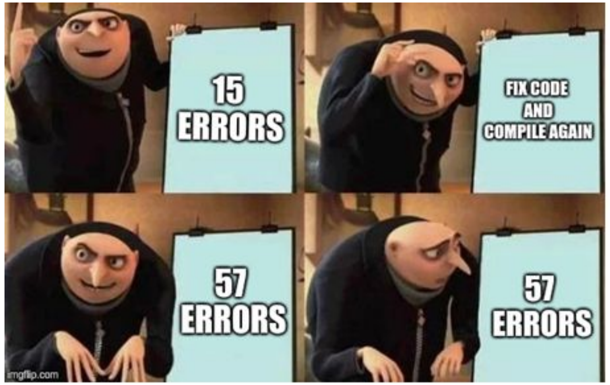
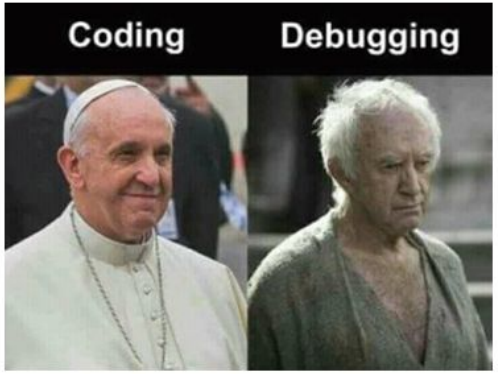
Debugging ML code

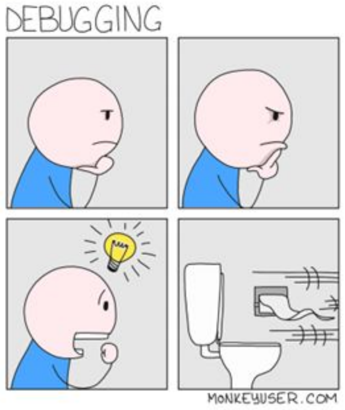
02476 Machine Learning Operations
Nicki Skafte Detlefsen

Debugging is a hard but necessary disiplin



DEBUGGING
I DON'T KNOW WHERE YOU ARE, I DON'T KNOW HOW YOU WORK, BUT I WILL FIND YOU, AND
I WILL FIX YOU





Debugging ML code is even harder

Bugs in ML code can be classical and ML specific

- Classic bugs: Code does not run
  - Use traditional debugger to find these

- ML specific bugs: My model is not converging
  - Need the correct approach

Let's start with the classics..

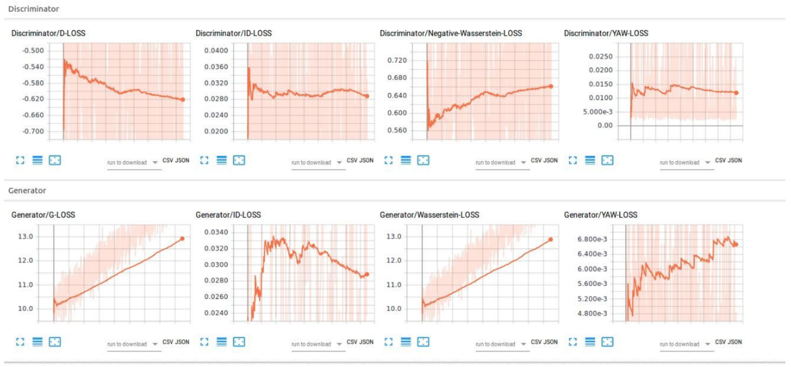


Image credit: https://discuss.pytorch.org/t/training-gan-divergence/26060

First step: Get a good environment

Jupyter notebooks a great at what they are meant for: exploring ideas and combining code + text into a full standalone document. However, it can be a pain debugging code in notebooks...

Spyder

PyCharm

VS Code

Torchstudio

Python debugging in general

1. Print statements

```python
print("x.shape = {}".format(x.shape))
```

2. Stop at an interesting point in your code and interact

```python
from IPython import embed; embed()
... # do your stuff interactively here
exit()  # exit ipython to let your code continue
```

