Lab 9: Hyperparameter Search and Knowledge Distillation

University of Washington

ECE 596

Spring 2021

Outline

Part 1: Hyperparameter Search

- Meta-optimization of a neural network
- Hyperparameter tuning methods
- Hyperparameter tuning with hyperopt

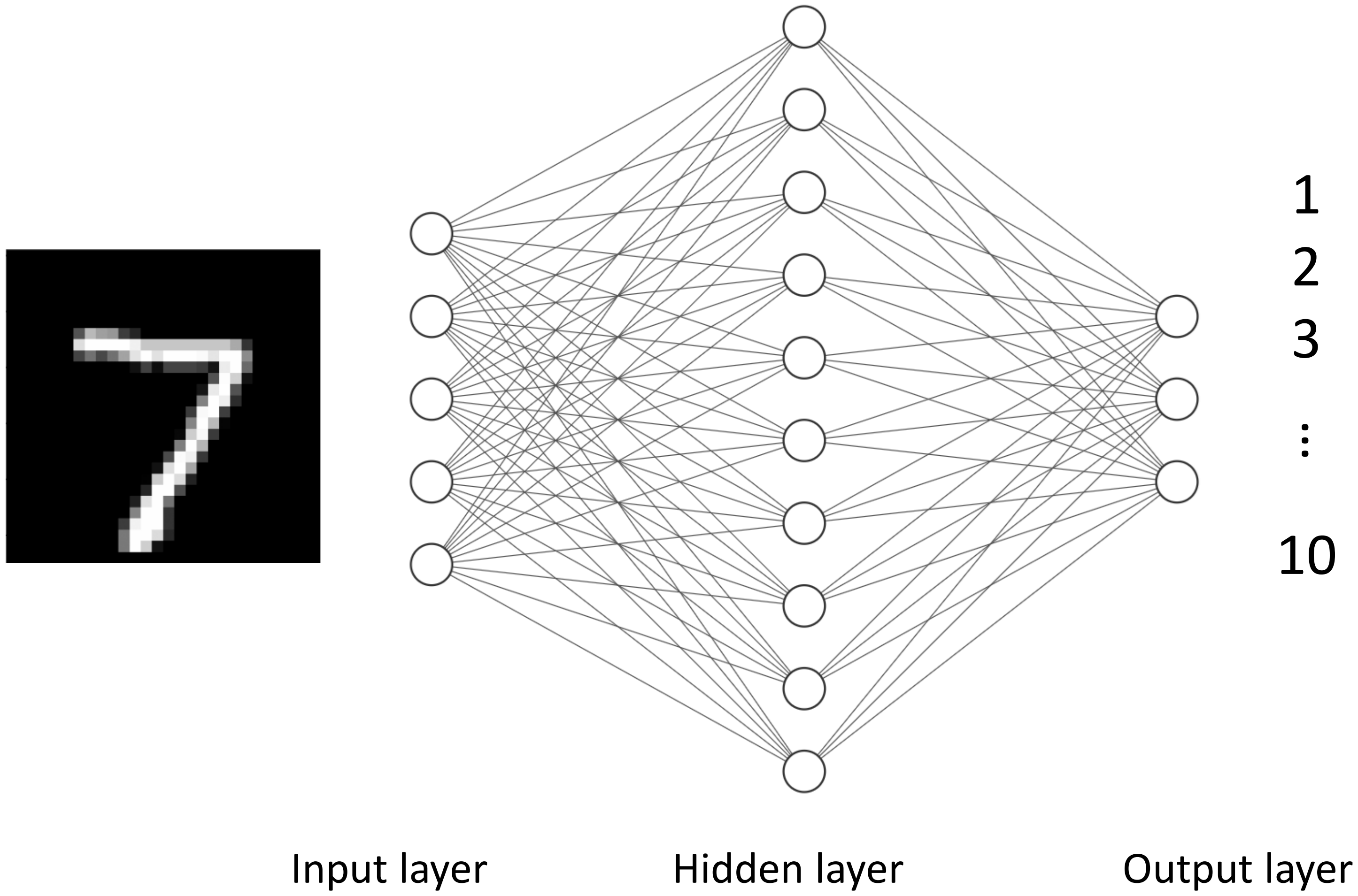
Part 2: Knowledge Distillation

- Motivation
- Distillation training
- Soft targets and temperature
- Example using MNIST dataset

