Untitled1

August 8, 2024

[1]:
```
import cv2
import numpy as np
from keras.datasets import cifar10
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense , Conv2D , MaxPooling2D ,Dropout, Flatten ,␣
  ↪BatchNormalization
from keras.regularizers import l2
from keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.models import load_model
import warnings
warnings.filterwarnings('ignore')
```

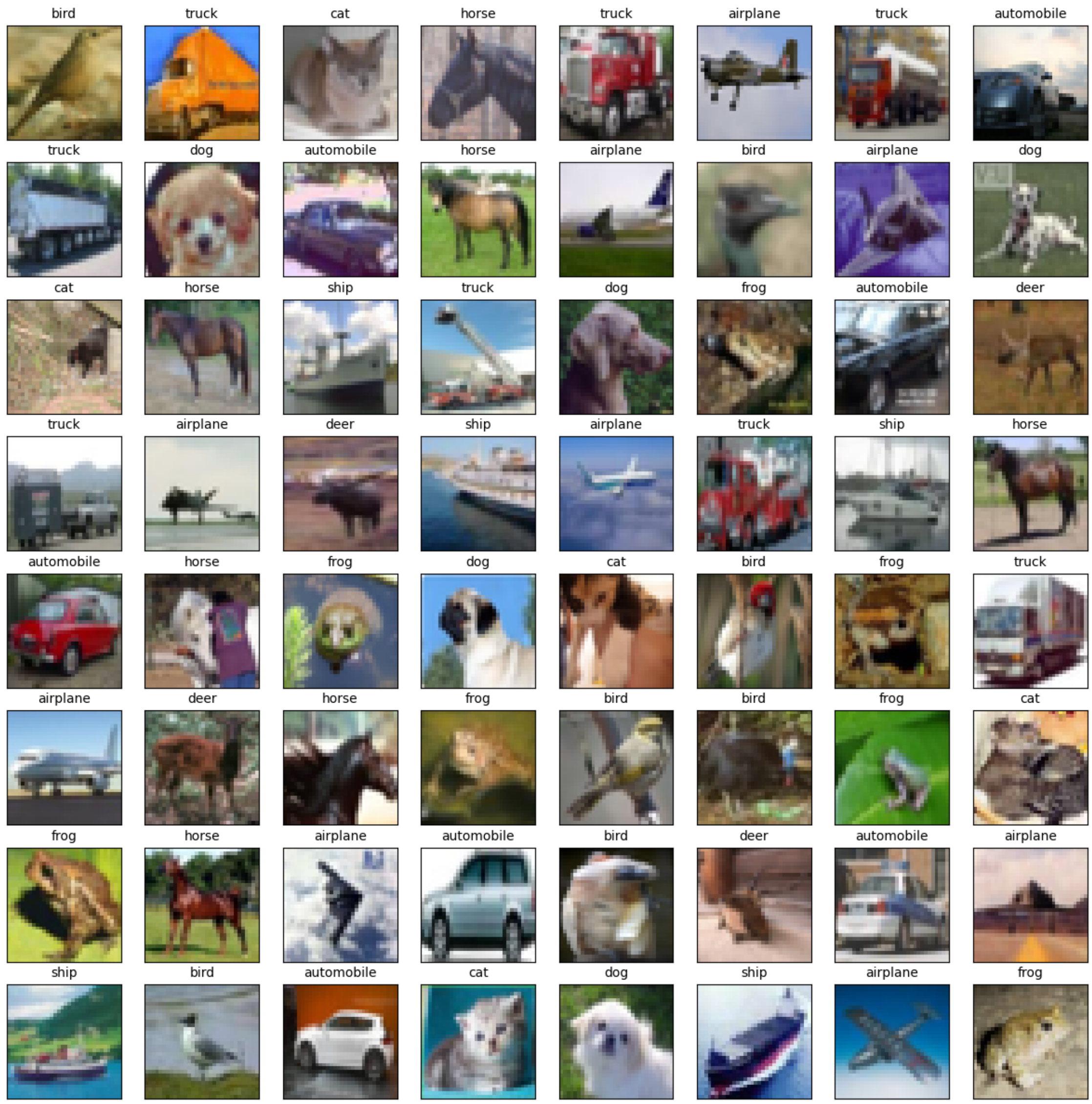
[2]:
```
(X_train ,y_train),(X_test , y_test) = cifar10.load_data()
```

[3]:
```
X_train ,X_valid ,y_train , y_valid = train_test_split(X_train ,y_train ,␣
  ↪test_size =0.1,random_state =0)
```

[4]:
```
print('Train Image Shape : ', X_train.shape)
print('Train labels :',y_train.shape)

print('Valid Image shape :',X_valid.shape)
print('valid Labels :', y_valid.shape)

print('test Image :',X_test.shape)
print('test labels :',y_test.shape)
```

```
Train Image Shape :  (45000, 32, 32, 3)
Train labels : (45000, 1)
Valid Image shape : (5000, 32, 32, 3)
valid Labels : (5000, 1)
test Image : (10000, 32, 32, 3)
test labels : (10000, 1)
```

[5]:
```
class_name = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',␣
 ↪'horse', 'ship', 'truck']
plt.figure(figsize=(15,15))

for i in range(64):

    plt.subplot(8,8,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)

    plt.imshow(X_train[i])

    plt.title(class_name[y_train[i][0]],fontsize =10)

plt.show()
```



#Normalization of Image

[6]:
```
X_train = X_train.astype('float32')
X_valid = X_valid.astype('float32')
X_test = X_test.astype('float32')
```

[7]:
```
#Calculate the mean and Standard deviation of the traning image
```

[8]:
```
mean = np.mean(X_train)
std = np.std(X_train)
```

[9]:
```
X_train = (X_train-mean)/(std+1e-7) # 1e-7 add for prevent the division by 0
X_test  = (X_test -mean)/(std+1e-7)
```

```
X_valid = (X_valid - mean)/(std+1e-7)
```

1 ONE - HOT ENCODING OF LABELS

#class_names: This is a list that maps the integer labels to human-readable names. #Each index in this list corresponds to a class label in the one-hot encoded vector. #For instance, class_names[3] is 'cat', so the one-hot vector [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] corresponds to the label 'cat'.

[10]:
```
y_train = to_categorical(y_train , 10)
y_valid = to_categorical(y_valid , 10)
y_test = to_categorical(y_test , 10)
```

2 Data Augmentation

is a powerful technique to enhance your dataset and improve model performance.

By applying various transformations to your data, you make your model more robust and capable of handling a wider range of real-world scenarios

Why Use Data Augmentation?

Increase Dataset Size: Helps in cases where you have a limited amount of data. By generating augmented versions of your existing data, you effectively increase the dataset size. ##### Improve Model Generalization: Helps the model generalize better to new, unseen data by providing more varied examples during training. ##### Reduce Overfitting: By introducing variations, the model is less likely to memorize the training data, which reduces overfitting.

The choice of data augmentation techniques often depends on the specific characteristics of the dataset and the problem at hand. The CIFAR-10 dataset comprises small color images of objects from 10 different classes. Given the nature of these images, some augmentation techniques are more applicable than others:

[11]:
```
# Data augmentation
data_generator = ImageDataGenerator(
    # Rotate images randomly by up to 15 degrees
    rotation_range=15,

    # Shift images horizontally by up to 12% of their width
    width_shift_range=0.12,

    # Shift images vertically by up to 12% of their height
    height_shift_range=0.12,

    # Randomly flip images horizontally
    horizontal_flip=True,

    # Zoom images in by up to 10%
```

```
    zoom_range=0.1,

    # Change brightness by up to 10%
    brightness_range=[0.9,1.1],

    # Shear intensity (shear angle in counter-clockwise direction in degrees)
    shear_range=10,

    # Channel shift intensity
    channel_shift_range=0.1,
)
```

3 Define CNN Model Architecture

The network begins with a pair of Conv2D layers, each with 32 filters of size 3x3. This is followed by a Batch Normalization layer which accelerates training and provides some level of regularization, helping to prevent overfitting.

The pairs of Conv2D layers are followed by a MaxPooling2D layer, which reduces the spatial dimensions (height and width), effectively providing a form of translation invariance and reducing computational complexity. This is followed by a Dropout layer that randomly sets a fraction (0.2 for the first dropout layer) of the input units to 0 at each update during training, helping to prevent overfitting.

