

Platforms GPU comparison

fast.ai v3 lesson 1

2019/02/02 Marcello Morchio

marcello@datascienceseed.com

Performances

| Platform | Video time 00:44:00 Resnet34 learn.fit_one_cycle(4) | Video time 01:16:00 Resnet34 learn.unfreeze() learn.fit_one_cylce(1) | Video time 01:25:00 Resnet34 learn.unfreeze() learn.fit_one_cycle(2, max_lr=slice(1e-6,1e-4)) | Video time 01:29:00 Resten50 learn.fit_one_cycle(8) |
|-------------------|---|---|---|---|
| Fast.ai video | 00:01:56 | 00:00:29 | 00:00:58 | 00:03:41 <i>(5 epochs, not 8)</i> |
| Gradient P5000 | 00:01:38 | 00:00:28 | 00:00:56 | 00:09:51 |
| Kaggle | 00:10:36 | 00:02:57 | 00:05:58 | 00:46:43 |
| Colab | 00:07:43 | 00:02:00 | 00:04:01 | 00:26:48 |
| Intel CPU | 01:11:26 one epoch | | | |

PS Paperspace: Cloud Machine Learning

course-v3/nbs/dl1/

lesson1-pets-MM

https://n52bt9x4.gradient.paperspace.com/notebooks/course-v3/nbs/dl1/lesson1-pets-MM.ipynb

New Tab

☆

📁

📄

🔍

👤

⋮

★ Bookmarks

🏠 Home - Internal

🔍 Google

📧 WebMail Aruba - Pos

📄 Servizio di Invio teler

🗨️ Community – Deep L

👤 delega

🔗 Copy of lesson1.ipynl

🔗 Copy of lesson1.ipynl

🖱️ Running Open AI Gyn

jupyter

lesson1-pets-MM

Last Checkpoint: 7 minuti fa (unsaved changes)

🐍

Logout

File

Edit

View

Insert

Cell

Kernel

Widgets

Help

📄

+

✂️

📄

📄

⬆️

⬆️

▶️ Run

⏏️

🔄

▶️▶️

Code

⌵

🗨️

(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

)

(1): BasicBlock(

(conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu): ReLU(inplace)

(conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

In [17]:

learn.fit_one_cycle(4)

Total time: 01:38

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 1.384465 | 0.341945 | 0.108254 |
| 2 | 0.555354 | 0.280853 | 0.095399 |
| 3 | 0.332476 | 0.232261 | 0.079161 |
| 4 | 0.254912 | 0.215045 | 0.073072 |

In []:

learn.save('stage-1')

Results

Let's see what results we have got.

We will first see which were the categories that the model most confused with one another. We will try to see if what the model predicted was reasonable or not. In this case the mistakes look reasonable (none of the mistakes seems obviously naive). This is an indicator that our classifier is working correctly.

Furthermore, when we plot the confusion matrix, we can see that the distribution is heavily skewed: the model makes the same mistakes over and over again but it rarely confuses other categories. This suggests that it just finds it difficult to distinguish some specific categories between each other; this is normal behaviour.



```
('staffordshire_bull_terrier', 'american_bulldog', 3),  
( 'staffordshire_bull_terrier', 'american_pit_bull_terrier', 3)]
```

Unfreezing, fine-tuning, and learning rates

Since our model is working as we expect it to, we will *unfreeze* our model and train some more.

```
In [24]: learn.unfreeze()
```

```
In [26]: learn.fit_one_cycle(1)
```

Total time: 00:28

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.423295 | 0.280155 | 0.095399 |

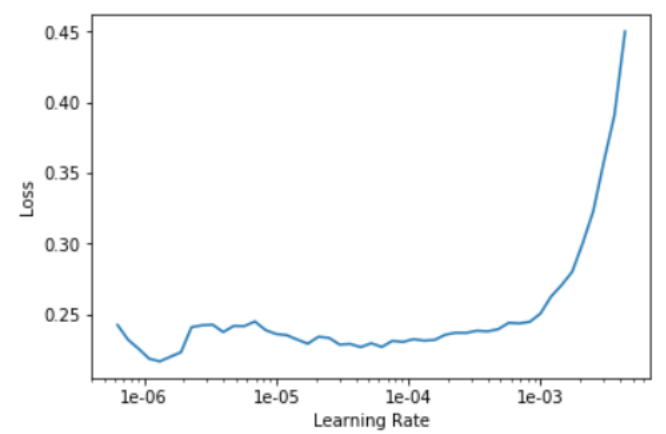
```
In [27]: learn.load('stage-1');
```

```
In [28]: learn.lr_find()
```

LR Finder is complete, type `{learner_name}.recorder.plot()` to see the graph.

```
In [29]: learn.recorder.plot()
```





```
In [30]: learn.unfreeze()
learn.fit_one_cycle(2, max_lr=slice(1e-6,1e-4))
```

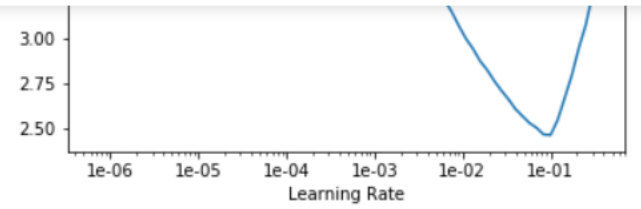
Total time: 00:56

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.233947 | 0.205685 | 0.072395 |
| 2 | 0.214571 | 0.204242 | 0.070365 |

That's a pretty accurate model!

Training: resnet50

Now we will train in the same way as before but with one caveat: instead of using resnet34 as our backbone we will use resnet50 (resnet34 is a 34 layer



In [34]: learn.fit_one_cycle(8)

Total time: 09:51

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.695744 | 0.302126 | 0.096076 |
| 2 | 0.405410 | 0.245322 | 0.082544 |
| 3 | 0.355702 | 0.216658 | 0.077131 |
| 4 | 0.287318 | 0.207957 | 0.064276 |
| 5 | 0.215132 | 0.201466 | 0.062246 |
| 6 | 0.154829 | 0.177019 | 0.056834 |
| 7 | 0.111082 | 0.178244 | 0.054127 |
| 8 | 0.084753 | 0.173119 | 0.051421 |

In []: learn.save('stage-1-50')

It's astonishing that it's possible to recognize pet breeds so accurately! Let's see if full fine-tuning helps:

In []: learn.unfreeze()
learn.fit_one_cycle(3, max_lr=slice(1e-6,1e-4))

Total time: 03:27

Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit your work.

100%| 0/300240/0/300240 [00.00<00.00, 90502140.0210/S]

Hide Input Output Markdown Code

```
learn.fit_one_cycle(4)
```

Total time 10:36

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 1.420245 | 0.333588 | 0.112314 |
| 2 | 0.562881 | 0.240066 | 0.078484 |
| 3 | 0.346455 | 0.208156 | 0.064953 |
| 4 | 0.269381 | 0.209507 | 0.070365 |

```
[ ]: learn.save('stage-1')
```

Results

Sessions

Interactive Session 2h:28m:12s / 6h

CPU 0% RAM 3.7GB/14GB

GPU On Disk 2.1GB/5.2GB

Versions

1 uncommitted draft

Marcello Morchio's draft

Draft Environment

No Data Sources

Connect your Kernel to our library of datasets

+ Add Data

Settings

Sharing Private, 0 collaborators

Language Python

Docker Latest available

GPU BETA GPU on

Internet BETA Internet connected

Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit your work.

```
( 'yorkshire_terrier', 'navanese', 2 )]
```

Unfreezing, fine-tuning, and learning rates

Since our model is working as we expect it to, we will *unfreeze* our model and train some more.

```
[41]: learn.unfreeze()
```

```
[42]: learn.fit_one_cycle(1)
```

Total time: 02:57

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.534654 | 0.304706 | 0.100812 |

```
[ ]: learn.load('stage-1');
```

```
[ ]: learn.lr_find()
```

Sessions

Interactive Session 3h:17m:49s / 6h

CPU 102% RAM 3.8GB/14GB

GPU On Disk 2.2GB/5.2GB

Versions

1 uncommitted draft

Marcello Morchio's draft

Draft Environment

No Data Sources

Connect your Kernel to our library of datasets

+ Add Data

Settings

Sharing Private, 0 collaborators

Language Python

Docker Latest available

GPU BETA GPU on

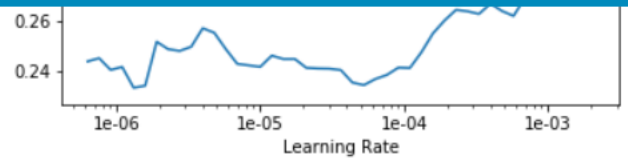
Internet BETA Internet connected

Packages Custom packages are not supported for GPU instances

Docs

> API

Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit your work.



```
[46]: learn.unfreeze()
learn.fit_one_cycle(2, max_lr=slice(1e-6, 1e-4))
```

Total time: 05:58

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.234137 | 0.197812 | 0.065629 |
| 2 | 0.227579 | 0.198307 | 0.067659 |

Markdown Code

B *I* “ 🔗 🖼️ ☰ ☷

That's a pretty accurate model!

Training: resnet50

Sessions

Interactive Session 3h:29m:37s / 6h
CPU 0% RAM 4.1GB/14GB
GPU On Disk 2.4GB/5.2GB

Versions

1 uncommitted draft
Marcello Morchio's draft

Draft Environment

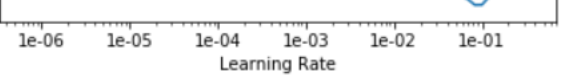
No Data Sources
Connect your Kernel to our library of datasets
+ Add Data

Settings

Sharing Private, 0 collaborators
Language Python
Docke Latest available
GPU BETA GPU on
Internet BETA Internet connected
Packages Custom packages are not supported for GPU instances

Docs API

Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit your work.



```
[52]: learn.fit_one_cycle(8)
```

Total time: 46:43

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.725932 | 0.259300 | 0.078484 |
| 2 | 0.395987 | 0.215787 | 0.071719 |
| 3 | 0.322443 | 0.194188 | 0.064953 |
| 4 | 0.276028 | 0.187555 | 0.062923 |
| 5 | 0.185594 | 0.160852 | 0.048714 |
| 6 | 0.156400 | 0.141862 | 0.050068 |
| 7 | 0.123473 | 0.131424 | 0.043302 |
| 8 | 0.086528 | 0.134478 | 0.045332 |

```
[ ]: learn.save('stage-1-50')
```

It's astonishing that it's possible to recognize pet breeds so accurately! Let's see if full fine-tuning helps:

Sessions
● Interactive Session 4h:45m:12s / 6h
CPU 0% RAM 4GB/14GB
GPU On Disk 2.4GB/5.2GB

Versions
1 uncommitted draft
Marcello Morchio's draft

Draft Environment
No Data Sources
Connect your Kernel to our library of datasets
[+ Add Data](#)

Settings
Sharing Private, 0 collaborators
Language Python
Docker Latest available
GPU BETA GPU on
Internet BETA Internet connected
Packages Custom packages are not supported for GPU instances
[Docs](#) [API](#)

co

Copy of lesson1-pets.ipynb

File Edit View Insert Runtime Tools Help

CODE

TEXT

CELL

CELL

CONNECTED

EDITING

COMMENT

SHARE

```
(1): Flatten()
(2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(3): Dropout(p=0.25)
(4): Linear(in_features=1024, out_features=512, bias=True)
(5): ReLU(inplace)
(6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(7): Dropout(p=0.5)
(8): Linear(in_features=512, out_features=37, bias=True)
)
)
```

[17]

learn.fit_one_cycle(4)

Total time: 07:43

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 1.376427 | 0.349496 | 0.108931 |
| 2 | 0.546538 | 0.270940 | 0.086604 |
| 3 | 0.343079 | 0.225992 | 0.076455 |
| 4 | 0.255855 | 0.213355 | 0.071719 |

[]

learn.save('stage-1')

Results

6 cells hidden

Unfreezing, fine-tuning, and learning rates

8 cells hidden

co

Copy of lesson1-pets.ipynb ☆

File Edit View Insert Runtime Tools Help

+ CODE + TEXT

↑ CELL ↓ CELL

COMMENT SHARE

CONNECTED EDITING

```
[ ] ('american_bulldog', 'staffordshire_bull_terrier', 3),
    ('basset_hound', 'beagle', 3),
    ('chihuahua', 'miniature_pinscher', 3),
    ('staffordshire_bull_terrier', 'american_bulldog', 3),
    ('staffordshire_bull_terrier', 'american_pit_bull_terrier', 3)]
```

Unfreezing, fine-tuning, and learning rates

Since our model is working as we expect it to, we will *unfreeze* our model and train some more.

```
[27] learn.unfreeze().
```

```
[28] learn.fit_one_cycle(1)
```

Total time: 02:00

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.556583 | 0.312836 | 0.108254 |

```
[29] learn.load('stage-1');
```

```
[30] learn.lr_find()
```

0.00% [0/2 00:00<00:00]

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
|-------|------------|------------|------------|

Interrupted

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

co

Copy of lesson1-pets.ipynb

File Edit View Insert Runtime Tools Help

CODE

TEXT

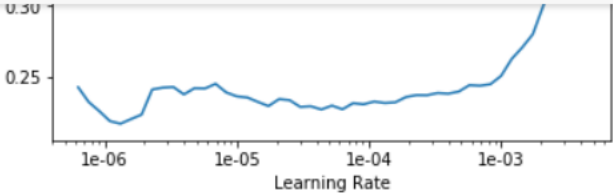
CELL

CELL

CONNECTED

EDITING

[]



[32]

```
learn.unfreeze()
learn.fit_one_cycle(2, max_lr=slice(1e-6,1e-4))
```

Total time: 04:01

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.226710 | 0.203354 | 0.066306 |
| 2 | 0.214740 | 0.198217 | 0.065629 |

learn.unfreeze()
learn.fit_one_cycle(2, max_lr=slice(1e-6,1e-4))

Total time: 00:53

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.242544 | 0.208489 | 0.067659 |
| 2 | 0.206940 | 0.204482 | 0.062246 |

That's a pretty accurate model!

