```
pip install --upgrade pip

Requirement already satisfied: pip in
/Users/sravya/anaconda3/lib/python3.10/site-packages (23.2.1)
Note: you may need to restart the kernel to use updated packages.

pip install wheel

Requirement already satisfied: wheel in
/Users/sravya/anaconda3/lib/python3.10/site-packages (0.38.4)
Note: you may need to restart the kernel to use updated packages.

pip install keras

Requirement already satisfied: keras in
/Users/sravya/anaconda3/lib/python3.10/site-packages (2.13.1)
Note: you may need to restart the kernel to use updated packages.

pip install tensorflow

Requirement already satisfied: tensorflow in
/Users/sravya/anaconda3/lib/python3.10/site-packages (2.13.0)
Requirement already satisfied: absl-py>=1.0.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(1.6.3)
Requirement already satisfied: flatbuffers>=23.1.21 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(3.7.0)
Requirement already satisfied: libclang>=13.0.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(16.0.6)
Requirement already satisfied: numpy<=1.24.3,>=1.22 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(3.3.0)
Requirement already satisfied: packaging in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
```

```
(23.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(4.24.2)
Requirement already satisfied: setuptools in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(68.1.2)
Requirement already satisfied: six>=1.12.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(2.3.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(4.4.0)
Requirement already satisfied: wrapt>=1.11.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(1.57.0)
Requirement already satisfied: tensorboard<2.14,>=2.13 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(2.13.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0
in /Users/sravya/anaconda3/lib/python3.10/site-packages (from
tensorflow) (2.13.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(2.13.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from tensorflow)
(0.33.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
astunparse>=1.6.0->tensorflow) (0.38.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
tensorboard<2.14,>=2.13->tensorflow) (2.22.0)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
tensorboard<2.14,>=2.13->tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
tensorboard<2.14,>=2.13->tensorflow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
```

```
tensorboard<2.14,>=2.13->tensorflow) (2.28.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /Users/sravya/anaconda3/lib/python3.10/site-packages (from
tensorboard<2.14,>=2.13->tensorflow) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
tensorboard<2.14,>=2.13->tensorflow) (2.2.2)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (4.9)
Requirement already satisfied: urllib3<2.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (1.26.14)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from google-
auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<3,>=2 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from
werkzeug>=1.0.1->tensorboard<2.14,>=2.13->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from pyasn1-
modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13-
>tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5-
>tensorboard<2.14,>=2.13->tensorflow) (3.2.2)
Note: you may need to restart the kernel to use updated packages.
```

```
import numpy as np

def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='latin')
```

```
    return dict

import os
def load_batch_file(batch_filename):
    filepath = os.path.join('cifar-10-batches-py/', batch_filename)
    unpickled = unpickle(filepath)
    return unpickled

train_batch_1 = load_batch_file('data_batch_1')
train_batch_2 = load_batch_file('data_batch_2')
train_batch_3 = load_batch_file('data_batch_3')
train_batch_4 = load_batch_file('data_batch_4')
train_batch_5 = load_batch_file('data_batch_5')
test_batch = load_batch_file('test_batch')

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

pip install np_utils
```

```
Requirement already satisfied: np_utils in
/Users/sravya/anaconda3/lib/python3.10/site-packages (0.6.0)
Requirement already satisfied: numpy>=1.0 in
/Users/sravya/anaconda3/lib/python3.10/site-packages (from np_utils)
(1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

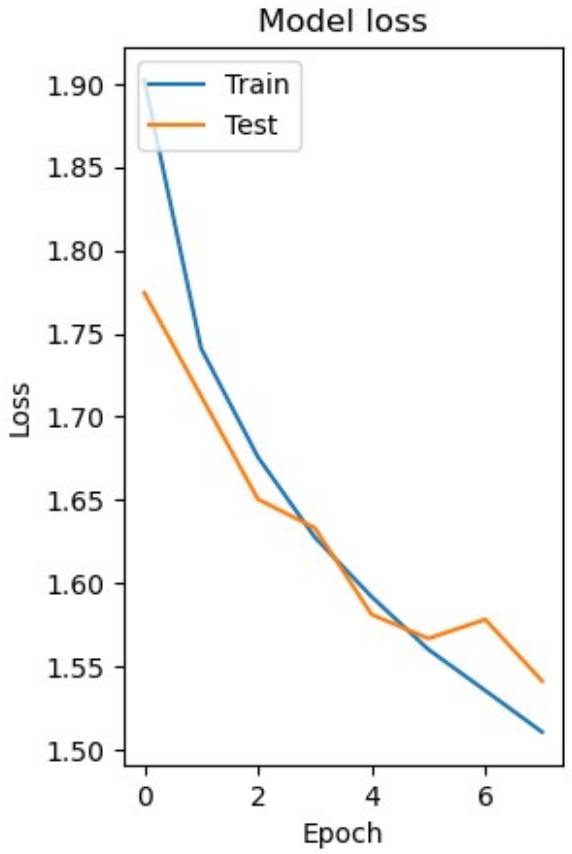
```
from keras.utils import to_categorical
#from keras.utils import np_utils
num_classes = 10
train_x = np.concatenate([train_batch_1['data'],
train_batch_2['data'], train_batch_3['data'], train_batch_4['data'],
train_batch_5['data']])
train_x = train_x.astype('float32') # this is necessary for the
division below
train_x /= 255
train_y = np.concatenate([to_categorical(labels, num_classes) for
labels in [train_batch_1['labels'], train_batch_2['labels'],
train_batch_3['labels'], train_batch_4['labels'],
train_batch_5['labels']]])

