

Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-7

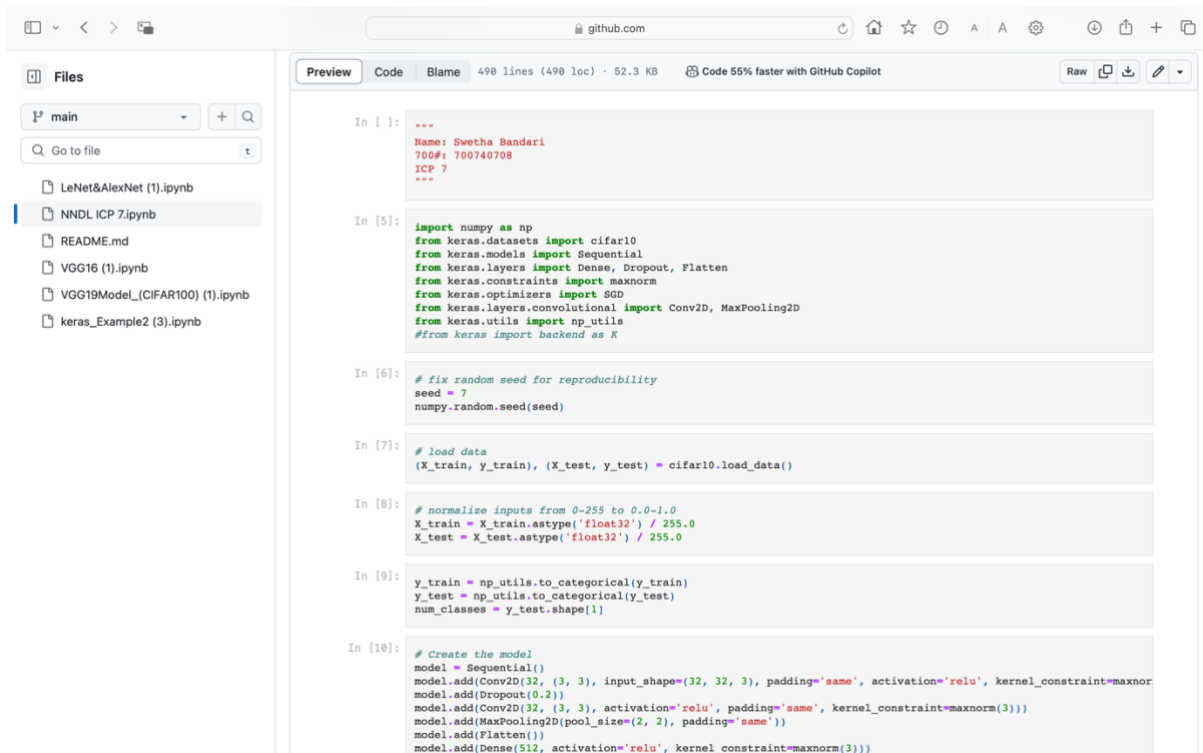
Name – Bandari Swetha

700# – 700740708

UCM ID – sxb07080

GitHub link –

<https://github.com/sxb07080/NNDL-ICP-7>



```
In [ ]: """
Name: Swetha Bandari
700#: 700740708
ICP 7
"""

In [5]: 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 maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
#from keras import backend as K

In [6]: # fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)

In [7]: # load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

In [8]: # 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

In [9]: y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]

In [10]: # 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), padding='same'))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
# ...
```

Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_(CIFAR100) (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

Raw

In [10]:

Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm))
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), padding='same'))
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'))

In [11]:

sgd = SGD(learning_rate=0.01, momentum=0.9, decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 32, 32, 32)	896
dropout_2 (Dropout)	(None, 32, 32, 32)	0
conv2d_3 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_2 (Dense)	(None, 512)	4194816
dropout_3 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 10)	5130

Total params: 4,210,090
Trainable params: 4,210,090
Non-trainable params: 0

None

Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_(CIFAR100) (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

Raw

None

In [12]:

epochs = 5
batch_size = 32
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size)
Fit the model

Epoch 1/5
1563/1563 [=====] - 19s 7ms/step - loss: 1.7232 - accuracy: 0.3746 - val_loss: 1.4776 - val_accuracy: 0.4563
Epoch 2/5
1563/1563 [=====] - 10s 6ms/step - loss: 1.3675 - accuracy: 0.5117 - val_loss: 1.2470 - val_accuracy: 0.5551
Epoch 3/5
1563/1563 [=====] - 10s 6ms/step - loss: 1.2071 - accuracy: 0.5716 - val_loss: 1.1232 - val_accuracy: 0.6047
Epoch 4/5
1563/1563 [=====] - 10s 7ms/step - loss: 1.0855 - accuracy: 0.6136 - val_loss: 1.1554 - val_accuracy: 0.5928
Epoch 5/5
1563/1563 [=====] - 10s 7ms/step - loss: 0.9709 - accuracy: 0.6583 - val_loss: 0.9986 - val_accuracy: 0.6550

