scpexample

March 28, 2023

[2]:
```
# Original Dataset: https://www.cs.toronto.edu/~kriz/cifar.html for more␣
 ↪information
# # Load of necessary libraries
import numpy as np
import tensorflow.keras
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.utils import to_categorical

# Modified originial example from importing keras to using tensoflow.keras to␣
 ↪run on chapman servers
```

[3]:
```
# to make the example replicable
np.random.seed(42)# Load of the dataset
(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()
```

[4]:
```
import matplotlib.pyplot as plt
class_names = ['airplane','automobile','bird','cat','deer',
               'dog','frog','horse','ship','truck']
fig = plt.figure(figsize=(8,3))
for i in range(len(class_names)):
  ax = fig.add_subplot(2, 5, 1 + i, xticks=[], yticks=[])
  idx = np.where(Y_train[:]==i)[0]
  features_idx = X_train[idx,::]
  img_num = np.random.randint(features_idx.shape[0])
  im = features_idx[img_num,::]
  ax.set_title(class_names[i])
  #im = np.transpose(features_idx[img_num,::], (1, 2, 0))
  plt.imshow(im)
plt.show()
```

[5]:
```
# Initializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# We add our␣
 ↪classificator
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Training of the model
model.fit(X_train, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test, to_categorical(Y_test)))# Evaluation of the␣
 ↪model
scores = model.evaluate(X_test, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
2023-03-28 09:52:54.479300: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not
creating XLA devices, tf_xla_enable_xla_devices not set
2023-03-28 09:52:54.480009: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcuda.so.1
2023-03-28 09:52:55.049698: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1747] Found device 0 with
```

```
properties:
pciBusID: 0000:87:00.0 name: NVIDIA A100-SXM4-80GB computeCapability: 8.0
coreClock: 1.41GHz coreCount: 108 deviceMemorySize: 79.18GiB
deviceMemoryBandwidth: 1.85TiB/s
2023-03-28 09:52:55.049734: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.11.0
2023-03-28 09:52:55.052466: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.11
2023-03-28 09:52:55.052494: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.11
2023-03-28 09:52:55.053161: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcufft.so.10
2023-03-28 09:52:55.053336: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcurand.so.10
2023-03-28 09:52:55.053685: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusolver.so.11
2023-03-28 09:52:55.054198: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusparse.so.11
2023-03-28 09:52:55.054293: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.8
2023-03-28 09:52:55.066325: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1889] Adding visible gpu
devices: 0
2023-03-28 09:52:55.081268: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not
creating XLA devices, tf_xla_enable_xla_devices not set
2023-03-28 09:52:55.088402: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1747] Found device 0 with
properties:
pciBusID: 0000:87:00.0 name: NVIDIA A100-SXM4-80GB computeCapability: 8.0
coreClock: 1.41GHz coreCount: 108 deviceMemorySize: 79.18GiB
deviceMemoryBandwidth: 1.85TiB/s
2023-03-28 09:52:55.088428: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.11.0
2023-03-28 09:52:55.088448: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.11
2023-03-28 09:52:55.088457: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.11
```

```
2023-03-28 09:52:55.088466: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcufft.so.10
2023-03-28 09:52:55.088474: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcurand.so.10
2023-03-28 09:52:55.088482: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusolver.so.11
2023-03-28 09:52:55.088490: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusparse.so.11
2023-03-28 09:52:55.088499: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.8
2023-03-28 09:52:55.101363: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1889] Adding visible gpu
devices: 0
2023-03-28 09:52:55.101392: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.11.0
2023-03-28 09:52:55.558948: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1287] Device interconnect
StreamExecutor with strength 1 edge matrix:
2023-03-28 09:52:55.559004: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1293]      0
2023-03-28 09:52:55.559010: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1306] 0:   N
2023-03-28 09:52:55.562720: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 78934 MB memory) -> physical
GPU (device: 0, name: NVIDIA A100-SXM4-80GB, pci bus id: 0000:87:00.0, compute
capability: 8.0)
2023-03-28 09:52:56.671009: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR
optimization passes are enabled (registered 2)
2023-03-28 09:52:56.690441: I
tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency:
2245870000 Hz

Epoch 1/10

2023-03-28 09:52:56.986101: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.11
2023-03-28 09:52:57.689706: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.11
2023-03-28 09:52:57.691525: I
```

```
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.8
2023-03-28 09:52:59.287915: I tensorflow/stream_executor/cuda/cuda_blas.cc:1838]
TensorFloat-32 will be used for the matrix multiplication. This will only be
logged once.

391/391 [==============================] - 11s 19ms/step - loss: 11.4878 -
accuracy: 0.3801 - val_loss: 1.1693 - val_accuracy: 0.5904
Epoch 2/10
391/391 [==============================] - 3s 8ms/step - loss: 0.8449 -
accuracy: 0.7152 - val_loss: 1.0675 - val_accuracy: 0.6299
Epoch 3/10
391/391 [==============================] - 3s 8ms/step - loss: 0.4033 -
accuracy: 0.8735 - val_loss: 1.1808 - val_accuracy: 0.6297
Epoch 4/10
391/391 [==============================] - 3s 8ms/step - loss: 0.1364 -
accuracy: 0.9667 - val_loss: 1.3439 - val_accuracy: 0.6416
Epoch 5/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0474 -
accuracy: 0.9908 - val_loss: 1.5645 - val_accuracy: 0.6380
Epoch 6/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0250 -
accuracy: 0.9954 - val_loss: 1.6233 - val_accuracy: 0.6438
Epoch 7/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0195 -
accuracy: 0.9964 - val_loss: 1.7052 - val_accuracy: 0.6391
Epoch 8/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0291 -
accuracy: 0.9927 - val_loss: 1.7616 - val_accuracy: 0.6217
Epoch 9/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0360 -
accuracy: 0.9905 - val_loss: 1.7786 - val_accuracy: 0.6329
Epoch 10/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0220 -
accuracy: 0.9945 - val_loss: 1.9077 - val_accuracy: 0.6386
313/313 [==============================] - 0s 1ms/step - loss: 1.9077 -
accuracy: 0.6386
Loss: 1.908
Accuracy: 0.639
```

