Fadilla Zennifa CNN apss

In [1]:

```
#loaddata
import sys
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D,
Dropout,BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [2]:

```
#make training data 50% of dataset
X_train,y_train = X_train[:int(0.5 * len(X_train))],y_train[:int(0.5 * len(y_train))]
```

In [3]:

```
print('Images Shape: {}'.format(X_train.shape))
print('Labels Shape: {}'.format(y_train.shape))
```

```
Images Shape: (25000, 32, 32, 3)
Labels Shape: (25000, 1)
```

In [4]:

```
#X_train.max()
X_train = X_train/255
X_test = X_test/255
```

In [5]:

```
#CNN MODEL
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', in
put_shape=(32, 32, 3)))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
```

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

```
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-
packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling
BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint i
s deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

In [6]:

```
#compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',
metrics=['sparse_categorical_accuracy'])
```

In [7]:

```
#model fitting
history = model.fit(X_train, y_train, epochs=100, verbose=1, validation_data=(X_test, y_test))

#history = model.fit(X_trainnew, y_trainnew, epochs=10,validation_data=(X_testnew, y_testnew))
```

```
Train on 25000 samples, validate on 10000 samples
Epoch 1/100
25000/25000 [==============================] - 312s 12ms/sample - loss: 1.8340 -
sparse_categorical_accuracy: 0.3754 - val_loss: 1.3752 - val_sparse_categorical_accuracy: 0.5026
Epoch 2/100
25000/25000 [==============================] - 347s 14ms/sample - loss: 1.3230 -
sparse_categorical_accuracy: 0.5245 - val_loss: 1.2799 - val_sparse_categorical_accuracy: 0.5492
Epoch 3/100
25000/25000 [==============================] - 346s 14ms/sample - loss: 1.1182 -
sparse_categorical_accuracy: 0.6054 - val_loss: 0.9454 - val_sparse_categorical_accuracy: 0.6652
Epoch 4/100
25000/25000 [==============================] - 331s 13ms/sample - loss: 0.9868 -
sparse_categorical_accuracy: 0.6562 - val_loss: 0.8951 - val_sparse_categorical_accuracy: 0.6888
Epoch 5/100
25000/25000 [==============================] - 345s 14ms/sample - loss: 0.8928 -
sparse_categorical_accuracy: 0.6872 - val_loss: 0.8305 - val_sparse_categorical_accuracy: 0.7102
Epoch 6/100
25000/25000 [==============================] - 336s 13ms/sample - loss: 0.8369 -
sparse_categorical_accuracy: 0.7097 - val_loss: 0.7957 - val_sparse_categorical_accuracy: 0.7250
Epoch 7/100
25000/25000 [==============================] - 321s 13ms/sample - loss: 0.7754 -
sparse_categorical_accuracy: 0.7301 - val_loss: 0.7998 - val_sparse_categorical_accuracy: 0.7259
Epoch 8/100
25000/25000 [==============================] - 330s 13ms/sample - loss: 0.7280 -
sparse_categorical_accuracy: 0.7497 - val_loss: 0.7562 - val_sparse_categorical_accuracy: 0.7426
Epoch 9/100
25000/25000 [==============================] - 1173s 47ms/sample - loss: 0.6843 -
sparse_categorical_accuracy: 0.7628 - val_loss: 0.6944 - val_sparse_categorical_accuracy: 0.7580
Epoch 10/100
25000/25000 [==============================] - 296s 12ms/sample - loss: 0.6386 -
sparse_categorical_accuracy: 0.7784 - val_loss: 0.6282 - val_sparse_categorical_accuracy: 0.7869
Epoch 11/100
25000/25000 [==============================] - 322s 13ms/sample - loss: 0.6096 -
sparse_categorical_accuracy: 0.7893 - val_loss: 0.6198 - val_sparse_categorical_accuracy: 0.7889
Epoch 12/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.5749 -
sparse_categorical_accuracy: 0.8016 - val_loss: 0.6061 - val_sparse_categorical_accuracy: 0.7958
Epoch 13/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.5487 -
sparse_categorical_accuracy: 0.8101 - val_loss: 0.6248 - val_sparse_categorical_accuracy: 0.7899
Epoch 14/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.5227 -
sparse_categorical_accuracy: 0.8189 - val_loss: 0.6030 - val_sparse_categorical_accuracy: 0.7985
Epoch 15/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.4986 -
sparse_categorical_accuracy: 0.8274 - val_loss: 0.6183 - val_sparse_categorical_accuracy: 0.7980
Epoch 16/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.4754 -
sparse_categorical_accuracy: 0.8341 - val_loss: 0.6265 - val_sparse_categorical_accuracy: 0.7968
Epoch 17/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.4581 -
```

```
sparse_categorical_accuracy: 0.8412 - val_loss: 0.6016 - val_sparse_categorical_accuracy: 0.8052