test_x = test_batch['data'].astype('float32') / 255
test_y = to_categorical(test_batch['labels'], num_classes)

from keras.models import Sequential
from keras.layers import Dense

img_rows = img_cols = 32
channels = 3
```

```
simple_model = Sequential()
simple_model.add(Dense(10_000,
input_shape=(img_rows*img_cols*channels,), activation='relu'))
simple_model.add(Dense(10, activation='softmax'))

simple_model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])
simple_model_history = simple_model.fit(train_x, train_y,
batch_size=100, epochs=8, validation_data=(test_x, test_y))
```

```
Epoch 1/8
500/500 [==============================] - 85s 168ms/step - loss:
1.9023 - accuracy: 0.3264 - val_loss: 1.7745 - val_accuracy: 0.3825
Epoch 2/8
500/500 [==============================] - 90s 178ms/step - loss:
1.7411 - accuracy: 0.3920 - val_loss: 1.7124 - val_accuracy: 0.3946
Epoch 3/8
500/500 [==============================] - 89s 179ms/step - loss:
1.6757 - accuracy: 0.4185 - val_loss: 1.6504 - val_accuracy: 0.4264
Epoch 4/8
500/500 [==============================] - 80s 159ms/step - loss:
1.6275 - accuracy: 0.4364 - val_loss: 1.6332 - val_accuracy: 0.4210
Epoch 5/8
500/500 [==============================] - 79s 158ms/step - loss:
1.5918 - accuracy: 0.4483 - val_loss: 1.5811 - val_accuracy: 0.4494
Epoch 6/8
500/500 [==============================] - 80s 161ms/step - loss:
1.5601 - accuracy: 0.4616 - val_loss: 1.5667 - val_accuracy: 0.4587
Epoch 7/8
500/500 [==============================] - 84s 167ms/step - loss:
1.5354 - accuracy: 0.4687 - val_loss: 1.5782 - val_accuracy: 0.4500
Epoch 8/8
500/500 [==============================] - 82s 163ms/step - loss:
1.5105 - accuracy: 0.4778 - val_loss: 1.5411 - val_accuracy: 0.4525
```

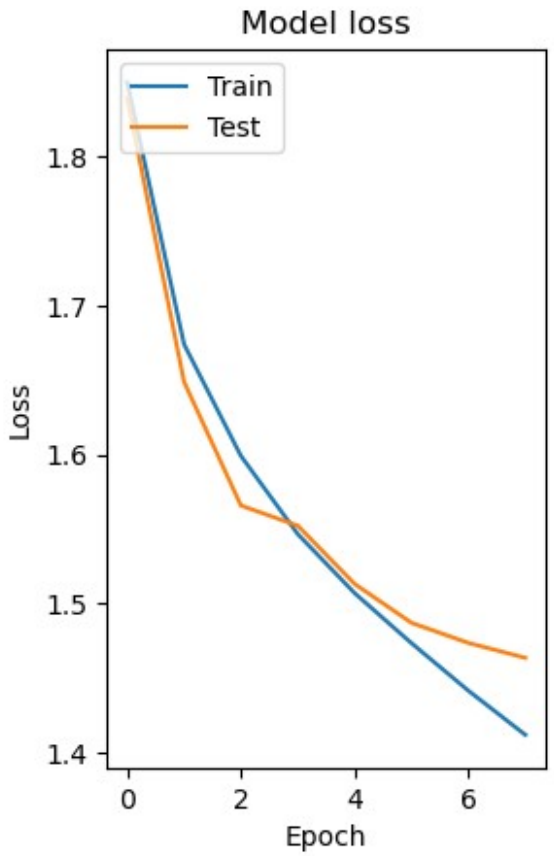
```
def plot_history(history, title):
    #plt.figure(figsize=(10,3))
    # Plot training & validation accuracy values
    #plt.subplot(121)
    #plt.plot(history.history['acc'])
    #plt.plot(history.history['val_acc'])
    #plt.title('Model accuracy')
    #plt.ylabel('Accuracy')
    #plt.xlabel('Epoch')
    #plt.legend(['Train', 'Test'], loc='upper left')