[6]:
```
# Cenetering the data
X_train_mean = np.mean(X_train, axis = 0)
X_train_cent = X_train - X_train_mean# Normalization
X_train_std = np.std(X_train, axis = 0)
X_train_norm = X_train_cent / X_train_std
```

[7]:
```
X_test_norm = (X_test - X_train_mean) / X_train_std
```

[8]:
```
# Initializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# We add our␣
 ↪classificator
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Training of the model
model.fit(X_train, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test, to_categorical(Y_test)))# Evaluation of the␣
 ↪model
scores = model.evaluate(X_test, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 3s 8ms/step - loss: 19.7605 -
accuracy: 0.3846 - val_loss: 1.2488 - val_accuracy: 0.5683
Epoch 2/10
391/391 [==============================] - 3s 8ms/step - loss: 0.9302 -
accuracy: 0.6770 - val_loss: 1.0765 - val_accuracy: 0.6269
Epoch 3/10
391/391 [==============================] - 3s 8ms/step - loss: 0.5316 -
accuracy: 0.8283 - val_loss: 1.1405 - val_accuracy: 0.6319
Epoch 4/10
391/391 [==============================] - 3s 8ms/step - loss: 0.2302 -
accuracy: 0.9336 - val_loss: 1.2439 - val_accuracy: 0.6442
Epoch 5/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0852 -
accuracy: 0.9796 - val_loss: 1.5134 - val_accuracy: 0.6430
Epoch 6/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0367 -
accuracy: 0.9926 - val_loss: 1.7248 - val_accuracy: 0.6355
Epoch 7/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0265 -
accuracy: 0.9948 - val_loss: 1.8047 - val_accuracy: 0.6401
Epoch 8/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0326 -
```

```
accuracy: 0.9920 - val_loss: 1.9313 - val_accuracy: 0.6269
Epoch 9/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0372 -
accuracy: 0.9899 - val_loss: 1.9774 - val_accuracy: 0.6331
Epoch 10/10
391/391 [==============================] - 3s 8ms/step - loss: 0.0358 -
accuracy: 0.9890 - val_loss: 2.0509 - val_accuracy: 0.6266
313/313 [==============================] - 1s 2ms/step - loss: 2.0509 -
accuracy: 0.6266
Loss: 2.051
Accuracy: 0.627
```

[9]:
```
# We Import Batch Normalizarion layer
from tensorflow.keras.layers import BatchNormalization, Activation#␣
 ↪Inizializting the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), input_shape=(32, 32, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Activation('relu'))# Defining a thirdd convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Activation('relu'))# We include our classifier
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Training the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test))) # aquí␣
 ↪deberíamos usar un conjunto distinto al de test!!!# Evaluating the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 12ms/step - loss: 2.3247 -
accuracy: 0.4261 - val_loss: 1.2374 - val_accuracy: 0.6088
Epoch 2/10
391/391 [==============================] - 4s 10ms/step - loss: 0.6996 -
accuracy: 0.7663 - val_loss: 1.0293 - val_accuracy: 0.6440
Epoch 3/10
```

```
391/391 [==============================] - 4s 10ms/step - loss: 0.3008 -
accuracy: 0.9254 - val_loss: 0.9748 - val_accuracy: 0.6725
Epoch 4/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0957 -
accuracy: 0.9890 - val_loss: 1.0104 - val_accuracy: 0.6808
Epoch 5/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0266 -
accuracy: 0.9996 - val_loss: 1.0112 - val_accuracy: 0.6970
Epoch 6/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0102 -
accuracy: 1.0000 - val_loss: 1.0470 - val_accuracy: 0.7028
Epoch 7/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0051 -
accuracy: 1.0000 - val_loss: 1.0802 - val_accuracy: 0.7053
Epoch 8/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0032 -
accuracy: 1.0000 - val_loss: 1.1110 - val_accuracy: 0.7042
Epoch 9/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0020 -
accuracy: 1.0000 - val_loss: 1.1395 - val_accuracy: 0.7051
Epoch 10/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0014 -
accuracy: 1.0000 - val_loss: 1.1627 - val_accuracy: 0.7072
313/313 [==============================] - 1s 2ms/step - loss: 1.1627 -
accuracy: 0.7073
Loss: 1.163
Accuracy: 0.707
```

[10]:
```
# L2 Regularization# Regularizer layer import
from tensorflow.keras.regularizers import l2# Inizializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Classifier␣
 ↪inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_regularizer=l2(0.01)))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
```

```
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 11ms/step - loss: 7.5383 -
accuracy: 0.3908 - val_loss: 1.5495 - val_accuracy: 0.5265
Epoch 2/10
391/391 [==============================] - 4s 11ms/step - loss: 1.4810 -
accuracy: 0.5539 - val_loss: 1.4154 - val_accuracy: 0.5722
Epoch 3/10
391/391 [==============================] - 4s 11ms/step - loss: 1.3562 -
accuracy: 0.6021 - val_loss: 1.2802 - val_accuracy: 0.6307
Epoch 4/10
391/391 [==============================] - 4s 11ms/step - loss: 1.2698 -
accuracy: 0.6355 - val_loss: 1.2694 - val_accuracy: 0.6336
Epoch 5/10
391/391 [==============================] - 4s 11ms/step - loss: 1.2059 -
accuracy: 0.6550 - val_loss: 1.2265 - val_accuracy: 0.6432
Epoch 6/10
391/391 [==============================] - 4s 11ms/step - loss: 1.1643 -
accuracy: 0.6737 - val_loss: 1.1851 - val_accuracy: 0.6652
Epoch 7/10
391/391 [==============================] - 4s 11ms/step - loss: 1.1176 -
accuracy: 0.6948 - val_loss: 1.1342 - val_accuracy: 0.6835
Epoch 8/10
391/391 [==============================] - 4s 11ms/step - loss: 1.0953 -
accuracy: 0.7010 - val_loss: 1.1472 - val_accuracy: 0.6773
Epoch 9/10
391/391 [==============================] - 4s 11ms/step - loss: 1.0668 -
accuracy: 0.7107 - val_loss: 1.1288 - val_accuracy: 0.6872
Epoch 10/10
391/391 [==============================] - 5s 12ms/step - loss: 1.0247 -
accuracy: 0.7280 - val_loss: 1.1020 - val_accuracy: 0.6994
313/313 [==============================] - 1s 2ms/step - loss: 1.1020 -
accuracy: 0.6994
Loss: 1.102
Accuracy: 0.699
```