Epoch 18/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.4341 -
sparse_categorical_accuracy: 0.8507 - val_loss: 0.5805 - val_sparse_categorical_accuracy: 0.8109
Epoch 19/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.4368 -
sparse_categorical_accuracy: 0.8510 - val_loss: 0.6339 - val_sparse_categorical_accuracy: 0.7941
Epoch 20/100
25000/25000 [==============================] - 299s 12ms/sample - loss: 0.4170 -
sparse_categorical_accuracy: 0.8548 - val_loss: 0.5718 - val_sparse_categorical_accuracy: 0.8155
Epoch 21/100
25000/25000 [==============================] - 291s 12ms/sample - loss: 0.3972 -
sparse_categorical_accuracy: 0.8636 - val_loss: 0.6579 - val_sparse_categorical_accuracy: 0.7948
Epoch 22/100
25000/25000 [==============================] - 294s 12ms/sample - loss: 0.3755 -
sparse_categorical_accuracy: 0.8686 - val_loss: 0.6879 - val_sparse_categorical_accuracy: 0.7848
Epoch 23/100
25000/25000 [==============================] - 317s 13ms/sample - loss: 0.3704 -
sparse_categorical_accuracy: 0.8702 - val_loss: 0.6372 - val_sparse_categorical_accuracy: 0.8042
Epoch 24/100
25000/25000 [==============================] - 298s 12ms/sample - loss: 0.3563 -
sparse_categorical_accuracy: 0.8771 - val_loss: 0.6021 - val_sparse_categorical_accuracy: 0.8120
Epoch 25/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.3432 -
sparse_categorical_accuracy: 0.8818 - val_loss: 0.6504 - val_sparse_categorical_accuracy: 0.8042
Epoch 26/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.3412 -
sparse_categorical_accuracy: 0.8817 - val_loss: 0.6168 - val_sparse_categorical_accuracy: 0.8152
Epoch 27/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.3239 -
sparse_categorical_accuracy: 0.8888 - val_loss: 0.5837 - val_sparse_categorical_accuracy: 0.8238
Epoch 28/100
25000/25000 [==============================] - 297s 12ms/sample - loss: 0.3190 -
sparse_categorical_accuracy: 0.8877 - val_loss: 0.6079 - val_sparse_categorical_accuracy: 0.8162
Epoch 29/100
25000/25000 [==============================] - 294s 12ms/sample - loss: 0.3068 -
sparse_categorical_accuracy: 0.8914 - val_loss: 0.6141 - val_sparse_categorical_accuracy: 0.8162
Epoch 30/100
25000/25000 [==============================] - 307s 12ms/sample - loss: 0.2986 -
sparse_categorical_accuracy: 0.8962 - val_loss: 0.6565 - val_sparse_categorical_accuracy: 0.8108
Epoch 31/100
25000/25000 [==============================] - 295s 12ms/sample - loss: 0.2973 -
sparse_categorical_accuracy: 0.8964 - val_loss: 0.6457 - val_sparse_categorical_accuracy: 0.8117
Epoch 32/100
25000/25000 [==============================] - 295s 12ms/sample - loss: 0.2943 -
sparse_categorical_accuracy: 0.8985 - val_loss: 0.6313 - val_sparse_categorical_accuracy: 0.8169
Epoch 33/100
25000/25000 [==============================] - 352s 14ms/sample - loss: 0.2855 -
sparse_categorical_accuracy: 0.9012 - val_loss: 0.6257 - val_sparse_categorical_accuracy: 0.8171
Epoch 34/100
25000/25000 [==============================] - 308s 12ms/sample - loss: 0.2821 -
sparse_categorical_accuracy: 0.9006 - val_loss: 0.6246 - val_sparse_categorical_accuracy: 0.8197
Epoch 35/100
25000/25000 [==============================] - 312s 12ms/sample - loss: 0.2682 -
sparse_categorical_accuracy: 0.9061 - val_loss: 0.6103 - val_sparse_categorical_accuracy: 0.8274
Epoch 36/100
25000/25000 [==============================] - 291s 12ms/sample - loss: 0.2682 -
sparse_categorical_accuracy: 0.9078 - val_loss: 0.5983 - val_sparse_categorical_accuracy: 0.8260
Epoch 37/100
25000/25000 [==============================] - 291s 12ms/sample - loss: 0.2536 -
sparse_categorical_accuracy: 0.9112 - val_loss: 0.6273 - val_sparse_categorical_accuracy: 0.8202
Epoch 38/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.2553 -
sparse_categorical_accuracy: 0.9116 - val_loss: 0.6495 - val_sparse_categorical_accuracy: 0.8142
Epoch 39/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.2522 -
sparse_categorical_accuracy: 0.9133 - val_loss: 0.6555 - val_sparse_categorical_accuracy: 0.8180
Epoch 40/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.2528 -
sparse_categorical_accuracy: 0.9125 - val_loss: 0.6162 - val_sparse_categorical_accuracy: 0.8264
Epoch 41/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.2449 -
sparse_categorical_accuracy: 0.9146 - val_loss: 0.6711 - val_sparse_categorical_accuracy: 0.8093
Epoch 42/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.2377 -
sparse_categorical_accuracy: 0.9174 - val_loss: 0.6411 - val_sparse_categorical_accuracy: 0.8200
Epoch 43/100
```

