Objective: To reach an accuracy on the test set higher than 45%.

Modify the DNN_warmup notebook as follows:

- Add suitable hidden layers to the skeleton model in the notebook.
- Add early stopping to the training process in the section titled “Train the model”.

```
from tensorflow import keras
from keras.layers import Flatten
from keras.models import Sequential
from keras.layers import Dense
from keras import layers,regularizers
from keras.layers import Dropout,BatchNormalization

tf.keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)


l2_norm = .0001

# Add a Dense layer with number of neurons equal to the number of classes, with softmax as activation funct
model = Sequential()
model.add(Flatten(input_shape=(32,32,3)))
## you may add here dense layers, using fo instance model.add(Dense(64, activation='relu'))
model.add(Dense(768,activation='relu',kernel_initializer='he_normal',
                kernel_regularizer=regularizers.l2(l2_norm)))
model.add(BatchNormalization())
model.add(Dropout(.2))
model.add(Dense(384,activation='relu',kernel_initializer='he_normal',
                kernel_regularizer=keras.regularizers.l2(l2_norm)))
model.add(BatchNormalization())
model.add(Dropout(.5))
model.add(Dense(num_classes, activation='softmax'))
```

```
from keras.optimizers import Adam, SGD, Adadelta, Adagrad, Adamax, Nadam, RMSprop

sgd = SGD(learning_rate=0.001, momentum=0.09, decay=0.0, nesterov=True)
loss = ['categorical_crossentropy']

metrics = ['accuracy','precision','recall']
```

✓ 0.1s Python

```
model.compile(optimizer=sgd,
              loss=loss[0],
              metrics=[metrics[0]],
              )
```

✓ 0.1s Python

```python
batch_size = 128
epochs = 200
from tensorflow import keras
from keras.callbacks import ModelCheckpoint, EarlyStopping
import os

es_callback = EarlyStopping(monitor='val_accuracy', patience=3,verbose=1)

checkpoint_path = "output1/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = ModelCheckpoint(checkpoint_path,monitor='val_accuracy',save_best_only=True)

history = model.fit(x_train, y_train, batch_size=batch_size, validation_data =(x_val, y_val),epochs=epochs,
                    callbacks=[es_callback,cp_callback])
```

✓ 20m 18.1s Python

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/200
293/293 [==============================] - ETA: 0s - loss: 3.2264 - accuracy: 0.1911INFO:tensorflow:Assets
written to: output1\cp.ckpt\assets
293/293 [==============================] - 14s 45ms/step - loss: 3.2264 - accuracy: 0.1911 - val_loss: 2.2748 -
val_accuracy: 0.3107
Epoch 2/200
292/293 [============================>.] - ETA: 0s - loss: 2.7906 - accuracy: 0.2574INFO:tensorflow:Assets
written to: output1\cp.ckpt\assets
293/293 [==============================] - 13s 44ms/step - loss: 2.7913 - accuracy: 0.2573 - val loss: 2.1151 -


293/293 [==============================] - 12s 40ms/step - loss: 1.5343 - accuracy: 0.5373 - val_loss: 1.5836 -
val_accuracy: 0.5233
Epoch 98/200
293/293 [==============================] - 11s 37ms/step - loss: 1.5306 - accuracy: 0.5370 - val_loss: 1.5813 -
val_accuracy: 0.5232
Epoch 98: early stopping
```

```python
keras.models.load_model('output1\\cp.ckpt')
```

✓ 0.7s Python

```
<keras.engine.sequential.Sequential at 0x249962e8f70>
```

Evaluate the model

```python
_, train_acc = model.evaluate(x_train, y_train, verbose=1)
_, val_acc = model.evaluate(x_val, y_val, verbose=1)

_, test_acc = model.evaluate(x_test, y_test, verbose=1)
print('Train: %.3f, val: %.3f,Test: %.3f' % (train_acc,val_acc, test_acc))
```

✓ 16.2s Python

```
1172/1172 [==============================] - 10s 8ms/step - loss: 1.3386 - accuracy: 0.6242
391/391 [==============================] - 3s 8ms/step - loss: 1.5813 - accuracy: 0.5232
313/313 [==============================] - 3s 8ms/step - loss: 1.5698 - accuracy: 0.5229
Train: 0.624, val: 0.523,Test: 0.523
```

Bonus: Improve your model to reach an accuracy higher that 50% of the test set.

```
from tensorflow import keras
from keras.layers import Flatten
from keras.models import Sequential
from keras.layers import Dense
from keras import layers,regularizers
from keras.layers import Dropout,Conv2D,Activation,MaxPooling2D,BatchNormalization

tf.keras.backend.clear_session()
np.random.seed(42)
tf.random.set_seed(42)


num_filters_1=32
num_filters_2=64
num_filters_3=128
filter_size=3
pool_size=2
```

```
l2_norm = .0001
model = Sequential()
model.add(Conv2D(num_filters_1,filter_size, padding='same',
                 kernel_regularizer=regularizers.l2(l2_norm), input_shape=x_train.shape[1:],activation='elu'))
model.add(BatchNormalization())

model.add(Conv2D(num_filters_1, filter_size, padding='same',
                 kernel_regularizer=regularizers.l2(l2_norm),activation='elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(0.2))
model.add(Conv2D(num_filters_2, filter_size, padding='same',
                 kernel_regularizer=regularizers.l2(l2_norm),activation='elu'))
model.add(BatchNormalization())

model.add(Conv2D(num_filters_2, filter_size, padding='same',
                 kernel_regularizer=regularizers.l2(l2_norm),activation='elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(0.3))
model.add(Conv2D(num_filters_3, filter_size, padding='same',
                 kernel_regularizer=regularizers.l2(l2_norm),activation='elu'))
model.add(BatchNormalization())
model.add(Conv2D(num_filters_3, filter_size, padding='same',
                 kernel_regularizer=regularizers.l2(l2_norm),activation='elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))
```

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

model.summary()
```

```python
from keras.optimizers import Adam, SGD, Adadelta, Adagrad, Adamax, Nadam, RMSprop

rms = RMSprop(learning_rate=0.001,rho=0.9,epsilon=None,decay=0.000001)
# Losses    https://keras.io/losses/
loss = ['categorical_crossentropy']

# Metrics    https://www.tensorflow.org/api_docs/python/tf/metrics
metrics = ['accuracy','precision','recall']

# Compile the model you created before using
# rms optimizer as optimizer
# categorical crossentropy as loss function
# accuracy as metric
def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 75:
        lrate = 0.0005
    if epoch > 100:
        lrate = 0.0003
    return lrate

model.compile(optimizer=rms,
              loss=loss[0],
              metrics=[metrics[0]],
              )
```

✓ 0.4s Python

```python
batch_size = 64
epochs = 50
from tensorflow import keras
from keras.callbacks import ModelCheckpoint, EarlyStopping
import os

from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator( rotation_range=90,
                    width_shift_range=0.1, height_shift_range=0.1,
                    horizontal_flip=True)
datagen.fit(x_train)

es_callback = EarlyStopping(monitor='val_accuracy', patience=10,verbose=1)

checkpoint_path = "output/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = ModelCheckpoint(checkpoint_path,monitor='val_accuracy',save_best_only=True)

history = model.fit(datagen.flow(x_train,y_train,batch_size=batch_size),
        validation_data=(x_val, y_val), epochs=epochs,
        callbacks= [keras.callbacks.LearningRateScheduler(lr_schedule),es_callback,cp_callback], verbose=1)
```

✓ 266m 15.6s Python

```
Epoch 39/50
586/586 [==============================] - 138s 235ms/step - loss: 0.9309 - accuracy: 0.7289 - val_loss: 0.9442
- val_accuracy: 0.7394 - lr: 0.0010
Epoch 40/50
586/586 [==============================] - 234s 399ms/step - loss: 0.9286 - accuracy: 0.7283 - val_loss: 0.9604
- val_accuracy: 0.7314 - lr: 0.0010
Epoch 41/50
586/586 [==============================] - 280s 477ms/step - loss: 0.9254 - accuracy: 0.7311 - val_loss: 0.9528
- val_accuracy: 0.7370 - lr: 0.0010
Epoch 42/50
586/586 [==============================] - 276s 471ms/step - loss: 0.9223 - accuracy: 0.7310 - val_loss: 0.8763
- val_accuracy: 0.7566 - lr: 0.0010
Epoch 43/50
586/586 [==============================] - 278s 474ms/step - loss: 0.9212 - accuracy: 0.7310 - val_loss: 0.8862
- val_accuracy: 0.7567 - lr: 0.0010
Epoch 43: early stopping
```

```
keras.models.load_model('output\\cp.ckpt')
```

✓ 0.3s Python

Evaluate the model

```
_, train_acc = model.evaluate(x_train, y_train, verbose=1)
_, val_acc = model.evaluate(x_val, y_val, verbose=1)

_, test_acc = model.evaluate(x_test, y_test, verbose=1)
print('Train: %.3f, val: %.3f,Test: %.3f' % (train_acc,val_acc, test_acc))
```

✓ 1m 52.7s Python

```
1172/1172 [==============================] - 72s 62ms/step - loss: 0.8200 - accuracy: 0.7703
391/391 [==============================] - 19s 49ms/step - loss: 0.8862 - accuracy: 0.7567
313/313 [==============================] - 16s 51ms/step - loss: 0.9013 - accuracy: 0.7488
Train: 0.770, val: 0.757,Test: 0.749
```