1 Answer

Sorted by: Highest score (default)

92

The validation generator works exactly like the training generator. You define how many batches it will wield per epoch.

- The training generator will yield `steps_per_epoch` batches.
- When the epoch ends, the validation generator will yield `validation_steps` batches.

But validation data has absolutely no relation to training data. There is no need to separate validation batches according to training batches (I would even say that there is no point in doing that, unless you have a very specific intention). Also, the total number of samples in training data is not related to the total number of samples in test data.

The point of having many batches is just to spare your computer's memory, so you test smaller packs one at a time. Probably, you find a batch size that will fit your memory or expected training time and use that size.

That said, Keras gives you a totally free method, so you can determine the training and the validation batches as you wish.

Epochs:

Ideally, you use all your validation data at once. If you use only part of your validation data, you will get different metrics for each batch, what may make you think that your model got worse or better when it actually didn't, you just measured different validation sets.

That's why they suggest `validation_steps = total_validation_samples // validation_batch_size`. Theoretically, you test your entire data every epoch, as you theoretically should also train your entire data every epoch.

So, theorethycally each epoch yields:

- `steps_per_epoch = TotalTrainingSamples / TrainingBatchSize`
- `validation_steps = TotalvalidationSamples / ValidationBatchSize`

Basically, the two vars are: how many batches per epoch you will yield.
This makes sure that at each epoch:

- You train exactly your entire training set
- You validate exactly your entire validation set

Nevertheless, it's totally up to you how you separate your training and validation data.

If you do want to have one different batch per epoch (epochs using less than your entire data), it's ok, just pass `steps_per_epoch=1` or `validation_steps=1`, for instance. The generator is not resetted after each epoch, so the second epoch will take the second batch, and so on, until it loops again to the first batch.

I prefer training the entire data per epoch, and if the time is too long, I use a `callback` that shows the logs at the end of each batch:

```
batch,logs:print(logs))]
```

Multiprocessing

I was never able to use `use_multiprocessing=True`, it freezes at the start of the first epoch.

I've noticed the `workers` are related to how many batches are preloaded from the generator. If you define `max_queue_size=1`, you will have exactly `workers` amount of batches preloaded.

They suggest you use keras Sequences when multiprocessing. The sequences work pretty much as a generator, but it keeps track of the order/position of each batch.

Share Follow

edited Jul 23, 2019 at 19:12

answered Aug 29, 2017 at 16:35

Daniel Möller
81.5k ● 17 ● 178 ● 200

thanks a lot. I'm aware that training and validation data are not directly related. i simply got confused by what the doc parameter descriptions really meant for me. also thanks for the clarification concerning the optimal use of validation batches and multiprocessing. – Philipp Lange Aug 29, 2017 at 17:57

1 I did some correction in the `step` vars above, they're divided by the batch size instead of the number of batches. All the idea is unchanged, just the formula was wrong. – Daniel Möller Aug 29, 2017 at 21:29

@DanielMöller Still I am confused with your answer. Lets say I set my `steps_per_epochs = 25 & epoch= 100 & validation_step = 3`. For every epoch, there were 25 steps and for each step, generator yielded training data of shape `X_train : (233, 100, 4) & Y_train : (233, 100, 2)` and training happens. The above process continues for every 25 steps and at the end of 25th step validation starts where the generator yield `X_validate: (33,100,4) & Y_validate : (33, 100, 2)` `3 times` and `validation acc & loss` printed in result. – Mari Jul 23, 2019 at 18:30

@DanielMöller My question is : 1. What will be `batch_size in my case (for both training & Validation)` ? 2. During validation, the generator yields `3 times X_validate & Y_validate arrays` ,since i have given `validation_steps = 3`. So how does loss and val_acc calculated ? Whether it will be calculated for every step & finally average the results ? or some other method ? – Mari Jul 23, 2019 at 18:44

Batch size = 233 and 33 respectively. I'm not sure how Keras calculates the loss. Probably average of each batch. – Daniel Möller Jul 23, 2019 at 18:48