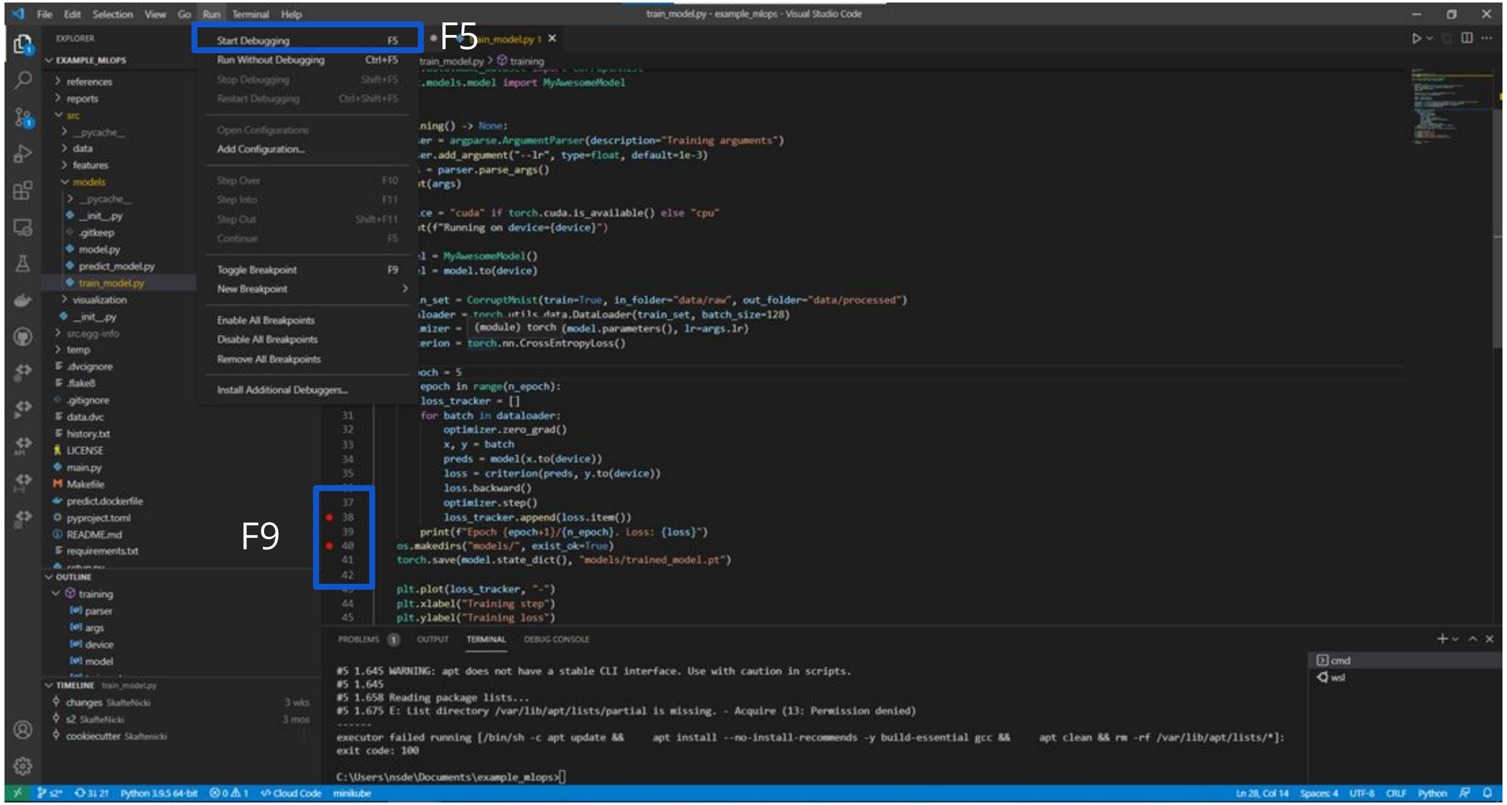
3. Work in an actual debugger

```python
import pdb; pdb.set_trace()
```

Learn more here:

