NATURAL LANGUAGE PROCESSING

Hyperparameter Tuning in Pytorch

goorm

KAIST AI
Graduate School of AI

DAVIAN

Contents

Type of Hyperparameters

- Model-free hyperparameters
  - Learning rate
  - Batch size per gpu
  - Training epoch
  - Learning rate scheduler(Warm up steps, lambda, step size ...)
  - Optimizer (beta1, beta2 in Adam)
  - Weight initialization
  - Early stop strategy
  - Regularization
  - Dropout
  - Perturbation or noise for an input
  - ...

Type of Hyperparameters

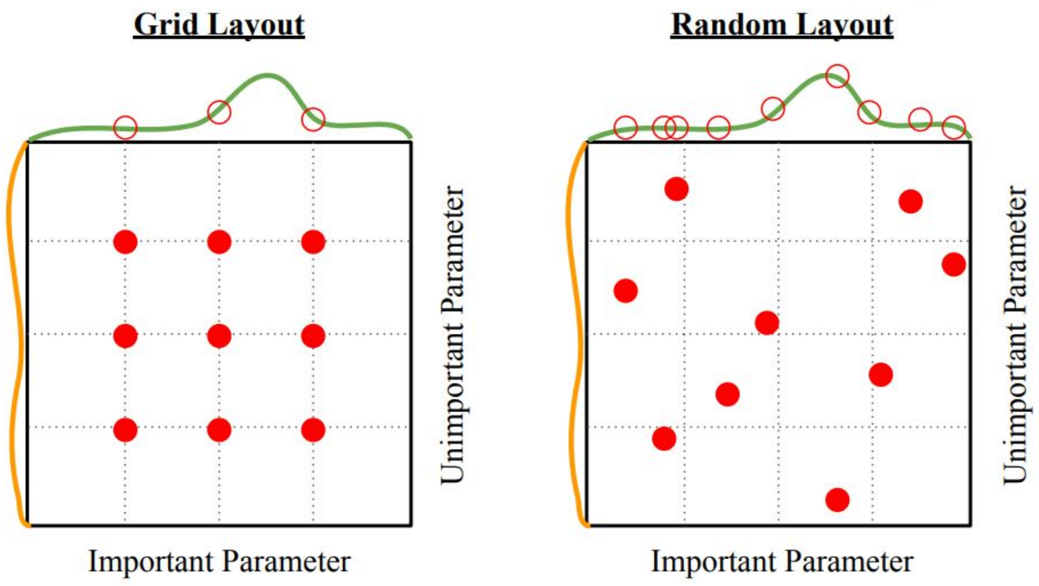
- Model hyperparameters
  - Kernel size
  - number of layer
  - number of hidden units
  - number of embedding units
  - pooling
  - activation function

Type of Hyperparameters

- Model free & Model hyperparameters
  - Learning rate x Batch size per gpu x Training epoch x Learning rate scheduler (Warmup steps, lambda, step size...) x Optimizer (+beta1, beta2 in Adam) x Weight initialization x Early stop strategy x Regularization x Dropout x Kernel size x number of layer x number of hidden units x number of embedding units x pooling layer x activation function x....
  - training time for a model

Hyperparameter Optimization

- Hyperparameters Optimization
  - Grid Search
  - Random Search
  - Bayesian



- (Maybe for researchers) Best practice for hyperparameters optimization
  - ***Learning rate, Learning rate, Learning rate***
  - ***Batch size***

Hyperparameter Optimization Steps

1. Check initial loss
2. Overfit a small sample
3. Find learning rate that makes loss go down
4. Coarse grid, train for 1~5 epochs
5. Refine grid, train longer than step 4
6. Look at loss curves

Hyperparameter Optimization Steps

1. **Check initial loss**
    a. Turn off learning rate scheduler and train a model