[24]:
```
# Batch Normalizarion + L2 Regularization
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), input_shape=(32, 32, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
```

```
model.add(BatchNormalization())
model.add(Activation('relu'))# Defining a thirdd convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Activation('relu'))# We include our classifier
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_regularizer=l2(0.01)))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Training the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test))) # aquí␣
 ↪deberíamos usar un conjunto distinto al de test!!!# Evaluating the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 6s 13ms/step - loss: 16.3932 -
accuracy: 0.4344 - val_loss: 4.9654 - val_accuracy: 0.6098
Epoch 2/10
391/391 [==============================] - 5s 12ms/step - loss: 3.4965 -
accuracy: 0.7548 - val_loss: 2.1779 - val_accuracy: 0.6663
Epoch 3/10
391/391 [==============================] - 5s 12ms/step - loss: 1.4450 -
accuracy: 0.8838 - val_loss: 1.7073 - val_accuracy: 0.6572
Epoch 4/10
391/391 [==============================] - 5s 12ms/step - loss: 0.9340 -
accuracy: 0.9422 - val_loss: 1.5223 - val_accuracy: 0.6864
Epoch 5/10
391/391 [==============================] - 5s 12ms/step - loss: 0.7544 -
accuracy: 0.9597 - val_loss: 1.5374 - val_accuracy: 0.6800
Epoch 6/10
391/391 [==============================] - 5s 12ms/step - loss: 0.7063 -
accuracy: 0.9656 - val_loss: 1.5613 - val_accuracy: 0.6788
Epoch 7/10
391/391 [==============================] - 5s 12ms/step - loss: 0.6974 -
accuracy: 0.9660 - val_loss: 1.5139 - val_accuracy: 0.6838
Epoch 8/10
391/391 [==============================] - 5s 12ms/step - loss: 0.6315 -
accuracy: 0.9712 - val_loss: 1.6310 - val_accuracy: 0.6695
Epoch 9/10
391/391 [==============================] - 5s 13ms/step - loss: 0.6497 -
```

```
accuracy: 0.9702 - val_loss: 1.5889 - val_accuracy: 0.6677
Epoch 10/10
391/391 [==============================] - 5s 12ms/step - loss: 0.5783 -
accuracy: 0.9745 - val_loss: 1.5475 - val_accuracy: 0.6754
313/313 [==============================] - 1s 2ms/step - loss: 1.5475 -
accuracy: 0.6754
Loss: 1.547
Accuracy: 0.675
```

[28]:
```
# L1 Regularization
# Regularizer layer import
from tensorflow.keras.regularizers import l1# Inizializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Classifier␣
 ↪inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_regularizer=l1(0.01)))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 12ms/step - loss: 910.8077 -
accuracy: 0.2171 - val_loss: 12.8589 - val_accuracy: 0.2750
Epoch 2/10
391/391 [==============================] - 4s 11ms/step - loss: 12.6013 -
accuracy: 0.2973 - val_loss: 12.2352 - val_accuracy: 0.3093
Epoch 3/10
391/391 [==============================] - 4s 11ms/step - loss: 12.1133 -
accuracy: 0.3482 - val_loss: 12.0791 - val_accuracy: 0.3580
Epoch 4/10
391/391 [==============================] - 4s 11ms/step - loss: 11.9664 -
accuracy: 0.3759 - val_loss: 12.0287 - val_accuracy: 0.3716
```

```
Epoch 5/10
391/391 [==============================] - 4s 11ms/step - loss: 11.8663 -
accuracy: 0.3987 - val_loss: 11.7456 - val_accuracy: 0.3903
Epoch 6/10
391/391 [==============================] - 4s 11ms/step - loss: 11.6739 -
accuracy: 0.4196 - val_loss: 11.5418 - val_accuracy: 0.4259
Epoch 7/10
391/391 [==============================] - 4s 11ms/step - loss: 11.5140 -
accuracy: 0.4322 - val_loss: 11.4020 - val_accuracy: 0.4401
Epoch 8/10
391/391 [==============================] - 4s 11ms/step - loss: 11.4058 -
accuracy: 0.4443 - val_loss: 11.3827 - val_accuracy: 0.4214
Epoch 9/10
391/391 [==============================] - 4s 11ms/step - loss: 11.3109 -
accuracy: 0.4468 - val_loss: 11.2783 - val_accuracy: 0.4569
Epoch 10/10
391/391 [==============================] - 4s 11ms/step - loss: 11.2459 -
accuracy: 0.4537 - val_loss: 11.1702 - val_accuracy: 0.4666
313/313 [==============================] - 1s 2ms/step - loss: 11.1702 -
accuracy: 0.4666
Loss: 11.170
Accuracy: 0.467
```