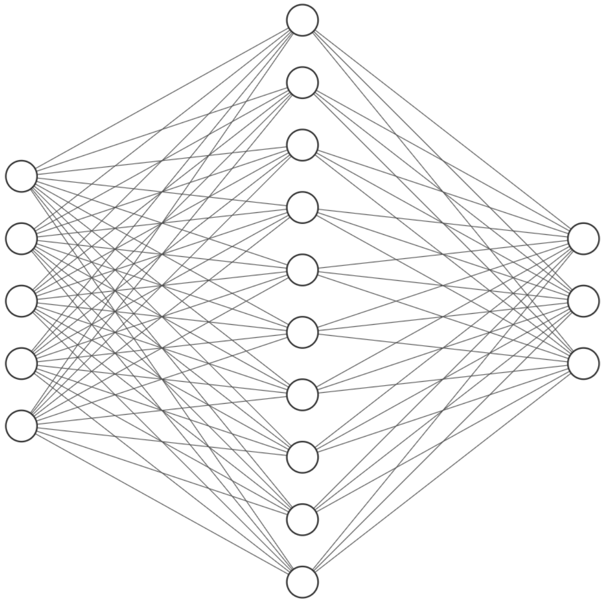
Lab Assignment

Part 1: Hyperparameter Search

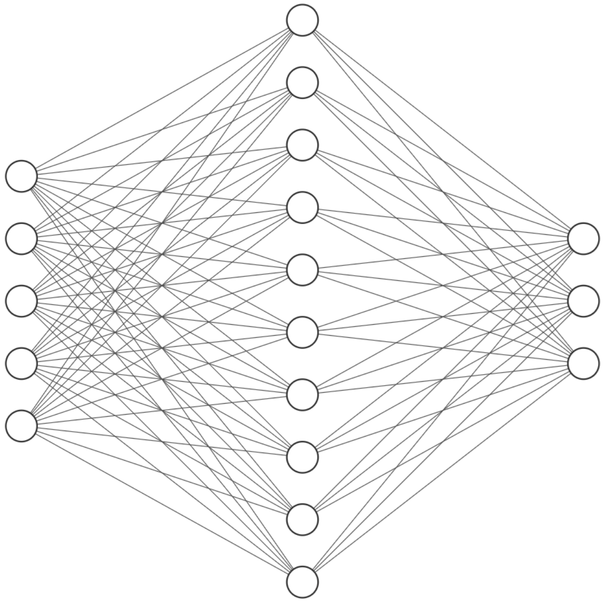
Meta optimization of a neural network



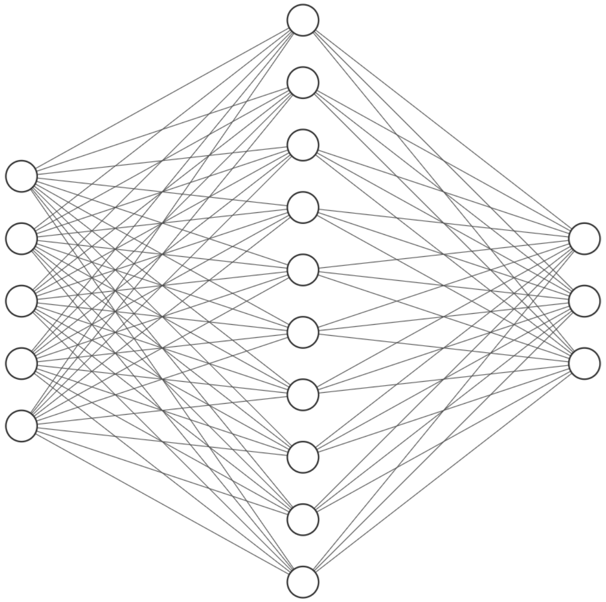
Meta optimization of a neural network