Training: resnet50

12 cells hidden

co

Copy of lesson1-pets.ipynb

File Edit View Insert Runtime Tools Help

+ CODE + TEXT

↑ CELL ↓ CELL

CONNECTED EDITING

COMMENT SHARE

Avatar

Learning Rate

2.50 2.75

1e-06 1e-05 1e-04 1e-03 1e-02 1e-01

learn.fit_one_cycle(8)

Total time: 26:48

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 0.742405 | 0.293544 | 0.099459 |
| 2 | 0.412172 | 0.261765 | 0.092016 |
| 3 | 0.373033 | 0.198678 | 0.069689 |
| 4 | 0.248154 | 0.203448 | 0.073072 |
| 5 | 0.220838 | 0.185997 | 0.063599 |
| 6 | 0.154433 | 0.145898 | 0.058863 |
| 7 | 0.117187 | 0.149072 | 0.055480 |
| 8 | 0.112777 | 0.141536 | 0.054804 |

learn.fit_one_cycle(8)

Total time: 06:59

| epoch | train_loss | valid_loss | error_rate | |
|-------|------------|------------|------------|---------|
| 1 | 0.548006 | 0.268912 | 0.076455 | (00:57) |
| 2 | 0.365533 | 0.193667 | 0.064953 | (00:51) |
| 3 | 0.336032 | 0.211020 | 0.073072 | (00:51) |
| 4 | 0.263173 | 0.212025 | 0.060893 | (00:51) |
| 5 | 0.217016 | 0.183195 | 0.063599 | (00:51) |
| 6 | 0.161002 | 0.167274 | 0.048038 | (00:51) |
| 7 | 0.086668 | 0.143490 | 0.044655 | (00:51) |
| 8 | 0.082288 | 0.154927 | 0.046008 | (00:51) |

```
(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(rel): ReLU(inplace)
(conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(1): BasicBlock(
  (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace)
  (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
```

In [*]: 1 learn.fit_one_cycle(4)

25.00% [1/4 1:11:26<3 34:18]

| epoch | train_loss | valid_loss | error_rate |
|-------|------------|------------|------------|
| 1 | 1.375314 | 0.332302 | 0.093369 |

0.00% [0/92 00:00<00:00]

In []: 1 learn.save('stage-1')

Windows edition

Windows 10 Enterprise

© 2017 Microsoft Corporation. All rights reserved.

System

| | |
|-------------------------|---|
| Manufacturer: | HP |
| Model: | HP EliteBook 850 G5 |
| Processor: | Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz |
| Installed memory (RAM): | 32.0 GB (31.9 GB usable) |
| System type: | 64-bit Operating System, x64-based processor |
| Pen and Touch: | No Pen or Touch Input is available for this Display |

Results

Let's see what results we have got.

We will first see which were the categories that the model most confused with one another. We will try to see if what the model predicted was reasonable or not. In this case the mistakes look reasonable (none of the mistakes seems obviously naive). This is an indicator that our classifier is working correctly.

Utilities

- Documentation visualization
 - Kaggle: OK - `doc(function)` pops up the text and the links
 - Colabs: Not OK - `doc(function)` does not pop up anything

Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit your work.



Sessions

Interactive Session 2h:46m:7s / 6h

| | | | |
|-----|----|------|-------------|
| CPU | 0% | RAM | 3.8GB/14GB |
| GPU | On | Disk | 2.2GB/5.2GB |

Versions

1 uncommitted draft

Marcello Morchio's draft

Draft Environment

No Data Sources

Connect your Kernel to our library of datasets

+ Add Data

Settings

| | |
|---------------|--------------------------|
| Sharing | Private, 0 collaborators |
| Language | Python |
| Docker | Latest available |
| GPU BETA | GPU on |
| Internet BETA | Internet connected |

```
[33]: doc(interp.plot_top_losses)
```

Doc references appear as expected



```
[ ]: interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```

plot_top_losses [source]

```
plot_top_losses('k', 'largest'='True', 'figsize'='(12, 12)')
```

Show images in top_losses along with their prediction, actual, loss, and probability of predicted class.

[Show in docs](#)



No doc reference appears in the notebook

`doc(interp.plot_top_losses) |`

[] `interp.plot_confusion_matrix(figsize=(12,12), dpi=60)`

