



CS 760: Machine Learning **Practical Tips when Training Neural Networks**

Kirthi Kandasamy

University of Wisconsin-Madison

March 8, 2023

Announcements

- **Midterm course evaluations.**
- **Homeworks:**
 - HW 4 due on Mar 20
 - HW 5 will be out early, due on Apr 3
- **Re-organizing lecture schedule** (will update website soon)
- **Midterm:**
 - 7:30 PM tomorrow (3/9/23). Will take 90 minutes.
 - B130, Van Vleck Hall
 - Arrive on time!
 - Bring scratch paper + cheat sheet + pen.

Training neural networks

Use existing DL libraries (PyTorch, Tensorflow etc.)

- Not worth implementing BP from scratch

First step: build a simple pipeline

- Set up data, model training, evaluation loop
- Overfit on one batch
 - Goal: see that we can get near-zero loss, catch any bugs
- Check that training loss reduces when you train.

Tips & Tricks: Data

- Shuffle the training data
 - In training ,usually don't select random examples, but rather go through the dataset for each epoch
 - Shuffle to avoid relationships between consecutive points
- Pay attention to your data
 - Missing values, NaNs, default values etc

Tips & Tricks: Initialization

Usually want to pick small random values to initialize weights

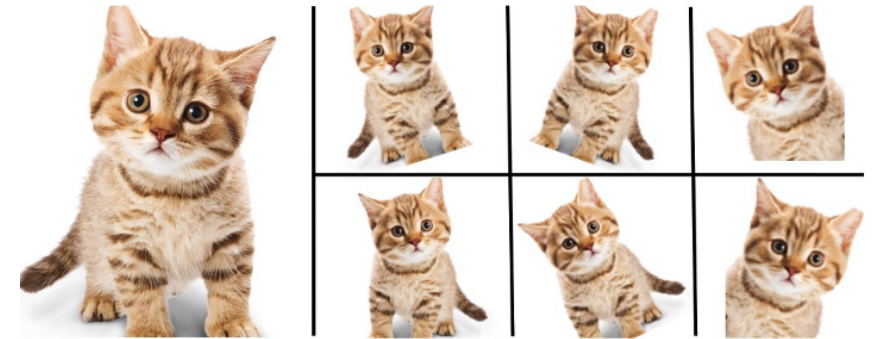
- Don't want the same value: symmetry means every weights has same gradient, hard to break out of
- Multiple methods: various rules of thumb
 - Sample from a normal distribution
 - Note that #inputs affects the variance... grows as d^2 for d inputs. Helps to normalize, when initializing.

Tips & Tricks: Learning Rate Schedule

- Simple ways:
 - Constant
 - Divide by a factor every fixed number of epochs (annealing)
 - Look at training/validation loss and reduce on plateau
- Also simple: use an optimizer like Adam that internally tracks learning rates
- Lots of variations available

Tips & Tricks: Regularizing

- Best thing to do: get more data!
 - Not always possible or cheap, but start here.
- Augmentation
 - But make sure you understand the transformations
- Use other strategies: dropout, weight decay, early stopping
 - Check each strategy **one-at-a-time**



Enlarge your Dataset

Nanonets

Tips & Tricks: Hyperparameter Tuning

Many solutions:

- **Grid Search:** pick candidate sets S_1, \dots, S_k for each hyperparameter, search over every combination in $S_1 \times S_2 \times \dots \times S_k$
- **Random Search**
- **Advanced approaches:**
Bayesian Optimization, Hyperband etc.