Learning rate = 0.1
Dropout = 0.1
Accuracy = 65%

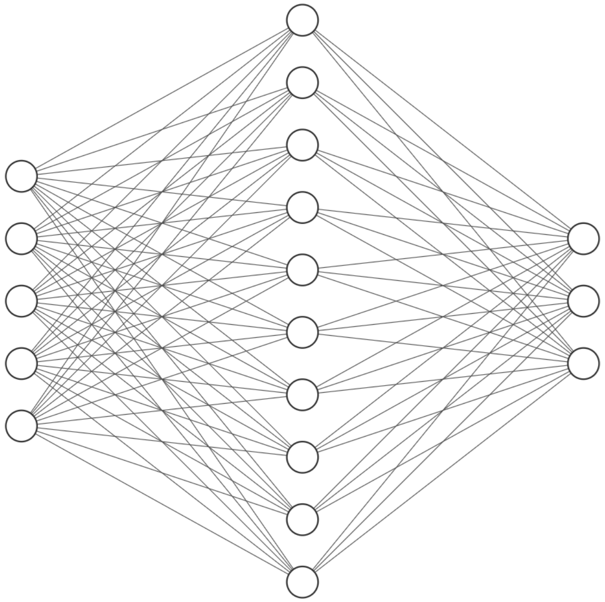


Learning rate = 0.0001
Dropout = 0.25
Accuracy = 85%

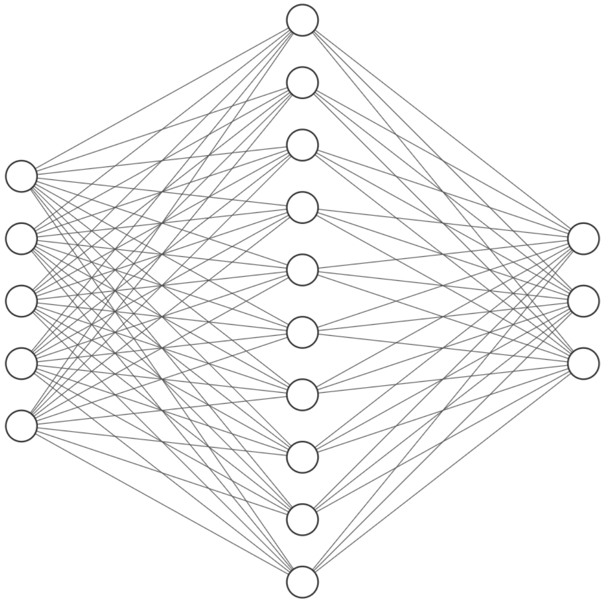


Learning rate = 0.01
Dropout = 0.5
