```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

```
batch_size = 50
img_width, img_height, img_num_channels  = 32, 32, 3
loss_function = sparse_categorical_crossentropy
no_classes = 10
no_epochs = 100
optimizer = Adam()
validation_split = 0.2
verbosity = 1
```

```
(input_train, target_train), (input_test, target_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
```

```
# input_shape = (img_width, img_height, img_num_channels)
input_shape = (32, 32, 3)
```

```
input_train = input_train.astype('float32')
input_test = input_test.astype('float32')
```

```
input_train = input_train / 255
input_test = input_test / 255
```

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dense(128, activation='relu'))

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

```
model.compile(loss=loss_function, optimizer=optimizer, metrics=['accuracy'])
```

```
history = model.fit(input_train, target_train, batch_size=batch_size, epochs=10, verbose=verbosity, validation_split=validation_split)
```

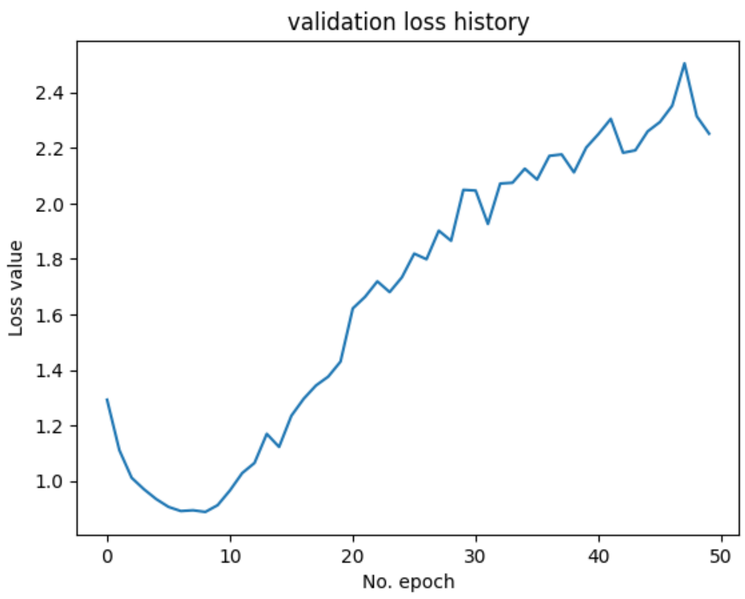
```
Epoch 1/50
800/800 [==============================] - 16s 6ms/step - loss: 1.5447 - accuracy: 0.4308 - val_loss: 1.2929 - val_accuracy: 0.5
Epoch 2/50
800/800 [==============================] - 5s 6ms/step - loss: 1.1855 - accuracy: 0.5760 - val_loss: 1.1113 - val_accuracy: 0.60
Epoch 3/50
800/800 [==============================] - 4s 5ms/step - loss: 1.0195 - accuracy: 0.6388 - val_loss: 1.0118 - val_accuracy: 0.64
Epoch 4/50
800/800 [==============================] - 4s 5ms/step - loss: 0.8995 - accuracy: 0.6804 - val_loss: 0.9705 - val_accuracy: 0.66
Epoch 5/50
800/800 [==============================] - 4s 6ms/step - loss: 0.8064 - accuracy: 0.7146 - val_loss: 0.9350 - val_accuracy: 0.68
Epoch 6/50
800/800 [==============================] - 4s 5ms/step - loss: 0.7366 - accuracy: 0.7395 - val_loss: 0.9069 - val_accuracy: 0.68
Epoch 7/50
800/800 [==============================] - 4s 5ms/step - loss: 0.6551 - accuracy: 0.7703 - val_loss: 0.8919 - val_accuracy: 0.69
Epoch 8/50
800/800 [==============================] - 5s 6ms/step - loss: 0.5966 - accuracy: 0.7885 - val_loss: 0.8946 - val_accuracy: 0.70
Epoch 9/50
800/800 [==============================] - 4s 5ms/step - loss: 0.5393 - accuracy: 0.8084 - val_loss: 0.8887 - val_accuracy: 0.71
Epoch 10/50
800/800 [==============================] - 4s 5ms/step - loss: 0.4829 - accuracy: 0.8295 - val_loss: 0.9131 - val_accuracy: 0.71
Epoch 11/50
800/800 [==============================] - 4s 6ms/step - loss: 0.4267 - accuracy: 0.8486 - val_loss: 0.9661 - val_accuracy: 0.71
Epoch 12/50
800/800 [==============================] - 4s 5ms/step - loss: 0.3808 - accuracy: 0.8671 - val_loss: 1.0287 - val_accuracy: 0.71
```

