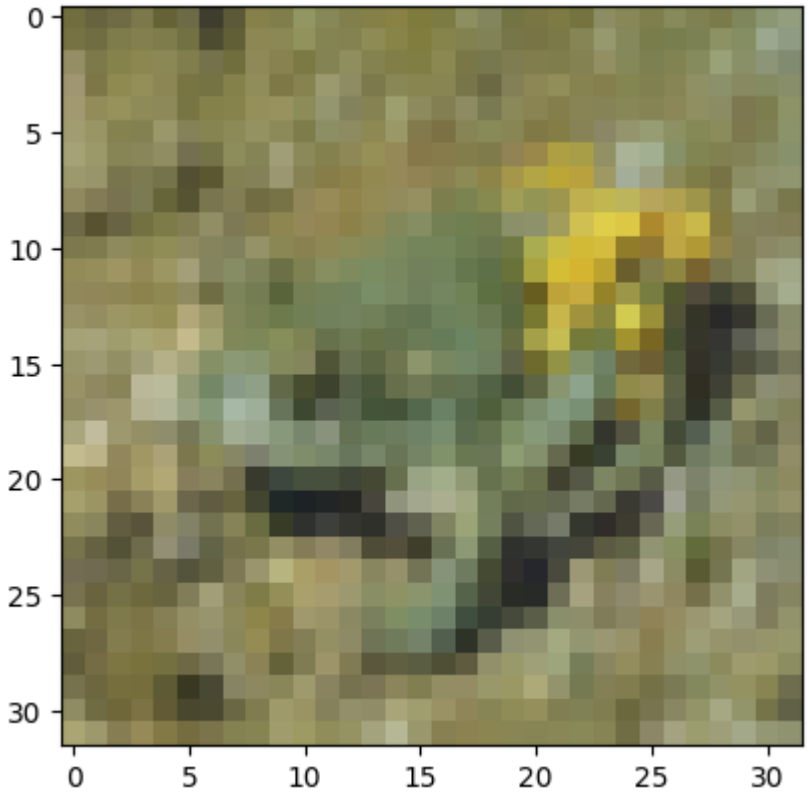
In [ ]:
```
from sklearn.metrics import classification_report

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

	precision	recall	f1-score	support
airplane	0.87	0.88	0.88	1014
automobile	0.91	0.94	0.92	1014
bird	0.81	0.79	0.80	952
cat	0.69	0.68	0.69	1016
deer	0.83	0.83	0.83	997
dog	0.79	0.76	0.77	1025
frog	0.88	0.89	0.89	980
horse	0.89	0.88	0.89	977
ship	0.92	0.92	0.92	1003
truck	0.89	0.92	0.90	1022
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	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 209ms/step
Result:  [[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]]
```