Accuracy = 78%

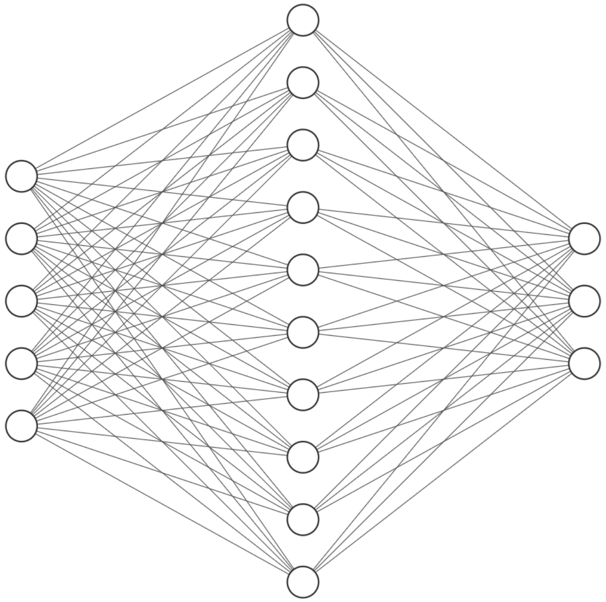
Meta optimization of a neural network



Learning rate = 0.1
Dropout = 0.1
Accuracy = 65%



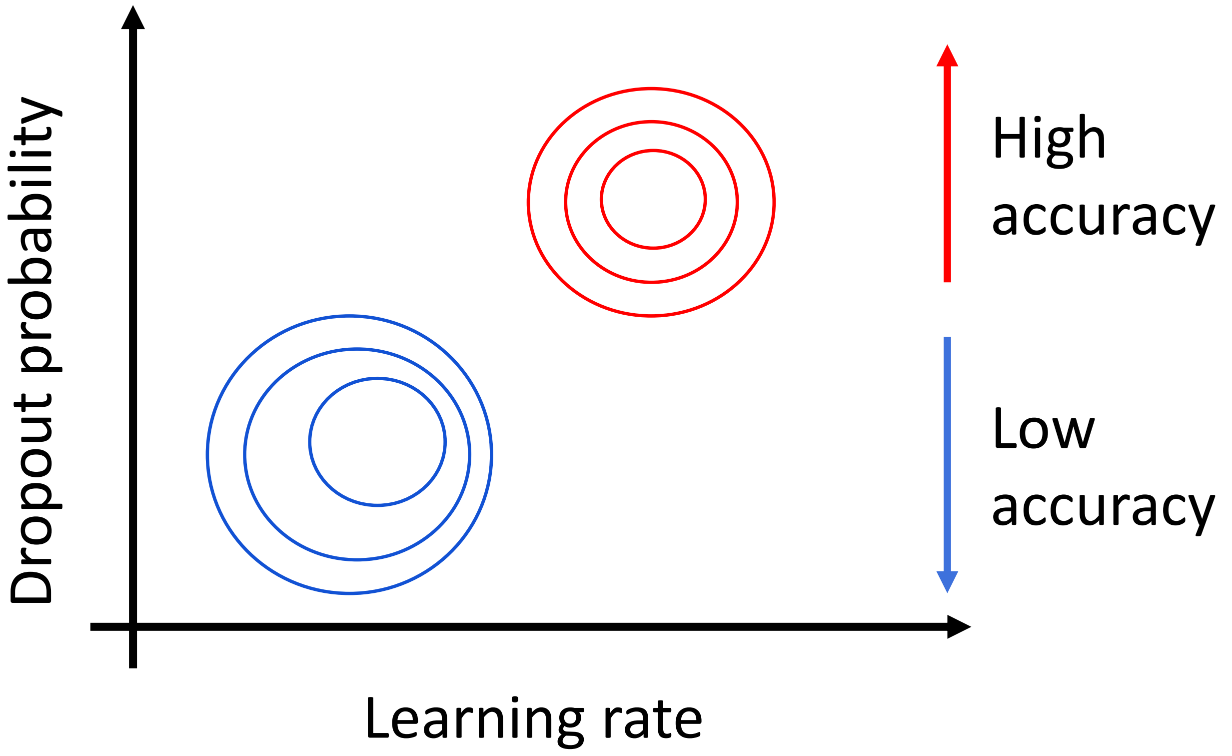
Learning rate = 0.0001
Dropout = 0.25
Accuracy = 85%