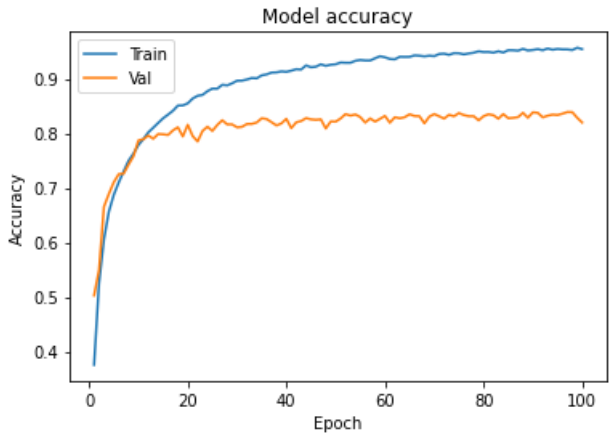
25000/25000 [==============================] - 291s 12ms/sample - loss: 0.2395 - sparse_categorical_accuracy: 0.9168 - val_loss: 0.6302 - val_sparse_categorical_accuracy: 0.8223
Epoch 44/100
25000/25000 [==============================] - 285s 11ms/sample - loss: 0.2222 - sparse_categorical_accuracy: 0.9244 - val_loss: 0.6384 - val_sparse_categorical_accuracy: 0.8277
Epoch 45/100
25000/25000 [==============================] - 296s 12ms/sample - loss: 0.2254 - sparse_categorical_accuracy: 0.9209 - val_loss: 0.6372 - val_sparse_categorical_accuracy: 0.8254
Epoch 46/100
25000/25000 [==============================] - 292s 12ms/sample - loss: 0.2293 - sparse_categorical_accuracy: 0.9220 - val_loss: 0.6292 - val_sparse_categorical_accuracy: 0.8246
Epoch 47/100
25000/25000 [==============================] - 304s 12ms/sample - loss: 0.2140 - sparse_categorical_accuracy: 0.9264 - val_loss: 0.6434 - val_sparse_categorical_accuracy: 0.8254
Epoch 48/100
25000/25000 [==============================] - 296s 12ms/sample - loss: 0.2177 - sparse_categorical_accuracy: 0.9235 - val_loss: 0.7095 - val_sparse_categorical_accuracy: 0.8086
Epoch 49/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.2130 - sparse_categorical_accuracy: 0.9256 - val_loss: 0.6648 - val_sparse_categorical_accuracy: 0.8216
Epoch 50/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.2089 - sparse_categorical_accuracy: 0.9264 - val_loss: 0.6609 - val_sparse_categorical_accuracy: 0.8212
Epoch 51/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.2014 - sparse_categorical_accuracy: 0.9296 - val_loss: 0.6626 - val_sparse_categorical_accuracy: 0.8270
Epoch 52/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.2039 - sparse_categorical_accuracy: 0.9292 - val_loss: 0.6212 - val_sparse_categorical_accuracy: 0.8351
Epoch 53/100
25000/25000 [==============================] - 287s 11ms/sample - loss: 0.2026 - sparse_categorical_accuracy: 0.9292 - val_loss: 0.6238 - val_sparse_categorical_accuracy: 0.8323
Epoch 54/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.1966 - sparse_categorical_accuracy: 0.9328 - val_loss: 0.6238 - val_sparse_categorical_accuracy: 0.8344
Epoch 55/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.1918 - sparse_categorical_accuracy: 0.9342 - val_loss: 0.6215 - val_sparse_categorical_accuracy: 0.8296
Epoch 56/100
25000/25000 [==============================] - 297s 12ms/sample - loss: 0.1952 - sparse_categorical_accuracy: 0.9334 - val_loss: 0.7003 - val_sparse_categorical_accuracy: 0.8195
Epoch 57/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.1911 - sparse_categorical_accuracy: 0.9333 - val_loss: 0.6587 - val_sparse_categorical_accuracy: 0.8271
Epoch 58/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.1822 - sparse_categorical_accuracy: 0.9374 - val_loss: 0.6793 - val_sparse_categorical_accuracy: 0.8213
Epoch 59/100
25000/25000 [==============================] - 311s 12ms/sample - loss: 0.1747 - sparse_categorical_accuracy: 0.9411 - val_loss: 0.6934 - val_sparse_categorical_accuracy: 0.8264
Epoch 60/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.1794 - sparse_categorical_accuracy: 0.9392 - val_loss: 0.6509 - val_sparse_categorical_accuracy: 0.8317
Epoch 61/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.1862 - sparse_categorical_accuracy: 0.9361 - val_loss: 0.7167 - val_sparse_categorical_accuracy: 0.8192
Epoch 62/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.1826 - sparse_categorical_accuracy: 0.9355 - val_loss: 0.6494 - val_sparse_categorical_accuracy: 0.8294
