Alessandro Allegranzi

BU MET CS 767

Assignment 3

5/27/2024

Table of Contents

[1. Modification of code to attempt improvement (2 page max) 1](#_Toc167383165)

[1.1 Description of changes and reason they *could reasonably be* an improvement (at most one page) 1](#_Toc167383166)

[1.2 Comparison of the result with the original output, with explanation 1](#_Toc167383167)

[1.3 URL of your Colab code 1](#_Toc167383168)

[2. Your CNN Application (3 pg max) 1](#_Toc167383170)

[2.1 Give 2-4 requirements for an application that you will implement 1](#_Toc167383171)

[2.2 Sample I/O 1](#_Toc167383173)

[2.3 The CNN Architecture 1](#_Toc167383175)

[2.3 Key code 2](#_Toc167383177)

[2.4 URL of your Colab code 2](#_Toc167383179)

[References 2](#_Toc167383181)

[Evaluation 2](#_Toc167383182)

[Appendix 1 2](#_Toc167383183)

[Appendix 2… 2](#_Toc167383184)

5/27/2024

MET CS 767 Assignment 3: CNN’s

*Alessandro Allegranzi*

Applied machine learning—after architecture selection—is largely a process of selecting parameter values. This requires an understanding of the parameters, and a systematic approach to dealing with parameter values. One purpose of this assignment is to give you experience with this. The other is to give you practice creating a deep convolutional neural network.

The overall instructions are the same as in prior assignments.

# 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)

## 

**Making model less effective:**

Use a simpler activation function:

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

# Next layer: max over 2x2 non-overlapping squares

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

# Typically increase kernel count with depth (to 64 here).

model.add(layers.Conv2D(64, (3, 3), activation='linear'))

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

model.add(layers.Conv2D(64, (3, 3), activation='linear'))

Increase learning rate:

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.1),

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

# how often predictions equal labels ("https://keras.io/api/metrics/")

metrics=['accuracy'])

history = model.fit(train\_images, train\_labels, epochs=2, #?? SET BACK TO 10

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

The above changes lowered the model accuracy to under 40%, which has more room to improve compared to the original.

print(test\_acc)

0.37790000438690186

**Making model more effective:**

Increase number of kernels and add layer:

model.add(layers.Conv2D(128, (3, 3), activation='relu', input\_shape=(32, 32, 3)))

# Next layer: max over 2x2 non-overlapping squares

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

# Typically increase kernel count with depth (to 64 here).

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

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

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

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

model.add(layers.Flatten())

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

model.add(layers.Dense(10))

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_8 (Conv2D) (None, 30, 30, 128) 3584

max\_pooling2d\_6 (MaxPoolin (None, 15, 15, 128) 0

g2D)

conv2d\_9 (Conv2D) (None, 13, 13, 256) 295168

max\_pooling2d\_7 (MaxPoolin (None, 6, 6, 256) 0

g2D)

conv2d\_10 (Conv2D) (None, 4, 4, 256) 590080

conv2d\_11 (Conv2D) (None, 2, 2, 256) 590080

max\_pooling2d\_8 (MaxPoolin (None, 1, 1, 256) 0

g2D)

flatten\_1 (Flatten) (None, 256) 0

dense\_2 (Dense) (None, 256) 65792

dense\_3 (Dense) (None, 10) 2570

=================================================================

Total params: 1547274 (5.90 MB)

Trainable params: 1547274 (5.90 MB)

Non-trainable params: 0 (0.00 Byte)

Using RMSProp as the optimizer function [2], increase epochs to 8, and use early stopping and model checkpoint callback functions. Early stopping stops the training if the trigger sets off, in this case I’ve set it to stop if the validation loss stops improving for 3 epochs. Model checkpoint saves the best model in case the training is interrupted [3]:

model.compile(optimizer=tf.keras.optimizers.RMSprop(),

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

# how often predictions equal labels ("https://keras.io/api/metrics/")

metrics=['accuracy'])

early\_stopping = EarlyStopping(monitor='val\_loss', patience=2)

model\_checkpoint = ModelCheckpoint('best\_model.h5', monitor='val\_loss', save\_best\_only=True)

history = model.fit(train\_images, train\_labels, epochs=8, #?? SET BACK TO 10

validation\_data=(test\_images, test\_labels), callbacks=[early\_stopping, model\_checkpoint])

The accuracy improved considerably to over 70%. However, the model took much longer to train.

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

Original Model:

313/313 - 4s - loss: 1.0287 - accuracy: 0.6383 - 4s/epoch - 12ms/step

A graph with a line and a line

Description automatically generated with medium confidence

Accuracy: 0.6287000179290771

Model After Improvements:

313/313 - 27s - loss: 1.1035 - accuracy: 0.7144 - 27s/epoch - 85ms/step

A graph with a line

Description automatically generated

Accuracy: 0.7143999934196472

The model stopped training after the 7th epoch due to earlyStopping.

## 1.3 URL of your Colab code

[Google Colab Code](https://colab.research.google.com/drive/1X06PQbgVkasBkGX-j8jx_ZVRhYqwWdx1?usp=sharing)

### >>AI generation for section 1 (or check: I did not use AI generation here \_X\_). Please collapse this.

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO AI-GENERATED MATERIAL

Your response replaces this.

YOUR PROMPT SEQUENCE

[1] Your first prompt replaces this.

[2]

Your response replaces this.

# 

# 2. Your CNN Application (3 pg max)

## 2.1 Give 2-4 requirements for an application that you will 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 Sample I/O

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

your response replaces this

## 2.3 The CNN Architecture

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

your response replaces this

## 2.3 Key code

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

your response replaces this

## 2.4 URL of your Colab code

your response replaces this

### >>AI generation for section 2 (or check: I did not use AI generation here \_\_). Please collapse this.

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO AI-GENERATED MATERIAL

Your response replaces this.

YOUR PROMPT SEQUENCE

[1] Your first prompt replaces this.

[2]

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] Geron, Aurelien. *Hands-On Machine Learning with Scikit-learn, Keras, & Tensorflow Third Edition*. “Chapter 10: Introduction to Artificial Neural Networks with Keras”. O’Reilly Media, Inc. October, 2022

[2] *RMSProp*. Keras Docs. https://keras.io/api/optimizers/rmsprop/

[3] *Early Stopping*. Keras Docs. https://keras.io/api/callbacks/early\_stopping/

# Evaluation



# Appendix 1

…

# Appendix 2…