    # Plot training & validation loss values
    plt.subplot(122)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
```

```
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

import matplotlib. pyplot as plt
plot_history(simple_model_history, 'Simple NN with 100 batch size')
```



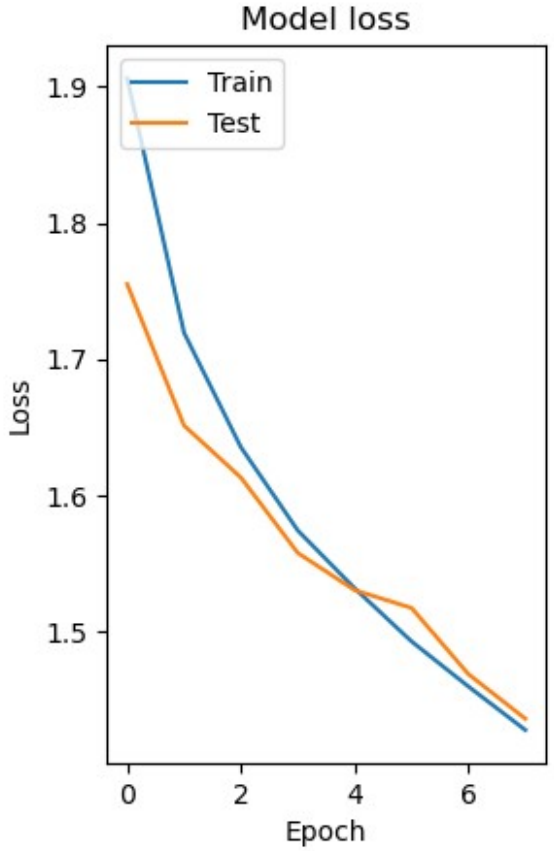

```
simple_model_smaller_batch = Sequential()
simple_model_smaller_batch.add(Dense(10_000,
input_shape=(img_rows*img_cols*channels,), activation='relu'))
simple_model_smaller_batch.add(Dense(10, activation='softmax'))

simple_model_smaller_batch.compile(optimizer='sgd',
loss='categorical_crossentropy', metrics=['accuracy'])
simple_model_smaller_batch_history =
simple_model_smaller_batch.fit(train_x, train_y, batch_size=50,
epochs=8, validation_data=(test_x, test_y))

Epoch 1/8
1000/1000 [==============================] - 104s 103ms/step - loss:
1.8500 - accuracy: 0.3428 - val_loss: 1.8394 - val_accuracy: 0.3348
```