Hyperparameter Optimization Steps

1. Check initial loss
2. **Overfit a small sample**
   a. Try to train to high training accuracy on a small sample of training data (~5-10 minibatches); fiddle with architecture, learning rate, weight initialization
   b. Loss not going down? LR too low, bad initialization
   c. Loss explodes to Inf or NaN? LR too high, bad initialization

Hyperparameter Optimization Steps

1. Check initial loss
2. Overfit a small sample
3. **Find learning rate that makes loss go down**
   a. Use the architecture from the previous step, use all training data, turn on small weight decay, find a learning rate that makes the loss drop significantly within ~100 iterations
   b. Good learning rates to try: 1e-1, 1e-2, 1e-3, 1e-4

Hyperparameter Optimization Steps

1. Check initial loss
2. Overfit a small sample
3. Find learning rate that makes loss go down
4. **Coarse grid, train for 1~5 epochs**
    a. Choose a few values of learning rate and weight decay around what worked from Step 3, train a few models for ~1-5 epochs.
    b. Good weight decay to try: 1e-4, 1e-5, 0

- additional
    - Make possible batch sizes larger

Hyperparameter Optimization Steps

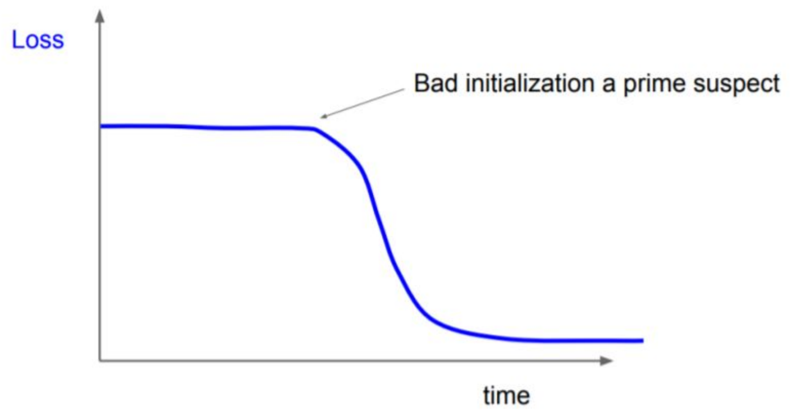
1. Check initial loss
2. Overfit a small sample
3. Find learning rate that makes loss go down
4. Coarse grid, train for 1~5 epochs
5. **Refine grid, train longer than step 4**
   a. Pick best models from Step 4, train them for longer (~10-20 epochs) without learning rate decay

