Choose optimal number of epochs to train a neural network in Keras

Difficulty Level : Easy ● Last Updated : 24 Aug, 2022

One of the critical issues while training a neural network on the sample data is **Overfitting**. When the number of epochs used to train a neural network model is more than necessary, the training model learns patterns that are specific to sample data to a great extent. This makes the model incapable to perform well on a new dataset. This model gives high accuracy on the training set (sample data) but fails to achieve good accuracy on the test set. In other words, the model loses generalization capacity by overfitting to the training data.

To mitigate overfitting and to increase the generalization capacity of the neural network, the model should be trained for an optimal number of epochs. A part of training data is dedicated for validation of the model, to check the performance of the model after each epoch of training. Loss and accuracy on the training set as well as on validation set are monitored to look over the epoch number after which the model starts overfitting.

keras.callbacks.callbacks.EarlyStopping()

Either loss/accuracy values can be monitored by Early stopping call back function. If the loss is being monitored, training comes to halt when there is an increment observed in loss values. Or, If accuracy is being monitored, training comes to halt when there is decrement observed in accuracy values.

Syntax with default values:

Understanding few important arguments:

- ***mode:*** *It is the mode in which change in the quantity monitored should be observed. This can be 'min' or 'max' or 'auto'. When the monitored value is loss, its value is 'min'. When the monitored value is accuracy, its value is 'max'. When the mode is set is 'auto', the function automatically monitors with the suitable mode.*
- ***min_delta:*** *The minimum value should be set for the change to be considered i.e., Change in the value being monitored should be higher than 'min_delta' value.*
- ***patience:*** *Patience is the number of epochs for the training to be continued after the first halt. The model waits for patience number of epochs for any improvement in the model.*
- ***verbose:*** *Verbose is an integer value-0, 1 or 2. This value is to select the way in which the progress is displayed while training.*
  - *Verbose = 0: Silent mode-Nothing is displayed in this mode.*
  - *Verbose = 1: A bar depicting the progress of training is displayed.*
  - *Verbose = 2: In this mode, one line per epoch, showing the progress of training per epoch is displayed.*
- ***restore_best_weights:*** *This is a boolean value. True value restores the weights which are optimal.*

Finding the optimal number of epochs to avoid overfitting on MNIST dataset.

Step 1: Loading dataset and preprocessing

```
import keras
from keras.utils.np_utils import to_categorical
from keras.datasets import mnist
# Loading data
(train_images, train_labels), (test_images, test_labels)= mnist.load_data()

# Reshaping data-Adding number of channels as 1 (Grayscale images)
```

```
                                    test_images.shape[1],
                                    test_images.shape[2], 1))
```



```
train_images = train_images.astype('float32')/255
test_images = test_images.astype('float32')/255

# Encoding labels to a binary class matrix
y_train = to_categorical(train_labels)
y_test = to_categorical(test_labels)
```

Step 2: Building a CNN model

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation ="relu",
                        input_shape =(28, 28, 1)))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Conv2D(64, (3, 3), activation ="relu"))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation ="relu"))
model.add(layers.Dense(10, activation ="softmax"))

model.summary()
```

Output: Summary of the model

```
[ ] model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                102464
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 121,930
Trainable params: 121,930
Non-trainable params: 0
_________________________________________________________________
```

Step 4: Compiling the model with RMSprop optimizer, categorical cross entropy loss function and accuracy as success metric

Step 5: Creating validation set and training set by partitioning the current training set

```
val_images = train_images[:10000]
partial_images = train_images[10000:]
val_labels = y_train[:10000]
partial_labels = y_train[10000:]
```

Step 6: Initializing earlystopping callback and training the model

```
from keras import callbacks
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

history = model.fit(partial_images, partial_labels, batch_size = 128,
                    epochs = 25, validation_data =(val_images, val_labels),
                    callbacks =[earlystopping])
```

