11/5/2024

MET CS 767 Assignment 3: CNN’s

Jake Giguere <giguere@bu.edu>

The purpose of this assignment is to give you experience with deep convolutional neural networks.

The overall instructions are the same as for Assignment 2 (red font, attachment, inserted comments about your noteworthy prompts, etc.).

# Modification of code to attempt improvement (2 page max)

Copy the implementation [here](https://colab.research.google.com/drive/1Yg-NXKlYzfvv9jI2MCxWZErovpR57_YC?usp=sharing) to your Google drive. Systematically modify the parameter values, attempting to improve the output, and report the results as below. Since the accuracy of the given implementation is already high, consider reducing the size of the CIFAR training set—or substituting parts of it so that the baseline implementation leaves more percentage room for improvement. If it adds to clarity, describe and explain changes that make the result worse.

## 1.1 Description of changes and reason they *could reasonably be* an improvement (at most one page)

1. The first change would be to increase the number of epochs. Epochs give more time for the model to learn and classify the images accurately.

2. Adding padding to the images will help the kernel create less bias to a group of pixels.

3. Another option would be to test the kernel size, and see how that impacts learning. The kernel size can be more accurate if smaller, at the cost of longer learning time. Potentially there could be no different from 3x3, 2x2, 1x1

## 1.2 Comparison of the result with the original output, with explanation

With a 3x3 kernel, no padding, and epochs set to 10, the accuracy increased to 80% compared to the original output of 58.12% with 2 epochs

After downloading the colab notebook. I was able to utilize my personal computers 8 cores to thread. The original output was training each epoch at roughly 83 seconds. With threading I was able to train on average 40 sec / epoch.

tf.config.threading.set\_intra\_op\_parallelism\_threads(8)

tf.config.threading.set\_inter\_op\_parallelism\_threads(8)

model.compile(optimizer='adam',

loss=keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

history = model.fit(train\_images, train\_labels, epochs=10,

validation\_data=(test\_images, test\_labels))

After adding padding to the Conv2D layers, the accuracy increased to 88.45%. Given 5 more epochs would expect +90% overall accuracy. This increased accuracy is due to each pixel is getting looked at evenly with padding. Without the padding the pixels in the middle will have more bias and have been looped through more than the outer pixels.

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3), padding='same'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))

With the power of M3 MAX chip and ChatGPT, I was also able to use tf to the fullest capacity.

ChatGPT prompt: *lets take advantage of this M3 Max chip*

ChatGPT responses:

# Enable memory growth to prevent TensorFlow from consuming all GPU memory

gpus = tf.config.list\_physical\_devices('GPU')

if gpus:

try:

for gpu in gpus:

tf.config.experimental.set\_memory\_growth(gpu, True)

print("Memory growth enabled for GPUs")

except RuntimeError as e:

print(e)

Also transforming data into a tf data set and using AUTOTUNE to automatically use the optimal number of threads for data loading.

# Convert data to TensorFlow Dataset

train\_ds = tf.data.Dataset.from\_tensor\_slices((train\_images, train\_labels))

test\_ds = tf.data.Dataset.from\_tensor\_slices((test\_images, test\_labels))

# Use parallel data loading and prefetching

AUTOTUNE = tf.data.AUTOTUNE

train\_ds = (

train\_ds

.shuffle(buffer\_size=10000)

.batch(64)

.map(lambda x, y: (tf.expand\_dims(x, -1), y), num\_parallel\_calls=AUTOTUNE)

.prefetch(buffer\_size=AUTOTUNE)

)

test\_ds = (

test\_ds

.batch(64)

.map(lambda x, y: (tf.expand\_dims(x, -1), y), num\_parallel\_calls=AUTOTUNE)

.prefetch(buffer\_size=AUTOTUNE)

)

## 1.3 URL of your Colab code

your response replaces this

# 2. Your CNN Application (3 pg max)

## 2.1 Give 2-4 requirements for a highly capable, unique application you’ll implement

## These describe *what* functionality your application will provide for the user, including the nature of inputs and outputs. This section should not include *how* you will design or code the application.

your response replaces this

## 2.2 Uniqueness of Your Application

## Explain what’s unique about your application (not the architecture).

your response replaces this

## 2.3 Sample I/O

## Give three varied input/outputs pairs for your *completed* application.

your response replaces this

## 2.4 The CNN Architecture

## Show your architecture in one or more annotated figures.

your response replaces this

## 2.5 Uniqueness of Your Architecture

## Explain what’s unique about your CNN architecture

your response replaces this

## 2.6 Key code

## Provide snippets of the essential core code of your implementation.

your response replaces this

## 2.7 URL of your Colab code

your response replaces this

# References

You are welcome to use the work of others—but only if you clearly indicate what work is theirs. Failure to do so is plagiarism. Each of your references should occur within the text. For example, [1] should occur below *and* within the body of your response at the relevant location. Include specific sections of the textbooks if used directly.

[1] your first reference replaces this

[2] …

# Evaluation



# Appendix 1

…

# Appendix 2…