This pattern of two Conv2D layers, followed by a Batch Normalization layer, a MaxPooling2D layer, and a Dropout layer, repeats three more times. The number of filters in the Conv2D layers doubles with each repetition, starting from 32 and going up to 64, 128, and then 256. This increasing pattern helps the network to learn more complex features at each level. The dropout rate also increases at each step, from 0.2 to 0.5.

After the convolutional and pooling layers, a Flatten layer is used to convert the 2D outputs of the preceding layer into a 1D vector.

Finally, a Dense (or fully connected) layer is used for classification. It has 10 units, each representing one of the 10 classes of the CIFAR-10 dataset, and a softmax activation function is used to convert the outputs to probability scores for each class.

[12]:
```
model = Sequential()

# Set the weight decay value for L2 regularization
weight_decay = 0.0001

# first Convolutional layer
model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay), input_shape=X_train.
↪shape[1:]))

# add batch normalization layer
model.add(BatchNormalization())
```

```python
# Second Convolutional layer (similar to first)
model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
# add batch normalization layer
model.add(BatchNormalization())

# adding first max pooling layer
model.add(MaxPooling2D(pool_size=(2,2)))

# add drop out layer
model.add(Dropout(rate=0.2))

# adding the third and fourth convolutional layers with 64 filters
model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())
model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())


# Add the second max pooling layer and increase dropout rate to 0.3
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.3))


# adding the fifth and sixth convolutional layers with 128 filters
model.add(Conv2D(filters=128, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())

# Add the third max pooling layer and increase dropout rate to 0.4
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.4))

# adding the seventh and eighth convolutional layers with 256 filters

model.add(Conv2D(filters=256, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3,3), padding='same',␣
↪activation='relu', kernel_regularizer=l2(weight_decay)))
```

```python
model.add(BatchNormalization())


# Add the fourth max pooling layer and increase dropout rate to 0.5
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.5))

# Flatten
model.add(Flatten())

model.add(Dense(10,activation='softmax'))
```

[13]:
```python
model.summary()
```

Model: "sequential"

```
 Layer (type)                    Output Shape              Param #

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

 batch_normalization             (None, 32, 32, 32)            128
 (BatchNormalization)

 conv2d_1 (Conv2D)               (None, 32, 32, 32)          9,248

 batch_normalization_1           (None, 32, 32, 32)            128
 (BatchNormalization)

 max_pooling2d (MaxPooling2D)    (None, 16, 16, 32)              0

 dropout (Dropout)               (None, 16, 16, 32)              0

 conv2d_2 (Conv2D)               (None, 16, 16, 64)         18,496

 batch_normalization_2           (None, 16, 16, 64)            256
 (BatchNormalization)

 conv2d_3 (Conv2D)               (None, 16, 16, 64)         36,928

 batch_normalization_3           (None, 16, 16, 64)            256
 (BatchNormalization)

 max_pooling2d_1 (MaxPooling2D)  (None, 8, 8, 64)                0

 dropout_1 (Dropout)             (None, 8, 8, 64)                0
```

```
conv2d_4 (Conv2D)              (None, 8, 8, 128)          73,856

batch_normalization_4          (None, 8, 8, 128)             512
(BatchNormalization)

conv2d_5 (Conv2D)              (None, 8, 8, 128)         147,584

batch_normalization_5          (None, 8, 8, 128)             512
(BatchNormalization)

max_pooling2d_2 (MaxPooling2D) (None, 4, 4, 128)               0

dropout_2 (Dropout)            (None, 4, 4, 128)               0

conv2d_6 (Conv2D)              (None, 4, 4, 256)         295,168

batch_normalization_6          (None, 4, 4, 256)           1,024
(BatchNormalization)

conv2d_7 (Conv2D)              (None, 4, 4, 256)         590,080

batch_normalization_7          (None, 4, 4, 256)           1,024
(BatchNormalization)

max_pooling2d_3 (MaxPooling2D) (None, 2, 2, 256)               0

dropout_3 (Dropout)            (None, 2, 2, 256)               0

flatten (Flatten)              (None, 1024)                    0

dense (Dense)                  (None, 10)                 10,250


Total params: 1,186,346 (4.53 MB)

Trainable params: 1,184,426 (4.52 MB)

Non-trainable params: 1,920 (7.50 KB)
```

4 Training the CNN model

The ReduceLROnPlateau callback is used to reduce the learning rate by half (factor=0.5) whenever the validation loss does not improve for 10 consecutive epochs. This helps to adjust the learning rate dynamically, allowing the model to get closer to the global minimum of the loss function when progress has plateaued. This strategy

can improve the convergence of the training process.

The EarlyStopping callback is employed to monitor the validation loss and halt the training process when there hasn't been any improvement for a certain number of epochs, ensuring that the model doesn't waste computational resources and time. Furthermore, this callback restores the best weights from the training process, ensuring we conclude with the optimal model configuration from the epochs.

[14]:
```
x_batch, y_batch = next(data_generator.flow(X_train, y_train,␣
 ↪batch_size=batch_size))
print(x_batch.shape, y_batch.shape)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 x_batch, y_batch = next(data_generator.flow(X_train, y_train,␣
 ↪batch_size=batch_size))
      2 print(x_batch.shape, y_batch.shape)

NameError: name 'batch_size' is not defined
```

