Spring 2024: CS5720
Neural Networks and Deep Learning - ICP-7
Yasaswini Majety (700747747)

Github Link: https://github.com/yasaswini8777/Neural_ICP_7

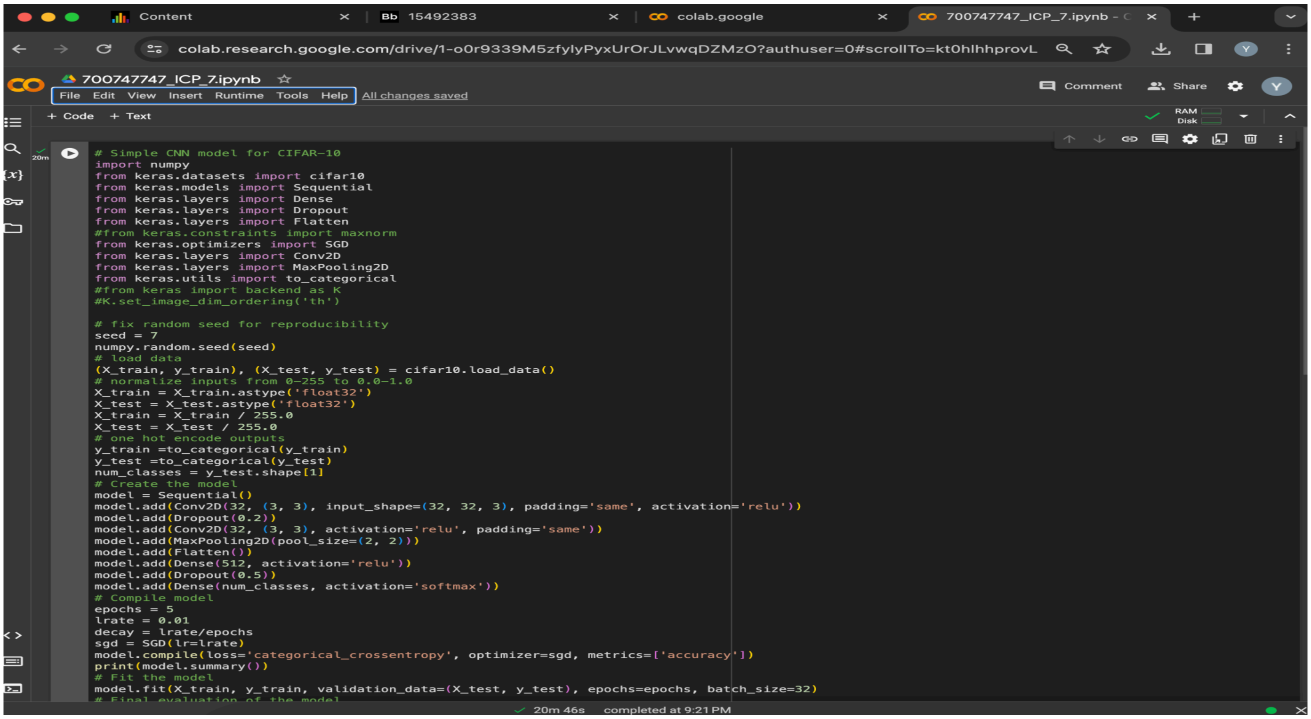
Use Case Description:
LeNet5, AlexNet, Vgg16, Vgg19
1. Training the model
2. Evaluating the model

Programming elements:
1. About CNN
2. Hyperparameters of CNN
3. Image classification with CNN

In class programming:

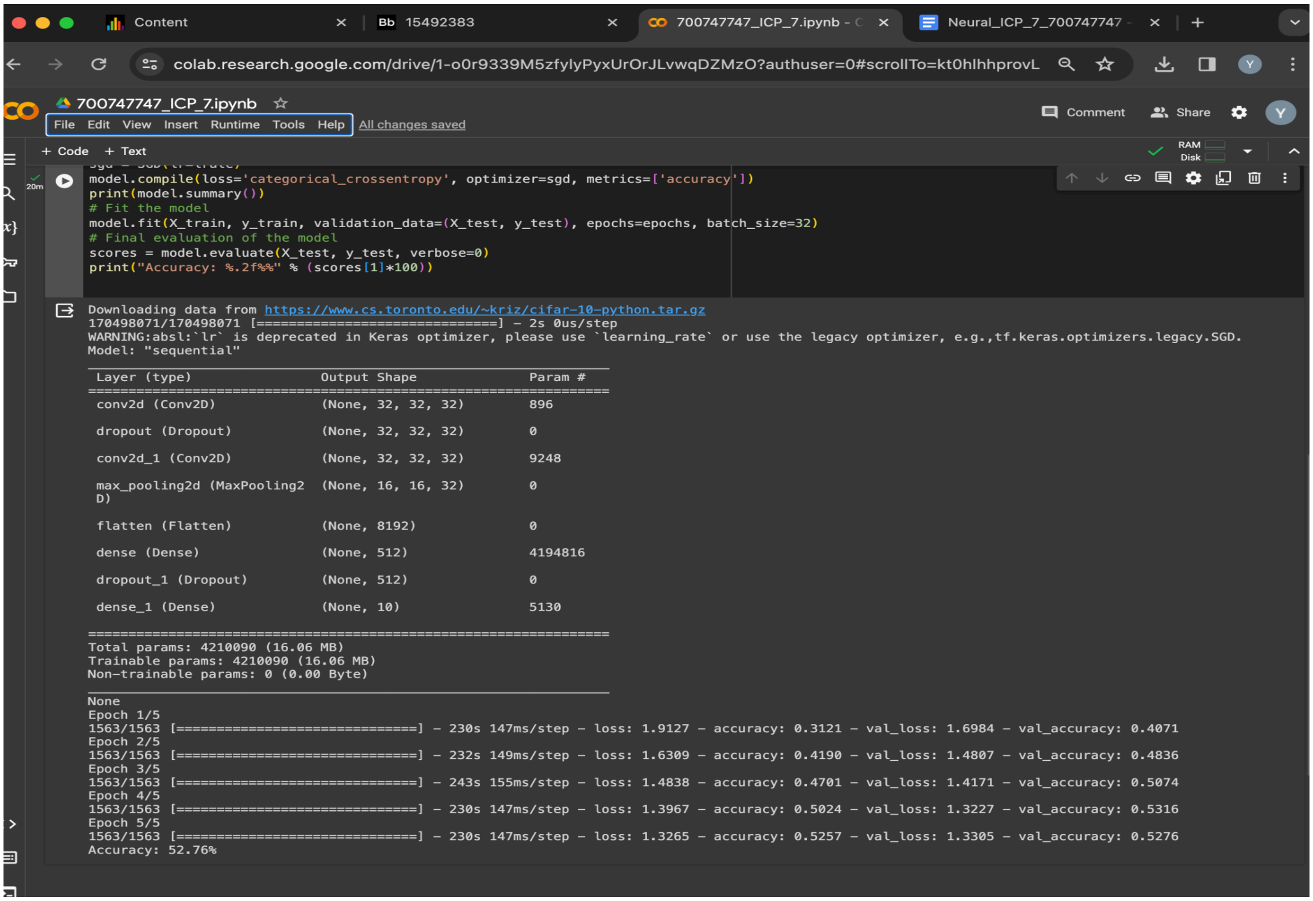
1. Follow the instruction below and then report how the performance changed.(apply all at once)
- Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Flatten layer.
- Dropout layer at 20%.
- Fully connected layer with 1024 units and a rectifier activation function.
- Dropout layer at 20%.
- Fully connected layer with 512 units and a rectifier activation function.
- Dropout layer at 20%.
- Fully connected output layer with 10 units and a Softmax activation function