Epoch 63/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.1740 - sparse_categorical_accuracy: 0.9401 - val_loss: 0.6501 - val_sparse_categorical_accuracy: 0.8274
Epoch 64/100
25000/25000 [==============================] - 299s 12ms/sample - loss: 0.1819 - sparse_categorical_accuracy: 0.9399 - val_loss: 0.6564 - val_sparse_categorical_accuracy: 0.8293
Epoch 65/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.1758 - sparse_categorical_accuracy: 0.9401 - val_loss: 0.6457 - val_sparse_categorical_accuracy: 0.8348
Epoch 66/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.1683 - sparse_categorical_accuracy: 0.9426 - val_loss: 0.6558 - val_sparse_categorical_accuracy: 0.8316
Epoch 67/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1678 - sparse_categorical_accuracy: 0.9421 - val_loss: 0.6774 - val_sparse_categorical_accuracy: 0.8312
Epoch 68/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.1683 - sparse_categorical_accuracy: 0.9409 - val_loss: 0.7339 - val_sparse_categorical_accuracy: 0.8181

```
Epoch 69/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.1669 -
sparse_categorical_accuracy: 0.9421 - val_loss: 0.6705 - val_sparse_categorical_accuracy: 0.8304
Epoch 70/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.1665 -
sparse_categorical_accuracy: 0.9413 - val_loss: 0.6611 - val_sparse_categorical_accuracy: 0.8352
Epoch 71/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1626 -
sparse_categorical_accuracy: 0.9452 - val_loss: 0.6790 - val_sparse_categorical_accuracy: 0.8310
Epoch 72/100
25000/25000 [==============================] - 323s 13ms/sample - loss: 0.1638 -
sparse_categorical_accuracy: 0.9458 - val_loss: 0.7217 - val_sparse_categorical_accuracy: 0.8268
Epoch 73/100
25000/25000 [==============================] - 279s 11ms/sample - loss: 0.1608 -
sparse_categorical_accuracy: 0.9437 - val_loss: 0.6542 - val_sparse_categorical_accuracy: 0.8338
Epoch 74/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.1541 -
sparse_categorical_accuracy: 0.9468 - val_loss: 0.6827 - val_sparse_categorical_accuracy: 0.8311
Epoch 75/100
25000/25000 [==============================] - 306s 12ms/sample - loss: 0.1524 -
sparse_categorical_accuracy: 0.9466 - val_loss: 0.6601 - val_sparse_categorical_accuracy: 0.8369
Epoch 76/100
25000/25000 [==============================] - 312s 12ms/sample - loss: 0.1597 -
sparse_categorical_accuracy: 0.9446 - val_loss: 0.6694 - val_sparse_categorical_accuracy: 0.8333
Epoch 77/100
25000/25000 [==============================] - 308s 12ms/sample - loss: 0.1524 -
sparse_categorical_accuracy: 0.9462 - val_loss: 0.6762 - val_sparse_categorical_accuracy: 0.8312
Epoch 78/100
25000/25000 [==============================] - 295s 12ms/sample - loss: 0.1537 -
sparse_categorical_accuracy: 0.9476 - val_loss: 0.6520 - val_sparse_categorical_accuracy: 0.8312
Epoch 79/100
25000/25000 [==============================] - 295s 12ms/sample - loss: 0.1477 -
sparse_categorical_accuracy: 0.9502 - val_loss: 0.7171 - val_sparse_categorical_accuracy: 0.8236
Epoch 80/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1499 -
sparse_categorical_accuracy: 0.9488 - val_loss: 0.6891 - val_sparse_categorical_accuracy: 0.8312
Epoch 81/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1491 -
sparse_categorical_accuracy: 0.9490 - val_loss: 0.6665 - val_sparse_categorical_accuracy: 0.8348
Epoch 82/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1501 -