https://switowski.com/blog/ipython-debugging

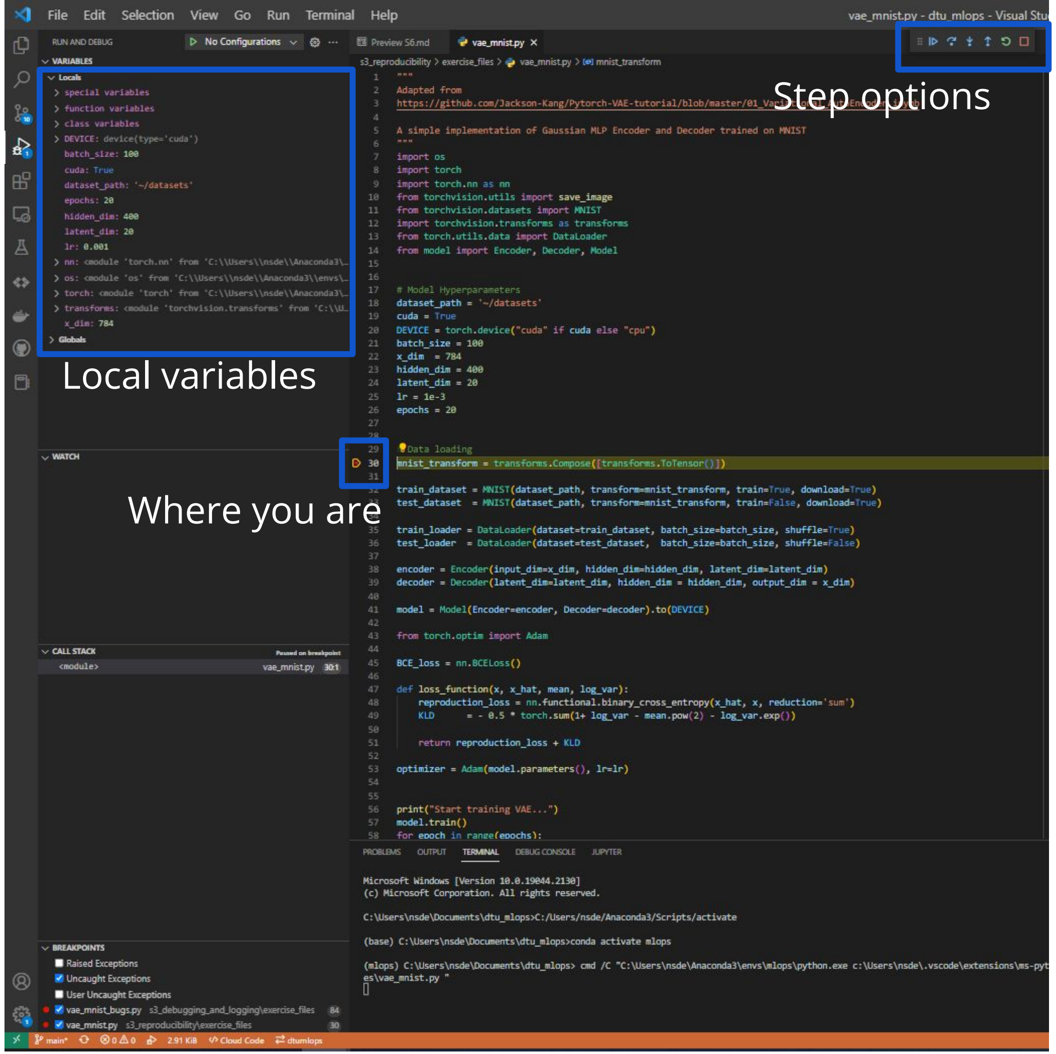
VS Code debugger



```
.models.model import MyAwesomeModel


ning() -> None:
er = argparse.ArgumentParser(description="Training arguments")
er.add_argument("--lr", type=float, default=1e-3)
 = parser.parse_args()
t(args)

ce = "cuda" if torch.cuda.is_available() else "cpu"
t(f"Running on device={device}")

l = MyAwesomeModel()
l = model.to(device)

n_set = CorruptMnist(train=True, in_folder="data/raw", out_folder="data/processed")
loader = torch.utils.data.DataLoader(train_set, batch_size=128)
mizer = (module) torch (model.parameters(), lr=args.lr)
erion = torch.nn.CrossEntropyLoss()

och = 5
epoch in range(n_epoch):
loss_tracker = []
for batch in dataloader:
    optimizer.zero_grad()
    x, y = batch
    preds = model(x.to(device))
    loss = criterion(preds, y.to(device))
    loss.backward()
    optimizer.step()
    loss_tracker.append(loss.item())
print(f"Epoch {epoch+1}/{n_epoch}. Loss: {loss}")
os.makedirs("models/", exist_ok=True)
torch.save(model.state_dict(), "models/trained_model.pt")

plt.plot(loss_tracker, "-")
plt.xlabel("Training step")
plt.ylabel("Training loss")
```

```
#5 1.645 WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
#5 1.645
#5 1.658 Reading package lists...
#5 1.675 E: List directory /var/lib/apt/lists/partial is missing. - Acquire (13: Permission denied)
------
executor failed running [/bin/sh -c apt update &&     apt install --no-install-recommends -y build-essential gcc &&     apt clean && rm -rf /var/lib/apt/lists/*]:
exit code: 100

C:\Users\nsde\Documents\example_mlops>
```

VS Code


File Edit Selection View Go Run Terminal Help
vae_mnist.py - dtu_mlops - Visual Stu
RUN AND DEBUG
No Configurations
Preview S6.md
vae_mnist.py
s3_reproducibility > exercise_files > vae_mnist.py > mnist_transform
Step options
Local variables
Where you are
WATCH
CALL STACK
Paused on breakpoint
<module> vae_mnist.py 30:1
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE JUPYTER
main* 0 0 2.91 KiB Cloud Code dtumlops

```
VARIABLES
Locals
  > special variables
  > function variables
  > class variables
  > DEVICE: device(type='cuda')
    batch_size: 100
    cuda: True
    dataset_path: '~/datasets'
    epochs: 20
    hidden_dim: 400
    latent_dim: 20
    lr: 0.001
  > nn: <module 'torch.nn' from 'C:\\Users\\nsde\\Anaconda3\…
  > os: <module 'os' from 'C:\\Users\\nsde\\Anaconda3\\envs\…
  > torch: <module 'torch' from 'C:\\Users\\nsde\\Anaconda3\…
  > transforms: <module 'torchvision.transforms' from 'C:\\U…
    x_dim: 784
> Globals
```

```python
"""
Adapted from
https://github.com/Jackson-Kang/Pytorch-VAE-tutorial/blob/master/01_Variational_AutoEncoder.ipynb

A simple implementation of Gaussian MLP Encoder and Decoder trained on MNIST
"""
import os
import torch
import torch.nn as nn
from torchvision.utils import save_image
from torchvision.datasets import MNIST
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from model import Encoder, Decoder, Model


# Model Hyperparameters
dataset_path = '~/datasets'
cuda = True
DEVICE = torch.device("cuda" if cuda else "cpu")
batch_size = 100
x_dim  = 784
hidden_dim = 400
latent_dim = 20
lr = 1e-3
epochs = 20


# Data loading
mnist_transform = transforms.Compose([transforms.ToTensor()])

train_dataset = MNIST(dataset_path, transform=mnist_transform, train=True, download=True)
test_dataset  = MNIST(dataset_path, transform=mnist_transform, train=False, download=True)

train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size, shuffle=True)
test_loader  = DataLoader(dataset=test_dataset,  batch_size=batch_size, shuffle=False)

encoder = Encoder(input_dim=x_dim, hidden_dim=hidden_dim, latent_dim=latent_dim)
decoder = Decoder(latent_dim=latent_dim, hidden_dim = hidden_dim, output_dim = x_dim)

model = Model(Encoder=encoder, Decoder=decoder).to(DEVICE)

from torch.optim import Adam

BCE_loss = nn.BCELoss()

def loss_function(x, x_hat, mean, log_var):
    reproduction_loss = nn.functional.binary_cross_entropy(x_hat, x, reduction='sum')
    KLD      = - 0.5 * torch.sum(1+ log_var - mean.pow(2) - log_var.exp())

    return reproduction_loss + KLD

optimizer = Adam(model.parameters(), lr=lr)


print("Start training VAE...")
model.train()
for epoch in range(epochs):
```

```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nsde\Documents\dtu_mlops>C:/Users/nsde/Anaconda3/Scripts/activate

(base) C:\Users\nsde\Documents\dtu_mlops>conda activate mlops

(mlops) C:\Users\nsde\Documents\dtu_mlops> cmd /C "C:\Users\nsde\Anaconda3\envs\mlops\python.exe c:\Users\nsde\.vscode\extensions\ms-pyt
es\vae_mnist.py "
```

BREAKPOINTS
- [ ] Raised Exceptions
- [x] Uncaught Exceptions
- [ ] User Uncaught Exceptions
- [x] vae_mnist_bugs.py s3_debugging_and_logging\exercise_files 84
- [x] vae_mnist.py s3_reproducibility\exercise_files 30

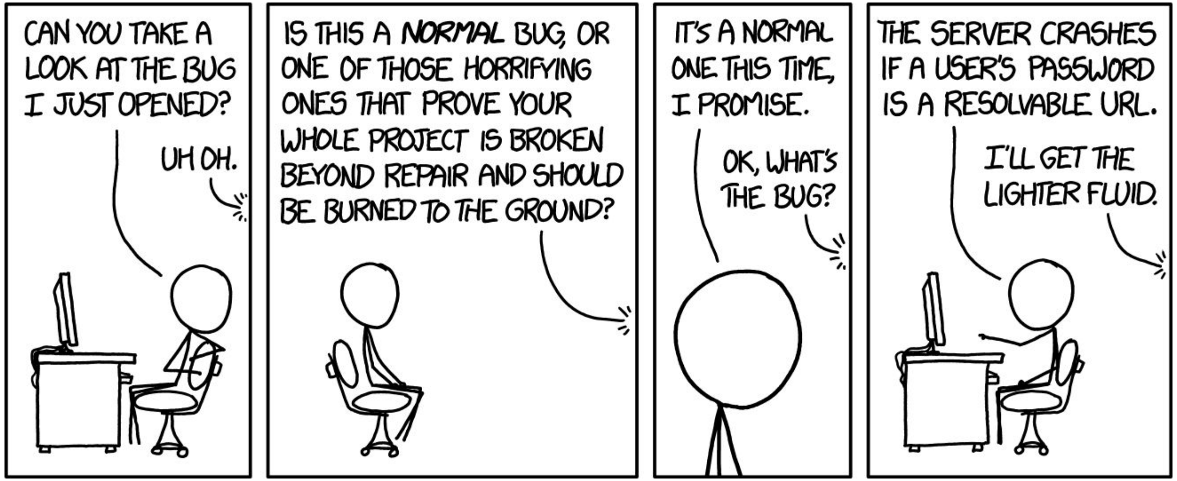
Step options:
-F5: next breakpoint
-F10: next line
-F11: step into
-Shift-F11: step out
-CTRL-Shift-F11: Restart
-Shift-F5: stop

https://xkcd.com/1700/

Back to these ML specific bugs

Or better known as

When everything is running, but results are wrong


CAN YOU TAKE A LOOK AT THE BUG I JUST OPENED?
UH OH.
IS THIS A NORMAL BUG, OR ONE OF THOSE HORRIFYING ONES THAT PROVE YOUR WHOLE PROJECT IS BROKEN BEYOND REPAIR AND SHOULD BE BURNED TO THE GROUND?
IT'S A NORMAL ONE THIS TIME, I PROMISE.
OK, WHAT'S THE BUG?
THE SERVER CRASHES IF A USER'S PASSWORD IS A RESOLVABLE URL.
I'LL GET THE LIGHTER FLUID.

Finding these buggers comes with experience

A potpourri of my findings over the last couple of years and others.

Image credit: https://xkcd.com/1700/

1. Check your data!

Your starting point should always be the data in ML!

Check

- Examine summary statistics (data normalized?)
- Look at label distributions (is it shuffled?)
- Visualize a few samples (are the as expected?)

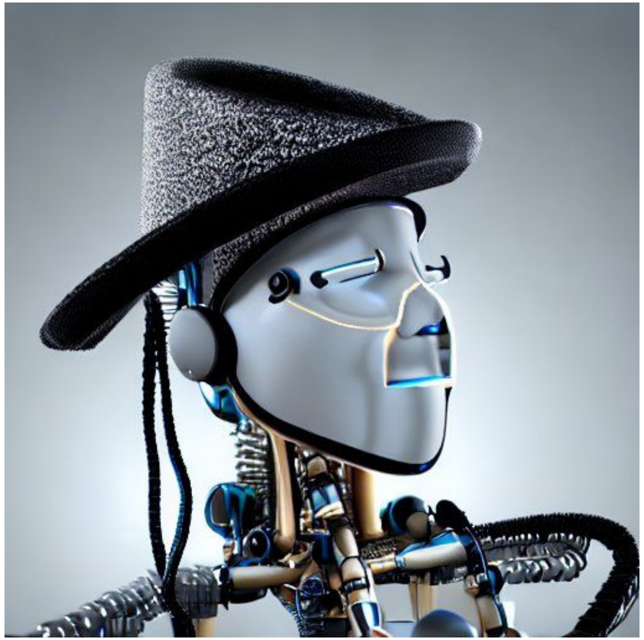
If you are working on dataset you know, you may skip these.

2. Start as simple as you can

Remove all fancy stuff.

- Mixed precision
- Regularization like dropout
- Early stopping
- Learning rate schedulers
- ...