Did the performance change?
2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.
3. Visualize Loss and Accuracy using the history object


Content
15492383
colab.google
700747747_ICP_7.ipynb
colab.research.google.com/drive/1-o0r9339M5zfylyPyxUrOrJLvwqDZMzO?authuser=0#scrollTo=kt0hlhhprovL
700747747_ICP_7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
20m

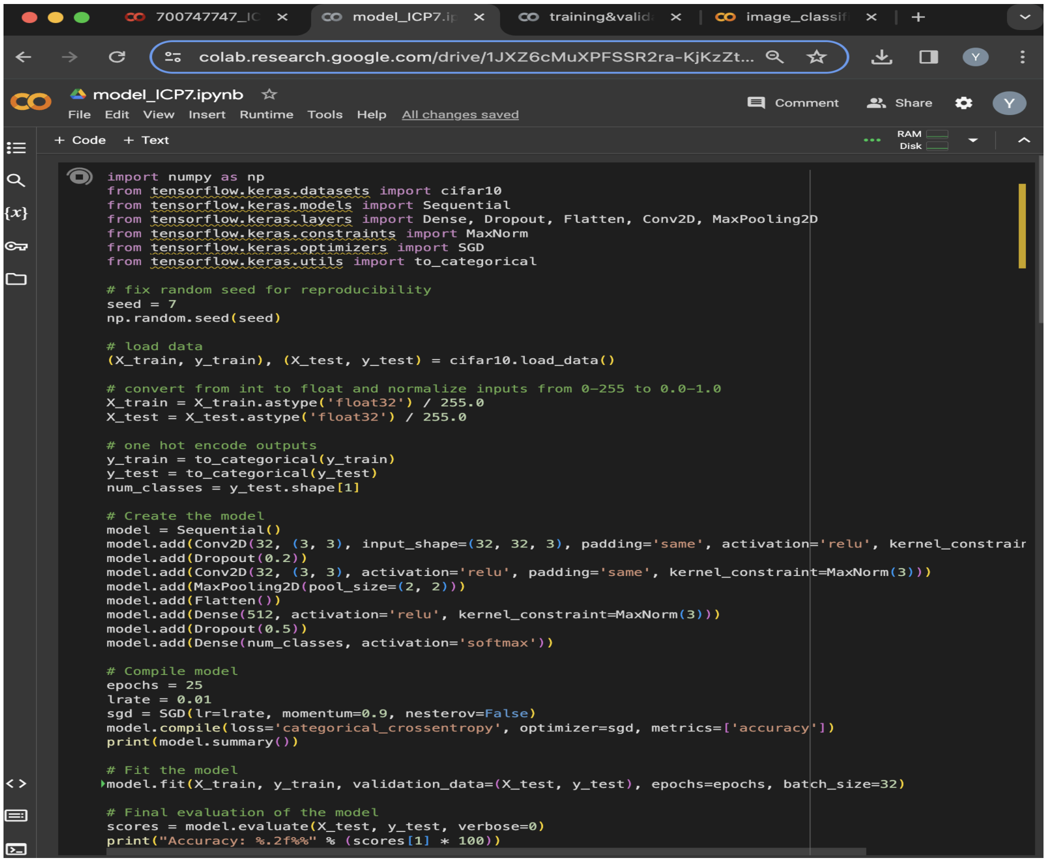
```
# Simple CNN model for CIFAR-10
import numpy
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
#from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.utils import to_categorical
#from keras import backend as K
#K.set_image_dim_ordering('th')

# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
# one hot encode outputs
y_train =to_categorical(y_train)
y_test =to_categorical(y_test)
num_classes = y_test.shape[1]
# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
# Compile model
epochs = 5
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
```