sparse_categorical_accuracy: 0.9476 - val_loss: 0.6823 - val_sparse_categorical_accuracy: 0.8330
Epoch 83/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1467 -
sparse_categorical_accuracy: 0.9502 - val_loss: 0.7162 - val_sparse_categorical_accuracy: 0.8266
Epoch 84/100
25000/25000 [==============================] - 314s 13ms/sample - loss: 0.1493 -
sparse_categorical_accuracy: 0.9478 - val_loss: 0.6730 - val_sparse_categorical_accuracy: 0.8351
Epoch 85/100
25000/25000 [==============================] - 291s 12ms/sample - loss: 0.1390 -
sparse_categorical_accuracy: 0.9522 - val_loss: 0.6931 - val_sparse_categorical_accuracy: 0.8275
Epoch 86/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1390 -
sparse_categorical_accuracy: 0.9521 - val_loss: 0.7020 - val_sparse_categorical_accuracy: 0.8282
Epoch 87/100
25000/25000 [==============================] - 286s 11ms/sample - loss: 0.1367 -
sparse_categorical_accuracy: 0.9514 - val_loss: 0.7024 - val_sparse_categorical_accuracy: 0.8295
Epoch 88/100
25000/25000 [==============================] - 288s 12ms/sample - loss: 0.1311 -
sparse_categorical_accuracy: 0.9546 - val_loss: 0.6732 - val_sparse_categorical_accuracy: 0.8374
Epoch 89/100
25000/25000 [==============================] - 297s 12ms/sample - loss: 0.1399 -
sparse_categorical_accuracy: 0.9517 - val_loss: 0.7147 - val_sparse_categorical_accuracy: 0.8283
Epoch 90/100
25000/25000 [==============================] - 295s 12ms/sample - loss: 0.1396 -
sparse_categorical_accuracy: 0.9526 - val_loss: 0.6707 - val_sparse_categorical_accuracy: 0.8383
Epoch 91/100
25000/25000 [==============================] - 287s 11ms/sample - loss: 0.1412 -
sparse_categorical_accuracy: 0.9542 - val_loss: 0.6610 - val_sparse_categorical_accuracy: 0.8374
Epoch 92/100
25000/25000 [==============================] - 291s 12ms/sample - loss: 0.1398 -
sparse_categorical_accuracy: 0.9522 - val_loss: 0.7053 - val_sparse_categorical_accuracy: 0.8290
Epoch 93/100
25000/25000 [==============================] - 286s 11ms/sample - loss: 0.1323 -
sparse_categorical_accuracy: 0.9552 - val_loss: 0.7220 - val_sparse_categorical_accuracy: 0.8321
Epoch 94/100
25000/25000 [==============================] - 292s 12ms/sample - loss: 0.1377 -
```

```
sparse_categorical_accuracy: 0.9531 - val_loss: 0.6903 - val_sparse_categorical_accuracy: 0.8341
Epoch 95/100
25000/25000 [==============================] - 290s 12ms/sample - loss: 0.1312 -
sparse_categorical_accuracy: 0.9545 - val_loss: 0.6745 - val_sparse_categorical_accuracy: 0.8334
Epoch 96/100
25000/25000 [==============================] - 293s 12ms/sample - loss: 0.1365 -
sparse_categorical_accuracy: 0.9537 - val_loss: 0.6827 - val_sparse_categorical_accuracy: 0.8358
Epoch 97/100
25000/25000 [==============================] - 312s 12ms/sample - loss: 0.1361 -
sparse_categorical_accuracy: 0.9536 - val_loss: 0.6550 - val_sparse_categorical_accuracy: 0.8388
Epoch 98/100
25000/25000 [==============================] - 289s 12ms/sample - loss: 0.1363 -
sparse_categorical_accuracy: 0.9526 - val_loss: 0.6561 - val_sparse_categorical_accuracy: 0.8386
Epoch 99/100
25000/25000 [==============================] - 287s 11ms/sample - loss: 0.1268 -
sparse_categorical_accuracy: 0.9567 - val_loss: 0.7249 - val_sparse_categorical_accuracy: 0.8283
Epoch 100/100
25000/25000 [==============================] - 297s 12ms/sample - loss: 0.1332 -
sparse_categorical_accuracy: 0.9542 - val_loss: 0.7496 - val_sparse_categorical_accuracy: 0.8196
```

