

Week 4

SUBFORUMS
All
Assignment: Exercise 4 - Multi-class classifier

← Week 4

SP Accuracy low

Sarvesh Perumunda · Assignment: Exercise 4 - Multi-class classifier · a year ago

I have tried a lot of different ways and even used the suggestions from the discussions but my accuracy rate is remaining low. I'm attaching my code with this message. It would be really helpful if someone please send their right code, it's been a long and frustrating time but there is no concrete solution I can come up with. Please help. Thanks in Advance

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(26, activation=tf.nn.softmax)])

# Compile model.
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.1),
              loss = 'sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the Model
history = model.fit_generator(train_datagen.flow(training_images, training_labels, batch_size=32),
                             steps_per_epoch=len(training_images) / 32,
                             epochs=2,
                             validation_data=validation_datagen.flow(testing_images, testing_labels, batch_size=32),
                             validation_steps=len(testing_images) / 32)

model.evaluate(testing_images, testing_labels)
```

6 Upvotes Reply Follow this discussion

Earliest Top Most Recent

ID **Indraneel Dabhade** · 4 months ago

Use SGD(learning_rate = 0.0001). The grader does not check for augmentation, hence advice relating to neglecting augmentation works. Optionally use 'tanh' for activation and kernel_regularizer='l1'. Also, there are 24 classes, not 26. Can simply replace 24.0 with 9.0. Also, don't hardcode the final layer density of neurons. Use len(set(labels)) and pass on as a variable. Lower dropout to 0.1.

0 Upvotes Reply




hamed abbaszadeh · 6 months ago

Thanks for the hints in this thread, I could solve my code's accuracy issue. However, I could not understand why

- 1. batch_size=32 and consequently //32 were used.
- 2. why 26 classes while len(set(training_labels)) gives me only 24 (came for

Any idea?

⬆ 0 Upvotes  Reply

SX **Soh Wei Xiang** · 6 months ago


1. Adding this "steps_per_epoch to be len(training_images) // 32" helps to improve the accuracy tremendously.

⬆ 0 Upvotes  Reply

CW **Chow Jun Wei** · 7 months ago


Yes remove image augmentation. I only use one set of Convolution of 256 and a Max Pooling Layer and the accuracy increases too. Try check model.summary() and see if you can alter the Convolution and Pooling layer so the input before Flatten is not too low.

P.S. I get 0.99 accuracy in 2 epoch (as I use lots of neurons, not to say that the more the better as overfitting is always the concern).

⬆ 4 Upvotes  Reply


GT **Gaurav Thakur** · 7 months ago

I tried on my PC and only after 25 epochs I was getting 0.92 accuracy

⬆ 0 Upvotes  Reply

BL **Bernardin LIGAN** · 8 months ago


None of these solutions worked for me. I think we have to increase the epochs because in addition I get a loss of about 100

⬆ 0 Upvotes  Reply

HT **HIMANSHU TYAGI** · 9 months ago

Why is output layer number is 26 and not 3

⬆ 0 Upvotes  Hide 3 Replies

 **Vishwa Prabhakar Singh** · 9 months ago

the lecture lead us to believe that the assignment was for rock paper scissors when it's actually for sign language, which has 24 categories (from [0..24] except for [9]) instead of 3 in RPS ,so we made the last layers with only 3 neurons instead of 24

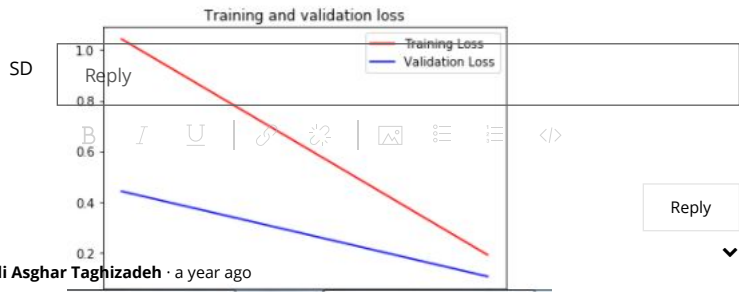
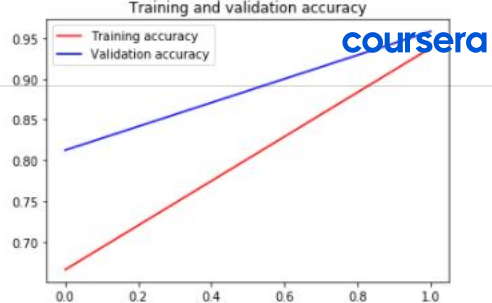
⬆ 3 Upvotes

DD **Dipankar Rahul Dey** · 6 months ago

Try with less image augmentation, I think more epochs are needed when use image augmentation, with 2 epochs it's not giving a good accuracy

⬆ 0 Upvotes

DD **Dipankar Rahul Dey** · 6 months ago



AT **Ali Asghar Taghizadeh** · a year ago

In my case, there was already a decrease in the training and validation loss, but the accuracy of the model was not improved. I solved the problem by increasing the number of epochs to more than 8 and the problem was solved.

0 Upvotes 0 Upvotes Reply



Derek Chia · a year ago

Here are some tips - considering that the train and test data are very similar:

1. Remove augmentation
2. try 128 and 64 number of filters for both Conv2D
3. Remove dropout
4. Dense layer to be 128 and 26
5. Use Adam optimizer with lr 0.001
6. steps_per_epoch to be len(training_images) // 32

Hope that helps.

12 Upvotes Hide 2 Replies

SH **SIRINE HEDFI** · 9 months ago

low accuracy even after applying the tips any suggestion ??

0 Upvotes

DW **David Woroniuk** · 7 months ago

Some more tips:

1) Add a second fully connected (Dense) layer, with double the number of neurons (think 512, 1024) prior to the final 'output' layer:

```
1 tf.keras.layers.Flatten(),
2 tf.keras.layers.Dropout(0.2),
3 tf.keras.layers.Dense(512, activation='relu'),
4 tf.keras.layers.Dense(1024, activation='relu'),
5 tf.keras.layers.Dense(25, activation='softmax')]
```

2) Remove augmentation

3) Adaptive movement (Adam) optimizer as opposed to RMSprop.

4) Vary the dropout value. Using 0.5, I managed to get a 90% validation set accuracy using 2 epochs.

0 Upvotes

SD Reply

B I U | 🔗 🔄 | 🖼️ ☰ ☷ </>



JS **Jayaram Subramanian** · a year ago

Remove data augumentation and try. It worked for me

↑ 1 Upvote Reply



MV **Malini Vyakaranam** · a year ago

Learning rate is a parameter that you can reduce..0.1 seems high for this dataset

↑ 1 Upvote Reply



SD

Reply

B *I* U | | </>

Reply



































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































