This if often the case if you start from another codebase

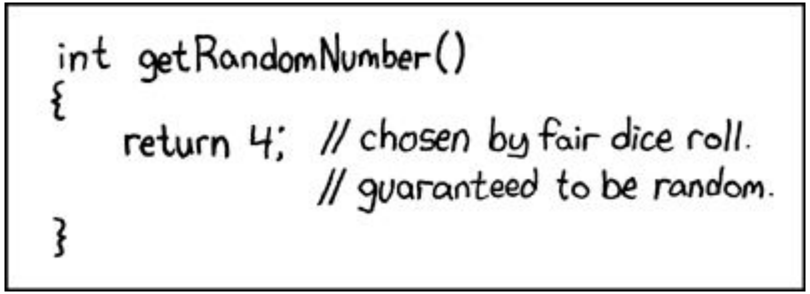


"Machine learning bot with a fancy hat", Stable diffusion

3. Make everything deterministic

Every machine learning run is by default random. Try fixing

- Seed everything and use same seed everywhere
- Remove all data augmentation
- Use only a single batch of data where you have a feeling of the outcome



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Image credit: https://xkcd.com/221/

4. Investigate the math

Debug the math!

- Go through your code, line by line
- There should be a one-to-one match between equations and lines of code
- Refactor if it is not clear

Check dimensions and annotate if necessary or use typing software

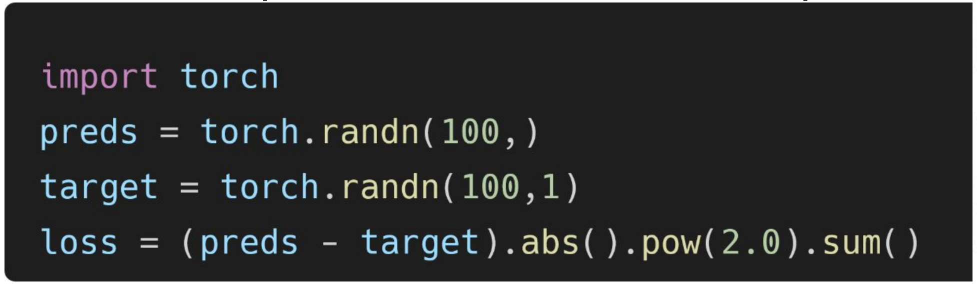
```python
# add shape comments
A = torch.randn(N, D)  # NxD
x = torch.randn(D)  # D
Ax = A.mv(x)  # N
```

4. Investigate the math (continue)

Lookout for broadcasting!

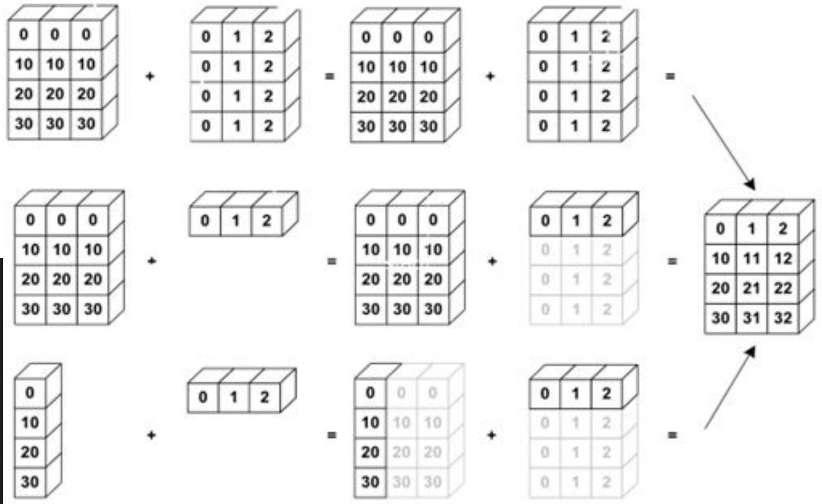
Broadcasting in python is both a blessing and a curse in python.

It can create problems (real life example)



```
import torch
preds = torch.randn(100,)
target = torch.randn(100,1)
loss = (preds - target).abs().pow(2.0).sum()
```

What is the problem here?



5. Overfitting is good?

Overfitting is usually seen as an bad thing but...