```
Epoch 2/8
1000/1000 [==============================] - 103s 102ms/step - loss:
1.6741 - accuracy: 0.4176 - val_loss: 1.6488 - val_accuracy: 0.4256
Epoch 3/8
1000/1000 [==============================] - 110s 110ms/step - loss:
1.5991 - accuracy: 0.4430 - val_loss: 1.5658 - val_accuracy: 0.4578
Epoch 4/8
1000/1000 [==============================] - 114s 113ms/step - loss:
1.5466 - accuracy: 0.4648 - val_loss: 1.5523 - val_accuracy: 0.4552
Epoch 5/8
1000/1000 [==============================] - 115s 115ms/step - loss:
1.5069 - accuracy: 0.4767 - val_loss: 1.5131 - val_accuracy: 0.4713
Epoch 6/8
1000/1000 [==============================] - 113s 113ms/step - loss:
1.4734 - accuracy: 0.4883 - val_loss: 1.4870 - val_accuracy: 0.4711
Epoch 7/8
1000/1000 [==============================] - 114s 114ms/step - loss:
1.4415 - accuracy: 0.5005 - val_loss: 1.4735 - val_accuracy: 0.4759
Epoch 8/8
1000/1000 [==============================] - 118s 118ms/step - loss:
1.4118 - accuracy: 0.5090 - val_loss: 1.4635 - val_accuracy: 0.4813

plot_history(simple_model_smaller_batch_history, 'Simple NN with 50
batch size')
```




```
simple_model_more_layers = Sequential()
simple_model_more_layers.add(Dense(10_000,
input_shape=(img_rows*img_cols*channels,), activation='relu'))
simple_model_more_layers.add(Dense(1_000, activation='relu'))
simple_model_more_layers.add(Dense(100, activation='relu'))
simple_model_more_layers.add(Dense(10, activation='softmax'))

simple_model_more_layers.compile(optimizer='sgd',
loss='categorical_crossentropy', metrics=['accuracy'])
simple_model_more_layers_history =
simple_model_more_layers.fit(train_x, train_y, batch_size=100,
epochs=8, validation_data=(test_x, test_y))

Epoch 1/8
500/500 [==============================] - 134s 266ms/step - loss:
1.9063 - accuracy: 0.3220 - val_loss: 1.7550 - val_accuracy: 0.3837
Epoch 2/8
500/500 [==============================] - 128s 255ms/step - loss:
1.7192 - accuracy: 0.3927 - val_loss: 1.6511 - val_accuracy: 0.4266
Epoch 3/8
500/500 [==============================] - 129s 258ms/step - loss:
1.6351 - accuracy: 0.4245 - val_loss: 1.6129 - val_accuracy: 0.4295
Epoch 4/8
```

```
500/500 [==============================] - 129s 257ms/step - loss:
1.5740 - accuracy: 0.4453 - val_loss: 1.5575 - val_accuracy: 0.4515
Epoch 5/8
500/500 [==============================] - 133s 266ms/step - loss:
1.5315 - accuracy: 0.4617 - val_loss: 1.5302 - val_accuracy: 0.4628
Epoch 6/8
500/500 [==============================] - 128s 257ms/step - loss:
1.4925 - accuracy: 0.4749 - val_loss: 1.5171 - val_accuracy: 0.4666
Epoch 7/8
500/500 [==============================] - 131s 262ms/step - loss:
1.4594 - accuracy: 0.4886 - val_loss: 1.4685 - val_accuracy: 0.4866
Epoch 8/8
500/500 [==============================] - 131s 261ms/step - loss:
1.4274 - accuracy: 0.4987 - val_loss: 1.4357 - val_accuracy: 0.4919

plot_history(simple_model_more_layers_history, 'Simple NN with more
layers')
```

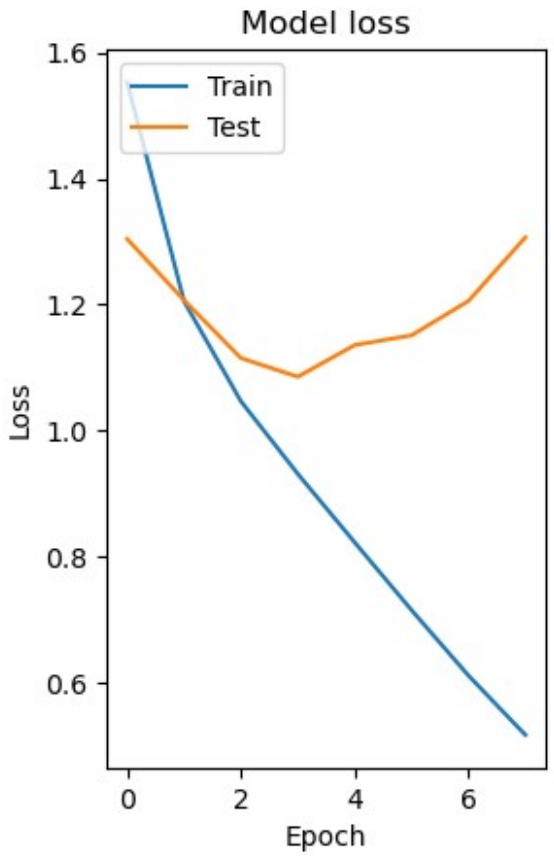



```
train_x_reshaped = train_x.reshape(len(train_x), img_rows, img_cols,
channels)
test_x_reshaped = test_x.reshape(len(test_x), img_rows, img_cols,
channels)
```

```
from keras.layers import Conv2D, Flatten
simple_cnn_model = Sequential()
simple_cnn_model.add(Conv2D(32, (3,3),
input_shape=(img_rows,img_cols,channels), activation='relu'))
simple_cnn_model.add(Conv2D(32, (3,3), activation='relu'))
simple_cnn_model.add(Conv2D(32, (3,3), activation='relu'))
simple_cnn_model.add(Flatten())
simple_cnn_model.add(Dense(10, activation='softmax'))

simple_cnn_model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
simple_cnn_model_history = simple_cnn_model.fit(train_x_reshaped,
train_y, batch_size=100, epochs=8, validation_data=(test_x_reshaped,
test_y))
```

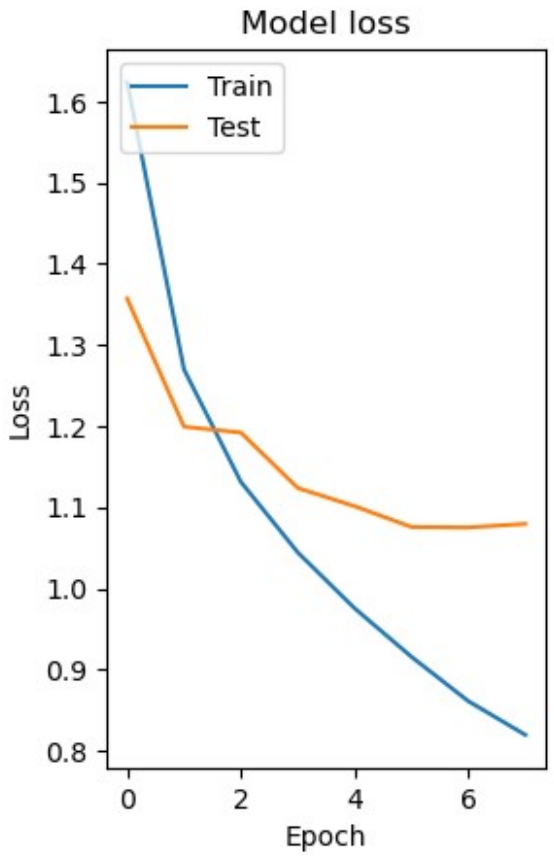
```
Epoch 1/8
500/500 [==============================] - 92s 183ms/step - loss:
1.5530 - accuracy: 0.4515 - val_loss: 1.3040 - val_accuracy: 0.5478
Epoch 2/8
500/500 [==============================] - 83s 165ms/step - loss:
1.2051 - accuracy: 0.5810 - val_loss: 1.2069 - val_accuracy: 0.5780
Epoch 3/8
500/500 [==============================] - 82s 165ms/step - loss:
1.0464 - accuracy: 0.6406 - val_loss: 1.1150 - val_accuracy: 0.6142
Epoch 4/8
500/500 [==============================] - 81s 163ms/step - loss:
0.9311 - accuracy: 0.6814 - val_loss: 1.0853 - val_accuracy: 0.6198
Epoch 5/8
500/500 [==============================] - 81s 162ms/step - loss:
0.8224 - accuracy: 0.7207 - val_loss: 1.1356 - val_accuracy: 0.6127
Epoch 6/8
500/500 [==============================] - 81s 163ms/step - loss:
0.7148 - accuracy: 0.7558 - val_loss: 1.1511 - val_accuracy: 0.6229
Epoch 7/8
500/500 [==============================] - 81s 161ms/step - loss:
0.6110 - accuracy: 0.7924 - val_loss: 1.2056 - val_accuracy: 0.6222
Epoch 8/8
500/500 [==============================] - 81s 162ms/step - loss:
0.5169 - accuracy: 0.8250 - val_loss: 1.3065 - val_accuracy: 0.6196
```

```
plot_history(simple_cnn_model_history, 'CNN with 3 convolution
layers')
```



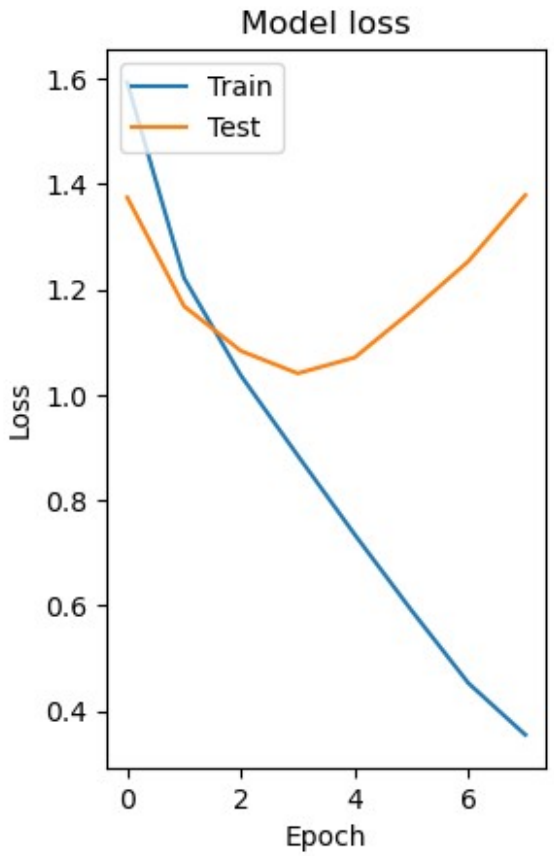

```
from keras.layers import Dropout
simple_cnn_model_2 = Sequential()
simple_cnn_model_2.add(Conv2D(32, (3,3),
input_shape=(img_rows,img_cols,channels), activation='relu'))
simple_cnn_model_2.add(Dropout(0.2))

simple_cnn_model_2.add(Conv2D(32, (3,3), activation='relu'))
simple_cnn_model_2.add(Dropout(0.2))

simple_cnn_model_2.add(Conv2D(32, (3,3), activation='relu'))
simple_cnn_model_2.add(Dropout(0.2))

simple_cnn_model_2.add(Flatten())
simple_cnn_model_2.add(Dense(10, activation='softmax'))

simple_cnn_model_2.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
simple_cnn_model_2_history = simple_cnn_model_2.fit(train_x_reshaped,
train_y, batch_size=100, epochs=8, validation_data=(test_x_reshaped,
test_y))