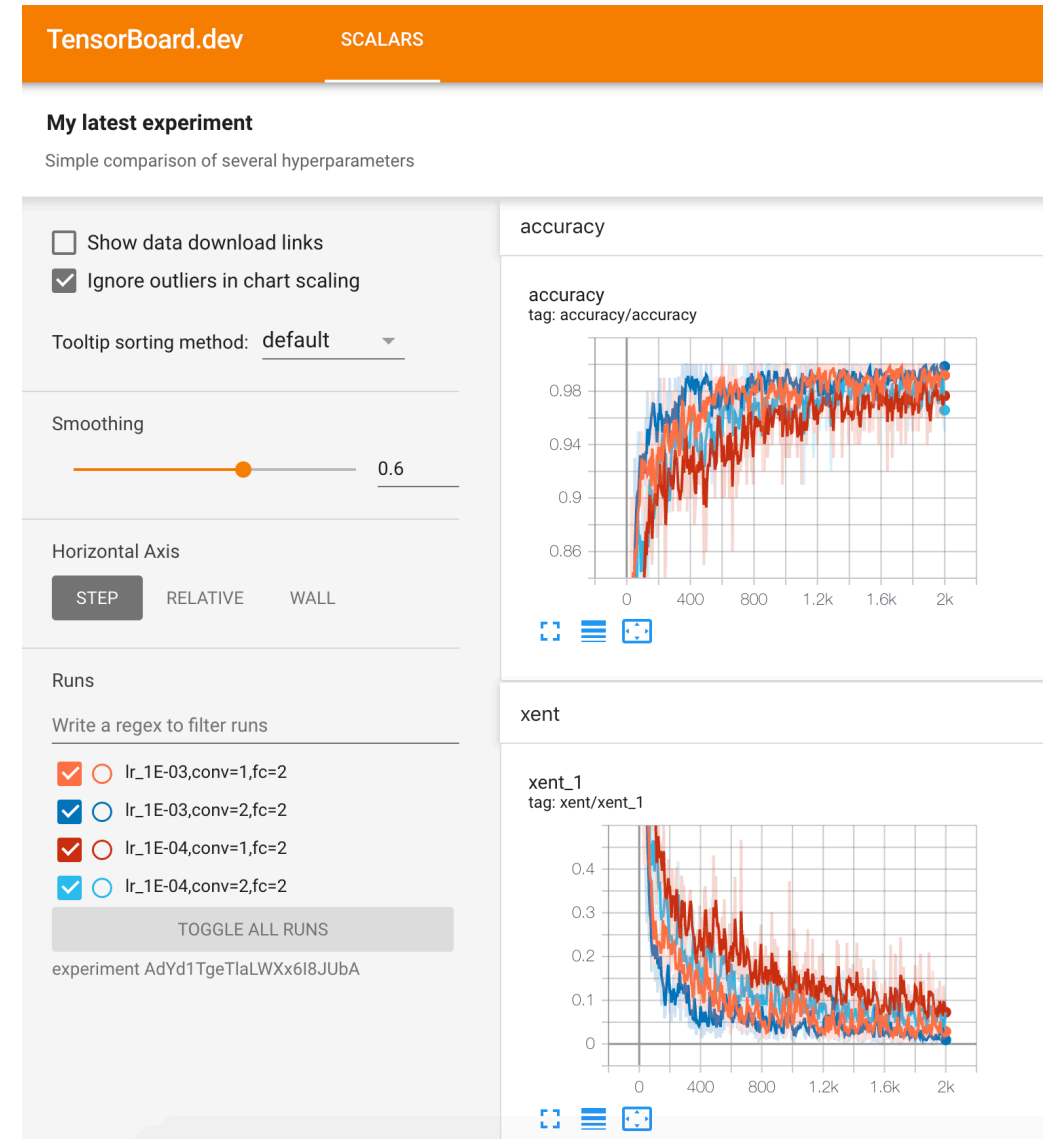
Tips & Tricks: Monitoring & Logging

- Checkpoint your models (save weights) regularly!
 - Training can crash
- Log information from training process
 - Keep track of train / test losses, time elapsed, current training settings. Log regularly.
 - Can use this for early stopping as well.

```
loading dataset /home/jitendra_gtbit11/...
building model...
WARNING:tensorflow: __init__ (from tensorflow.python.ops.init_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.initializers.variance_scaling instead with distribution=uniform to get equivalent behavior.
WARNING:tensorflow:From /home/jitendra_gtbit11/.local/lib/python2.7/site-packages/tensorflow/python/ops/init_ops.py:113:
deprecation: 'keep_dims' is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
2018-09-27 19:49:34.298676: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
start training...
- emotions = 7
- model = B
- optimizer = 'momentum'
- learning_rate = 0.016
- learning_rate_decay = 0.864
- optimizer_param (momentum) = 0.95
- keep_prob = 0.956
- epochs = 1500
- use landmarks = True
- use hog + landmarks = True
- use hog sliding window + landmarks = True
- use batchnorm after conv = True
- use batchnorm after fc = False
-----
Run id: 70MNF9
Log directory: logs/
[?]251-----
Training samples: 3436
Validation samples: 56
--
Training Step: 1 | time: 1.971s
[?]2K
| Momentum | epoch: 001 | loss: 0.00000 - acc: 0.0000 -- iter: 0128/3436
[?]A[?]ATraining Step: 2 | total loss: [?]1m[?]32m1.81674[?]0m[?]0m | time: 3.367s
[?]2K
| Momentum | epoch: 001 | loss: 1.81674 - acc: 0.0914 -- iter: 0256/3436
[?]A[?]ATraining Step: 3 | total loss: [?]1m[?]32m1.96555[?]0m[?]0m | time: 4.868s
[?]2K
| Momentum | epoch: 001 | loss: 1.96555 - acc: 0.1700 -- iter: 0384/3436
[?]A[?]ATraining Step: 4 | total loss: [?]1m[?]32m2.20454[?]0m[?]0m | time: 6.358s
[?]2K
| Momentum | epoch: 001 | loss: 2.20454 - acc: 0.1363 -- iter: 0512/3436
[?]A[?]ATraining Step: 5 | total loss: [?]1m[?]32m2.05230[?]0m[?]0m | time: 7.837s
[?]2K
| Momentum | epoch: 001 | loss: 2.05230 - acc: 0.1122 -- iter: 0640/3436
[?]A[?]ATraining Step: 6 | total loss: [?]1m[?]32m1.97573[?]0m[?]0m | time: 9.321s
[?]2K
```

Tips & Tricks: Monitoring & Logging

- Log information from training process
 - Use software packages
 - Also have built-in visualization
 - Example: TensorBoard



Finally,

- You don't always have to use the newest fanciest ML model.
- Sometimes simple models work well (esp in low data regimes)
 - E.g: Simple regression models with handcrafted features, kernel density estimation
 - Complex models may require expertise to get to work well.
 - Easier to interpret, incorporate domain expertise, and quantify uncertainty
- Incorporate domain expertise