20m 46s completed at 9:21 PM

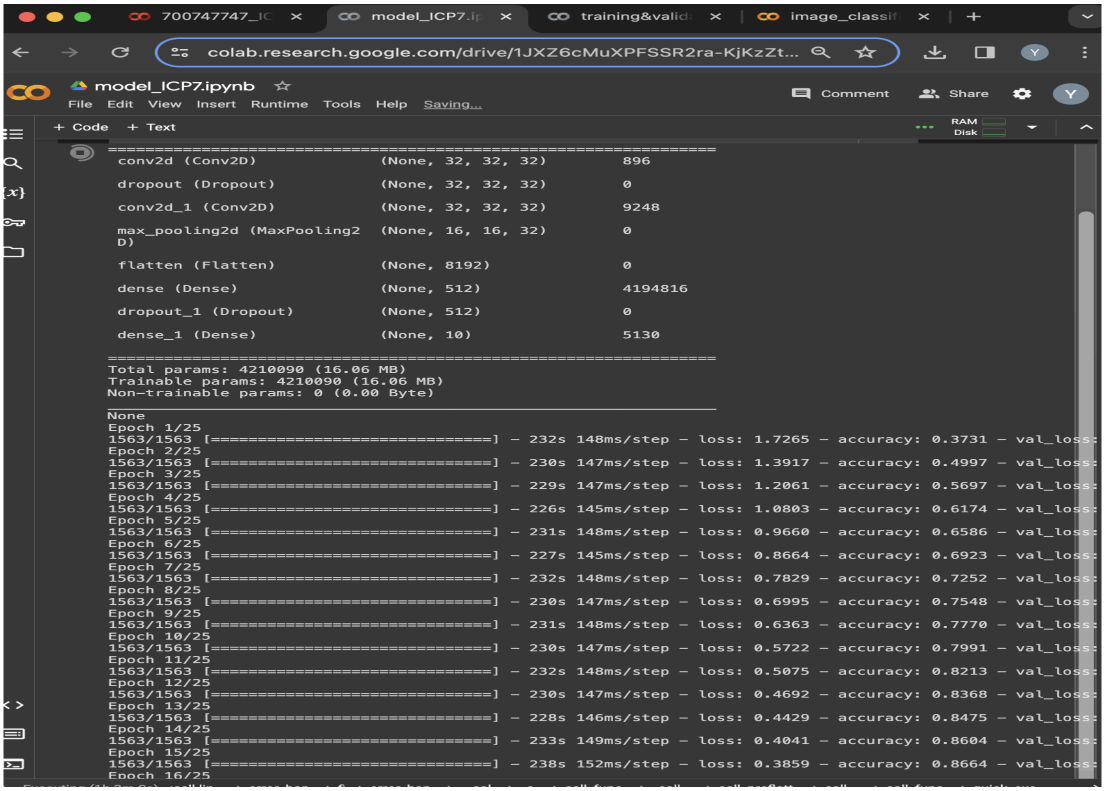

Content
15492383
700747747_ICP_7.ipynb
Neural_ICP_7_700747747
colab.research.google.com/drive/1-o0r9339M5zfylyPyxUrOrJLvwqDZMzO?authuser=0#scrollTo=kt0hlhhprovL
700747747_ICP_7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
20m

```
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

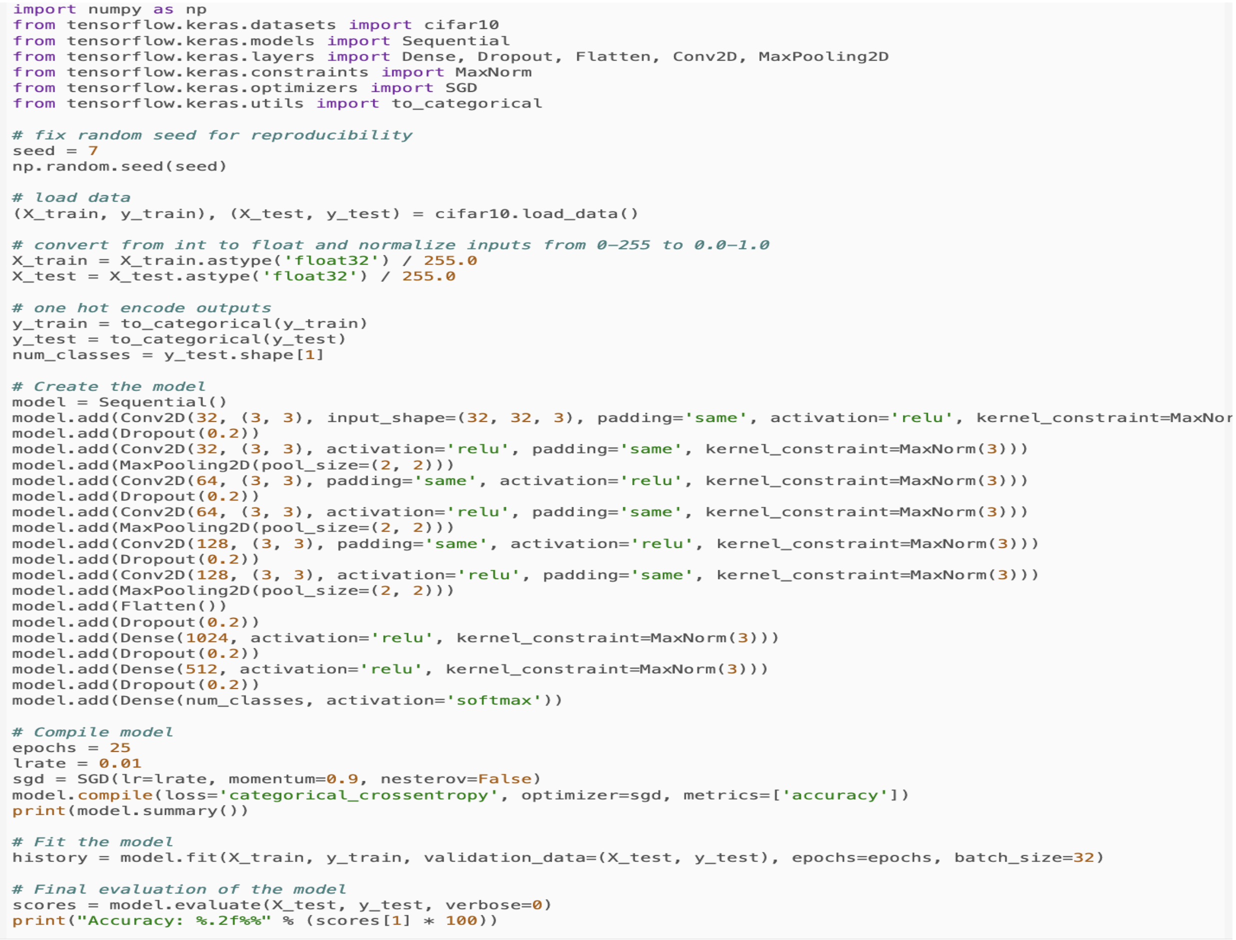
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 4210090 (16.06 MB)
Trainable params: 4210090 (16.06 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/5
1563/1563 [==============================] - 230s 147ms/step - loss: 1.9127 - accuracy: 0.3121 - val_loss: 1.6984 - val_accuracy: 0.4071
Epoch 2/5
1563/1563 [==============================] - 232s 149ms/step - loss: 1.6309 - accuracy: 0.4190 - val_loss: 1.4807 - val_accuracy: 0.4836
Epoch 3/5
1563/1563 [==============================] - 243s 155ms/step - loss: 1.4838 - accuracy: 0.4701 - val_loss: 1.4171 - val_accuracy: 0.5074
Epoch 4/5
1563/1563 [==============================] - 230s 147ms/step - loss: 1.3967 - accuracy: 0.5024 - val_loss: 1.3227 - val_accuracy: 0.5316
Epoch 5/5
1563/1563 [==============================] - 230s 147ms/step - loss: 1.3265 - accuracy: 0.5257 - val_loss: 1.3305 - val_accuracy: 0.5276
Accuracy: 52.76%
```