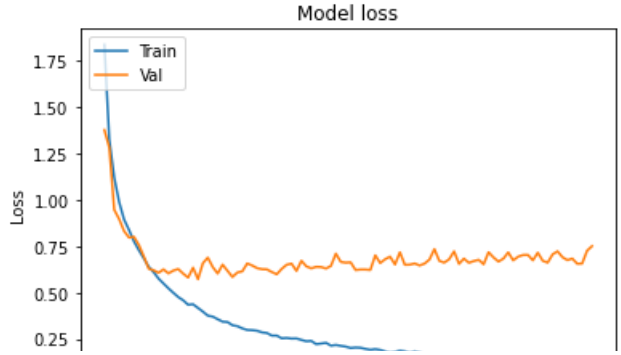
In [8]:

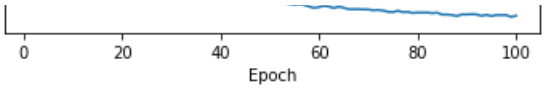
```
# Plot training & validation accuracy values
epoch_range = range(1, 101)
plt.plot(epoch_range, history.history['sparse_categorical_accuracy'])
plt.plot(epoch_range, history.history['val_sparse_categorical_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```



In [9]:

```
# Plot training & validation loss values
plt.plot(epoch_range, history.history['loss'])
plt.plot(epoch_range, history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```