- Models should be able to memorize one batch

- Train on one batch, if loss is 0 move on to larger models else debug

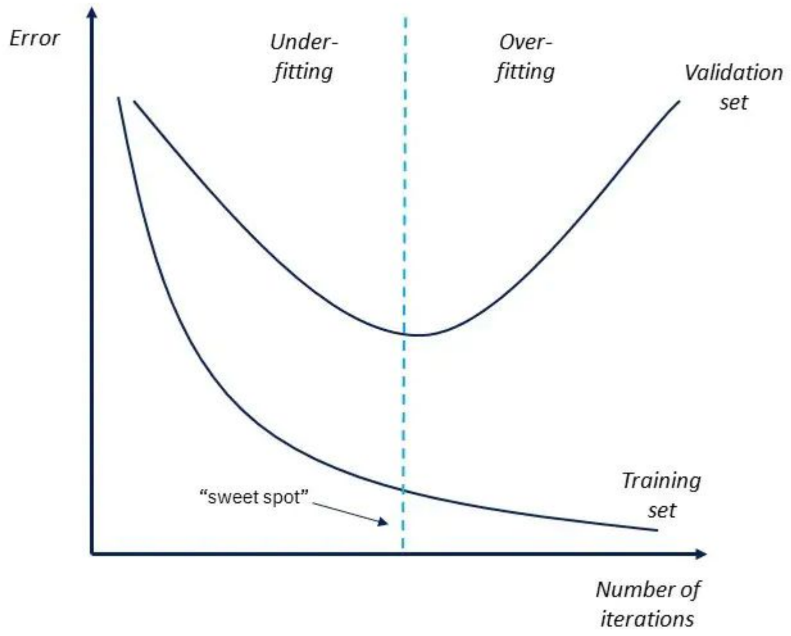


Image credit: https://www.ibm.com/cloud/learn/overfitting

6. Really look at your loss

Assuming your model is training without errors, but your loss is not behaving as it should.

- Are you printing/logging the results correctly?
- Did you remember loss.backward, optimizer.step and optimizer.zero_grad ?
- What about your learning rate and batch size?

For your loss, if possible calculate in log-space

```
a = 1
for x in data:
  a = a * x
```

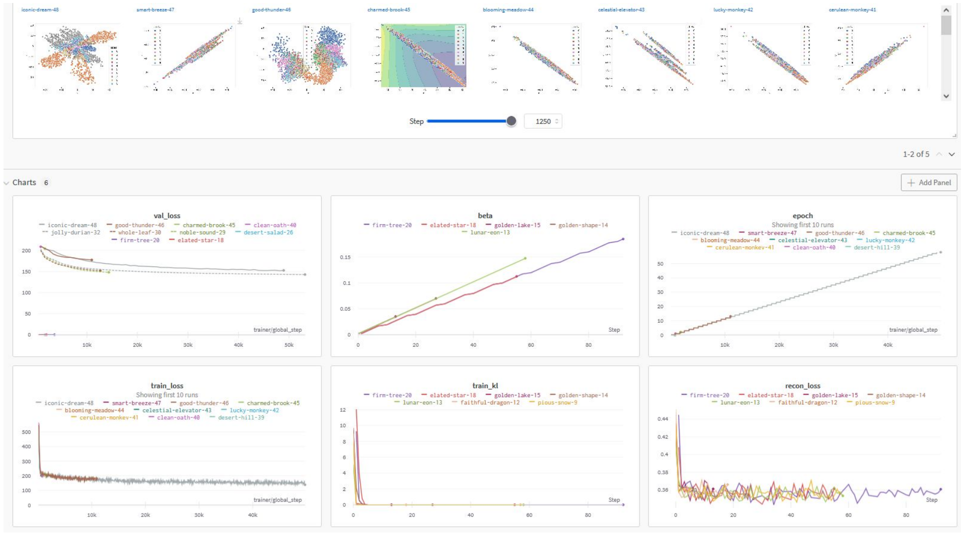
Trade-off between precision and stability

```
log_a = 0
for x in data:
  log_a = log_a + log(x)
```

7. Visualizations are your friend

Log and visualize everything.

- Training loss is really decreasing right?
- Can you add additional metrics?
- Log dynamic changing hyperparameters (lr with lr schedulers)
- Plot data, predictions, reconstructions etc. over time



5. Add complexity in steps

If you are still good, then

- Get a baseline that just works (=train on more data data)
- Stop overfitting:
  - Add back regularization
  - Add back data augmentations
- Tune hyper parameters