[29]:
```
# Elastic Net Regularization (L1 + L2)
# Regularizer layer import
from tensorflow.keras.regularizers import l1_l2# Inizializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Classifier␣
 ↪inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_regularizer=l1_l2(0.01, 0.01)))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
```

```
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 6s 13ms/step - loss: 915.2005 -
accuracy: 0.2150 - val_loss: 12.7535 - val_accuracy: 0.2899
Epoch 2/10
391/391 [==============================] - 5s 13ms/step - loss: 12.5977 -
accuracy: 0.2925 - val_loss: 12.2359 - val_accuracy: 0.3512
Epoch 3/10
391/391 [==============================] - 5s 13ms/step - loss: 12.1759 -
accuracy: 0.3544 - val_loss: 12.0488 - val_accuracy: 0.3820
Epoch 4/10
391/391 [==============================] - 5s 13ms/step - loss: 12.0097 -
accuracy: 0.3735 - val_loss: 11.8079 - val_accuracy: 0.3849
Epoch 5/10
391/391 [==============================] - 5s 13ms/step - loss: 11.7471 -
accuracy: 0.4061 - val_loss: 11.5707 - val_accuracy: 0.4050
Epoch 6/10
391/391 [==============================] - 5s 13ms/step - loss: 11.5527 -
accuracy: 0.4246 - val_loss: 11.5193 - val_accuracy: 0.4288
Epoch 7/10
391/391 [==============================] - 5s 13ms/step - loss: 11.4579 -
accuracy: 0.4312 - val_loss: 11.3383 - val_accuracy: 0.4481
Epoch 8/10
391/391 [==============================] - 5s 13ms/step - loss: 11.3135 -
accuracy: 0.4450 - val_loss: 11.2319 - val_accuracy: 0.4438
Epoch 9/10
391/391 [==============================] - 5s 13ms/step - loss: 11.2442 -
accuracy: 0.4459 - val_loss: 11.1345 - val_accuracy: 0.4468
Epoch 10/10
391/391 [==============================] - 5s 13ms/step - loss: 11.1670 -
accuracy: 0.4542 - val_loss: 11.2002 - val_accuracy: 0.4451
313/313 [==============================] - 1s 3ms/step - loss: 11.2002 -
accuracy: 0.4451
Loss: 11.200
Accuracy: 0.445
```

[30]:
```
# max norm variant
# Elastic Net Regularization (L1 + L2)# Regularizer layer import
from tensorflow.keras.constraints import max_norm# Inizializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Classifier␣
 ↪inclusion
```

```
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 11ms/step - loss: 1.5886 -
accuracy: 0.4327 - val_loss: 1.0912 - val_accuracy: 0.6198
Epoch 2/10
391/391 [==============================] - 4s 10ms/step - loss: 0.9850 -
accuracy: 0.6606 - val_loss: 0.9805 - val_accuracy: 0.6647
Epoch 3/10
391/391 [==============================] - 4s 10ms/step - loss: 0.7399 -
accuracy: 0.7502 - val_loss: 0.9369 - val_accuracy: 0.6770
Epoch 4/10
391/391 [==============================] - 4s 10ms/step - loss: 0.5421 -
accuracy: 0.8201 - val_loss: 0.9529 - val_accuracy: 0.6810
Epoch 5/10
391/391 [==============================] - 4s 10ms/step - loss: 0.3652 -
accuracy: 0.8865 - val_loss: 1.0124 - val_accuracy: 0.6802
Epoch 6/10
391/391 [==============================] - 4s 10ms/step - loss: 0.1988 -
accuracy: 0.9454 - val_loss: 1.1365 - val_accuracy: 0.6820
Epoch 7/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0938 -
accuracy: 0.9802 - val_loss: 1.2079 - val_accuracy: 0.6945
Epoch 8/10
391/391 [==============================] - 4s 11ms/step - loss: 0.0453 -
accuracy: 0.9930 - val_loss: 1.3889 - val_accuracy: 0.6822
Epoch 9/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0283 -
accuracy: 0.9957 - val_loss: 1.4559 - val_accuracy: 0.6948
Epoch 10/10
391/391 [==============================] - 4s 10ms/step - loss: 0.0221 -
accuracy: 0.9964 - val_loss: 1.4617 - val_accuracy: 0.6942
313/313 [==============================] - 1s 2ms/step - loss: 1.4617 -
```

```
accuracy: 0.6942
Loss: 1.462
Accuracy: 0.694
```

[31]:
```
# Dropout
# Dropout layer import
from tensorflow.keras.layers import Dropout# Inizializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))
model.add(Dropout(0.25))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.25))# Defining a third convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.25))# Classifier inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 4s 10ms/step - loss: 1.8205 -
accuracy: 0.3443 - val_loss: 1.2875 - val_accuracy: 0.5419
Epoch 2/10
391/391 [==============================] - 4s 10ms/step - loss: 1.2560 -
accuracy: 0.5567 - val_loss: 1.0667 - val_accuracy: 0.6281
Epoch 3/10
391/391 [==============================] - 4s 10ms/step - loss: 1.0224 -
accuracy: 0.6409 - val_loss: 0.9466 - val_accuracy: 0.6652
Epoch 4/10
391/391 [==============================] - 4s 10ms/step - loss: 0.8687 -
accuracy: 0.6966 - val_loss: 0.8854 - val_accuracy: 0.6927
Epoch 5/10
391/391 [==============================] - 4s 10ms/step - loss: 0.7355 -
accuracy: 0.7467 - val_loss: 0.8569 - val_accuracy: 0.7028
Epoch 6/10
```

```
391/391 [==============================] - 4s 10ms/step - loss: 0.6066 -
accuracy: 0.7924 - val_loss: 0.8367 - val_accuracy: 0.7123
Epoch 7/10
391/391 [==============================] - 4s 10ms/step - loss: 0.4968 -
accuracy: 0.8296 - val_loss: 0.8282 - val_accuracy: 0.7222
Epoch 8/10
391/391 [==============================] - 4s 10ms/step - loss: 0.3976 -
accuracy: 0.8645 - val_loss: 0.8707 - val_accuracy: 0.7207
Epoch 9/10
391/391 [==============================] - 4s 10ms/step - loss: 0.3109 -
accuracy: 0.8987 - val_loss: 0.8980 - val_accuracy: 0.7185
Epoch 10/10
391/391 [==============================] - 4s 10ms/step - loss: 0.2446 -
accuracy: 0.9179 - val_loss: 0.9533 - val_accuracy: 0.7206
313/313 [==============================] - 1s 2ms/step - loss: 0.9533 -
accuracy: 0.7205
Loss: 0.953
Accuracy: 0.720
```