Out[12]: <keras.callbacks.History at 0x7f689d6d65e0>

In [13]:

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

Accuracy: 65.50%

In [14]:

import numpy as np
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.constraints import maxnorm
from keras.utils import np_utils
from keras.optimizers import SGD

Fix random seed for reproducibility
np.random.seed(7)

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

Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_(CIFAR100) (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

Raw

```
# 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') / 255.0
X_test = X_test.astype('float32') / 255.0

# One hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.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), activation='relu', padding='same', 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), activation='relu', padding='same', 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 = 5
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'])
print(model.summary())

# Fit the model
history = 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)
```

Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_(CIFAR100) (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

Raw

```
print(model.summary())

# Fit the model
history = 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))
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 32)	896
dropout_4 (Dropout)	(None, 32, 32, 32)	0
conv2d_5 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 32)	0
conv2d_6 (Conv2D)	(None, 16, 16, 64)	18496
dropout_5 (Dropout)	(None, 16, 16, 64)	0
conv2d_7 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_3 (MaxPooling 2D)	(None, 8, 8, 64)	0
conv2d_8 (Conv2D)	(None, 8, 8, 128)	73856
dropout_6 (Dropout)	(None, 8, 8, 128)	0
conv2d_9 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_4 (MaxPooling 2D)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dropout_7 (Dropout)	(None, 2048)	0
dense_4 (Dense)	(None, 1024)	2098176
dropout_8 (Dropout)	(None, 1024)	0

Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_CIFAR100 (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

Raw

conv2d_9 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_4 (MaxPooling 2D)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dropout_7 (Dropout)	(None, 2048)	0
dense_4 (Dense)	(None, 1024)	2098176
dropout_8 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 512)	524800
dropout_9 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 10)	5130

Total params: 2,915,114
Trainable params: 2,915,114
Non-trainable params: 0

None
Epoch 1/5
1563/1563 [=====] - 15s 9ms/step - loss: 1.9322 - accuracy: 0.2796 - val_loss: 1.6108 - val_accuracy: 0.4168
Epoch 2/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.5375 - accuracy: 0.4379 - val_loss: 1.4261 - val_accuracy: 0.4795
Epoch 3/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.3979 - accuracy: 0.4918 - val_loss: 1.3406 - val_accuracy: 0.5164
Epoch 4/5
1563/1563 [=====] - 13s 8ms/step - loss: 1.3128 - accuracy: 0.5217 - val_loss: 1.2901 - val_accuracy: 0.5367
Epoch 5/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.2504 - accuracy: 0.5459 - val_loss: 1.1804 - val_accuracy: 0.5735
Accuracy: 57.35%

In [18]:
Predict the first 4 images of the test data
predictions = model.predict(X_test[:4])
Convert the predictions to class labels
predicted_labels = numpy.argmax(predictions, axis=1)

Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_CIFAR100 (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

Raw


Epoch 5/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.2504 - accuracy: 0.5459 - val_loss: 1.1804 - val_accuracy: 0.5735
Accuracy: 57.35%

In [18]:
Predict the first 4 images of the test data
predictions = model.predict(X_test[:4])
Convert the predictions to class labels
predicted_labels = numpy.argmax(predictions, axis=1)
Convert the actual labels to class labels
actual_labels = numpy.argmax(y_test[:4], axis=1)
Print the predicted and actual labels for the first 4 images
print("Predicted labels:", predicted_labels)
print("Actual labels: ", actual_labels)

1/1 [=====] - 0s 21ms/step
Predicted labels: [3 8 8 8]
Actual labels: [3 8 8 0]

In [19]:
import matplotlib.pyplot as plt
Plot the training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()

Plot the training and validation accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='lower right')
plt.show()



Files

main

Go to file

LeNet&AlexNet (1).ipynb

NNDL ICP 7.ipynb

README.md

VGG16 (1).ipynb

VGG19Model_(CIFAR100) (1).ipynb

keras_Example2 (3).ipynb

NNDL-ICP-7 / NNDL ICP 7.ipynb

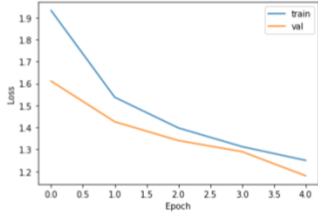
498 lines (498 loc) · 52.3 KB

Code 55% faster with GitHub Copilot

RawDownloadEdit

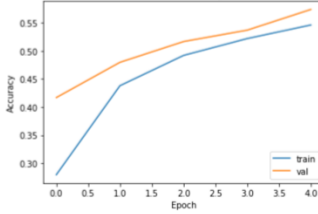
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='lower right')
plt.show()
```

Model Loss



Epoch	train	val
0.0	1.90	1.60
1.0	1.55	1.45
2.0	1.45	1.35
3.0	1.35	1.25
4.0	1.25	1.15

Model Accuracy



Epoch	train	val
0.0	0.28	0.42
1.0	0.45	0.48
2.0	0.50	0.52
3.0	0.52	0.54
4.0	0.54	0.57