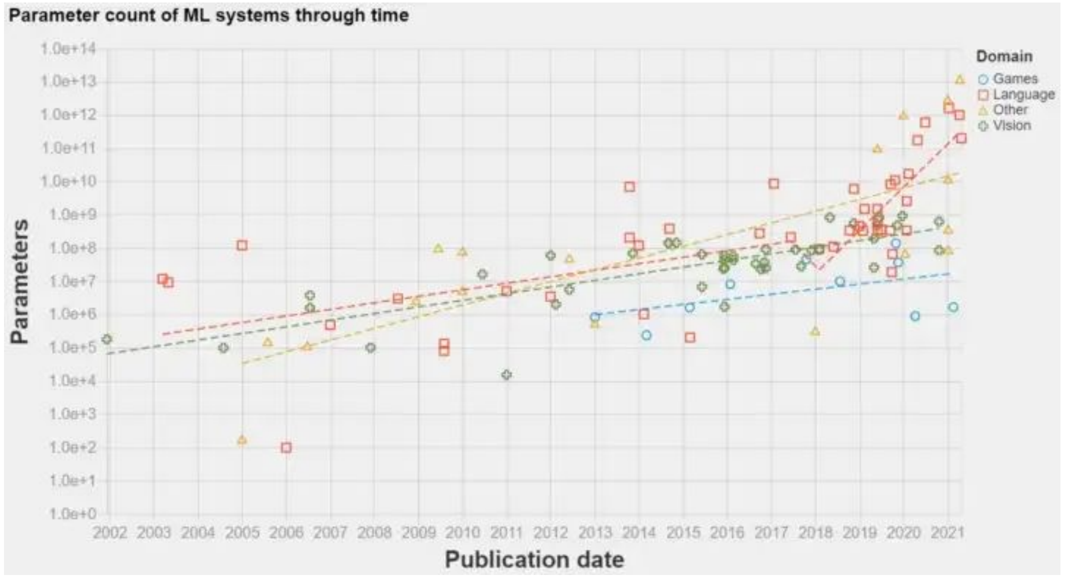
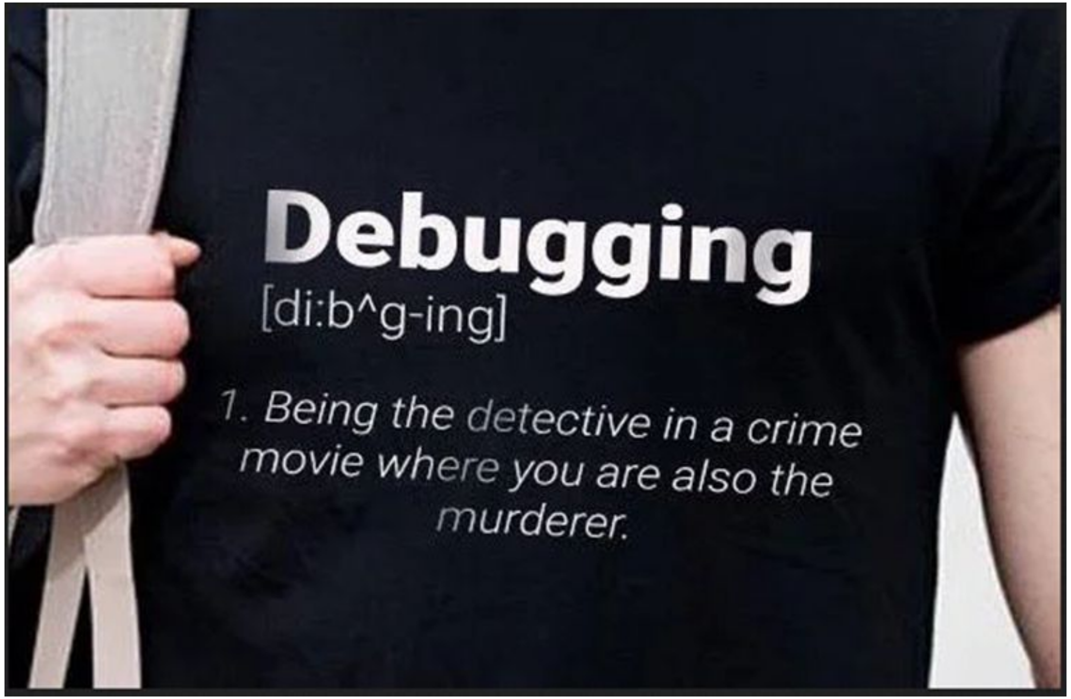


Image credit: https://towardsdatascience.com/parameter-counts-in-machine-learning-a312dc4753d0

Summary of ML debugging

1. Check your data
2. Start as simple as you can
3. Make everything deterministic
4. Investigate the math
5. Overfit to your data
6. Look at your loss
7. Add complexity in steps

Meme of the day


Debugging
[di:b^g-ing]
1. Being the detective in a crime movie where you are also the murderer.