[32]:
```
# Dropout & Max Norm
from tensorflow.keras.constraints import max_norm# Inizializing the model
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))
model.add(Dropout(0.25))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.25))# Defining a third convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.25))# Classifier inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 13ms/step - loss: 1.7897 -
```

```
accuracy: 0.3560 - val_loss: 1.2524 - val_accuracy: 0.5556
Epoch 2/10
391/391 [==============================] - 5s 12ms/step - loss: 1.2401 -
accuracy: 0.5608 - val_loss: 1.0757 - val_accuracy: 0.6196
Epoch 3/10
391/391 [==============================] - 5s 12ms/step - loss: 1.0183 -
accuracy: 0.6437 - val_loss: 0.9525 - val_accuracy: 0.6675
Epoch 4/10
391/391 [==============================] - 5s 12ms/step - loss: 0.8593 -
accuracy: 0.6979 - val_loss: 0.8876 - val_accuracy: 0.6885
Epoch 5/10
391/391 [==============================] - 5s 13ms/step - loss: 0.7242 -
accuracy: 0.7489 - val_loss: 0.8724 - val_accuracy: 0.7008
Epoch 6/10
391/391 [==============================] - 5s 12ms/step - loss: 0.5942 -
accuracy: 0.7973 - val_loss: 0.8394 - val_accuracy: 0.7132
Epoch 7/10
391/391 [==============================] - 5s 12ms/step - loss: 0.4826 -
accuracy: 0.8360 - val_loss: 0.8614 - val_accuracy: 0.7129
Epoch 8/10
391/391 [==============================] - 5s 12ms/step - loss: 0.3779 -
accuracy: 0.8735 - val_loss: 0.8866 - val_accuracy: 0.7223
Epoch 9/10
391/391 [==============================] - 5s 12ms/step - loss: 0.2903 -
accuracy: 0.9058 - val_loss: 0.9443 - val_accuracy: 0.7191
Epoch 10/10
391/391 [==============================] - 5s 12ms/step - loss: 0.2272 -
accuracy: 0.9242 - val_loss: 0.9764 - val_accuracy: 0.7219
313/313 [==============================] - 0s 1ms/step - loss: 0.9764 -
accuracy: 0.7218
Loss: 0.976
Accuracy: 0.722
```

[33]:
```
# Kernel Size change of (5, 5)
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(5, 5), activation='relu', input_shape=(32,␣
 ↪32, 3)))
model.add(Dropout(0.25))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(5, 5), activation='relu'))
model.add(Dropout(0.25))# Defining a third convolutional layer
model.add(Conv2D(128, kernel_size=(5, 5), activation='relu'))
model.add(Dropout(0.25))# Classifier inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
```

```
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 11ms/step - loss: 1.7889 -
accuracy: 0.3485 - val_loss: 1.2708 - val_accuracy: 0.5475
Epoch 2/10
391/391 [==============================] - 4s 11ms/step - loss: 1.2408 -
accuracy: 0.5640 - val_loss: 1.0715 - val_accuracy: 0.6303
Epoch 3/10
391/391 [==============================] - 4s 11ms/step - loss: 0.9985 -
accuracy: 0.6541 - val_loss: 0.9324 - val_accuracy: 0.6752
Epoch 4/10
391/391 [==============================] - 4s 11ms/step - loss: 0.8239 -
accuracy: 0.7150 - val_loss: 0.8468 - val_accuracy: 0.7092
Epoch 5/10
391/391 [==============================] - 4s 11ms/step - loss: 0.7015 -
accuracy: 0.7600 - val_loss: 0.8083 - val_accuracy: 0.7269
Epoch 6/10
391/391 [==============================] - 4s 11ms/step - loss: 0.5829 -
accuracy: 0.7973 - val_loss: 0.7842 - val_accuracy: 0.7326
Epoch 7/10
391/391 [==============================] - 4s 11ms/step - loss: 0.4808 -
accuracy: 0.8342 - val_loss: 0.7756 - val_accuracy: 0.7446
Epoch 8/10
391/391 [==============================] - 4s 11ms/step - loss: 0.3776 -
accuracy: 0.8705 - val_loss: 0.8178 - val_accuracy: 0.7393
Epoch 9/10
391/391 [==============================] - 4s 11ms/step - loss: 0.2939 -
accuracy: 0.9014 - val_loss: 0.8342 - val_accuracy: 0.7461
Epoch 10/10
391/391 [==============================] - 4s 11ms/step - loss: 0.2303 -
accuracy: 0.9242 - val_loss: 0.8503 - val_accuracy: 0.7462
313/313 [==============================] - 1s 2ms/step - loss: 0.8503 -
accuracy: 0.7462
Loss: 0.850
Accuracy: 0.746
```

[34]:
```
# Kernel Size change of (10, 10)
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(10, 10), activation='relu', input_shape=(32,␣
 ↪32, 3)))
model.add(Dropout(0.25))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(10, 10), activation='relu'))
model.add(Dropout(0.25))# Defining a third convolutional layer
model.add(Conv2D(128, kernel_size=(10, 10), activation='relu'))
model.add(Dropout(0.25))# Classifier inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 3s 7ms/step - loss: 1.8863 -
accuracy: 0.3003 - val_loss: 1.4515 - val_accuracy: 0.4793
Epoch 2/10
391/391 [==============================] - 3s 7ms/step - loss: 1.3988 -
accuracy: 0.4984 - val_loss: 1.3045 - val_accuracy: 0.5361
Epoch 3/10
391/391 [==============================] - 3s 7ms/step - loss: 1.2143 -
accuracy: 0.5732 - val_loss: 1.1447 - val_accuracy: 0.6008
Epoch 4/10
391/391 [==============================] - 3s 7ms/step - loss: 1.0887 -
accuracy: 0.6169 - val_loss: 1.0629 - val_accuracy: 0.6282
Epoch 5/10
391/391 [==============================] - 3s 7ms/step - loss: 0.9757 -
accuracy: 0.6557 - val_loss: 1.0113 - val_accuracy: 0.6497
Epoch 6/10
391/391 [==============================] - 3s 7ms/step - loss: 0.8902 -
accuracy: 0.6871 - val_loss: 0.9787 - val_accuracy: 0.6616
Epoch 7/10
391/391 [==============================] - 3s 7ms/step - loss: 0.8161 -
accuracy: 0.7203 - val_loss: 0.9546 - val_accuracy: 0.6736
```

```
Epoch 8/10
391/391 [==============================] - 3s 7ms/step - loss: 0.7528 -
accuracy: 0.7392 - val_loss: 0.9412 - val_accuracy: 0.6783
Epoch 9/10
391/391 [==============================] - 3s 7ms/step - loss: 0.6841 -
accuracy: 0.7600 - val_loss: 0.9197 - val_accuracy: 0.6916
Epoch 10/10
391/391 [==============================] - 3s 7ms/step - loss: 0.6251 -
accuracy: 0.7813 - val_loss: 0.9344 - val_accuracy: 0.6881
313/313 [==============================] - 1s 2ms/step - loss: 0.9344 -
accuracy: 0.6880
Loss: 0.934
Accuracy: 0.688
```

