Choosing number of Steps per Epoch

Asked 4 years, 3 months ago Modified 1 year, 8 months ago Viewed 55k times

29

25

If I want to train a model with train_generator, is there a significant difference between choosing

- 10 Epochs with 500 Steps each

and

- 100 Epochs with 50 Steps each

Currently I am training for 10 epochs, because each epoch takes a long time, but any graph showing improvement looks very "jumpy" because I only have 10 datapoints. I figure I can get a smoother graph if I use 100 Epochs, but I want to know first if there is any downside to this

tensorflow machine-learning keras deep-learning neural-network

Share Improve this question

Follow

edited Nov 9, 2020 at 17:09

Chris Farr

3,291 ● 1 ● 20 ● 23

asked Apr 19, 2018 at 13:23

n.st

355 ● 1 ● 5 ● 10

3 Answers

Sorted by:

Trending sort available

Highest score (default)

62

Based on what you said it sounds like you need a larger `batch_size`, and of course there are implications with that which could impact the steps_per_epoch and number of epochs.

To solve for jumping-around

- **A larger batch size** will give you a better gradient and will help to prevent jumping around
- You may also want to consider a smaller learning rate, or a learning rate scheduler (or decay) to allow the network to "settle in" as it trains

Implications of a larger batch-size

- Too large of a batch_size can produce memory problems, especially if you are using a GPU. Once you exceed the limit, dial it back until it works. This will help you find the max batch-size that your system can work with.
- Too large of a batch size can get you stuck in a local minima, so if your training get stuck, I would reduce it some. Imagine here you are over-correcting the *jumping-around* and it's not jumping around enough to further minimize the loss function.

When to reduce epochs

- If your train error is very low, yet your test/validation is very high, then you have over-fit the model with too many epochs.
- The best way to find the right balance is to use early-stopping with a validation test set. Here you can specify when to stop training, and save the weights for the network that gives you the best validation loss. (I highly recommend using this always)

When to adjust steps-per-epoch

- Traditionally, the steps per epoch is calculated as train_length // batch_size, since this will use all of the data points, one batch size worth at a time.
- If you are augmenting the data, then you can stretch this a tad (sometimes I multiply that function above by 2 or 3 etc. But, if it's already training for too long, then I would just stick with the traditional approach.

Share Improve this answer Follow

answered Apr 19, 2018 at 15:14

Chris Farr

3,291 ● 1 ● 20 ● 23

Yes I have used a relatively small batch size because of OOM errors. I will try to inch towards the maximum. As for your last point, are you saying that I should have to many steps that I iterate over every sample at least once per epoch? I do use data augmentation, but not during training. I have augmented my training and validation data by *10, and am using this new larger set as my training and validation set. Is this approach worse than using my original sets and then augmenting "live" during training? – n.st Apr 20, 2018 at 8:07

I'm getting a little confused so I'll just lay out how I would use data augmentation along with validation. Augment your train only, your validation and test data are left unaltered. Calculate steps_per_epoch as (train_lenth//batchsize) multiplied by any number >=1 if desired. Create a validation set of reasonable size but this should be separate from your train and test sets. – Chris Farr Apr 20, 2018 at 19:47

1 What I was asking is: do you use data augmentation by creating an ImageDataGenerator and passing it to your fit function, or did you already augment your training data beforehand and save it on your disk, for example by using a script? – n.st Apr 21, 2018 at 10:02

2 Ah, I augment live at least initially. This can however create a bottleneck in the training so augmenting and saving can also speed up the training. – Chris Farr Apr 21, 2018 at 12:37

I have similar questions about using Keras' ImageDataGenerator `train_gen = ImageDataGenerator(width_shift_range=0.3,height_shift_range=0.3,zoom_range=0.4)` and passing it into Keras' fit method (e.g. `model.fit(train_gen.flow(train_data,train_labels,batch_size=batch_size),steps_per_epoch=int(len(train_data)/batch_size),epochs=100)`). Say `len(train_data) = 1000` and `batch_size=20`. The `train_gen.flow(...)` call will generate augmented images so that in effect to the system it appears that I have more than 1000 train images. Can/should I set `steps_per_epoch` to a much larger size? – user3731622 Nov 22, 2018 at 22:52

Steps per epoch does not connect to epochs.

3

Naturally what you want if to 1 epoch your generator pass through all of your training data one time. To achieve this you should provide steps per epoch equal to number of batches like this:

```
steps_per_epoch = int( np.ceil(x_train.shape[0] / batch_size) )
```

as from above equation the largest the `batch_size`, the lower the `steps_per_epoch`.

Next you will choose epoch based on chosen validation. (choose what you think best)

Share Improve this answer Follow

edited May 26, 2018 at 14:04

seralouk
27.8k ● 8 ● 101 ● 119

answered Apr 19, 2018 at 14:21

Ioannis Nasios
7,968 ● 4 ● 29 ● 54

Do you know what happens if the above equality is not followed? stackoverflow.com/questions/55377309/... – deadcode Mar 27, 2019 at 20:46

Does it matter if `floor` is used instead of `ceil`? How will it work in the `model.fit`? – Joe Nov 8, 2019 at 7:12

1

The Steps per epoch denote the number of batches to be selected for one epoch. If 500 steps are selected then the network will train for 500 batches to complete one epoch. If we select the large number of epochs it can be computational

Share Improve this answer Follow

answered Jul 5, 2020 at 5:56

Manish Vasandnani
29 ● 2

3 This does not answer the question and should be deleted or placed as a comment. – NelsonGon Jul 31, 2020 at 15:14