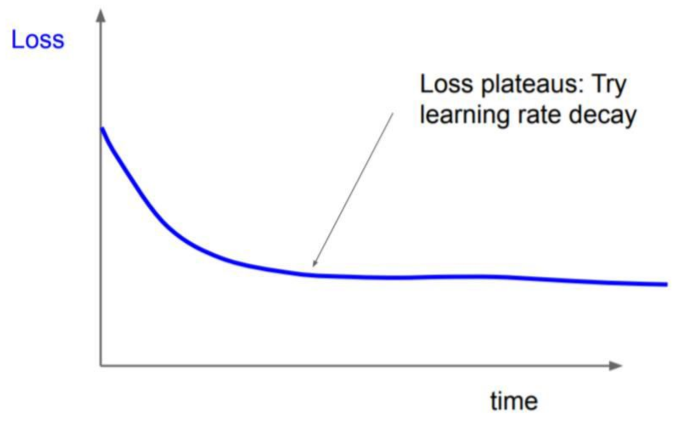
Hyperparameter Optimization Steps

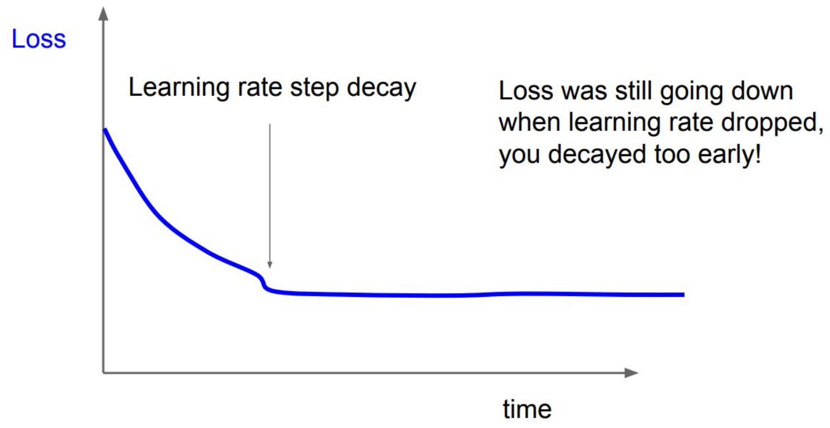
1. Check initial loss
2. Overfit a small sample
3. Find learning rate that makes loss go down
4. Coarse grid, train for 1~5 epochs
5. Refine grid, train longer than step 4
6. **Look at loss curves**

Look at Learning Curves

1) Look at the training learning curves

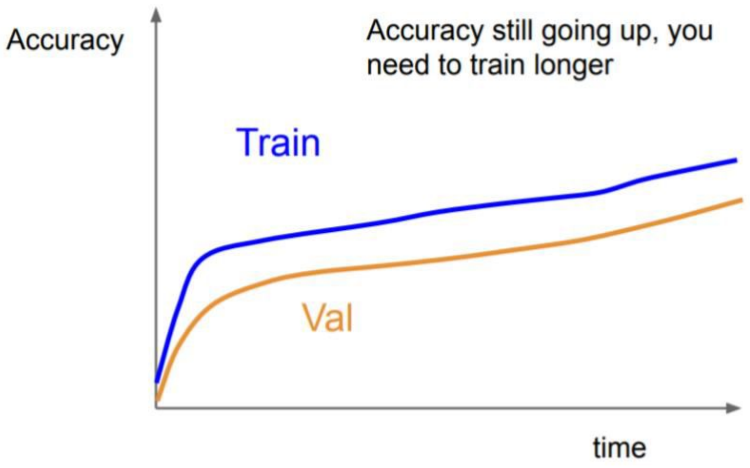


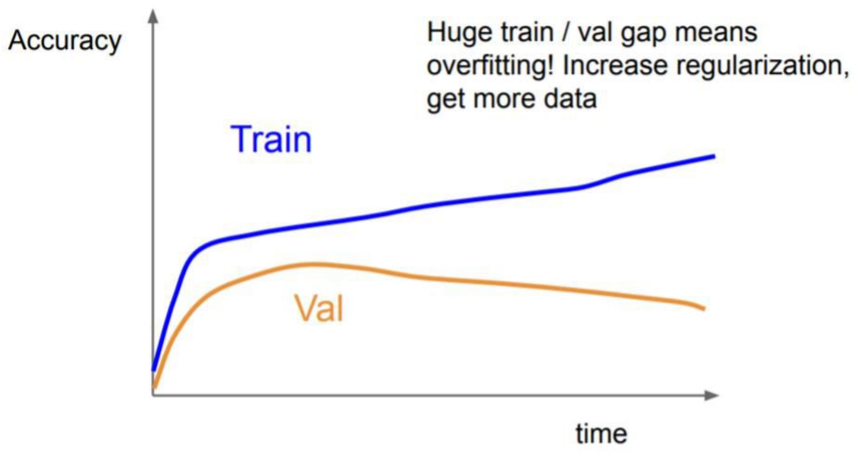


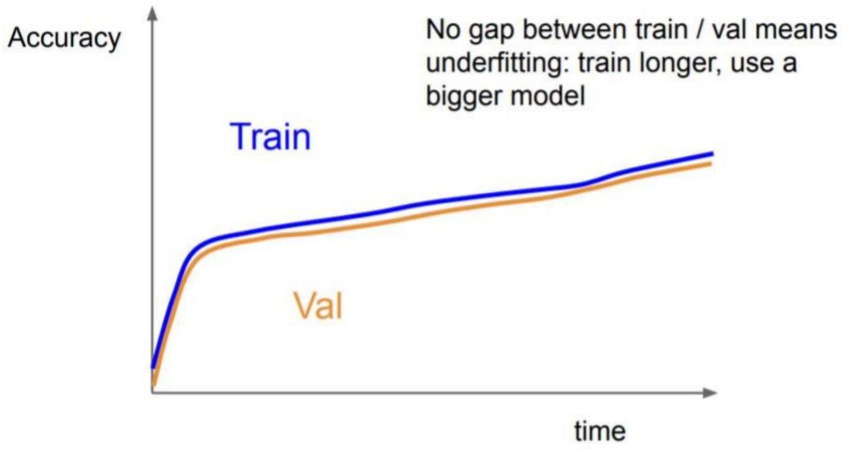


Look at Learning Curves

2) Look at the training & validation learning curves







Hyperparameter Optimization

- Tips for Geeks who struggle for state of the art performance or want to beat competitors
  - Train your model with the largest batch size that memory allows in single gpu
  - Evaluate as many checkpoints as possible
  - Before the test, combine train set and validation set and train the model with the combined dataset
  - Do ensemble
  - The hyperparameters configuration of competitors who use similar model is a good starting point
  - Use MLOps tool (e.g., wandb)

Further Hyperparameters

1) Accumulation steps
2) Random seed
3) Number of evaluation

Further Hyperparameters

1) Accumulation steps

- 16 batch with no accumulation vs 4 batch with 4 accumulation step, which can result in better performance?

```python
predictions = model(inputs)                 # Forward pass
loss = loss_function(predictions, labels)   # Compute loss function
loss.backward()                             # Backward pass
optimizer.step()                            # Optimizer step
predictions = model(inputs)                 # Forward pass with new parameters
```

pytorch_training.py hosted with ❤ by GitHub

```python
model.zero_grad()                                   # Reset gradients tensors
for i, (inputs, labels) in enumerate(training_set):
    predictions = model(inputs)                     # Forward pass
    loss = loss_function(predictions, labels)       # Compute loss function
    loss = loss / accumulation_steps                # Normalize our loss (if averaged)
    loss.backward()                                 # Backward pass
    if (i+1) % accumulation_steps == 0:             # Wait for several backward steps
        optimizer.step()                            # Now we can do an optimizer step
        model.zero_grad()                           # Reset gradients tensors
        if (i+1) % evaluation_steps == 0:           # Evaluate the model when we...
            evaluate_model()                        # ...have no gradients accumulated
```

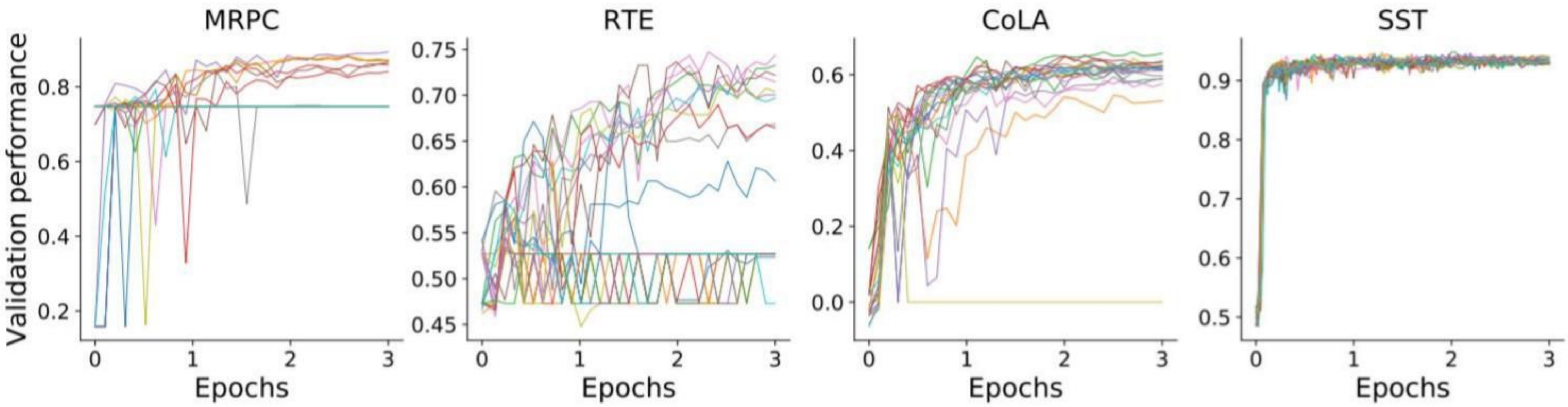
gradient_accumulation.py hosted with ❤ by GitHub

https://medium.com/huggingface/training-larger-batches-practical-tips-on-1-gpu-multi-gpu-distributed-setups-ec88c3e51255

Further Hyperparameters

2) Random Seed

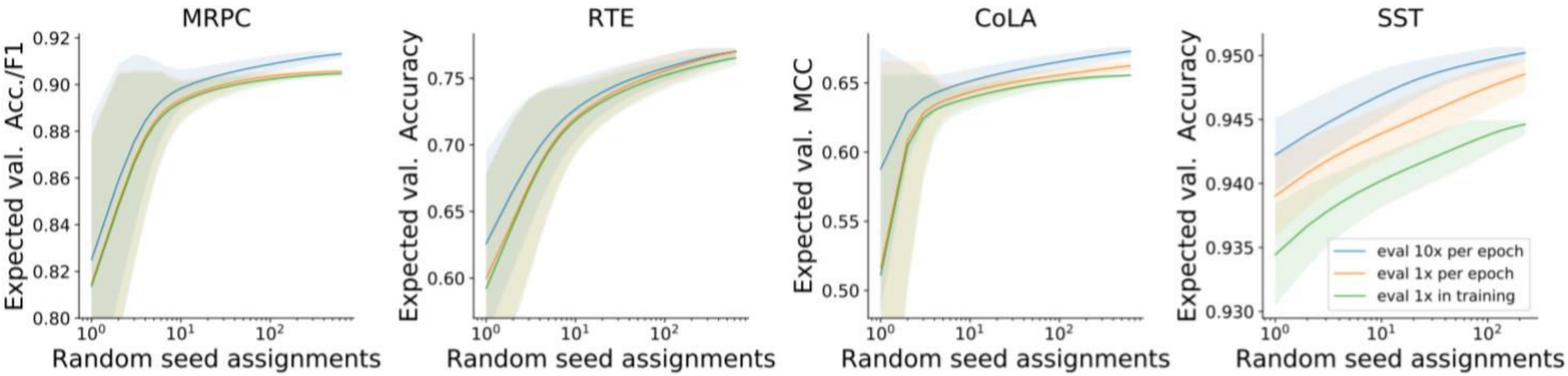
- There is a promising seeds
- These seeds can be distinguished early in training



Further Hyperparameters

3) Number of evaluation

- Expected validation performance as the number of evaluation increases

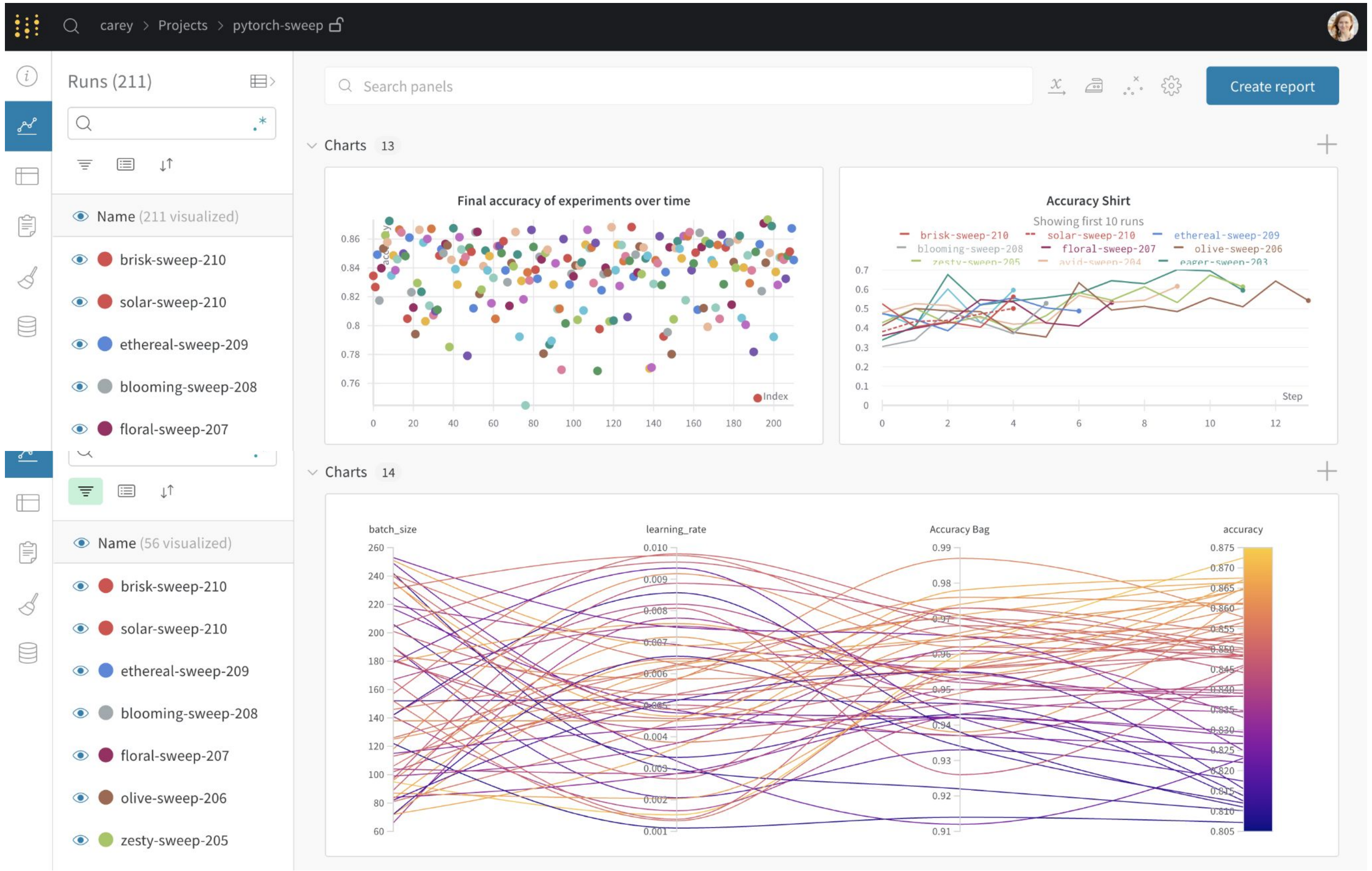


How to control randomness?

- random.seed()
- np.random.seed()
- torch.manual_seed()
- torch.cuda.manual_seed() / torch.cuda.manual_seed_all()
- torch.backends.cudnn.deterministic = True
- torch.backends.cudnn.benchmark = False
- torch.use_deterministic_algorithms(True)
- If you use CUDA tensors, we need to set the environment variable CUBLAS_WORKSPACE_CONFIG according to CUDA documentation

Wandb

- Experiment Tracking: https://docs.wandb.ai/quickstart
- Hyperparameter tuning: https://docs.wandb.ai/guides/sweeps



References

- http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture08.pdf