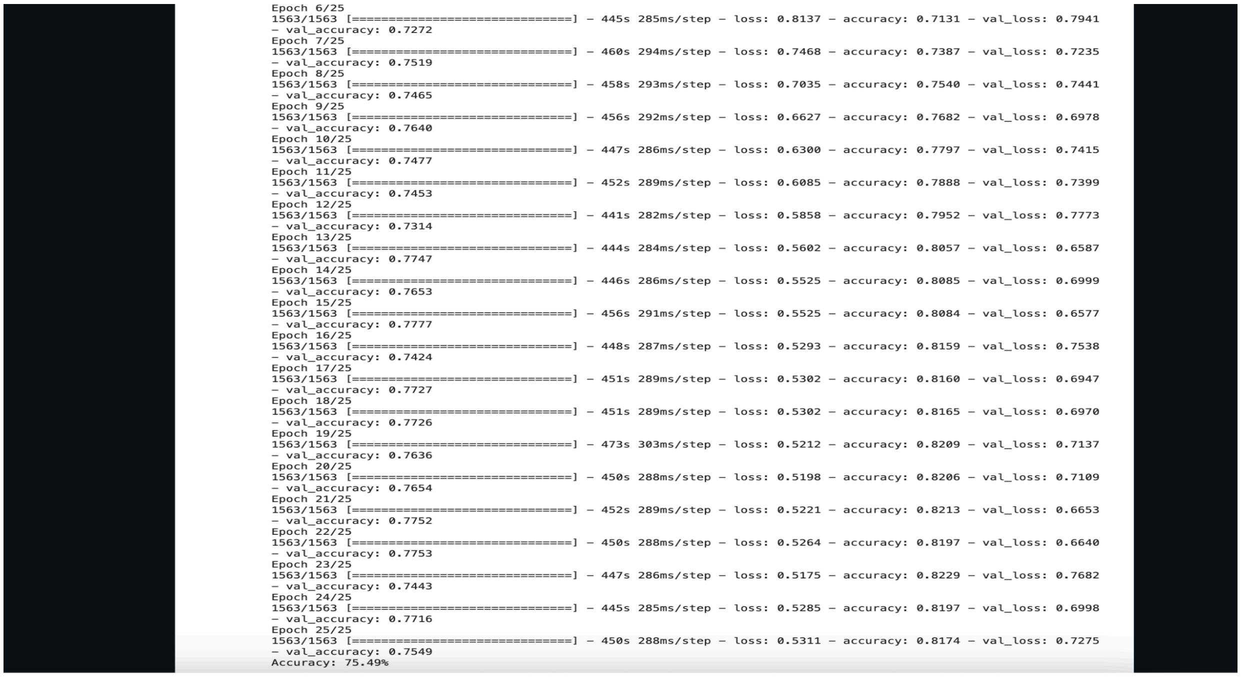
```python
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.constraints import MaxNorm
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# convert from int to float and normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# one hot encode outputs
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constrair
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
sgd = SGD(lr=lrate, momentum=0.9, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))
```



```
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 4210090 (16.06 MB)
Trainable params: 4210090 (16.06 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/25
1563/1563 [==============================] - 232s 148ms/step - loss: 1.7265 - accuracy: 0.3731 - val_loss:
Epoch 2/25
1563/1563 [==============================] - 230s 147ms/step - loss: 1.3917 - accuracy: 0.4997 - val_loss:
Epoch 3/25
1563/1563 [==============================] - 229s 147ms/step - loss: 1.2061 - accuracy: 0.5697 - val_loss:
Epoch 4/25
1563/1563 [==============================] - 226s 145ms/step - loss: 1.0803 - accuracy: 0.6174 - val_loss:
Epoch 5/25
1563/1563 [==============================] - 231s 148ms/step - loss: 0.9660 - accuracy: 0.6586 - val_loss:
Epoch 6/25
1563/1563 [==============================] - 227s 145ms/step - loss: 0.8664 - accuracy: 0.6923 - val_loss:
Epoch 7/25
1563/1563 [==============================] - 232s 148ms/step - loss: 0.7829 - accuracy: 0.7252 - val_loss:
Epoch 8/25
1563/1563 [==============================] - 230s 147ms/step - loss: 0.6995 - accuracy: 0.7548 - val_loss:
Epoch 9/25
1563/1563 [==============================] - 231s 148ms/step - loss: 0.6363 - accuracy: 0.7770 - val_loss:
Epoch 10/25
1563/1563 [==============================] - 230s 147ms/step - loss: 0.5722 - accuracy: 0.7991 - val_loss:
Epoch 11/25
1563/1563 [==============================] - 232s 148ms/step - loss: 0.5075 - accuracy: 0.8213 - val_loss:
Epoch 12/25
1563/1563 [==============================] - 230s 147ms/step - loss: 0.4692 - accuracy: 0.8368 - val_loss:
Epoch 13/25
1563/1563 [==============================] - 228s 146ms/step - loss: 0.4429 - accuracy: 0.8475 - val_loss:
Epoch 14/25
1563/1563 [==============================] - 233s 149ms/step - loss: 0.4041 - accuracy: 0.8604 - val_loss:
Epoch 15/25
1563/1563 [==============================] - 238s 152ms/step - loss: 0.3859 - accuracy: 0.8664 - val_loss:
Epoch 16/25
```



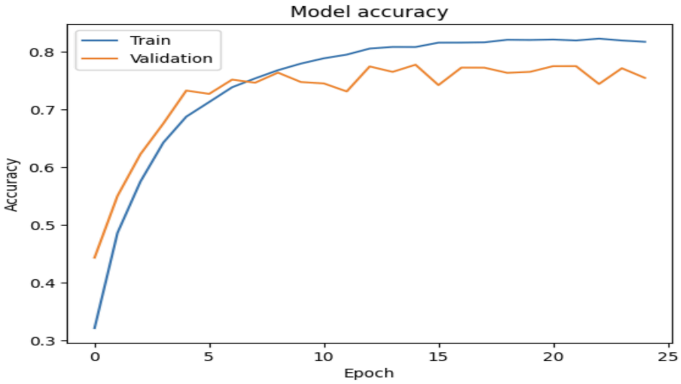
```python
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.constraints import MaxNorm
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# convert from int to float and normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# one hot encode outputs
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=MaxNor
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
sgd = SGD(lr=lrate, momentum=0.9, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

# Fit the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))
```

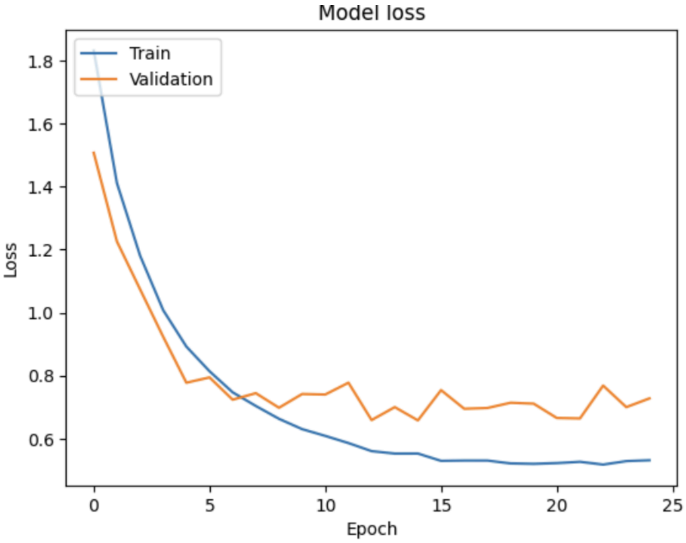


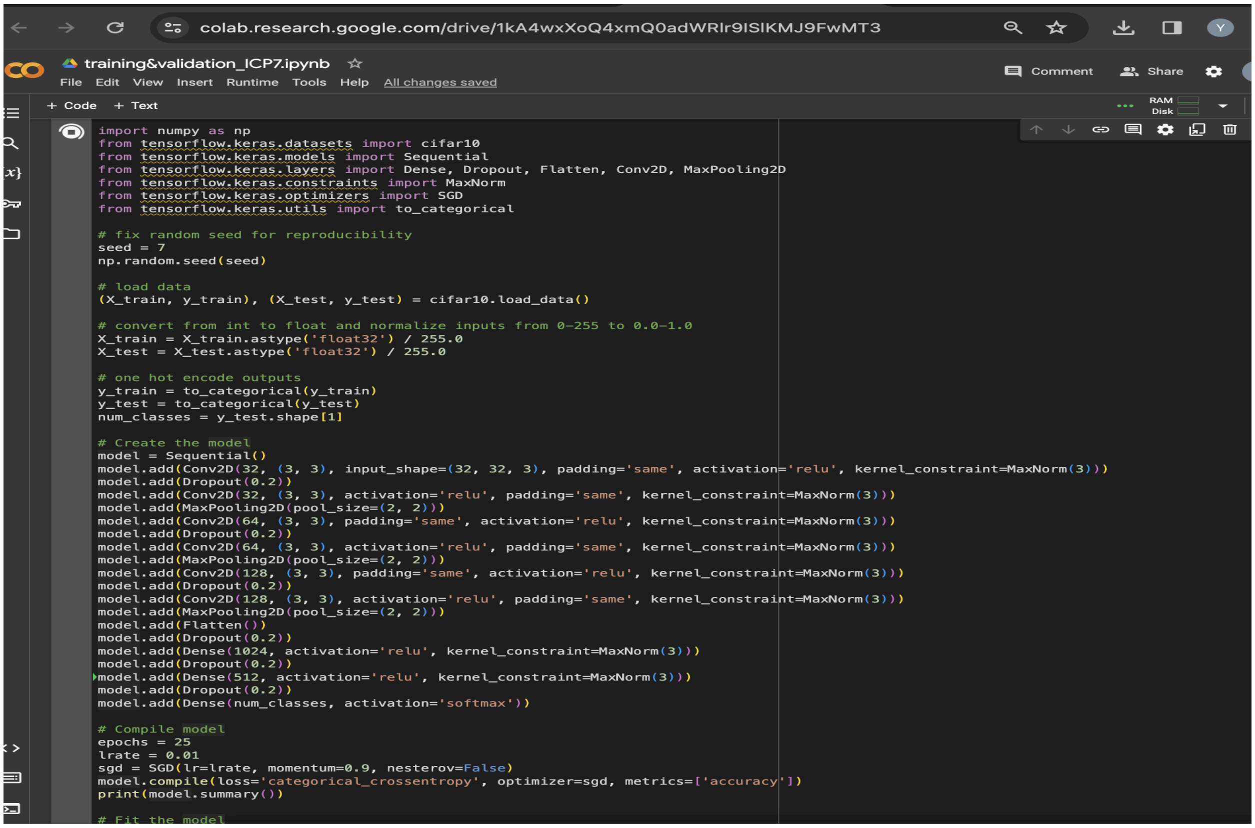
```
Epoch 6/25
1563/1563 [==============================] - 445s 285ms/step - loss: 0.8137 - accuracy: 0.7131 - val_loss: 0.7941
- val_accuracy: 0.7272
Epoch 7/25
1563/1563 [==============================] - 460s 294ms/step - loss: 0.7468 - accuracy: 0.7387 - val_loss: 0.7235
- val_accuracy: 0.7519
Epoch 8/25
1563/1563 [==============================] - 458s 293ms/step - loss: 0.7035 - accuracy: 0.7540 - val_loss: 0.7441
- val_accuracy: 0.7465
Epoch 9/25
1563/1563 [==============================] - 456s 292ms/step - loss: 0.6627 - accuracy: 0.7682 - val_loss: 0.6978
- val_accuracy: 0.7640
Epoch 10/25
1563/1563 [==============================] - 447s 286ms/step - loss: 0.6300 - accuracy: 0.7797 - val_loss: 0.7415
- val_accuracy: 0.7477
Epoch 11/25
1563/1563 [==============================] - 452s 289ms/step - loss: 0.6085 - accuracy: 0.7888 - val_loss: 0.7399
- val_accuracy: 0.7453
Epoch 12/25
1563/1563 [==============================] - 441s 282ms/step - loss: 0.5858 - accuracy: 0.7952 - val_loss: 0.7773
- val_accuracy: 0.7314
Epoch 13/25
1563/1563 [==============================] - 444s 284ms/step - loss: 0.5602 - accuracy: 0.8057 - val_loss: 0.6587
- val_accuracy: 0.7747
Epoch 14/25
1563/1563 [==============================] - 446s 286ms/step - loss: 0.5525 - accuracy: 0.8085 - val_loss: 0.6999
- val_accuracy: 0.7653
Epoch 15/25
1563/1563 [==============================] - 456s 291ms/step - loss: 0.5525 - accuracy: 0.8084 - val_loss: 0.6577
- val_accuracy: 0.7777
Epoch 16/25
1563/1563 [==============================] - 448s 287ms/step - loss: 0.5293 - accuracy: 0.8159 - val_loss: 0.7538
- val_accuracy: 0.7424
Epoch 17/25
1563/1563 [==============================] - 451s 289ms/step - loss: 0.5302 - accuracy: 0.8160 - val_loss: 0.6947
- val_accuracy: 0.7727
Epoch 18/25
1563/1563 [==============================] - 451s 289ms/step - loss: 0.5302 - accuracy: 0.8165 - val_loss: 0.6970
- val_accuracy: 0.7726
Epoch 19/25
1563/1563 [==============================] - 473s 303ms/step - loss: 0.5212 - accuracy: 0.8209 - val_loss: 0.7137
- val_accuracy: 0.7636
Epoch 20/25
1563/1563 [==============================] - 450s 288ms/step - loss: 0.5198 - accuracy: 0.8206 - val_loss: 0.7109
- val_accuracy: 0.7654
Epoch 21/25
1563/1563 [==============================] - 452s 289ms/step - loss: 0.5221 - accuracy: 0.8213 - val_loss: 0.6653
- val_accuracy: 0.7752
Epoch 22/25
1563/1563 [==============================] - 450s 288ms/step - loss: 0.5264 - accuracy: 0.8197 - val_loss: 0.6640
- val_accuracy: 0.7753
Epoch 23/25
1563/1563 [==============================] - 447s 286ms/step - loss: 0.5175 - accuracy: 0.8229 - val_loss: 0.7682
- val_accuracy: 0.7443
Epoch 24/25
1563/1563 [==============================] - 445s 285ms/step - loss: 0.5285 - accuracy: 0.8197 - val_loss: 0.6998
- val_accuracy: 0.7716
Epoch 25/25
1563/1563 [==============================] - 450s 288ms/step - loss: 0.5311 - accuracy: 0.8174 - val_loss: 0.7275
- val_accuracy: 0.7549
Accuracy: 75.49%
```

In [4]:

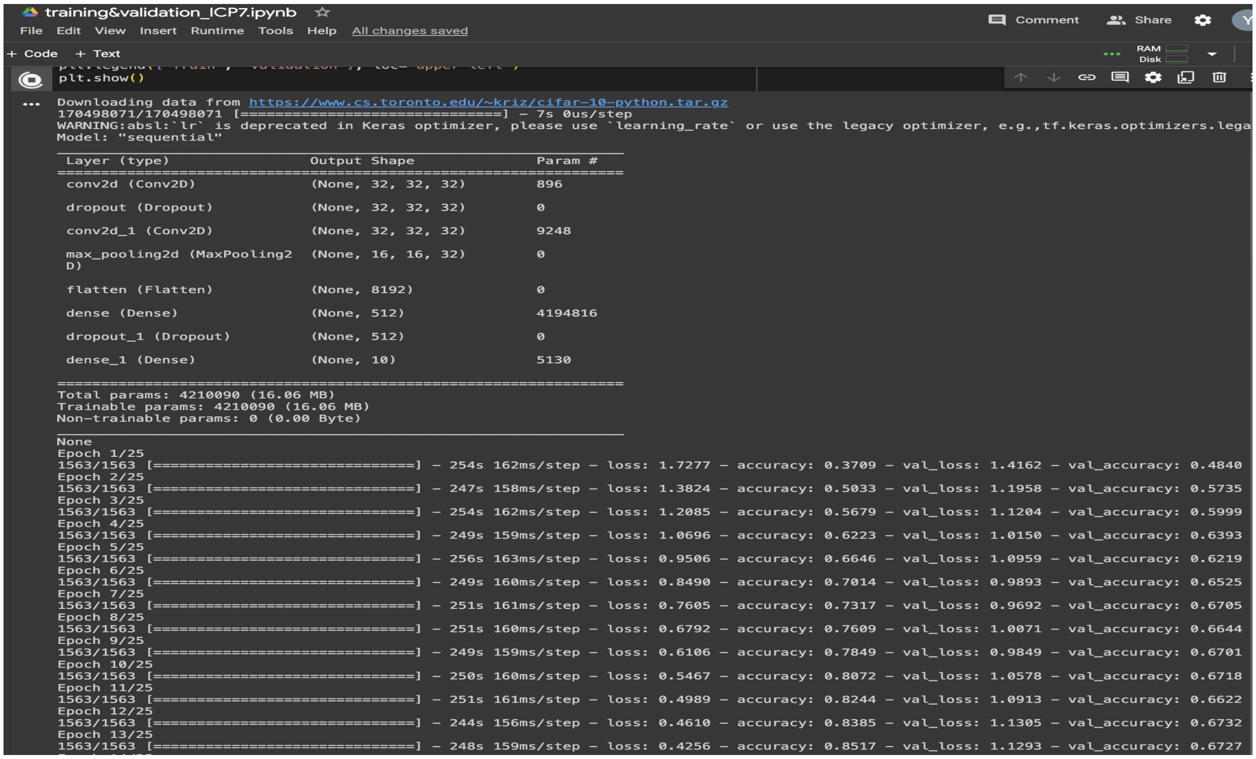
```
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

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





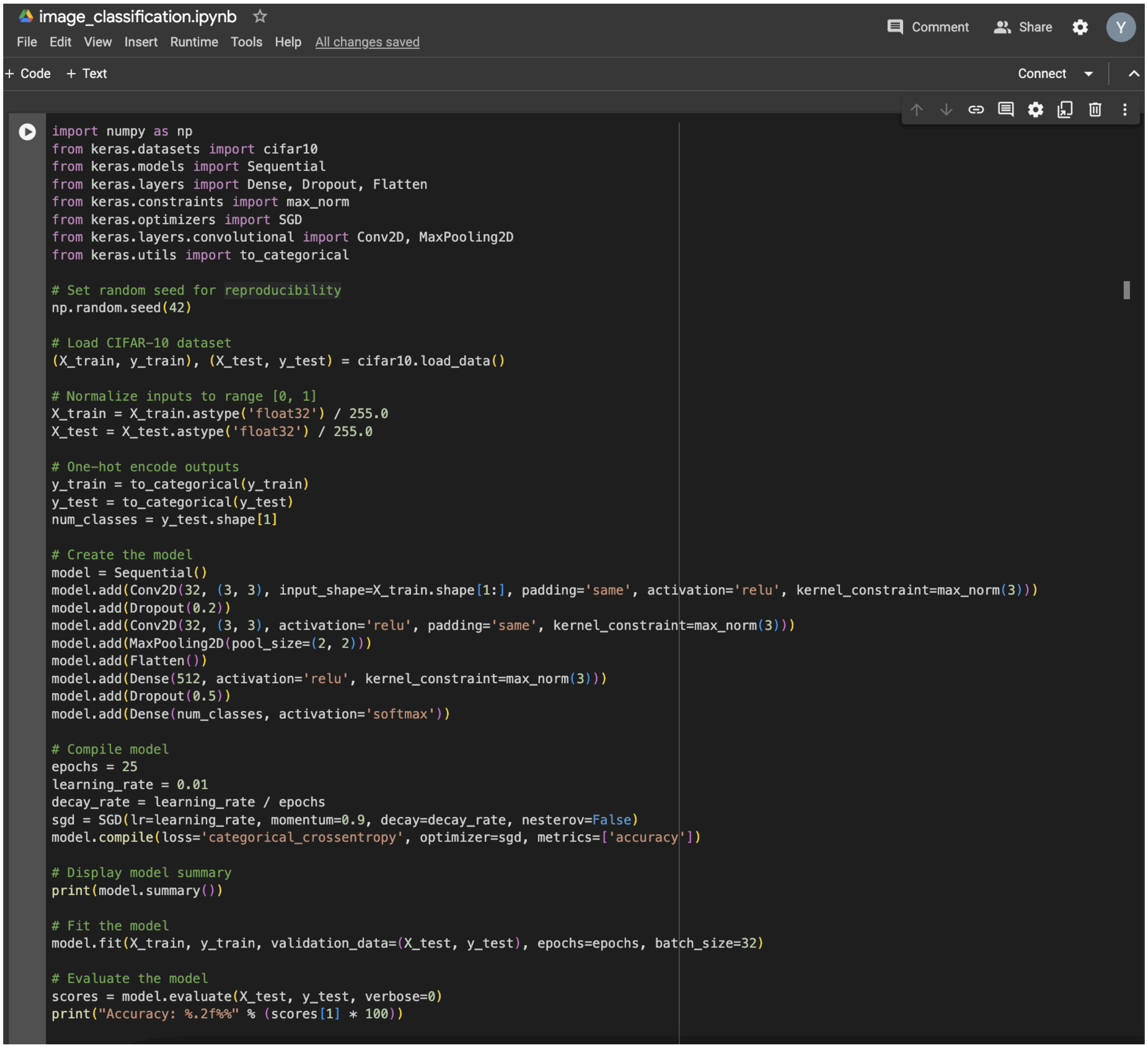

colab.research.google.com/drive/1kA4wxXoQ4xmQ0adWRlr9ISIKMJ9FwMT3
training&validation_ICP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM
Disk

```
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.constraints import MaxNorm
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# convert from int to float and normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# one hot encode outputs
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
sgd = SGD(lr=lrate, momentum=0.9, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

# Fit the model
```


training&validation_ICP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM
Disk

```
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 7s 0us/step
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.lega
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 4210090 (16.06 MB)
Trainable params: 4210090 (16.06 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/25
1563/1563 [==============================] - 254s 162ms/step - loss: 1.7277 - accuracy: 0.3709 - val_loss: 1.4162 - val_accuracy: 0.4840
Epoch 2/25
1563/1563 [==============================] - 247s 158ms/step - loss: 1.3824 - accuracy: 0.5033 - val_loss: 1.1958 - val_accuracy: 0.5735
Epoch 3/25
1563/1563 [==============================] - 254s 162ms/step - loss: 1.2085 - accuracy: 0.5679 - val_loss: 1.1204 - val_accuracy: 0.5999
Epoch 4/25
1563/1563 [==============================] - 249s 159ms/step - loss: 1.0696 - accuracy: 0.6223 - val_loss: 1.0150 - val_accuracy: 0.6393
Epoch 5/25
1563/1563 [==============================] - 256s 163ms/step - loss: 0.9506 - accuracy: 0.6646 - val_loss: 1.0959 - val_accuracy: 0.6219
Epoch 6/25
1563/1563 [==============================] - 249s 160ms/step - loss: 0.8490 - accuracy: 0.7014 - val_loss: 0.9893 - val_accuracy: 0.6525
Epoch 7/25
1563/1563 [==============================] - 251s 161ms/step - loss: 0.7605 - accuracy: 0.7317 - val_loss: 0.9692 - val_accuracy: 0.6705
Epoch 8/25
1563/1563 [==============================] - 251s 160ms/step - loss: 0.6792 - accuracy: 0.7609 - val_loss: 1.0071 - val_accuracy: 0.6644
Epoch 9/25
1563/1563 [==============================] - 249s 159ms/step - loss: 0.6106 - accuracy: 0.7849 - val_loss: 0.9849 - val_accuracy: 0.6701
Epoch 10/25
1563/1563 [==============================] - 250s 160ms/step - loss: 0.5467 - accuracy: 0.8072 - val_loss: 1.0578 - val_accuracy: 0.6718
Epoch 11/25
1563/1563 [==============================] - 251s 161ms/step - loss: 0.4989 - accuracy: 0.8244 - val_loss: 1.0913 - val_accuracy: 0.6622
Epoch 12/25
1563/1563 [==============================] - 244s 156ms/step - loss: 0.4610 - accuracy: 0.8385 - val_loss: 1.1305 - val_accuracy: 0.6732
Epoch 13/25
1563/1563 [==============================] - 248s 159ms/step - loss: 0.4256 - accuracy: 0.8517 - val_loss: 1.1293 - val_accuracy: 0.6727
```



```python
import numpy as np
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.constraints import max_norm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import to_categorical

# Set random seed for reproducibility
np.random.seed(42)

# Load CIFAR-10 dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Normalize inputs to range [0, 1]
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# One-hot encode outputs
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=X_train.shape[1:], padding='same', activation='relu', kernel_constraint=max_norm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=max_norm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=max_norm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
learning_rate = 0.01
decay_rate = learning_rate / epochs
sgd = SGD(lr=learning_rate, momentum=0.9, decay=decay_rate, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# Display model summary
print(model.summary())

# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)

# Evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))
```