In [10]:

```
# Save the entire model as a SavedModel.
!mkdir -p saved_model
model.save('saved_model/ridgehomedilla')
```

In [12]:

```
new_model = tf.keras.models.load_model('saved_model/ridgehomedilla')
```

```
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-
packages/tensorflow_core/python/ops/init_ops.py:97: calling VarianceScaling.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-
packages/tensorflow_core/python/ops/init_ops.py:97: calling Zeros.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-
packages/tensorflow_core/python/ops/init_ops.py:97: calling Ones.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-
packages/tensorflow_core/python/ops/init_ops.py:97: calling GlorotUniform.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
```

In [14]:

```
#evaluate model
from IPython.display import Image
Image("deer.png")
#print('Images Shape: {}'.format(X_train.shape))
```

Out[14]:

In [15]:

```
init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)


img = tf.read_file("deer.png")
img = tf.image.decode_jpeg(img, channels=3)
img.set_shape([None, None, 3])
img = tf.image.resize_images(img, (32, 32))
img = img.eval(session=sess) # convert to numpy array
img = np.expand_dims(img, 0) # make 'batch' of 1
# prepare pixel data
img = img.astype('float32')
img = img / 255.0

#pred = model.predict(img)
result = new_model.predict_classes(img)
print(result[0])
#acc

#class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
  #              'dog', 'frog', 'horse', 'ship', 'truck']
```

```
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-
packages/tensorflow_core/python/util/tf_should_use.py:198: initialize_all_variables (from
tensorflow.python.ops.variables) is deprecated and will be removed after 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
4
```

In [26]:

```
img = tf.read_file("Aeroplan.jpg")
img = tf.image.decode_jpeg(img, channels=3)
img.set_shape([None, None, 3])
img = tf.image.resize_images(img, (32, 32))
img = img.eval(session=sess) # convert to numpy array
img = np.expand_dims(img, 0) # make 'batch' of 1
# prepare pixel data
img = img.astype('float32')
img = img / 255.0

#pred = model.predict(img)
result = new_model.predict_classes(img)
print(result[0])
Image("Aeroplan.jpg")
```

```
0
```

Out[26]:

In [27]:

```
img = tf.read_file("cat1.jpg")
img = tf.image.decode_jpeg(img, channels=3)
img.set_shape([None, None, 3])
img = tf.image.resize_images(img, (32, 32))
img = img.eval(session=sess) # convert to numpy array
img = np.expand_dims(img, 0) # make 'batch' of 1
# prepare pixel data
img = img.astype('float32')
img = img / 255.0

#pred = model.predict(img)
result = new_model.predict_classes(img)
print(result[0])
Image("cat1.jpg")
```

3

Out[27]: