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()
```

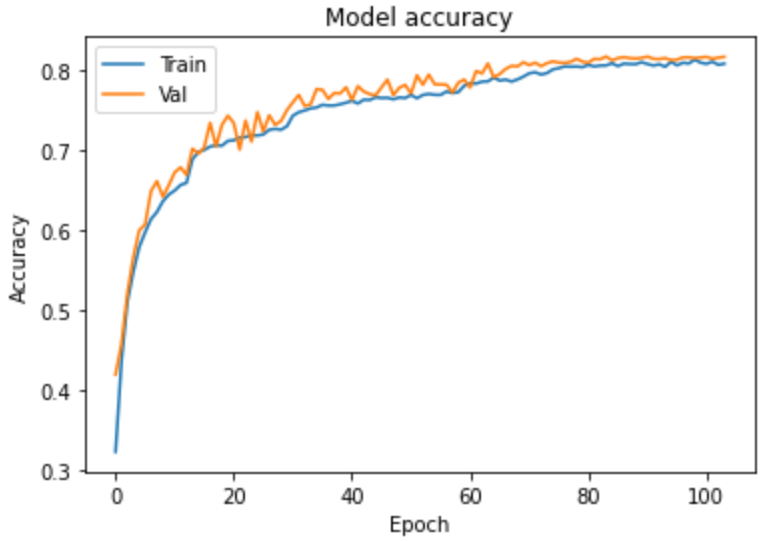
```
from keras.optimizers import Adam, SGD, Adadelta, Adagrad, Adamax, Nadam, RMSprop

adamax = Adamax(lr=0.002,beta_1=0.9,beta_2=0.999,epsilon=None,decay=0.0)
nadam = Nadam(lr=0.002,beta_1=0.9,beta_2=0.999,epsilon=None,schedule_decay=0.004)
# 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

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

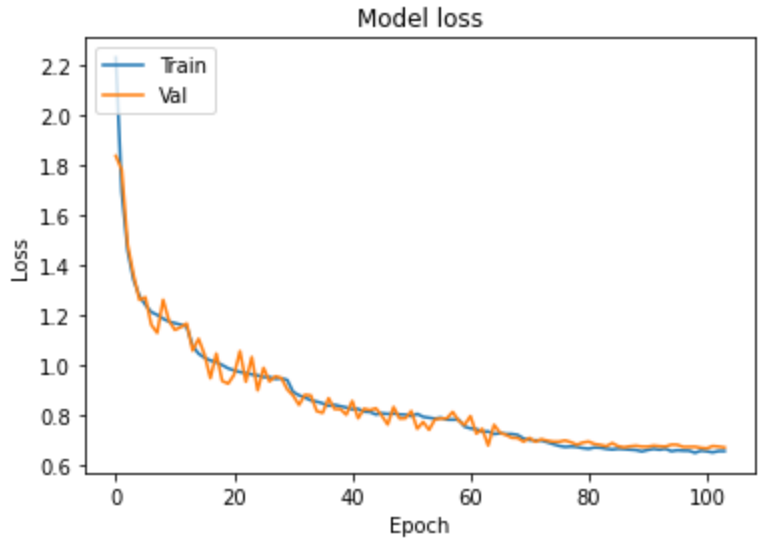
```
batch_size = 64
epochs = 200
from tensorflow import keras
from keras.callbacks import ModelCheckpoint, EarlyStopping
import os

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=20,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)
lr_scheduler = tf.keras.callbacks.ReduceLROnPlateau(factor=0.5, patience=5)
#lr_scheduler=keras.callbacks.LearningRateScheduler(lr_schedule)
history = model.fit(datagen.flow(x_train,y_train,batch_size=batch_size),
          validation_data=(x_val, y_val), epochs=epochs,
          callbacks= [lr_scheduler,es_callback,cp_callback], verbose=1)
```

```
Epoch 92/200
586/586 [==============================] - 158s 269ms/step - loss: 0.6652 - accuracy: 0.8059 - val_loss: 0.6782 - val_accuracy: 0.8143 - lr: 3.1250e-
05
Epoch 93/200
586/586 [==============================] - 158s 270ms/step - loss: 0.6620 - accuracy: 0.8074 - val_loss: 0.6758 - val_accuracy: 0.8139 - lr: 1.5625e-
05
Epoch 94/200
586/586 [==============================] - 158s 270ms/step - loss: 0.6659 - accuracy: 0.8043 - val_loss: 0.6746 - val_accuracy: 0.8151 - lr: 1.5625e-
05
Epoch 95/200
586/586 [==============================] - 160s 273ms/step - loss: 0.6561 - accuracy: 0.8106 - val_loss: 0.6822 - val_accuracy: 0.8126 - lr: 1.5625e-
05
Epoch 96/200
586/586 [==============================] - 160s 272ms/step - loss: 0.6598 - accuracy: 0.8061 - val_loss: 0.6818 - val_accuracy: 0.8132 - lr: 1.5625e-
05
...
586/586 [==============================] - 157s 268ms/step - loss: 0.6566 - accuracy: 0.8067 - val_loss: 0.6742 - val_accuracy: 0.8158 - lr: 7.8125e-
06
Epoch 104/200
586/586 [==============================] - 158s 270ms/step - loss: 0.6566 - accuracy: 0.8082 - val_loss: 0.6710 - val_accuracy: 0.8167 - lr: 7.8125e-
06
Epoch 104: 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))
```

```
1172/1172 [==============================] - 45s 39ms/step - loss: 0.5646 - accuracy: 0.8434
391/391 [==============================] - 17s 43ms/step - loss: 0.6710 - accuracy: 0.8167
313/313 [==============================] - 12s 39ms/step - loss: 0.6983 - accuracy: 0.8064
Train: 0.843, val: 0.817,Test: 0.806
```