Epoch 1/8
500/500 [==============================] - 114s 225ms/step - loss:
```

```
1.6238 - accuracy: 0.4268 - val_loss: 1.3571 - val_accuracy: 0.5245
Epoch 2/8
500/500 [==============================] - 107s 213ms/step - loss:
1.2697 - accuracy: 0.5567 - val_loss: 1.1993 - val_accuracy: 0.5765
Epoch 3/8
500/500 [==============================] - 107s 214ms/step - loss:
1.1312 - accuracy: 0.6069 - val_loss: 1.1921 - val_accuracy: 0.5765
Epoch 4/8
500/500 [==============================] - 105s 210ms/step - loss:
1.0443 - accuracy: 0.6378 - val_loss: 1.1238 - val_accuracy: 0.6046
Epoch 5/8
500/500 [==============================] - 106s 212ms/step - loss:
0.9755 - accuracy: 0.6617 - val_loss: 1.1013 - val_accuracy: 0.6171
Epoch 6/8
500/500 [==============================] - 104s 208ms/step - loss:
0.9157 - accuracy: 0.6832 - val_loss: 1.0757 - val_accuracy: 0.6258
Epoch 7/8
500/500 [==============================] - 104s 208ms/step - loss:
0.8610 - accuracy: 0.7025 - val_loss: 1.0751 - val_accuracy: 0.6294
Epoch 8/8
500/500 [==============================] - 103s 206ms/step - loss:
0.8196 - accuracy: 0.7164 - val_loss: 1.0796 - val_accuracy: 0.6299

plot_history(simple_cnn_model_2_history, '3 convolution layers with
dropout')
```




```
simple_cnn_model_3 = Sequential()
simple_cnn_model_3.add(Conv2D(64, (3,3),
input_shape=(img_rows,img_cols,channels), activation='relu'))
simple_cnn_model_3.add(Dropout(0.2))

simple_cnn_model_3.add(Conv2D(64, (3,3), activation='relu'))
simple_cnn_model_3.add(Dropout(0.2))

simple_cnn_model_3.add(Conv2D(64, (3,3), activation='relu'))
simple_cnn_model_3.add(Dropout(0.2))

simple_cnn_model_3.add(Flatten())
simple_cnn_model_3.add(Dense(128, activation='relu'))
simple_cnn_model_3.add(Dense(10, activation='softmax'))

simple_cnn_model_3.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
simple_cnn_model_3_history = simple_cnn_model_3.fit(train_x_reshaped,
train_y, batch_size=100, epochs=8, validation_data=(test_x_reshaped,
test_y))

Epoch 1/8
500/500 [==============================] - 266s 531ms/step - loss:
```

```
1.5930 - accuracy: 0.4349 - val_loss: 1.3744 - val_accuracy: 0.5126
Epoch 2/8
500/500 [==============================] - 269s 538ms/step - loss:
1.2216 - accuracy: 0.5731 - val_loss: 1.1680 - val_accuracy: 0.5894
Epoch 3/8
500/500 [==============================] - 269s 539ms/step - loss:
1.0364 - accuracy: 0.6366 - val_loss: 1.0832 - val_accuracy: 0.6236
Epoch 4/8
500/500 [==============================] - 268s 537ms/step - loss:
0.8847 - accuracy: 0.6932 - val_loss: 1.0406 - val_accuracy: 0.6466
Epoch 5/8
500/500 [==============================] - 265s 529ms/step - loss:
0.7350 - accuracy: 0.7417 - val_loss: 1.0707 - val_accuracy: 0.6432
Epoch 6/8
500/500 [==============================] - 255s 509ms/step - loss:
0.5910 - accuracy: 0.7936 - val_loss: 1.1589 - val_accuracy: 0.6409
Epoch 7/8
500/500 [==============================] - 252s 504ms/step - loss:
0.4531 - accuracy: 0.8407 - val_loss: 1.2533 - val_accuracy: 0.6325
Epoch 8/8
500/500 [==============================] - 240s 479ms/step - loss:
0.3550 - accuracy: 0.8753 - val_loss: 1.3789 - val_accuracy: 0.6304

plot_history(simple_cnn_model_3_history, 'CNN with more layers and
filters')
```


Model loss
Train
Test
Loss
Epoch
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
2
4
6