[16]:
```
# Set the batch size for the training
batch_size = 64

# Set the maximum number of epochs for the training
epochs = 300

# Define the optimizer (Adam)
optimizer = Adam()

# Compile the model with the defined optimizer, loss function, and metrics
model.compile(optimizer=optimizer, loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Add ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10,␣
 ↪min_lr=0.00001)

# Add EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=40,␣
 ↪restore_best_weights=True, verbose=1)

# Fit the model on the training data
model.fit(data_generator.flow(X_train, y_train, batch_size=batch_size),
          epochs=epochs,
          validation_data=(X_valid, y_valid),
          callbacks=[reduce_lr, early_stopping],
```

```
            verbose=1)
```

```
Epoch 1/300
704/704               238s 334ms/step -
accuracy: 0.3041 - loss: 2.6676 - val_accuracy: 0.4736 - val_loss: 1.7924 -
learning_rate: 0.0010
Epoch 2/300
704/704               193s 273ms/step -
accuracy: 0.4650 - loss: 1.7757 - val_accuracy: 0.5748 - val_loss: 1.3070 -
learning_rate: 0.0010
Epoch 3/300
704/704               188s 267ms/step -
accuracy: 0.5513 - loss: 1.4684 - val_accuracy: 0.6454 - val_loss: 1.1327 -
learning_rate: 0.0010
Epoch 4/300
704/704               200s 283ms/step -
accuracy: 0.6011 - loss: 1.3191 - val_accuracy: 0.6650 - val_loss: 1.0865 -
learning_rate: 0.0010
Epoch 5/300
704/704               195s 277ms/step -
accuracy: 0.6349 - loss: 1.2150 - val_accuracy: 0.7018 - val_loss: 1.0288 -
learning_rate: 0.0010
Epoch 6/300
704/704               205s 291ms/step -
accuracy: 0.6561 - loss: 1.1821 - val_accuracy: 0.7064 - val_loss: 1.0679 -
learning_rate: 0.0010
Epoch 7/300
704/704               193s 274ms/step -
accuracy: 0.6800 - loss: 1.1257 - val_accuracy: 0.6870 - val_loss: 1.2464 -
learning_rate: 0.0010
Epoch 8/300
704/704               222s 315ms/step -
accuracy: 0.6868 - loss: 1.1209 - val_accuracy: 0.7512 - val_loss: 0.9348 -
learning_rate: 0.0010
Epoch 9/300
704/704               197s 279ms/step -
accuracy: 0.7086 - loss: 1.0665 - val_accuracy: 0.7304 - val_loss: 1.0037 -
learning_rate: 0.0010
Epoch 10/300
704/704               203s 288ms/step -
accuracy: 0.7189 - loss: 1.0679 - val_accuracy: 0.7660 - val_loss: 0.9290 -
learning_rate: 0.0010
Epoch 11/300
704/704               211s 300ms/step -
accuracy: 0.7225 - loss: 1.0547 - val_accuracy: 0.7820 - val_loss: 0.8818 -
learning_rate: 0.0010
Epoch 12/300
704/704               217s 308ms/step -
```

```
accuracy: 0.7316 - loss: 1.0452 - val_accuracy: 0.7742 - val_loss: 0.9115 -
learning_rate: 0.0010
Epoch 13/300
704/704            209s 297ms/step -
accuracy: 0.7438 - loss: 1.0155 - val_accuracy: 0.7908 - val_loss: 0.8716 -
learning_rate: 0.0010
Epoch 14/300
704/704            227s 322ms/step -
accuracy: 0.7476 - loss: 1.0029 - val_accuracy: 0.7944 - val_loss: 0.8753 -
learning_rate: 0.0010
Epoch 15/300
704/704            213s 301ms/step -
accuracy: 0.7548 - loss: 0.9897 - val_accuracy: 0.7762 - val_loss: 0.9498 -
learning_rate: 0.0010
Epoch 16/300
704/704            215s 305ms/step -
accuracy: 0.7570 - loss: 0.9891 - val_accuracy: 0.7822 - val_loss: 0.8936 -
learning_rate: 0.0010
Epoch 17/300
704/704            207s 293ms/step -
accuracy: 0.7662 - loss: 0.9575 - val_accuracy: 0.7840 - val_loss: 0.9399 -
learning_rate: 0.0010
Epoch 18/300
704/704            207s 294ms/step -
accuracy: 0.7672 - loss: 0.9664 - val_accuracy: 0.8146 - val_loss: 0.8485 -
learning_rate: 0.0010
Epoch 19/300
704/704            198s 281ms/step -
accuracy: 0.7794 - loss: 0.9243 - val_accuracy: 0.8158 - val_loss: 0.8319 -
learning_rate: 0.0010
Epoch 20/300
704/704            218s 310ms/step -
accuracy: 0.7785 - loss: 0.9294 - val_accuracy: 0.8174 - val_loss: 0.8391 -
learning_rate: 0.0010
Epoch 21/300
704/704            230s 326ms/step -
accuracy: 0.7817 - loss: 0.9054 - val_accuracy: 0.7918 - val_loss: 0.8881 -
learning_rate: 0.0010
Epoch 22/300
704/704            215s 305ms/step -
accuracy: 0.7824 - loss: 0.9092 - val_accuracy: 0.7988 - val_loss: 0.9094 -
learning_rate: 0.0010
Epoch 23/300
704/704            217s 307ms/step -
accuracy: 0.7896 - loss: 0.8930 - val_accuracy: 0.7938 - val_loss: 0.8955 -
learning_rate: 0.0010
Epoch 24/300
704/704            226s 320ms/step -
```

```
accuracy: 0.7945 - loss: 0.8901 - val_accuracy: 0.8134 - val_loss: 0.8520 -
learning_rate: 0.0010
Epoch 25/300
704/704             220s 312ms/step -
accuracy: 0.8014 - loss: 0.8602 - val_accuracy: 0.8288 - val_loss: 0.7839 -
learning_rate: 0.0010
Epoch 26/300
704/704             218s 309ms/step -
accuracy: 0.8027 - loss: 0.8577 - val_accuracy: 0.8294 - val_loss: 0.7845 -
learning_rate: 0.0010
Epoch 27/300
704/704             219s 311ms/step -
accuracy: 0.7994 - loss: 0.8634 - val_accuracy: 0.8242 - val_loss: 0.8207 -
learning_rate: 0.0010
Epoch 28/300
704/704             217s 308ms/step -
accuracy: 0.8021 - loss: 0.8539 - val_accuracy: 0.8352 - val_loss: 0.7633 -
learning_rate: 0.0010
Epoch 29/300
704/704             224s 317ms/step -
accuracy: 0.8058 - loss: 0.8391 - val_accuracy: 0.8174 - val_loss: 0.8196 -
learning_rate: 0.0010
Epoch 30/300
704/704             217s 308ms/step -
accuracy: 0.8020 - loss: 0.8495 - val_accuracy: 0.8266 - val_loss: 0.8085 -
learning_rate: 0.0010
Epoch 31/300
704/704             217s 308ms/step -
accuracy: 0.8104 - loss: 0.8281 - val_accuracy: 0.8320 - val_loss: 0.7780 -
learning_rate: 0.0010
Epoch 32/300
704/704             219s 311ms/step -
accuracy: 0.8065 - loss: 0.8429 - val_accuracy: 0.8310 - val_loss: 0.7722 -
learning_rate: 0.0010
Epoch 33/300
704/704             220s 312ms/step -
accuracy: 0.8117 - loss: 0.8247 - val_accuracy: 0.8318 - val_loss: 0.7791 -
learning_rate: 0.0010
Epoch 34/300
704/704             212s 300ms/step -
accuracy: 0.8132 - loss: 0.8184 - val_accuracy: 0.8474 - val_loss: 0.7269 -
learning_rate: 0.0010
Epoch 35/300
704/704             230s 327ms/step -
accuracy: 0.8126 - loss: 0.8169 - val_accuracy: 0.8140 - val_loss: 0.8086 -
learning_rate: 0.0010
Epoch 36/300
704/704             234s 332ms/step -
```

```
accuracy: 0.8190 - loss: 0.8080 - val_accuracy: 0.8288 - val_loss: 0.7843 -
learning_rate: 0.0010
Epoch 37/300
704/704             206s 292ms/step -
accuracy: 0.8175 - loss: 0.8130 - val_accuracy: 0.8368 - val_loss: 0.7696 -
learning_rate: 0.0010
Epoch 38/300
704/704             207s 294ms/step -
accuracy: 0.8204 - loss: 0.7988 - val_accuracy: 0.8450 - val_loss: 0.7274 -
learning_rate: 0.0010
Epoch 39/300
704/704             216s 307ms/step -
accuracy: 0.8167 - loss: 0.8060 - val_accuracy: 0.8530 - val_loss: 0.7068 -
learning_rate: 0.0010
Epoch 40/300
704/704             192s 272ms/step -
accuracy: 0.8189 - loss: 0.7895 - val_accuracy: 0.8398 - val_loss: 0.7440 -
learning_rate: 0.0010
Epoch 41/300
704/704             217s 308ms/step -
accuracy: 0.8209 - loss: 0.7926 - val_accuracy: 0.8546 - val_loss: 0.6921 -
learning_rate: 0.0010
Epoch 42/300
704/704             239s 339ms/step -
accuracy: 0.8196 - loss: 0.7865 - val_accuracy: 0.8380 - val_loss: 0.7318 -
learning_rate: 0.0010
Epoch 43/300
704/704             220s 312ms/step -
accuracy: 0.8224 - loss: 0.7809 - val_accuracy: 0.8570 - val_loss: 0.6928 -
learning_rate: 0.0010
Epoch 44/300
704/704             213s 302ms/step -
accuracy: 0.8220 - loss: 0.7845 - val_accuracy: 0.8350 - val_loss: 0.7819 -
learning_rate: 0.0010
Epoch 45/300
704/704             197s 280ms/step -
accuracy: 0.8252 - loss: 0.7789 - val_accuracy: 0.8538 - val_loss: 0.6973 -
learning_rate: 0.0010
Epoch 46/300
704/704             196s 278ms/step -
accuracy: 0.8233 - loss: 0.7845 - val_accuracy: 0.8460 - val_loss: 0.7322 -
learning_rate: 0.0010
Epoch 47/300
704/704             205s 291ms/step -
accuracy: 0.8263 - loss: 0.7728 - val_accuracy: 0.8552 - val_loss: 0.7019 -
learning_rate: 0.0010
Epoch 48/300
704/704             207s 294ms/step -
```

accuracy: 0.8264 - loss: 0.7742 - val_accuracy: 0.8380 - val_loss: 0.7416 -
learning_rate: 0.0010
Epoch 49/300
704/704 **211s** 299ms/step -
accuracy: 0.8290 - loss: 0.7570 - val_accuracy: 0.8512 - val_loss: 0.7007 -
learning_rate: 0.0010
Epoch 50/300
704/704 **209s** 297ms/step -
accuracy: 0.8266 - loss: 0.7740 - val_accuracy: 0.8390 - val_loss: 0.7608 -
learning_rate: 0.0010
Epoch 51/300
704/704 **230s** 326ms/step -
accuracy: 0.8295 - loss: 0.7664 - val_accuracy: 0.8456 - val_loss: 0.7248 -
learning_rate: 0.0010
Epoch 52/300
704/704 **265s** 376ms/step -
accuracy: 0.8400 - loss: 0.7318 - val_accuracy: 0.8678 - val_loss: 0.6342 -
learning_rate: 5.0000e-04
Epoch 53/300
704/704 **264s** 374ms/step -
accuracy: 0.8514 - loss: 0.6886 - val_accuracy: 0.8700 - val_loss: 0.6298 -
learning_rate: 5.0000e-04
Epoch 54/300
704/704 **209s** 296ms/step -
accuracy: 0.8529 - loss: 0.6705 - val_accuracy: 0.8704 - val_loss: 0.6146 -
learning_rate: 5.0000e-04
Epoch 55/300
704/704 **206s** 292ms/step -
accuracy: 0.8557 - loss: 0.6546 - val_accuracy: 0.8664 - val_loss: 0.6237 -
learning_rate: 5.0000e-04
Epoch 56/300
704/704 **199s** 283ms/step -
accuracy: 0.8554 - loss: 0.6581 - val_accuracy: 0.8720 - val_loss: 0.6067 -
learning_rate: 5.0000e-04
Epoch 57/300
704/704 **196s** 279ms/step -
accuracy: 0.8578 - loss: 0.6373 - val_accuracy: 0.8746 - val_loss: 0.5855 -
learning_rate: 5.0000e-04
Epoch 58/300
704/704 **199s** 282ms/step -
accuracy: 0.8572 - loss: 0.6366 - val_accuracy: 0.8688 - val_loss: 0.6040 -
learning_rate: 5.0000e-04
Epoch 59/300
704/704 **311s** 442ms/step -
accuracy: 0.8586 - loss: 0.6304 - val_accuracy: 0.8596 - val_loss: 0.6374 -
learning_rate: 5.0000e-04
Epoch 60/300
704/704 **401s** 568ms/step -

```
accuracy: 0.8571 - loss: 0.6237 - val_accuracy: 0.8688 - val_loss: 0.6159 -
learning_rate: 5.0000e-04
Epoch 61/300
704/704             296s 420ms/step -
accuracy: 0.8598 - loss: 0.6233 - val_accuracy: 0.8690 - val_loss: 0.5969 -
learning_rate: 5.0000e-04
Epoch 62/300
704/704             240s 341ms/step -
accuracy: 0.8620 - loss: 0.6116 - val_accuracy: 0.8732 - val_loss: 0.5929 -
learning_rate: 5.0000e-04
Epoch 63/300
704/704             255s 362ms/step -
accuracy: 0.8632 - loss: 0.6026 - val_accuracy: 0.8782 - val_loss: 0.5756 -
learning_rate: 5.0000e-04
Epoch 64/300
704/704             356s 505ms/step -
accuracy: 0.8645 - loss: 0.6010 - val_accuracy: 0.8844 - val_loss: 0.5431 -
learning_rate: 5.0000e-04
Epoch 65/300
704/704             266s 377ms/step -
accuracy: 0.8648 - loss: 0.5991 - val_accuracy: 0.8756 - val_loss: 0.5817 -
learning_rate: 5.0000e-04
Epoch 66/300
704/704             245s 348ms/step -
accuracy: 0.8652 - loss: 0.5991 - val_accuracy: 0.8742 - val_loss: 0.5846 -
learning_rate: 5.0000e-04
Epoch 67/300
704/704             231s 328ms/step -
accuracy: 0.8640 - loss: 0.5915 - val_accuracy: 0.8690 - val_loss: 0.5822 -
learning_rate: 5.0000e-04
Epoch 68/300
704/704             246s 349ms/step -
accuracy: 0.8657 - loss: 0.5881 - val_accuracy: 0.8774 - val_loss: 0.5693 -
learning_rate: 5.0000e-04
Epoch 69/300
704/704             236s 335ms/step -
accuracy: 0.8673 - loss: 0.5918 - val_accuracy: 0.8738 - val_loss: 0.5787 -
learning_rate: 5.0000e-04
Epoch 70/300
704/704             239s 338ms/step -
accuracy: 0.8666 - loss: 0.5850 - val_accuracy: 0.8726 - val_loss: 0.5922 -
learning_rate: 5.0000e-04
Epoch 71/300
704/704             232s 329ms/step -
accuracy: 0.8642 - loss: 0.5855 - val_accuracy: 0.8728 - val_loss: 0.5840 -
learning_rate: 5.0000e-04
Epoch 72/300
704/704             241s 342ms/step -
```

accuracy: 0.8639 - loss: 0.5868 - val_accuracy: 0.8814 - val_loss: 0.5457 -
learning_rate: 5.0000e-04
Epoch 73/300
704/704 **236s** 335ms/step -
accuracy: 0.8652 - loss: 0.5867 - val_accuracy: 0.8664 - val_loss: 0.5951 -
learning_rate: 5.0000e-04
Epoch 74/300
704/704 **235s** 333ms/step -
accuracy: 0.8660 - loss: 0.5853 - val_accuracy: 0.8716 - val_loss: 0.5889 -
learning_rate: 5.0000e-04
Epoch 75/300
704/704 **255s** 362ms/step -
accuracy: 0.8712 - loss: 0.5638 - val_accuracy: 0.8852 - val_loss: 0.5415 -
learning_rate: 2.5000e-04
Epoch 76/300
704/704 **255s** 361ms/step -
accuracy: 0.8795 - loss: 0.5450 - val_accuracy: 0.8826 - val_loss: 0.5328 -
learning_rate: 2.5000e-04
Epoch 77/300
704/704 **273s** 388ms/step -
accuracy: 0.8810 - loss: 0.5315 - val_accuracy: 0.8910 - val_loss: 0.5198 -
learning_rate: 2.5000e-04
Epoch 78/300
704/704 **290s** 411ms/step -
accuracy: 0.8799 - loss: 0.5271 - val_accuracy: 0.8892 - val_loss: 0.5250 -
learning_rate: 2.5000e-04
Epoch 79/300
704/704 **262s** 372ms/step -
accuracy: 0.8804 - loss: 0.5216 - val_accuracy: 0.8824 - val_loss: 0.5407 -
learning_rate: 2.5000e-04
Epoch 80/300
704/704 **246s** 349ms/step -
accuracy: 0.8832 - loss: 0.5142 - val_accuracy: 0.8832 - val_loss: 0.5255 -
learning_rate: 2.5000e-04
Epoch 81/300
704/704 **266s** 377ms/step -
accuracy: 0.8842 - loss: 0.5090 - val_accuracy: 0.8892 - val_loss: 0.5079 -
learning_rate: 2.5000e-04
Epoch 82/300
704/704 **241s** 341ms/step -
accuracy: 0.8838 - loss: 0.5095 - val_accuracy: 0.8840 - val_loss: 0.5228 -
learning_rate: 2.5000e-04
Epoch 83/300
704/704 **241s** 340ms/step -
accuracy: 0.8870 - loss: 0.4990 - val_accuracy: 0.8872 - val_loss: 0.5118 -
learning_rate: 2.5000e-04
Epoch 84/300
704/704 **239s** 340ms/step -

```
accuracy: 0.8886 - loss: 0.4911 - val_accuracy: 0.8812 - val_loss: 0.5378 -
learning_rate: 2.5000e-04
Epoch 85/300
704/704            242s 343ms/step -
accuracy: 0.8863 - loss: 0.4859 - val_accuracy: 0.8928 - val_loss: 0.5024 -
learning_rate: 2.5000e-04
Epoch 86/300
704/704            267s 380ms/step -
accuracy: 0.8866 - loss: 0.4916 - val_accuracy: 0.8920 - val_loss: 0.5026 -
learning_rate: 2.5000e-04
Epoch 87/300
704/704            250s 355ms/step -
accuracy: 0.8865 - loss: 0.4946 - val_accuracy: 0.8946 - val_loss: 0.4759 -
learning_rate: 2.5000e-04
Epoch 88/300
704/704            229s 325ms/step -
accuracy: 0.8825 - loss: 0.4983 - val_accuracy: 0.8924 - val_loss: 0.4971 -
learning_rate: 2.5000e-04
Epoch 89/300
704/704            230s 326ms/step -
accuracy: 0.8889 - loss: 0.4829 - val_accuracy: 0.8840 - val_loss: 0.5184 -
learning_rate: 2.5000e-04
Epoch 90/300
704/704            250s 354ms/step -
accuracy: 0.8910 - loss: 0.4789 - val_accuracy: 0.8952 - val_loss: 0.4753 -
learning_rate: 2.5000e-04
Epoch 91/300
704/704            282s 400ms/step -
accuracy: 0.8890 - loss: 0.4829 - val_accuracy: 0.8912 - val_loss: 0.4944 -
learning_rate: 2.5000e-04
Epoch 92/300
704/704            274s 389ms/step -
accuracy: 0.8895 - loss: 0.4807 - val_accuracy: 0.8906 - val_loss: 0.5134 -
learning_rate: 2.5000e-04
Epoch 93/300
704/704            229s 326ms/step -
accuracy: 0.8905 - loss: 0.4742 - val_accuracy: 0.8974 - val_loss: 0.4687 -
learning_rate: 2.5000e-04
Epoch 94/300
704/704            265s 376ms/step -
accuracy: 0.8913 - loss: 0.4728 - val_accuracy: 0.8944 - val_loss: 0.4761 -
learning_rate: 2.5000e-04
Epoch 95/300
704/704            261s 371ms/step -
accuracy: 0.8908 - loss: 0.4767 - val_accuracy: 0.8874 - val_loss: 0.4912 -
learning_rate: 2.5000e-04
Epoch 96/300
704/704            225s 320ms/step -
```

accuracy: 0.8907 - loss: 0.4703 - val_accuracy: 0.8918 - val_loss: 0.4967 -
learning_rate: 2.5000e-04
Epoch 97/300
704/704 **242s** 344ms/step -
accuracy: 0.8894 - loss: 0.4733 - val_accuracy: 0.8910 - val_loss: 0.4878 -
learning_rate: 2.5000e-04
Epoch 98/300
704/704 **242s** 343ms/step -
accuracy: 0.8895 - loss: 0.4720 - val_accuracy: 0.8898 - val_loss: 0.5073 -
learning_rate: 2.5000e-04
Epoch 99/300
704/704 **243s** 344ms/step -
accuracy: 0.8896 - loss: 0.4733 - val_accuracy: 0.8896 - val_loss: 0.4911 -
learning_rate: 2.5000e-04
Epoch 100/300
704/704 **205s** 291ms/step -
accuracy: 0.8885 - loss: 0.4731 - val_accuracy: 0.8858 - val_loss: 0.5016 -
learning_rate: 2.5000e-04
Epoch 101/300
704/704 **221s** 314ms/step -
accuracy: 0.8927 - loss: 0.4663 - val_accuracy: 0.8946 - val_loss: 0.4724 -
learning_rate: 2.5000e-04
Epoch 102/300
704/704 **206s** 292ms/step -
accuracy: 0.8908 - loss: 0.4641 - val_accuracy: 0.8926 - val_loss: 0.4803 -
learning_rate: 2.5000e-04
Epoch 103/300
704/704 **206s** 292ms/step -
accuracy: 0.8897 - loss: 0.4663 - val_accuracy: 0.8848 - val_loss: 0.5063 -
learning_rate: 2.5000e-04
Epoch 104/300
704/704 **203s** 288ms/step -
accuracy: 0.8932 - loss: 0.4596 - val_accuracy: 0.8956 - val_loss: 0.4675 -
learning_rate: 1.2500e-04
Epoch 105/300
704/704 **256s** 364ms/step -
accuracy: 0.8985 - loss: 0.4428 - val_accuracy: 0.8984 - val_loss: 0.4804 -
learning_rate: 1.2500e-04
Epoch 106/300
704/704 **267s** 380ms/step -
accuracy: 0.8987 - loss: 0.4375 - val_accuracy: 0.8950 - val_loss: 0.4905 -
learning_rate: 1.2500e-04
Epoch 107/300
704/704 **240s** 341ms/step -
accuracy: 0.8992 - loss: 0.4323 - val_accuracy: 0.8986 - val_loss: 0.4660 -
learning_rate: 1.2500e-04
Epoch 108/300
704/704 **193s** 273ms/step -

```
accuracy: 0.8988 - loss: 0.4337 - val_accuracy: 0.8974 - val_loss: 0.4628 -
learning_rate: 1.2500e-04
Epoch 109/300
704/704            201s 285ms/step -
accuracy: 0.9011 - loss: 0.4258 - val_accuracy: 0.8970 - val_loss: 0.4682 -
learning_rate: 1.2500e-04
Epoch 110/300
704/704            206s 293ms/step -
accuracy: 0.8989 - loss: 0.4264 - val_accuracy: 0.8950 - val_loss: 0.4755 -
learning_rate: 1.2500e-04
Epoch 111/300
704/704            206s 293ms/step -
accuracy: 0.9018 - loss: 0.4191 - val_accuracy: 0.8992 - val_loss: 0.4570 -
learning_rate: 1.2500e-04
Epoch 112/300
704/704            207s 294ms/step -
accuracy: 0.9039 - loss: 0.4157 - val_accuracy: 0.9024 - val_loss: 0.4463 -
learning_rate: 1.2500e-04
Epoch 113/300
704/704            194s 275ms/step -
accuracy: 0.9023 - loss: 0.4144 - val_accuracy: 0.9020 - val_loss: 0.4422 -
learning_rate: 1.2500e-04
Epoch 114/300
704/704            243s 346ms/step -
accuracy: 0.9032 - loss: 0.4171 - val_accuracy: 0.9016 - val_loss: 0.4446 -
learning_rate: 1.2500e-04
Epoch 115/300
704/704            220s 312ms/step -
accuracy: 0.9057 - loss: 0.4072 - val_accuracy: 0.8952 - val_loss: 0.4677 -
learning_rate: 1.2500e-04
Epoch 116/300
704/704            238s 338ms/step -
accuracy: 0.9053 - loss: 0.4125 - val_accuracy: 0.9046 - val_loss: 0.4332 -
learning_rate: 1.2500e-04
Epoch 117/300
704/704            226s 321ms/step -
accuracy: 0.9042 - loss: 0.4076 - val_accuracy: 0.8992 - val_loss: 0.4549 -
learning_rate: 1.2500e-04
Epoch 118/300
704/704            264s 375ms/step -
accuracy: 0.9068 - loss: 0.4018 - val_accuracy: 0.8998 - val_loss: 0.4573 -
learning_rate: 1.2500e-04
Epoch 119/300
704/704            249s 354ms/step -
accuracy: 0.9052 - loss: 0.4081 - val_accuracy: 0.9034 - val_loss: 0.4361 -
learning_rate: 1.2500e-04
Epoch 120/300
704/704            268s 380ms/step -
```

```
accuracy: 0.9071 - loss: 0.3989 - val_accuracy: 0.8982 - val_loss: 0.4467 -
learning_rate: 1.2500e-04
Epoch 121/300
704/704           351s 498ms/step -
accuracy: 0.9047 - loss: 0.4042 - val_accuracy: 0.9022 - val_loss: 0.4304 -
learning_rate: 1.2500e-04
Epoch 122/300
704/704           324s 460ms/step -
accuracy: 0.9110 - loss: 0.3926 - val_accuracy: 0.9044 - val_loss: 0.4397 -
learning_rate: 1.2500e-04
Epoch 123/300
704/704           269s 382ms/step -
accuracy: 0.9049 - loss: 0.4036 - val_accuracy: 0.9022 - val_loss: 0.4355 -
learning_rate: 1.2500e-04
Epoch 124/300
704/704           264s 375ms/step -
accuracy: 0.9076 - loss: 0.3996 - val_accuracy: 0.9012 - val_loss: 0.4366 -
learning_rate: 1.2500e-04
Epoch 125/300
704/704           208s 295ms/step -
accuracy: 0.9035 - loss: 0.4015 - val_accuracy: 0.8984 - val_loss: 0.4441 -
learning_rate: 1.2500e-04
Epoch 126/300
704/704           209s 296ms/step -
accuracy: 0.9103 - loss: 0.3894 - val_accuracy: 0.8972 - val_loss: 0.4479 -
learning_rate: 1.2500e-04
Epoch 127/300
704/704           194s 275ms/step -
accuracy: 0.9065 - loss: 0.3960 - val_accuracy: 0.9008 - val_loss: 0.4334 -
learning_rate: 1.2500e-04
Epoch 128/300
704/704           193s 274ms/step -
accuracy: 0.9099 - loss: 0.3884 - val_accuracy: 0.8982 - val_loss: 0.4401 -
learning_rate: 1.2500e-04
Epoch 129/300
704/704           203s 287ms/step -
accuracy: 0.9083 - loss: 0.3914 - val_accuracy: 0.9056 - val_loss: 0.4210 -
learning_rate: 1.2500e-04
Epoch 130/300
704/704           200s 284ms/step -
accuracy: 0.9076 - loss: 0.3898 - val_accuracy: 0.9050 - val_loss: 0.4351 -
learning_rate: 1.2500e-04
Epoch 131/300
704/704           204s 289ms/step -
accuracy: 0.9077 - loss: 0.3907 - val_accuracy: 0.8974 - val_loss: 0.4500 -
learning_rate: 1.2500e-04
Epoch 132/300
704/704           202s 287ms/step -
```

```
accuracy: 0.9099 - loss: 0.3932 - val_accuracy: 0.8996 - val_loss: 0.4451 -
learning_rate: 1.2500e-04
Epoch 133/300
704/704             205s 291ms/step -
accuracy: 0.9047 - loss: 0.3993 - val_accuracy: 0.9010 - val_loss: 0.4428 -
learning_rate: 1.2500e-04
Epoch 134/300
704/704             206s 292ms/step -
accuracy: 0.9088 - loss: 0.3879 - val_accuracy: 0.9002 - val_loss: 0.4460 -
learning_rate: 1.2500e-04
Epoch 135/300
704/704             266s 378ms/step -
accuracy: 0.9049 - loss: 0.3935 - val_accuracy: 0.9038 - val_loss: 0.4337 -
learning_rate: 1.2500e-04
Epoch 136/300
704/704             266s 377ms/step -
accuracy: 0.9091 - loss: 0.3840 - val_accuracy: 0.9014 - val_loss: 0.4336 -
learning_rate: 1.2500e-04
Epoch 137/300
704/704             226s 321ms/step -
accuracy: 0.9098 - loss: 0.3834 - val_accuracy: 0.9004 - val_loss: 0.4427 -
learning_rate: 1.2500e-04
Epoch 138/300
704/704             195s 277ms/step -
accuracy: 0.9097 - loss: 0.3807 - val_accuracy: 0.8992 - val_loss: 0.4445 -
learning_rate: 1.2500e-04
Epoch 139/300
704/704             192s 273ms/step -
accuracy: 0.9073 - loss: 0.3891 - val_accuracy: 0.8974 - val_loss: 0.4433 -
learning_rate: 1.2500e-04
Epoch 140/300
704/704             193s 274ms/step -
accuracy: 0.9073 - loss: 0.3870 - val_accuracy: 0.8986 - val_loss: 0.4444 -
learning_rate: 6.2500e-05
Epoch 141/300
704/704             192s 272ms/step -
accuracy: 0.9132 - loss: 0.3696 - val_accuracy: 0.9060 - val_loss: 0.4116 -
learning_rate: 6.2500e-05
Epoch 142/300
704/704             203s 288ms/step -
accuracy: 0.9101 - loss: 0.3702 - val_accuracy: 0.9014 - val_loss: 0.4293 -
learning_rate: 6.2500e-05
Epoch 143/300
704/704             197s 279ms/step -
accuracy: 0.9151 - loss: 0.3671 - val_accuracy: 0.9056 - val_loss: 0.4208 -
learning_rate: 6.2500e-05
Epoch 144/300
704/704             199s 283ms/step -
```

```
accuracy: 0.9176 - loss: 0.3617 - val_accuracy: 0.9040 - val_loss: 0.4257 -
learning_rate: 6.2500e-05
Epoch 145/300
704/704              200s 284ms/step -
accuracy: 0.9156 - loss: 0.3649 - val_accuracy: 0.9066 - val_loss: 0.4215 -
learning_rate: 6.2500e-05
Epoch 146/300
704/704              197s 280ms/step -
accuracy: 0.9158 - loss: 0.3572 - val_accuracy: 0.9060 - val_loss: 0.4192 -
learning_rate: 6.2500e-05
Epoch 147/300
704/704              234s 332ms/step -
accuracy: 0.9143 - loss: 0.3613 - val_accuracy: 0.9050 - val_loss: 0.4194 -
learning_rate: 6.2500e-05
Epoch 148/300
704/704              243s 344ms/step -
accuracy: 0.9149 - loss: 0.3640 - val_accuracy: 0.9054 - val_loss: 0.4104 -
learning_rate: 6.2500e-05
Epoch 149/300
704/704              214s 303ms/step -
accuracy: 0.9162 - loss: 0.3580 - val_accuracy: 0.9070 - val_loss: 0.4113 -
learning_rate: 6.2500e-05
Epoch 150/300
704/704              203s 289ms/step -
accuracy: 0.9189 - loss: 0.3567 - val_accuracy: 0.9074 - val_loss: 0.4143 -
learning_rate: 6.2500e-05
Epoch 151/300
704/704              236s 334ms/step -
accuracy: 0.9151 - loss: 0.3601 - val_accuracy: 0.9078 - val_loss: 0.4104 -
learning_rate: 6.2500e-05
Epoch 152/300
704/704              248s 352ms/step -
accuracy: 0.9163 - loss: 0.3544 - val_accuracy: 0.9058 - val_loss: 0.4170 -
learning_rate: 6.2500e-05
Epoch 153/300
704/704              207s 294ms/step -
accuracy: 0.9187 - loss: 0.3527 - val_accuracy: 0.9064 - val_loss: 0.4223 -
learning_rate: 6.2500e-05
Epoch 154/300
704/704              221s 313ms/step -
accuracy: 0.9165 - loss: 0.3542 - val_accuracy: 0.9066 - val_loss: 0.4141 -
learning_rate: 6.2500e-05
Epoch 155/300
704/704              220s 312ms/step -
accuracy: 0.9158 - loss: 0.3539 - val_accuracy: 0.9088 - val_loss: 0.4016 -
learning_rate: 6.2500e-05
Epoch 156/300
704/704              211s 300ms/step -
```

accuracy: 0.9151 - loss: 0.3554 - val_accuracy: 0.9058 - val_loss: 0.4120 -
learning_rate: 6.2500e-05
Epoch 157/300
704/704 **206s** 293ms/step -
accuracy: 0.9159 - loss: 0.3558 - val_accuracy: 0.9060 - val_loss: 0.4113 -
learning_rate: 6.2500e-05
Epoch 158/300
704/704 **267s** 379ms/step -
accuracy: 0.9181 - loss: 0.3525 - val_accuracy: 0.9044 - val_loss: 0.4078 -
learning_rate: 6.2500e-05
Epoch 159/300
704/704 **256s** 362ms/step -
accuracy: 0.9202 - loss: 0.3431 - val_accuracy: 0.9068 - val_loss: 0.4116 -
learning_rate: 6.2500e-05
Epoch 160/300
704/704 **260s** 369ms/step -
accuracy: 0.9164 - loss: 0.3521 - val_accuracy: 0.9044 - val_loss: 0.4153 -
learning_rate: 6.2500e-05
Epoch 161/300
704/704 **241s** 342ms/step -
accuracy: 0.9169 - loss: 0.3491 - val_accuracy: 0.9084 - val_loss: 0.4121 -
learning_rate: 6.2500e-05
Epoch 162/300
704/704 **219s** 311ms/step -
accuracy: 0.9203 - loss: 0.3442 - val_accuracy: 0.9074 - val_loss: 0.4094 -
learning_rate: 6.2500e-05
Epoch 163/300
704/704 **214s** 303ms/step -
accuracy: 0.9158 - loss: 0.3501 - val_accuracy: 0.9062 - val_loss: 0.4046 -
learning_rate: 6.2500e-05
Epoch 164/300
704/704 **223s** 317ms/step -
accuracy: 0.9202 - loss: 0.3447 - val_accuracy: 0.9062 - val_loss: 0.4033 -
learning_rate: 6.2500e-05
Epoch 165/300
704/704 **228s** 324ms/step -
accuracy: 0.9190 - loss: 0.3429 - val_accuracy: 0.9066 - val_loss: 0.4133 -
learning_rate: 6.2500e-05
Epoch 166/300
704/704 **246s** 349ms/step -
accuracy: 0.9206 - loss: 0.3394 - val_accuracy: 0.9074 - val_loss: 0.4137 -
learning_rate: 3.1250e-05
Epoch 167/300
704/704 **220s** 312ms/step -
accuracy: 0.9217 - loss: 0.3350 - val_accuracy: 0.9104 - val_loss: 0.4051 -
learning_rate: 3.1250e-05
Epoch 168/300
704/704 **221s** 314ms/step -

```
accuracy: 0.9227 - loss: 0.3393 - val_accuracy: 0.9094 - val_loss: 0.4034 -
learning_rate: 3.1250e-05
Epoch 169/300
704/704             202s 286ms/step -
accuracy: 0.9182 - loss: 0.3417 - val_accuracy: 0.9082 - val_loss: 0.4040 -
learning_rate: 3.1250e-05
Epoch 170/300
704/704             240s 341ms/step -
accuracy: 0.9229 - loss: 0.3345 - val_accuracy: 0.9076 - val_loss: 0.4127 -
learning_rate: 3.1250e-05
Epoch 171/300
704/704             259s 368ms/step -
accuracy: 0.9190 - loss: 0.3447 - val_accuracy: 0.9070 - val_loss: 0.4044 -
learning_rate: 3.1250e-05
Epoch 172/300
704/704             236s 335ms/step -
accuracy: 0.9211 - loss: 0.3388 - val_accuracy: 0.9096 - val_loss: 0.4086 -
learning_rate: 3.1250e-05
Epoch 173/300
704/704             227s 322ms/step -
accuracy: 0.9205 - loss: 0.3370 - val_accuracy: 0.9100 - val_loss: 0.3987 -
learning_rate: 3.1250e-05
Epoch 174/300
704/704             239s 339ms/step -
accuracy: 0.9211 - loss: 0.3360 - val_accuracy: 0.9088 - val_loss: 0.4021 -
learning_rate: 3.1250e-05
Epoch 175/300
704/704             228s 324ms/step -
accuracy: 0.9227 - loss: 0.3321 - val_accuracy: 0.9108 - val_loss: 0.3932 -
learning_rate: 3.1250e-05
Epoch 176/300
704/704             230s 327ms/step -
accuracy: 0.9199 - loss: 0.3383 - val_accuracy: 0.9110 - val_loss: 0.4026 -
learning_rate: 3.1250e-05
Epoch 177/300
704/704             225s 319ms/step -
accuracy: 0.9241 - loss: 0.3296 - val_accuracy: 0.9114 - val_loss: 0.4010 -
learning_rate: 3.1250e-05
Epoch 178/300
704/704             236s 335ms/step -
accuracy: 0.9225 - loss: 0.3330 - val_accuracy: 0.9098 - val_loss: 0.4015 -
learning_rate: 3.1250e-05
Epoch 179/300
704/704             240s 341ms/step -
accuracy: 0.9261 - loss: 0.3219 - val_accuracy: 0.9090 - val_loss: 0.3958 -
learning_rate: 3.1250e-05
Epoch 180/300
704/704             247s 351ms/step -
```

```
accuracy: 0.9257 - loss: 0.3271 - val_accuracy: 0.9118 - val_loss: 0.3961 -
learning_rate: 3.1250e-05
Epoch 181/300
704/704            222s 315ms/step -
accuracy: 0.9272 - loss: 0.3222 - val_accuracy: 0.9076 - val_loss: 0.4029 -
learning_rate: 3.1250e-05
Epoch 182/300
704/704            223s 317ms/step -
accuracy: 0.9231 - loss: 0.3309 - val_accuracy: 0.9082 - val_loss: 0.3987 -
learning_rate: 3.1250e-05
Epoch 183/300
704/704            232s 329ms/step -
accuracy: 0.9234 - loss: 0.3308 - val_accuracy: 0.9058 - val_loss: 0.4108 -
learning_rate: 3.1250e-05
Epoch 184/300
704/704            248s 352ms/step -
accuracy: 0.9250 - loss: 0.3268 - val_accuracy: 0.9064 - val_loss: 0.4004 -
learning_rate: 3.1250e-05
Epoch 185/300
704/704            213s 303ms/step -
accuracy: 0.9225 - loss: 0.3291 - val_accuracy: 0.9086 - val_loss: 0.3936 -
learning_rate: 3.1250e-05
Epoch 186/300
704/704            202s 286ms/step -
accuracy: 0.9231 - loss: 0.3242 - val_accuracy: 0.9084 - val_loss: 0.3984 -
learning_rate: 1.5625e-05
Epoch 187/300
704/704            205s 291ms/step -
accuracy: 0.9225 - loss: 0.3259 - val_accuracy: 0.9106 - val_loss: 0.3947 -
learning_rate: 1.5625e-05
Epoch 188/300
704/704            208s 295ms/step -
accuracy: 0.9220 - loss: 0.3303 - val_accuracy: 0.9104 - val_loss: 0.3982 -
learning_rate: 1.5625e-05
Epoch 189/300
704/704            203s 288ms/step -
accuracy: 0.9224 - loss: 0.3280 - val_accuracy: 0.9094 - val_loss: 0.3951 -
learning_rate: 1.5625e-05
Epoch 190/300
704/704            209s 297ms/step -
accuracy: 0.9265 - loss: 0.3248 - val_accuracy: 0.9104 - val_loss: 0.3977 -
learning_rate: 1.5625e-05
Epoch 191/300
704/704            204s 290ms/step -
accuracy: 0.9241 - loss: 0.3265 - val_accuracy: 0.9102 - val_loss: 0.3955 -
learning_rate: 1.5625e-05
Epoch 192/300
704/704            202s 287ms/step -
```

accuracy: 0.9276 - loss: 0.3139 - val_accuracy: 0.9100 - val_loss: 0.3979 -
learning_rate: 1.5625e-05
Epoch 193/300
704/704 **206s** 292ms/step -
accuracy: 0.9226 - loss: 0.3298 - val_accuracy: 0.9110 - val_loss: 0.3955 -
learning_rate: 1.5625e-05
Epoch 194/300
704/704 **269s** 382ms/step -
accuracy: 0.9232 - loss: 0.3303 - val_accuracy: 0.9112 - val_loss: 0.3914 -
learning_rate: 1.5625e-05
Epoch 195/300
704/704 **279s** 396ms/step -
accuracy: 0.9234 - loss: 0.3233 - val_accuracy: 0.9104 - val_loss: 0.3982 -
learning_rate: 1.5625e-05
Epoch 196/300
704/704 **236s** 334ms/step -
accuracy: 0.9234 - loss: 0.3260 - val_accuracy: 0.9118 - val_loss: 0.3945 -
learning_rate: 1.5625e-05
Epoch 197/300
704/704 **201s** 285ms/step -
accuracy: 0.9251 - loss: 0.3211 - val_accuracy: 0.9118 - val_loss: 0.3924 -
learning_rate: 1.5625e-05
Epoch 198/300
704/704 **198s** 281ms/step -
accuracy: 0.9218 - loss: 0.3246 - val_accuracy: 0.9116 - val_loss: 0.3937 -
learning_rate: 1.5625e-05
Epoch 199/300
704/704 **199s** 282ms/step -
accuracy: 0.9237 - loss: 0.3251 - val_accuracy: 0.9112 - val_loss: 0.3938 -
learning_rate: 1.5625e-05
Epoch 200/300
704/704 **205s** 290ms/step -
accuracy: 0.9262 - loss: 0.3155 - val_accuracy: 0.9108 - val_loss: 0.3952 -
learning_rate: 1.5625e-05
Epoch 201/300
704/704 **201s** 285ms/step -
accuracy: 0.9260 - loss: 0.3171 - val_accuracy: 0.9100 - val_loss: 0.3971 -
learning_rate: 1.5625e-05
Epoch 202/300
704/704 **199s** 283ms/step -
accuracy: 0.9227 - loss: 0.3276 - val_accuracy: 0.9118 - val_loss: 0.3886 -
learning_rate: 1.5625e-05
Epoch 203/300
704/704 **200s** 285ms/step -
accuracy: 0.9262 - loss: 0.3141 - val_accuracy: 0.9104 - val_loss: 0.3963 -
learning_rate: 1.5625e-05
Epoch 204/300
704/704 **201s** 285ms/step -

accuracy: 0.9275 - loss: 0.3142 - val_accuracy: 0.9090 - val_loss: 0.3980 -
learning_rate: 1.5625e-05
Epoch 205/300
704/704 **202s** 287ms/step -
accuracy: 0.9262 - loss: 0.3205 - val_accuracy: 0.9110 - val_loss: 0.3921 -
learning_rate: 1.5625e-05
Epoch 206/300
704/704 **199s** 283ms/step -
accuracy: 0.9267 - loss: 0.3159 - val_accuracy: 0.9114 - val_loss: 0.4010 -
learning_rate: 1.5625e-05
Epoch 207/300
704/704 **202s** 286ms/step -
accuracy: 0.9268 - loss: 0.3173 - val_accuracy: 0.9108 - val_loss: 0.3982 -
learning_rate: 1.5625e-05
Epoch 208/300
704/704 **199s** 283ms/step -
accuracy: 0.9272 - loss: 0.3201 - val_accuracy: 0.9120 - val_loss: 0.3949 -
learning_rate: 1.5625e-05
Epoch 209/300
704/704 **208s** 295ms/step -
accuracy: 0.9284 - loss: 0.3126 - val_accuracy: 0.9098 - val_loss: 0.3962 -
learning_rate: 1.5625e-05
Epoch 210/300
704/704 **202s** 286ms/step -
accuracy: 0.9271 - loss: 0.3129 - val_accuracy: 0.9094 - val_loss: 0.3990 -
learning_rate: 1.5625e-05
Epoch 211/300
704/704 **224s** 317ms/step -
accuracy: 0.9282 - loss: 0.3121 - val_accuracy: 0.9108 - val_loss: 0.3932 -
learning_rate: 1.5625e-05
Epoch 212/300
704/704 **281s** 399ms/step -
accuracy: 0.9252 - loss: 0.3203 - val_accuracy: 0.9102 - val_loss: 0.3941 -
learning_rate: 1.5625e-05
Epoch 213/300
704/704 **233s** 331ms/step -
accuracy: 0.9273 - loss: 0.3162 - val_accuracy: 0.9106 - val_loss: 0.3925 -
learning_rate: 1.0000e-05
Epoch 214/300
704/704 **231s** 327ms/step -
accuracy: 0.9266 - loss: 0.3153 - val_accuracy: 0.9098 - val_loss: 0.3953 -
learning_rate: 1.0000e-05
Epoch 215/300
704/704 **227s** 322ms/step -
accuracy: 0.9274 - loss: 0.3137 - val_accuracy: 0.9116 - val_loss: 0.3897 -
learning_rate: 1.0000e-05
Epoch 216/300
704/704 **228s** 324ms/step -

accuracy: 0.9280 - loss: 0.3129 - val_accuracy: 0.9092 - val_loss: 0.3986 -
learning_rate: 1.0000e-05
Epoch 217/300
704/704 **246s** 349ms/step -
accuracy: 0.9266 - loss: 0.3163 - val_accuracy: 0.9114 - val_loss: 0.3922 -
learning_rate: 1.0000e-05
Epoch 218/300
704/704 **232s** 329ms/step -
accuracy: 0.9237 - loss: 0.3202 - val_accuracy: 0.9118 - val_loss: 0.3907 -
learning_rate: 1.0000e-05
Epoch 219/300
704/704 **230s** 326ms/step -
accuracy: 0.9278 - loss: 0.3103 - val_accuracy: 0.9098 - val_loss: 0.3958 -
learning_rate: 1.0000e-05
Epoch 220/300
704/704 **228s** 324ms/step -
accuracy: 0.9274 - loss: 0.3117 - val_accuracy: 0.9110 - val_loss: 0.3947 -
learning_rate: 1.0000e-05
Epoch 221/300
704/704 **233s** 331ms/step -
accuracy: 0.9286 - loss: 0.3104 - val_accuracy: 0.9122 - val_loss: 0.3985 -
learning_rate: 1.0000e-05
Epoch 222/300
704/704 **232s** 330ms/step -
accuracy: 0.9262 - loss: 0.3128 - val_accuracy: 0.9112 - val_loss: 0.3935 -
learning_rate: 1.0000e-05
Epoch 223/300
704/704 **235s** 334ms/step -
accuracy: 0.9276 - loss: 0.3123 - val_accuracy: 0.9102 - val_loss: 0.3945 -
learning_rate: 1.0000e-05
Epoch 224/300
704/704 **233s** 331ms/step -
accuracy: 0.9288 - loss: 0.3064 - val_accuracy: 0.9120 - val_loss: 0.3925 -
learning_rate: 1.0000e-05
Epoch 225/300
704/704 **252s** 357ms/step -
accuracy: 0.9270 - loss: 0.3150 - val_accuracy: 0.9118 - val_loss: 0.3907 -
learning_rate: 1.0000e-05
Epoch 226/300
704/704 **245s** 348ms/step -
accuracy: 0.9264 - loss: 0.3136 - val_accuracy: 0.9108 - val_loss: 0.3938 -
learning_rate: 1.0000e-05
Epoch 227/300
704/704 **243s** 345ms/step -
accuracy: 0.9264 - loss: 0.3158 - val_accuracy: 0.9104 - val_loss: 0.3918 -
learning_rate: 1.0000e-05
Epoch 228/300
704/704 **245s** 348ms/step -

```
accuracy: 0.9262 - loss: 0.3146 - val_accuracy: 0.9114 - val_loss: 0.3925 -
learning_rate: 1.0000e-05
Epoch 229/300
704/704            239s 339ms/step -
accuracy: 0.9272 - loss: 0.3116 - val_accuracy: 0.9118 - val_loss: 0.3942 -
learning_rate: 1.0000e-05
Epoch 230/300
704/704            247s 350ms/step -
accuracy: 0.9253 - loss: 0.3184 - val_accuracy: 0.9120 - val_loss: 0.3934 -
learning_rate: 1.0000e-05
Epoch 231/300
704/704            256s 363ms/step -
accuracy: 0.9297 - loss: 0.3076 - val_accuracy: 0.9114 - val_loss: 0.3987 -
learning_rate: 1.0000e-05
Epoch 232/300
704/704            255s 362ms/step -
accuracy: 0.9280 - loss: 0.3078 - val_accuracy: 0.9132 - val_loss: 0.3921 -
learning_rate: 1.0000e-05
Epoch 233/300
704/704            206s 293ms/step -
accuracy: 0.9259 - loss: 0.3173 - val_accuracy: 0.9114 - val_loss: 0.3958 -
learning_rate: 1.0000e-05
Epoch 234/300
704/704            205s 291ms/step -
accuracy: 0.9288 - loss: 0.3056 - val_accuracy: 0.9128 - val_loss: 0.3904 -
learning_rate: 1.0000e-05
Epoch 235/300
704/704            210s 298ms/step -
accuracy: 0.9276 - loss: 0.3131 - val_accuracy: 0.9098 - val_loss: 0.3934 -
learning_rate: 1.0000e-05
Epoch 236/300
704/704            207s 294ms/step -
accuracy: 0.9244 - loss: 0.3194 - val_accuracy: 0.9110 - val_loss: 0.3934 -
learning_rate: 1.0000e-05
Epoch 237/300
704/704            204s 290ms/step -
accuracy: 0.9262 - loss: 0.3121 - val_accuracy: 0.9114 - val_loss: 0.3935 -
learning_rate: 1.0000e-05
Epoch 238/300
704/704            204s 290ms/step -
accuracy: 0.9272 - loss: 0.3135 - val_accuracy: 0.9104 - val_loss: 0.3940 -
learning_rate: 1.0000e-05
Epoch 239/300
704/704            205s 291ms/step -
accuracy: 0.9282 - loss: 0.3112 - val_accuracy: 0.9102 - val_loss: 0.3915 -
learning_rate: 1.0000e-05
Epoch 240/300
704/704            214s 303ms/step -
```

```
accuracy: 0.9270 - loss: 0.3120 - val_accuracy: 0.9106 - val_loss: 0.3915 -
learning_rate: 1.0000e-05
Epoch 241/300
704/704              213s 303ms/step -
accuracy: 0.9284 - loss: 0.3111 - val_accuracy: 0.9092 - val_loss: 0.3943 -
learning_rate: 1.0000e-05
Epoch 242/300
704/704              206s 292ms/step -
accuracy: 0.9260 - loss: 0.3137 - val_accuracy: 0.9098 - val_loss: 0.3945 -
learning_rate: 1.0000e-05
Epoch 242: early stopping
Restoring model weights from the end of the best epoch: 202.
```

[16]:
```
<keras.src.callbacks.history.History at 0x3173f1a10>
```

[17]:
```python
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=1)

print('\nTest Accuracy:', test_acc)
print('Test Loss:     ', test_loss)
```

```
313/313              12s 38ms/step -
accuracy: 0.9101 - loss: 0.3975

Test Accuracy: 0.9083999991416931
Test Loss:     0.40374499559402466
```

[ ]:
```

```