[35]:
```
# Kernel Size change of (10, 10)
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(10, 10), activation='relu', input_shape=(32,␣
 ↪32, 3)))
model.add(Dropout(0.25))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(10, 10), activation='relu'))
model.add(Dropout(0.25))# Defining a third convolutional layer
model.add(Conv2D(128, kernel_size=(10, 10), activation='relu'))
model.add(Dropout(0.25))# Classifier inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 3s 7ms/step - loss: 1.8903 -
accuracy: 0.2949 - val_loss: 1.4063 - val_accuracy: 0.4976
Epoch 2/10
391/391 [==============================] - 3s 7ms/step - loss: 1.3849 -
accuracy: 0.5039 - val_loss: 1.2457 - val_accuracy: 0.5613
Epoch 3/10
391/391 [==============================] - 3s 7ms/step - loss: 1.1955 -
```

```
accuracy: 0.5805 - val_loss: 1.1370 - val_accuracy: 0.6061
Epoch 4/10
391/391 [==============================] - 3s 7ms/step - loss: 1.0846 -
accuracy: 0.6230 - val_loss: 1.0505 - val_accuracy: 0.6351
Epoch 5/10
391/391 [==============================] - 3s 7ms/step - loss: 0.9788 -
accuracy: 0.6575 - val_loss: 1.0269 - val_accuracy: 0.6470
Epoch 6/10
391/391 [==============================] - 3s 7ms/step - loss: 0.8948 -
accuracy: 0.6850 - val_loss: 1.0117 - val_accuracy: 0.6501
Epoch 7/10
391/391 [==============================] - 3s 7ms/step - loss: 0.8182 -
accuracy: 0.7119 - val_loss: 0.9691 - val_accuracy: 0.6746
Epoch 8/10
391/391 [==============================] - 3s 7ms/step - loss: 0.7430 -
accuracy: 0.7379 - val_loss: 0.9336 - val_accuracy: 0.6816
Epoch 9/10
391/391 [==============================] - 3s 7ms/step - loss: 0.6833 -
accuracy: 0.7600 - val_loss: 0.9382 - val_accuracy: 0.6881
Epoch 10/10
391/391 [==============================] - 3s 7ms/step - loss: 0.6237 -
accuracy: 0.7807 - val_loss: 0.9120 - val_accuracy: 0.7024
313/313 [==============================] - 1s 2ms/step - loss: 0.9120 -
accuracy: 0.7025
Loss: 0.912
Accuracy: 0.702
```

[36]:

```
# max norm variant and adding more dense layers in nn
model = Sequential()# Defining a convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=(32,␣
 ↪32, 3)))# Defining a second convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Defining a third␣
 ↪convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))# Classifier␣
 ↪inclusion
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dense(512, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dense(256, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dense(128, activation='relu', kernel_constraint=max_norm(3.)))
model.add(Dense(10, activation='softmax'))# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001, decay=1e-6),
              metrics=['accuracy'])# Traning the model
model.fit(X_train_norm, to_categorical(Y_train),
          batch_size=128,
          shuffle=True,
```

```
          epochs=10,
          validation_data=(X_test_norm, to_categorical(Y_test)))# Evaluating␣
 ↪the model
scores = model.evaluate(X_test_norm, to_categorical(Y_test))
print('Loss: %.3f' % scores[0])
print('Accuracy: %.3f' % scores[1])
```

```
Epoch 1/10
391/391 [==============================] - 5s 11ms/step - loss: 1.6861 -
accuracy: 0.3941 - val_loss: 1.1870 - val_accuracy: 0.5780
Epoch 2/10
391/391 [==============================] - 4s 11ms/step - loss: 1.0920 -
accuracy: 0.6134 - val_loss: 1.0224 - val_accuracy: 0.6427
Epoch 3/10
391/391 [==============================] - 4s 11ms/step - loss: 0.8321 -
accuracy: 0.7103 - val_loss: 0.9535 - val_accuracy: 0.6738
Epoch 4/10
391/391 [==============================] - 4s 11ms/step - loss: 0.6325 -
accuracy: 0.7858 - val_loss: 0.9306 - val_accuracy: 0.6841
Epoch 5/10
391/391 [==============================] - 4s 11ms/step - loss: 0.4252 -
accuracy: 0.8583 - val_loss: 0.9433 - val_accuracy: 0.6949
Epoch 6/10
391/391 [==============================] - 4s 11ms/step - loss: 0.2374 -
accuracy: 0.9259 - val_loss: 1.0665 - val_accuracy: 0.6952
Epoch 7/10
391/391 [==============================] - 4s 11ms/step - loss: 0.1052 -
accuracy: 0.9717 - val_loss: 1.2431 - val_accuracy: 0.6953
Epoch 8/10
391/391 [==============================] - 4s 11ms/step - loss: 0.0535 -
accuracy: 0.9861 - val_loss: 1.3665 - val_accuracy: 0.6905
Epoch 9/10
391/391 [==============================] - 4s 11ms/step - loss: 0.0405 -
accuracy: 0.9896 - val_loss: 1.5245 - val_accuracy: 0.6879
Epoch 10/10
391/391 [==============================] - 4s 11ms/step - loss: 0.0420 -
accuracy: 0.9888 - val_loss: 1.5635 - val_accuracy: 0.6922
313/313 [==============================] - 1s 2ms/step - loss: 1.5635 -
accuracy: 0.6922
Loss: 1.564
Accuracy: 0.692
```

```
[ ]:
```