```
Epoch 13/50
800/800 [==============================] - 4s 5ms/step - loss: 0.3401 - accuracy: 0.8782 - val_loss: 1.0651 - val_accuracy: 0.71
Epoch 14/50
800/800 [==============================] - 4s 6ms/step - loss: 0.3020 - accuracy: 0.8923 - val_loss: 1.1700 - val_accuracy: 0.70
Epoch 15/50
800/800 [==============================] - 4s 5ms/step - loss: 0.2719 - accuracy: 0.9030 - val_loss: 1.1229 - val_accuracy: 0.71
Epoch 16/50
800/800 [==============================] - 4s 5ms/step - loss: 0.2379 - accuracy: 0.9148 - val_loss: 1.2351 - val_accuracy: 0.70
Epoch 17/50
800/800 [==============================] - 5s 6ms/step - loss: 0.2196 - accuracy: 0.9224 - val_loss: 1.2961 - val_accuracy: 0.71
Epoch 18/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1954 - accuracy: 0.9313 - val_loss: 1.3441 - val_accuracy: 0.70
Epoch 19/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1815 - accuracy: 0.9370 - val_loss: 1.3761 - val_accuracy: 0.69
Epoch 20/50
800/800 [==============================] - 5s 6ms/step - loss: 0.1846 - accuracy: 0.9345 - val_loss: 1.4301 - val_accuracy: 0.70
Epoch 21/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1564 - accuracy: 0.9424 - val_loss: 1.6218 - val_accuracy: 0.68
Epoch 22/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1424 - accuracy: 0.9497 - val_loss: 1.6636 - val_accuracy: 0.70
Epoch 23/50
800/800 [==============================] - 5s 6ms/step - loss: 0.1327 - accuracy: 0.9520 - val_loss: 1.7193 - val_accuracy: 0.70
Epoch 24/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1428 - accuracy: 0.9496 - val_loss: 1.6807 - val_accuracy: 0.70
Epoch 25/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1301 - accuracy: 0.9546 - val_loss: 1.7340 - val_accuracy: 0.69
Epoch 26/50
800/800 [==============================] - 4s 6ms/step - loss: 0.1211 - accuracy: 0.9578 - val_loss: 1.8193 - val_accuracy: 0.70
Epoch 27/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1184 - accuracy: 0.9588 - val_loss: 1.7989 - val_accuracy: 0.69
Epoch 28/50
800/800 [==============================] - 4s 5ms/step - loss: 0.1179 - accuracy: 0.9589 - val_loss: 1.9019 - val_accuracy: 0.69
Epoch 29/50
```

```
score = model.evaluate(input_test, target_test, verbose=0)
print(f'Test loss: {score[0]}/ Test accuracy: {score[1]}')
```

```
Test loss: 2.3247320652008057/ Test accuracy: 0.6887000203132629
```

```
plt.plot(history.history['val_loss'])
plt.title('validation loss history')
plt.ylabel('Loss value')
plt.xlabel('No. epoch')
plt.show()
```



```
model.compile(loss=loss_function, optimizer=sgd, metrics=['accuracy'])
```

```
history = model.fit(input_train, target_train, batch_size=batch_size, epochs=50, verbose=verbosity, validation_split=validation_split)
```

```
score = model.evaluate(input_test, target_test, verbose=0)
print(f'Test loss: {score[0]}/ Test accuracy: {score[1]}')
```

```
plt.plot(history.history['val_loss'])
plt.title('validation loss history')
plt.ylabel('Loss value')
plt.xlabel('No. epoch')
plt.show()
```