```
[27] from keras import callbacks
```

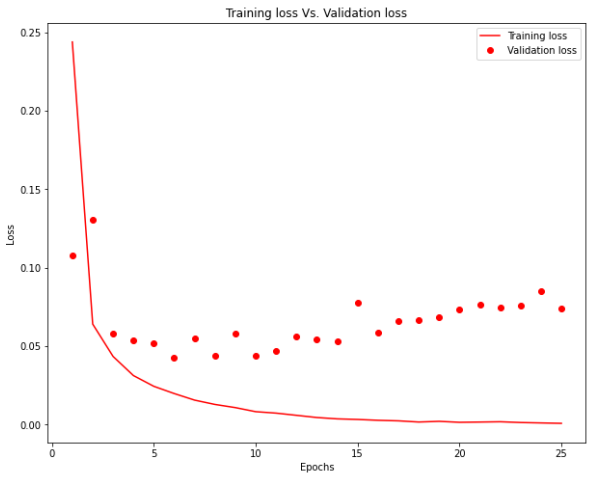
```
Epoch 1/25
50000/50000 [==============================] - 2s 39us/step - loss: 0.2556 - accuracy: 0.9213 - val_loss: 0.0874 - val_accuracy: 0.9745
Epoch 2/25
50000/50000 [==============================] - 2s 37us/step - loss: 0.0652 - accuracy: 0.9803 - val_loss: 0.0639 - val_accuracy: 0.9796
Epoch 3/25
50000/50000 [==============================] - 2s 38us/step - loss: 0.0435 - accuracy: 0.9864 - val_loss: 0.0455 - val_accuracy: 0.9860
Epoch 4/25
50000/50000 [==============================] - 2s 37us/step - loss: 0.0318 - accuracy: 0.9900 - val_loss: 0.0434 - val_accuracy: 0.9863
Epoch 5/25
50000/50000 [==============================] - 2s 36us/step - loss: 0.0251 - accuracy: 0.9917 - val_loss: 0.0471 - val_accuracy: 0.9871
Epoch 6/25
50000/50000 [==============================] - 2s 36us/step - loss: 0.0207 - accuracy: 0.9938 - val_loss: 0.0374 - val_accuracy: 0.9895
Epoch 7/25
50000/50000 [==============================] - 2s 37us/step - loss: 0.0164 - accuracy: 0.9949 - val_loss: 0.0491 - val_accuracy: 0.9870
Epoch 8/25
50000/50000 [==============================] - 2s 36us/step - loss: 0.0128 - accuracy: 0.9961 - val_loss: 0.0462 - val_accuracy: 0.9888
Epoch 9/25
50000/50000 [==============================] - 2s 36us/step - loss: 0.0108 - accuracy: 0.9967 - val_loss: 0.0481 - val_accuracy: 0.9882
Epoch 10/25
50000/50000 [==============================] - 2s 36us/step - loss: 0.0085 - accuracy: 0.9974 - val_loss: 0.0443 - val_accuracy: 0.9887
Epoch 11/25
50000/50000 [==============================] - 2s 36us/step - loss: 0.0066 - accuracy: 0.9980 - val_loss: 0.0435 - val_accuracy: 0.9899
Restoring model weights from the end of the best epoch
Epoch 00011: early stopping
```

Training stopped at 11th epoch i.e., the model will start overfitting from 12th epoch. **Therefore, the optimal number of epochs to train most dataset is 11.**

Observing loss values without using Early Stopping call back function:

Train the model up until 25 epochs and plot the training loss values and validation loss values against number of epochs. The plot looks like:

```
[42] <Figure size 432x288 with 0 Axes>
```



Inference:

As the number of epochs increases beyond 11, training set loss decreases and becomes nearly zero. Whereas, validation loss increases depicting the overfitting of the model on training data.

References:

Next

keras.fit() and keras.fit_generator()

Related Articles

1. keras.fit() and keras.fit_generator()
2. Python Keras | keras.utils.to_categorical()
3. Building a Generative Adversarial Network using Keras
4. How to train a network using trainers in PyBrain
5. Introduction to Artificial Neural Network | Set 2
6. A single neuron neural network in Python
7. Applying Convolutional Neural Network on mnist dataset
8. Importance of Convolutional Neural Network | ML
9. Neural Network Advances
10. ML - Neural Network Implementation in C++ From Scratch

manmayi

Read Discuss Courses Practice Video

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Article Tags : Deep-Learning, Neural Network, Machine Learning, Python

Practice Tags : Machine Learning, python

Improve Article Report Issue

A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

News

Work & Career

Business

Finance

Lifestyle

Knowledge

Languages

CPP

Golang

C#

SQL

Kotlin

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

NodeJS

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship