Hyperparameter Optimization on DNN with various Hyperparameter tuning processes

Turja Kundu
856511795

What are Hyperparameters?

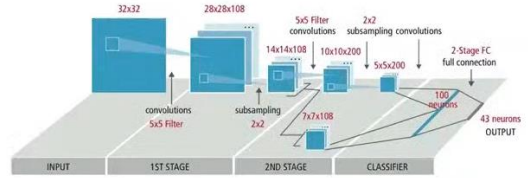
- Model parameters – internal values in a model determided from data
  - In neural networks: weights and bias
- Model hyperparameters – External values to model that determine model capacity
  - In neural Networks

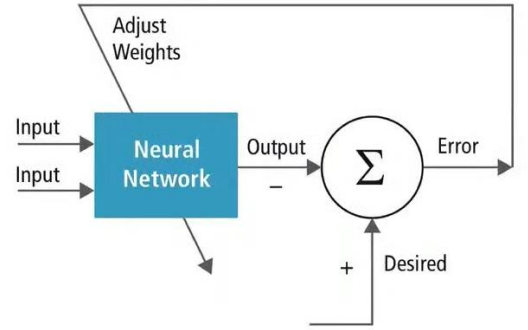
-- Topology

- Number of neurons in fully connected layers
- Filters, kernel sizes, convolution/pooling strides
- Nonlinearity: logistic, ReLU, tanh

-- Training

- Learning rate, batch size, momentum
- Dropout probability, batch normalization
- Optimizers(SGD, Adam, RMSProp, AdaGrad, etc)

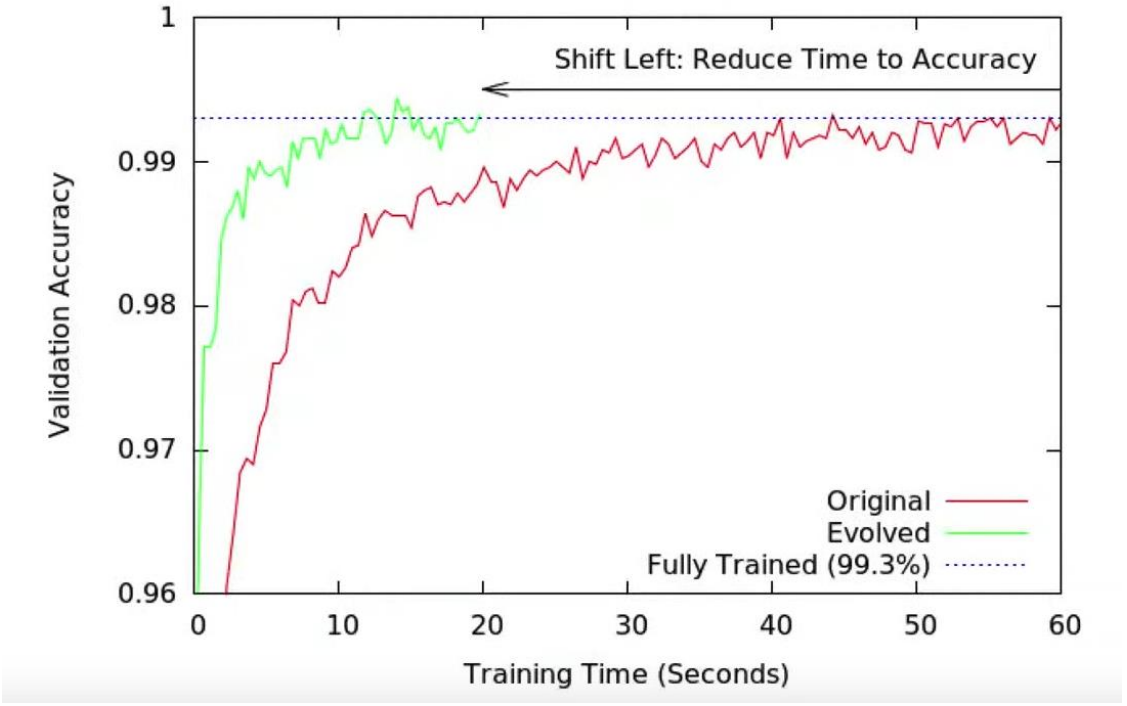




Why optimize hyperparameters?

- Finding a good set of hyperparameters can have a big impact:
  - Accuracy
  - Time-to-accuracy
  - Preventing underfitting / overfitting

- Hyperparameter optimization consistently highlighted as important part of ML workflows



How are Hyperparameter optimized?

- By hand
  - Hyperparameters are selected and tuned manually
  - Guided by intuition and rules of thumb
- Automatic hyperparameter optimization
  - Brute-force of entice search space intractable
  - Evaluate a subspace

    - Grid search
    - Random search
    - Bayesian
    - Genetic / evolutionary algorithms

Tuning By hand

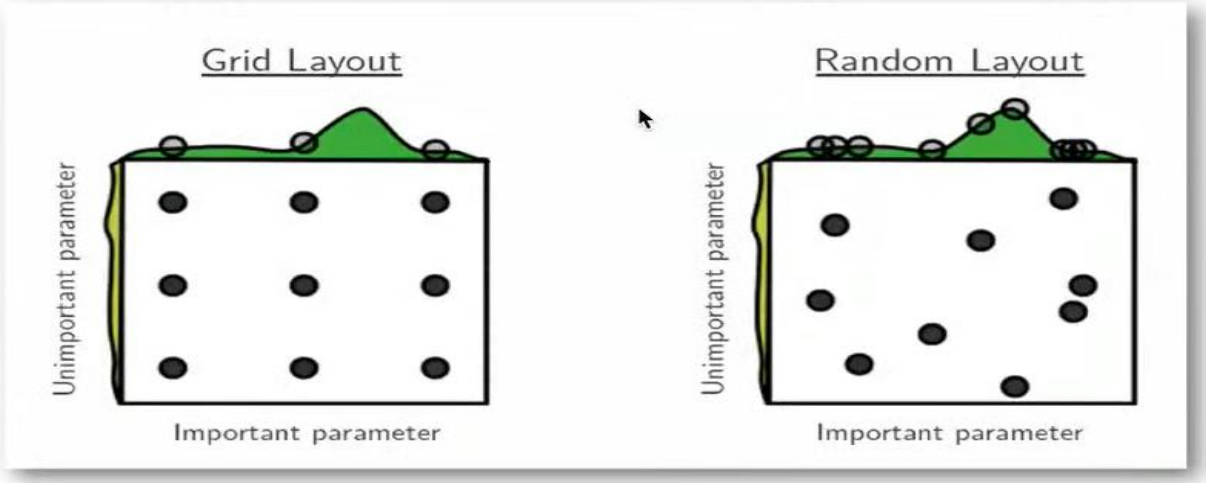
- Just fiddle with the parameters until you get the results you want
- Probably the most common type of hyperparameter optimization
- Upsides: the results are generally pretty good
- Downsides: lots of effort, and not theoretical guarantees

Grid Search

- Define some grid of parameters
- Try all the parameter values in the grid
  - By running the whole system for each setting of parameters
- Then choose the setting with the best result
- Essentially a brute force method
- Downsides: As the number of parameters increases, the cost of grid search increases exponentially
  - Early stopping to the rescue: Can run all the grid points for one epoch, then discard the half that performed worse, then run for another epoch, discard half, and continue.
  - Can take advantage of parallelism: Run all the different parameter settings independently on different servers in a cluster. doesn't reduce the energy cost

Random Search

- This is just grid search, but with randomly chosen points instead of points on a grid.
- This solves the curse of dimensionality
  - Don’t need to increase the number of grid points exponentially as the number of dimensions increases.

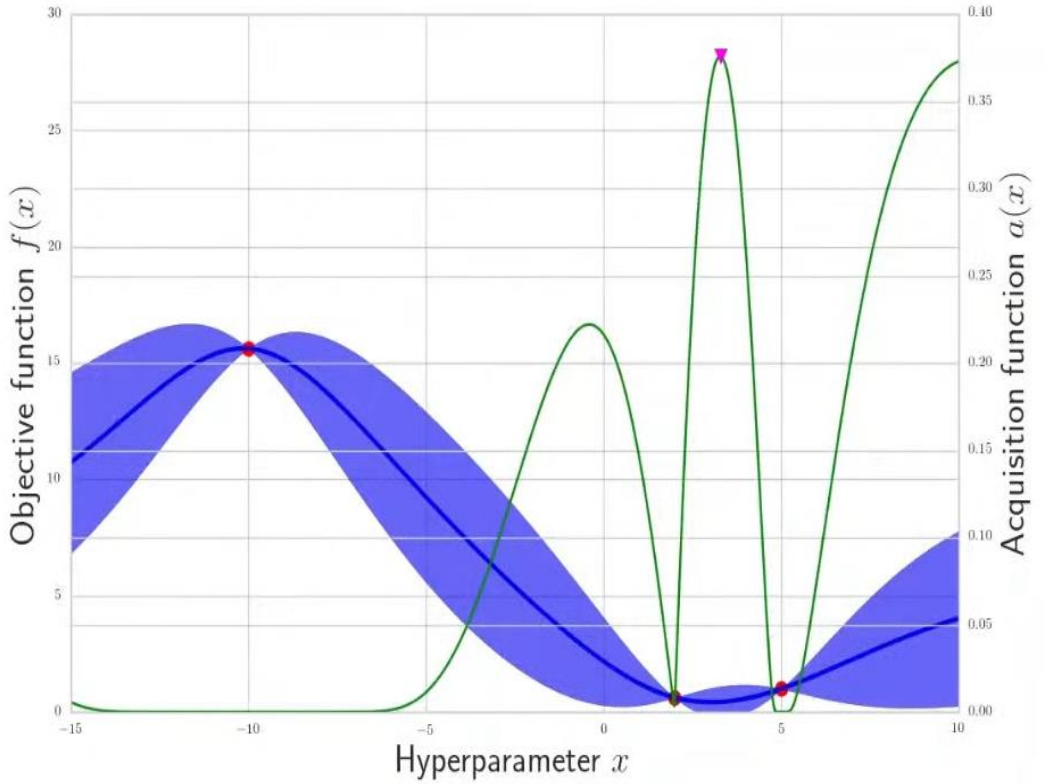


Bayesian Optimization

- Statistical approach for minimizing noisy black-box functions.
- Fit a surrogate model(Gaussian process) to observations
- Compute the acquisition fuction and select new points accordingly
- Evaluate new points
- Repeat

- Upside: empirically it has been demonstrated to get better results in fewer Experiments Compared with grid search and random search

- Downside: it's a pretty heavyweight method The updates are not as simple-to-implement as grid search



Genetic search

- Inspired by biological systems found in nature
  - Each hyperparameter a "gene"
  - Each distinct set of hyperparameters values an individual organism

- Combines exploration and exploitation
  - Fittest individual from a generation used to generate individuals in next generation
  - Crossover from parents in previous generations to generate net childred in new generation
  - Mutation to encourage exploration
  - Separate populations, occasionally combined to encourage exploration of different areas of hyperparameter space

Hyperparameter optimization libraries

	Ray Tune	Optuna	Hyperopt	Sckit-Optimize	Microsoft's NNI
Open sourced	Yes	Yes	Yes	Yes	Yes
Algorithms	Ax/Botorch, HyperOpt, and Bayesian Optimization	AxSearch, DragonflySearch, HyperOptSearch, OptunaSearch, BayesOptSearch	Random Search, Tree of Parzen Estimators, Adaptive TPE	Bayesian Hyperparameter Optimization	Bayesian optimization, Heuristic search, Exhaustive search,
Various supported frameworkst	Pytorch,Tensorflow, XGBoost, LIghtGBM, Scikit-Learn, and Keras	Any ML or Deep Learrning framework, PyTorch, TensorFlow, Keras, MXNet, Scikit-Learn, LightGBMt	sklearn, xgboost, Tensorflow, pytorch, etc	Machine Learning algorithms offered by the scikit-learn library	Pytorch, Tensorflow, Keras, Theano, Caffe2

- For my experiment I use NNI HPO library with Pytorch

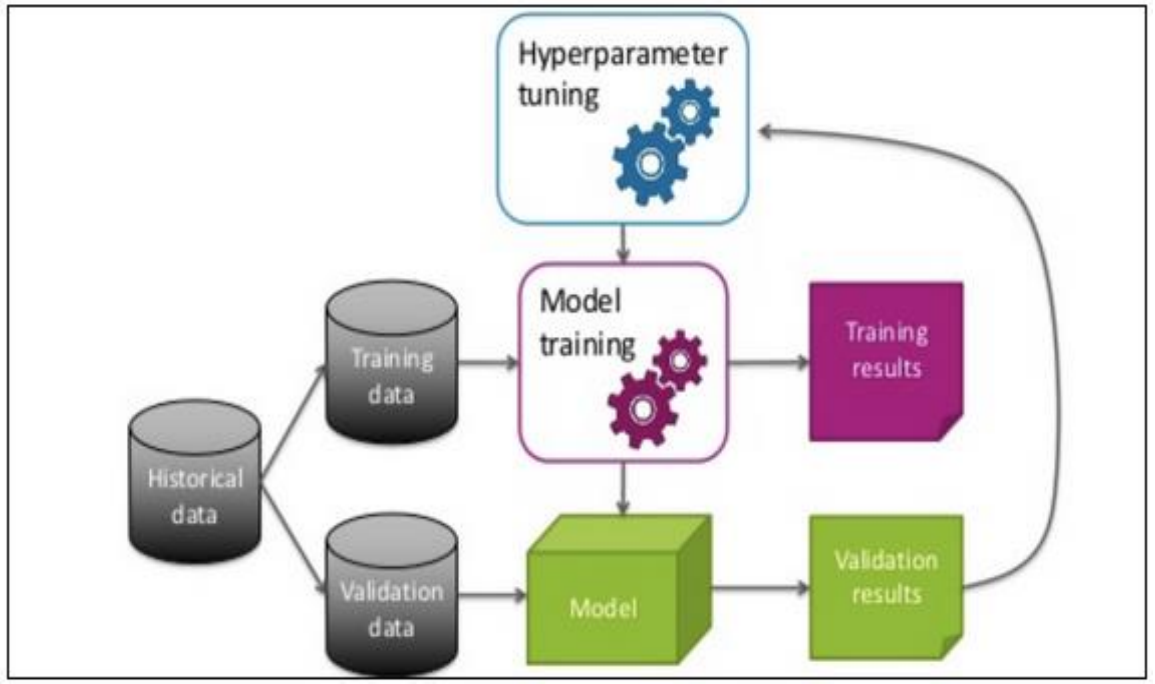
My Experiment Details

- Dataset: FashionMNIST
- Hyperparameter to search:
  -- Batch size
  -- Number of Nodes in a layer
  -- Learning rate
  -- Momentum
  -- Dropout


```
In [160]: search_space = {
              "batch_size": {"_type":"choice", "_value": [16, 32,64,128]},
              'features': {'_type': 'choice', '_value': [16,32,64,128, 256, 512, 1024]},
              'lr': {'_type': 'loguniform', '_value': [0.0001, 0.1]},
              'momentum': {'_type': 'uniform', '_value': [0, 1]},
              'dropout': {'_type': 'uniform', '_value': [0.1, .7]},
          }
```

My Experiment Details

- Hyperparameter search Algorithm used
  - Grid search
  - Random search
  - Evolutionary search (Heuristic search)
  - TPE (Bayesian optimization)
- Metric used: accuracy
- Process:
  - Get hyperparameters from NNI
  - Train model with hyperparameters with training data
  - Test model with test data
  - Report test accuracy to NNI and request optimized hyperparameters
  - Repeat steps



My Experiment Details : Model

```
56
57 train_dataloader = DataLoader(training_data, batch_size=params['batch_size'])
58 test_dataloader = DataLoader(test_data, batch_size=params['batch_size'])
59
60 # %%
61 # Build model with hyperparameters
62 # --------------------------------
63 device = "cuda" if torch.cuda.is_available() else "cpu"
64 print(f"Using {device} device")
65
66 class NeuralNetwork(nn.Module):
67     def __init__(self):
68         super(NeuralNetwork, self).__init__()
69         self.flatten = nn.Flatten()
70         self.linear_relu_stack = nn.Sequential(
71             nn.Linear(28*28, params['features']),
72             nn.ReLU(),
73             nn.Linear(params['features'], params['features']),
74             nn.ReLU(),
75             nn.Dropout(params['dropout']),
76             nn.Linear(params['features'], 128),
77             nn.ReLU(),
78             nn.Linear(128, 10)
79         )
80
81     def forward(self, x):
82         x = self.flatten(x)
83         logits = self.linear_relu_stack(x)
84         return logits
85
86 model = NeuralNetwork().to(device)
87
88 loss_fn = nn.CrossEntropyLoss()
89 optimizer = torch.optim.SGD(model.parameters(), lr=params['lr'], momentum=params['momentum'])
```

Result: Grid Seach

- Max trial Number : 30
- Train and Test Model for 5 epochs

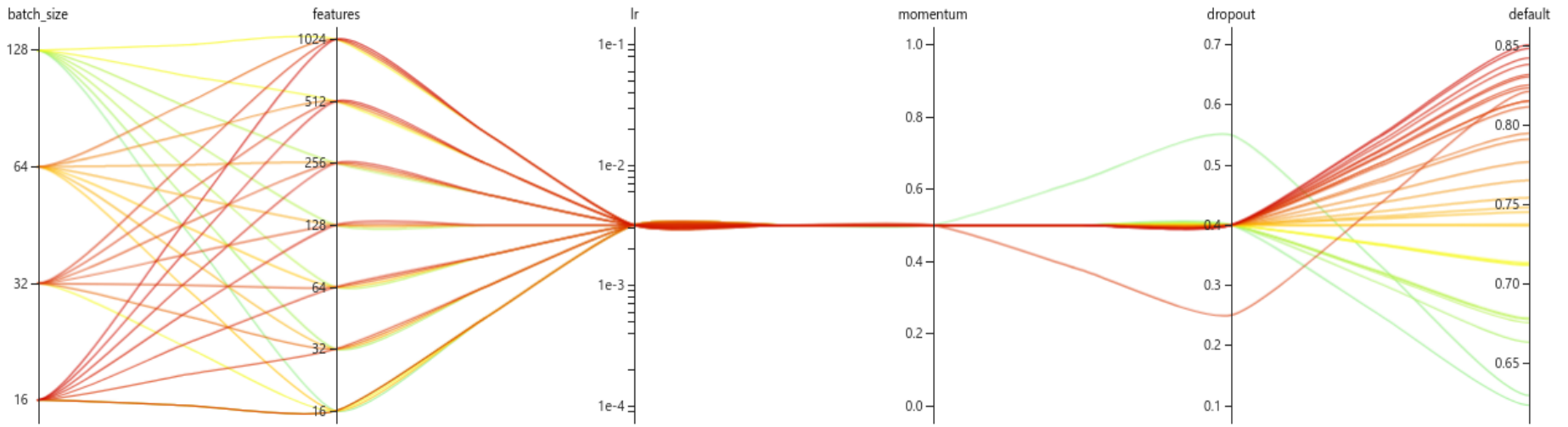
Best Accuracy	0.850500
Total Duration	11m 20s

- Hyperparameters of Top accuracy

Batch size	Nodes in Layer	Learning rate	Momentum	Dropout
16	1024	0.0031622	0.5	0.4

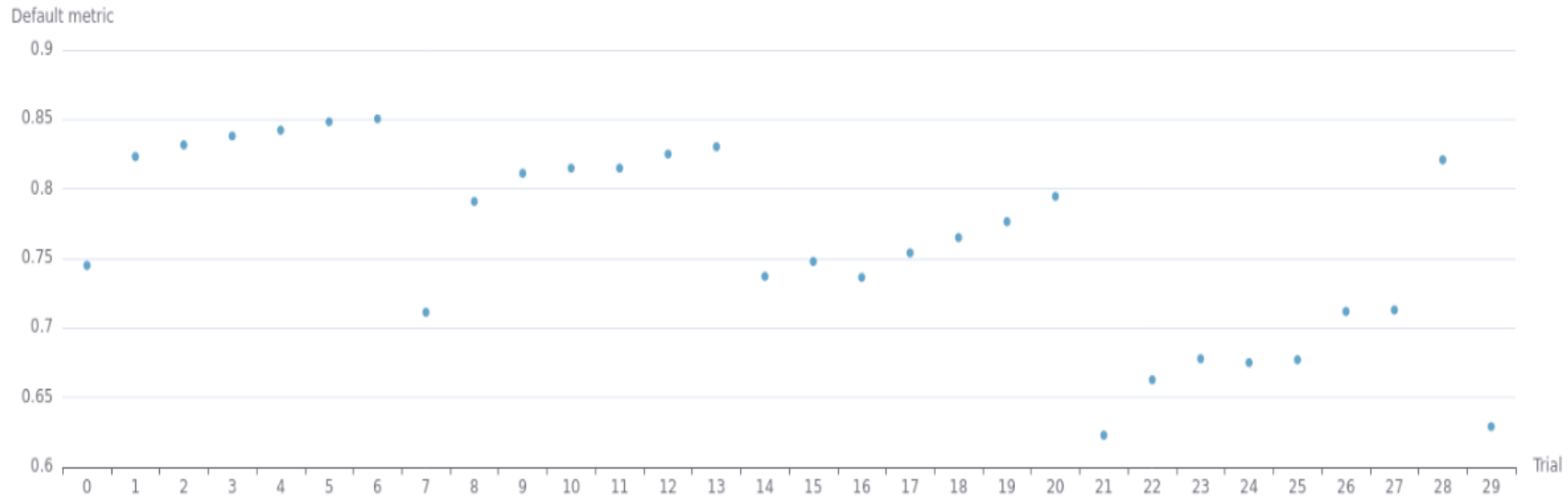
Result: Grid Seach

- Hyperparameters



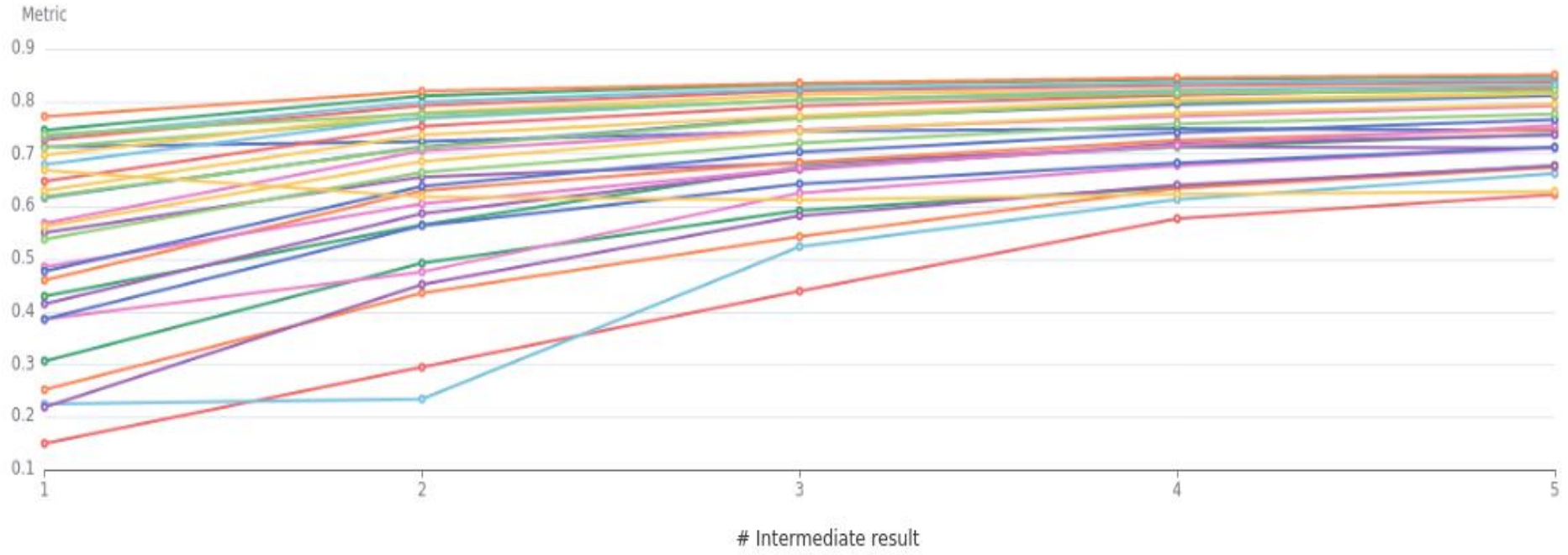
Result: Grid Seach

- Accuracy in different trials



Result: Grid Seach

- Intermediate result



Result: Random Seach

- Max trial Number : 30
- Train and Test Model for 5 epochs

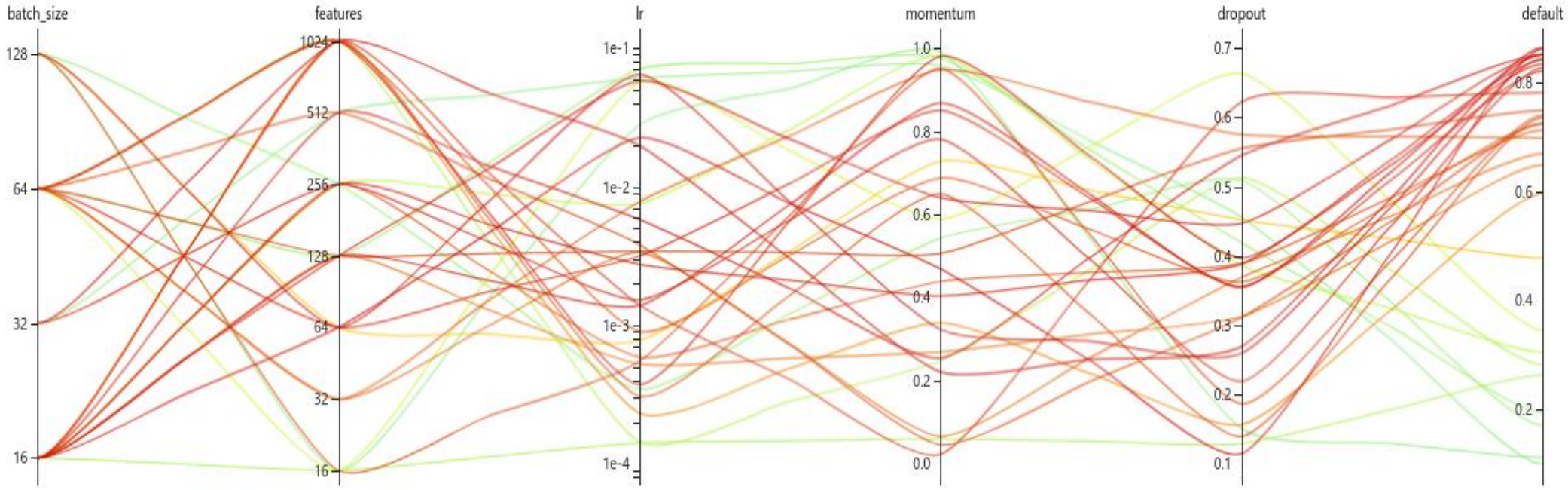
Best Accuracy	0.863600
Total Duration	11m 40s

- Hyperparameters of Top accuracy

Batch size	Nodes in Layer	Learning rate	Momentum	Dropout
32	1024	0.01969	0.2213	0.269621

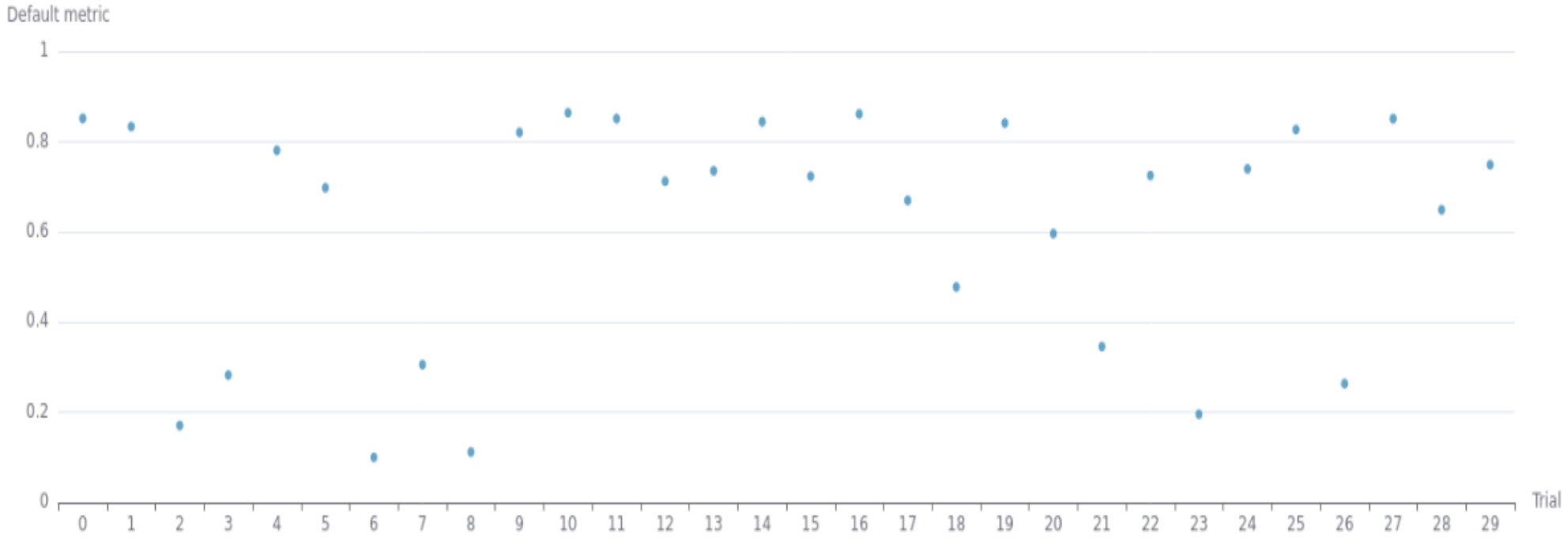
Result: Random Seach

- Hyperparameters



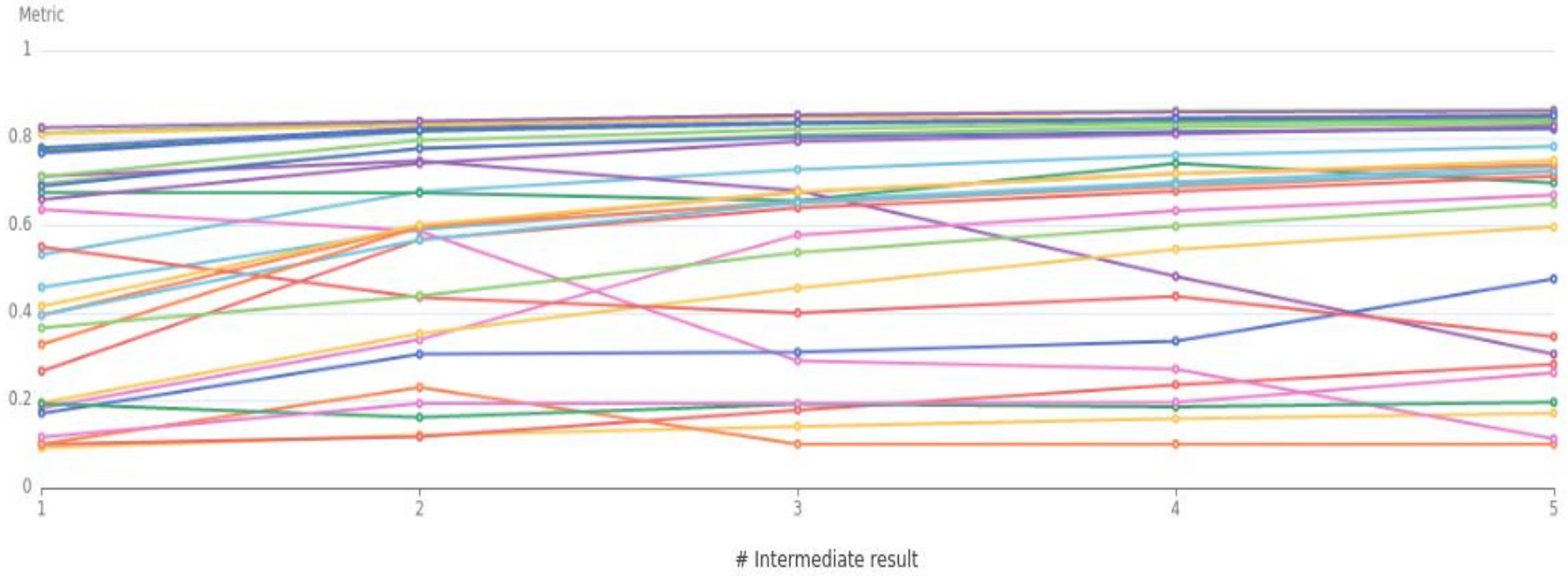
Result: Random Seach

- Accuracy in different trials



Result: Random Seach

- Intermediate result



Result: Evolutionary Seach

- Max trial Number : 30
- Train and Test Model for 5 epochs

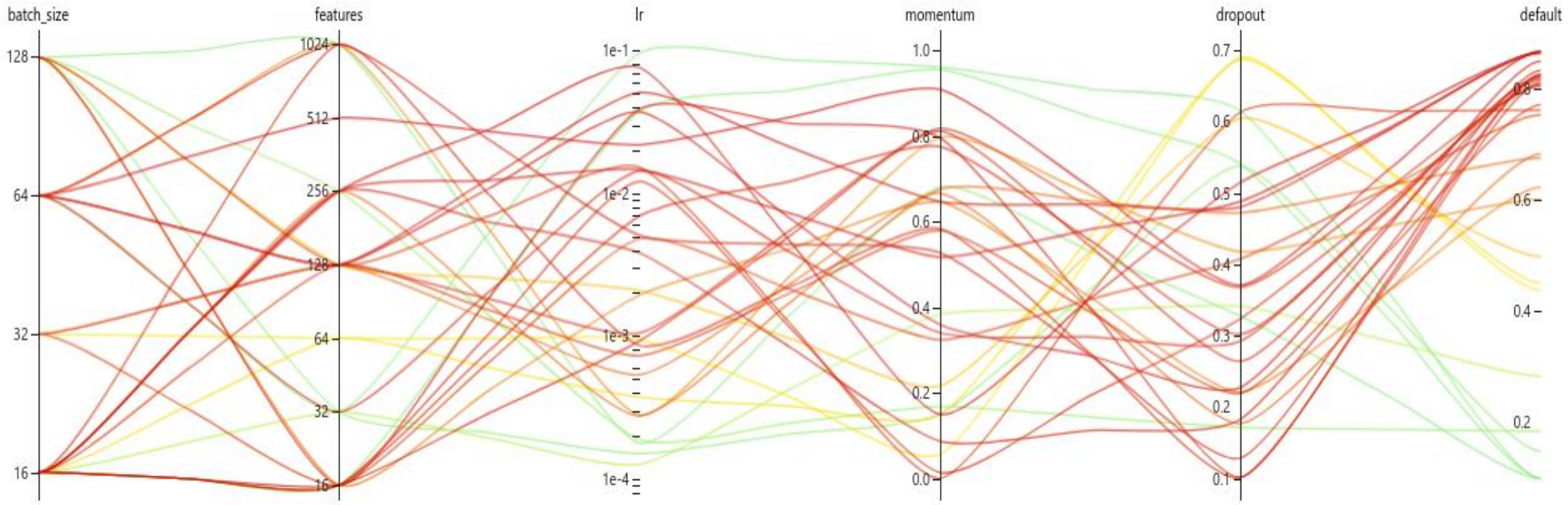
Best Accuracy	0.867000
Total Duration	11m 8s

- Hyperparameters of Top accuracy

Batch size	Nodes in Layer	Learning rate	Momentum	Dropout
64	512	0.0219234	0.9108	0.3715

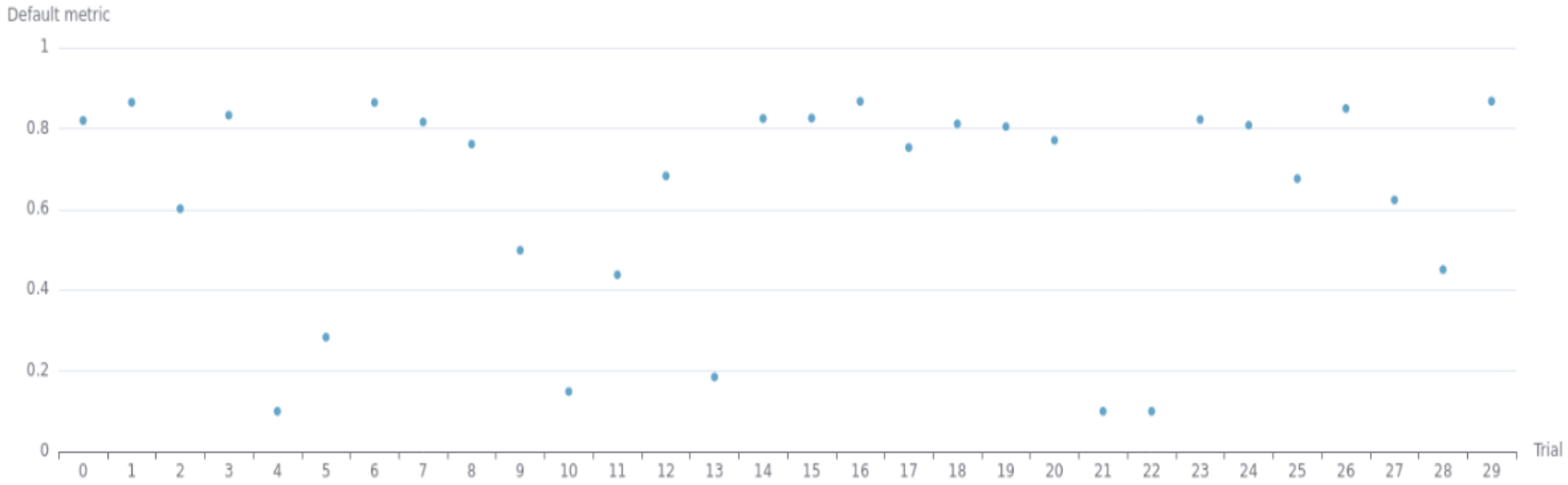
Result: Evolutionary Seach

- Hyperparameters



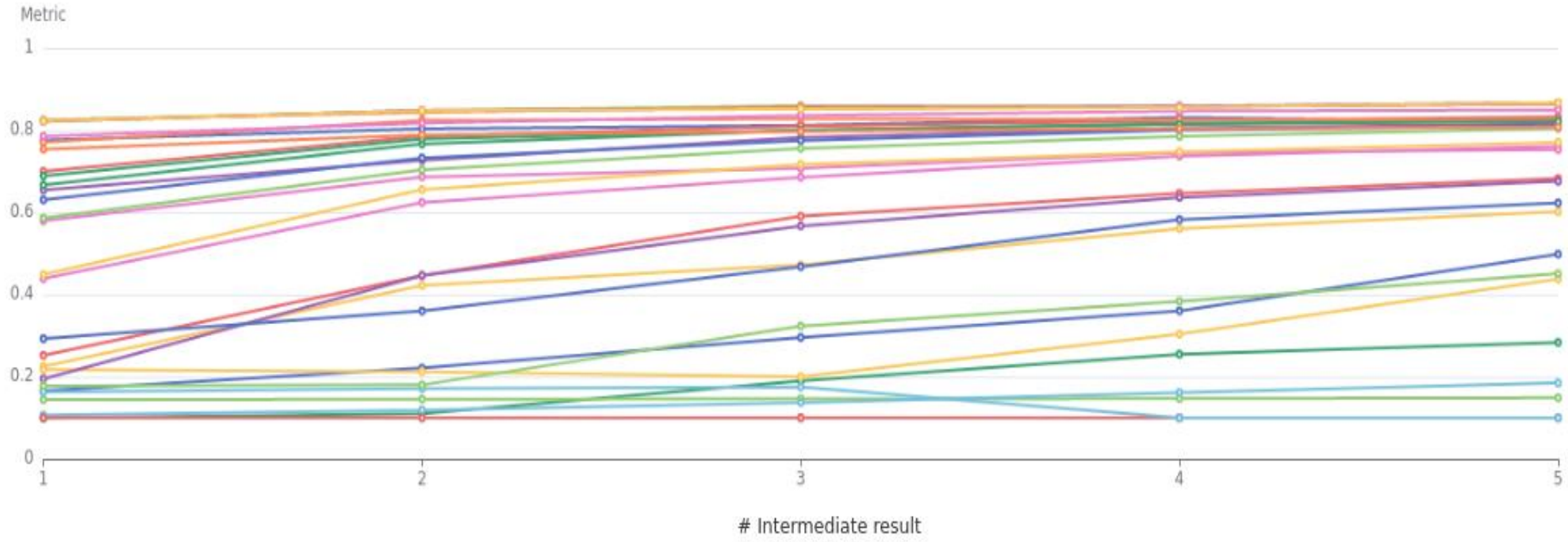
Result: Evolutionary Seach

- Accuracy in different trials



Result: Evolutionary Seach

- Intermediate result



Result: TPE(Bayesian Optimization)

- Max trial Number : 30
- Train and Test Model for 5 epochs

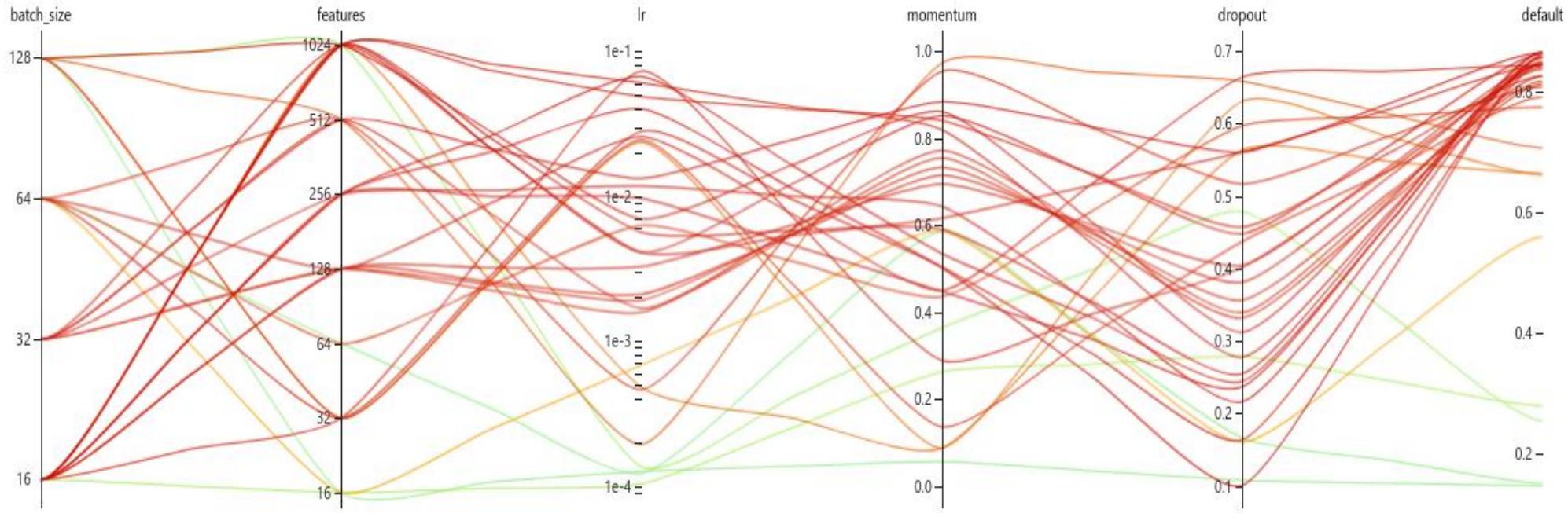
Best Accuracy	0.866300
Total Duration	12m

- Hyperparameters of Top accuracy

Batch size	Nodes in Layer	Learning rate	Momentum	Dropout
16	1024	0.005675	0.59179	0.2552

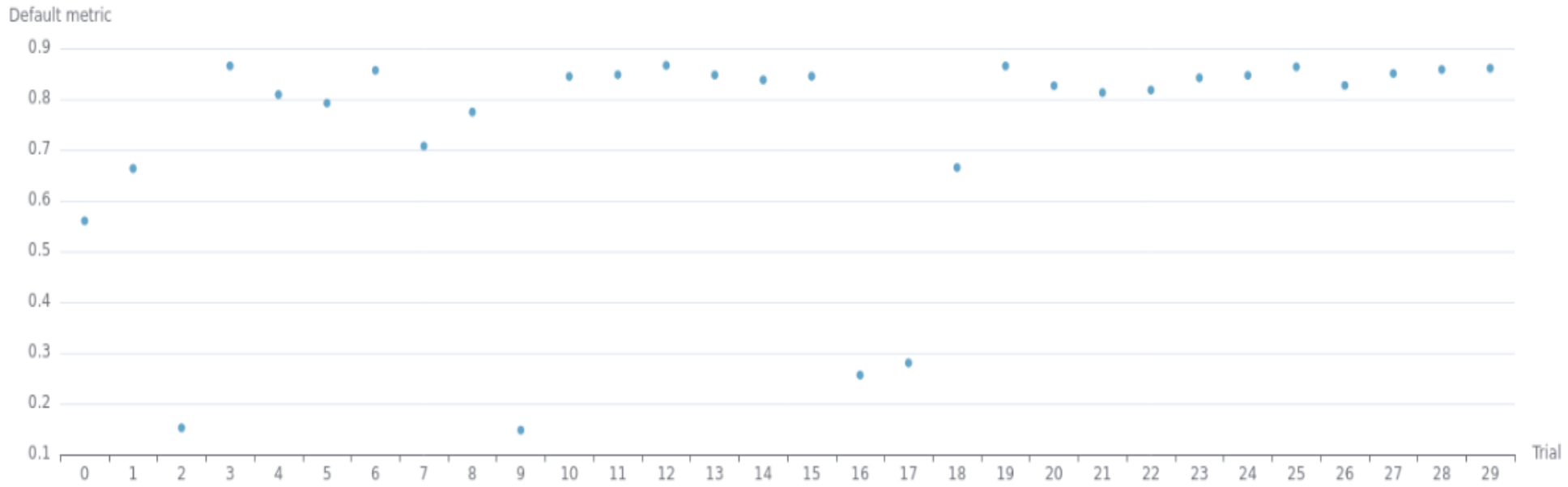
Result: TPE(Bayesian Optimization)

- Hyperparameters



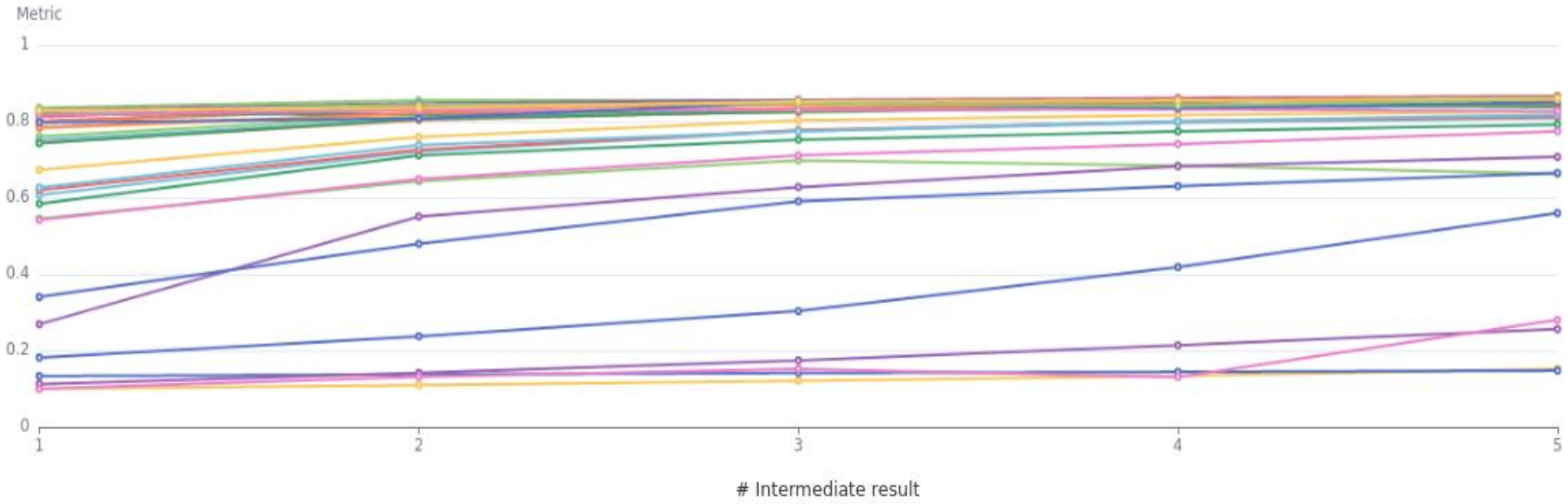
Result: TPE(Bayesian Optimization)

- Accuracy in different trials



Result: TPE(Bayesian Optimization)

- Intermediate result



Summary Results

Hyperparameter Optimization Methods	Best Accuracy
Grid Search	0.850500
Random Search	0.863600
Evolutionary Search	0.867000
TPE(Bayesian Optimization)	0.866300

Thanks