Learning rate = 0.01
Dropout = 0.5
Accuracy = 78%

Baby sitting neural network is hard and inefficient

Hyperparameter tuning methods



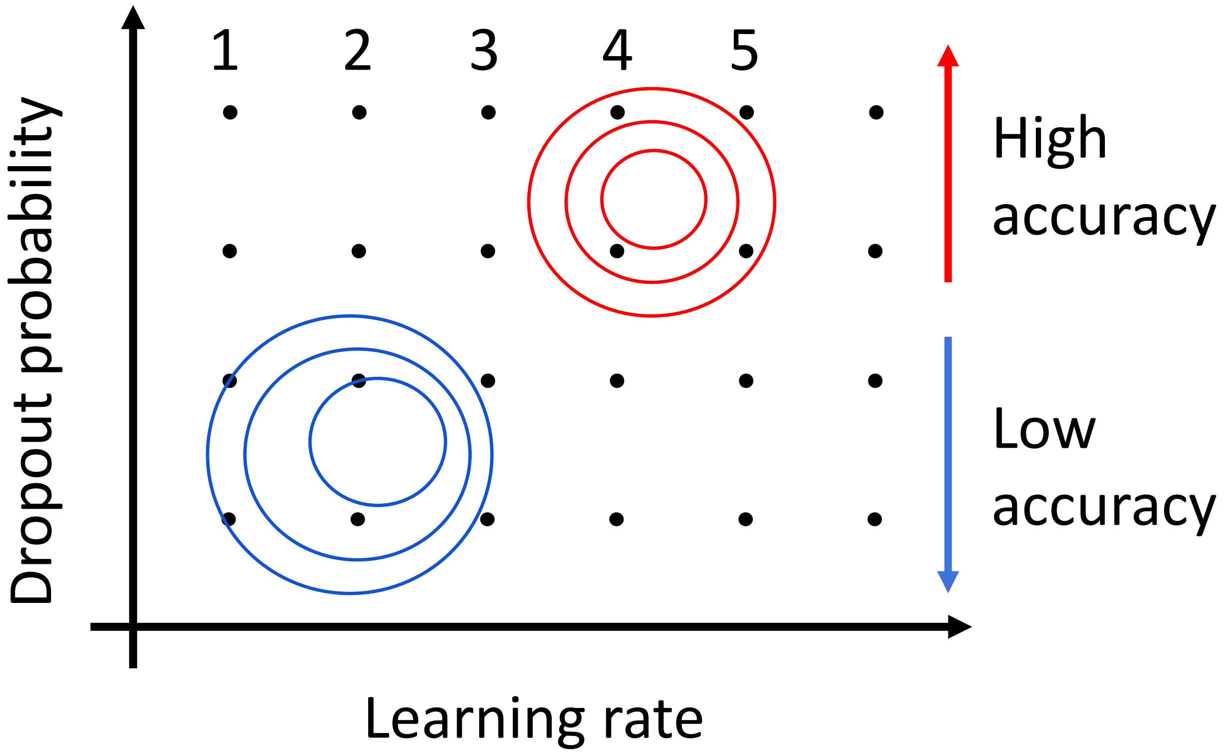
Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods



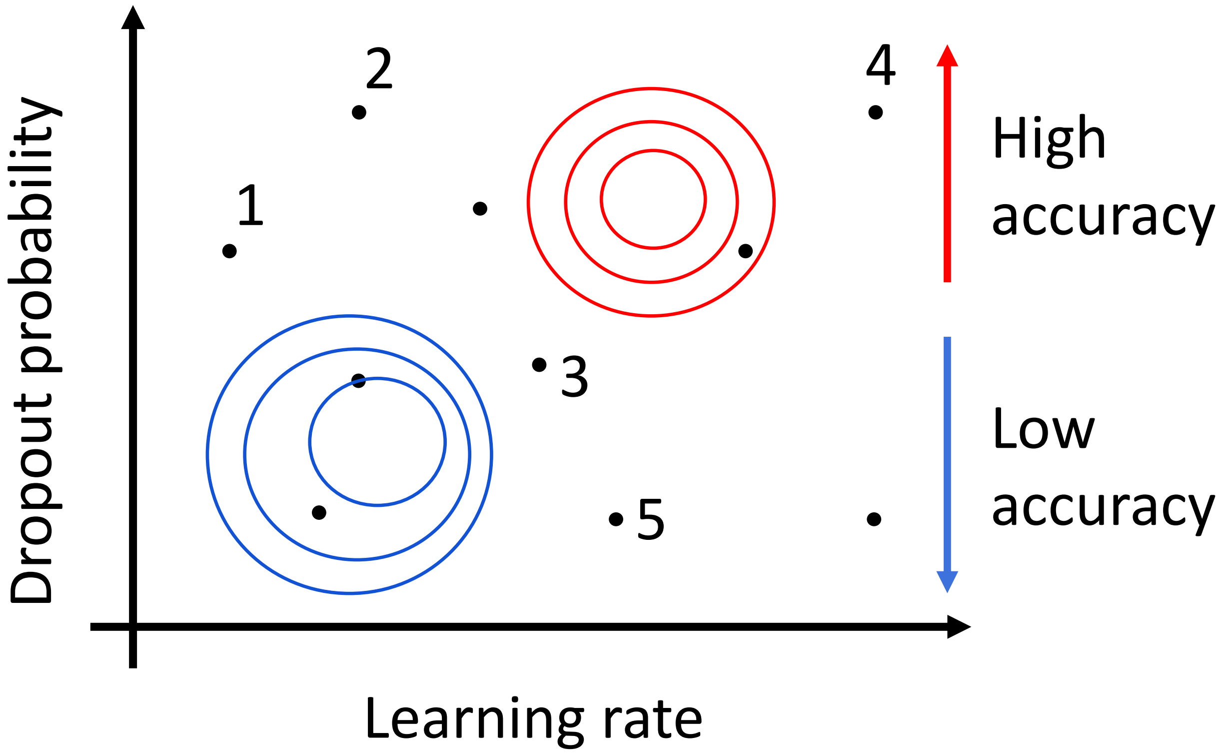
Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods



Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods

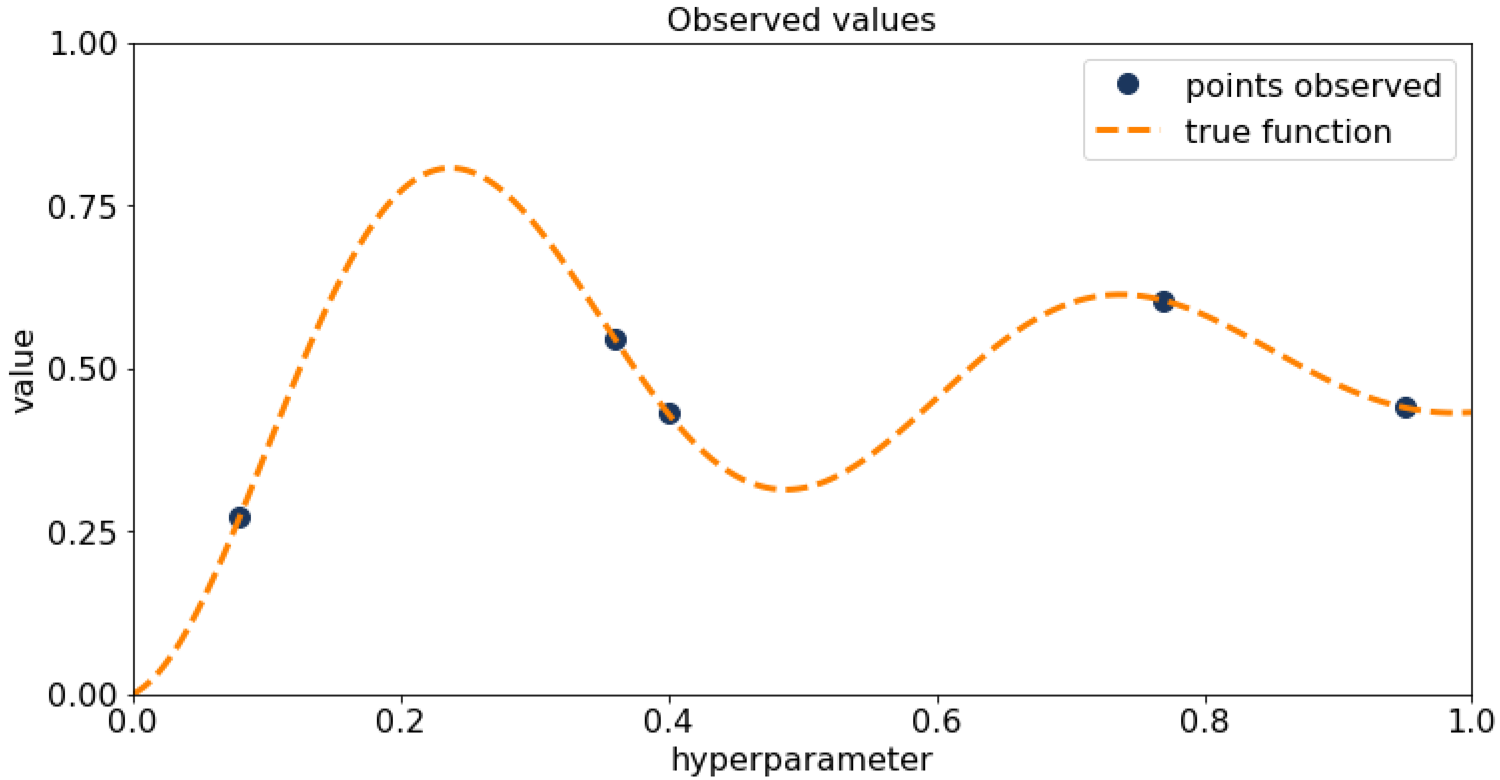


Image credit: MLConf

Step 1

Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods

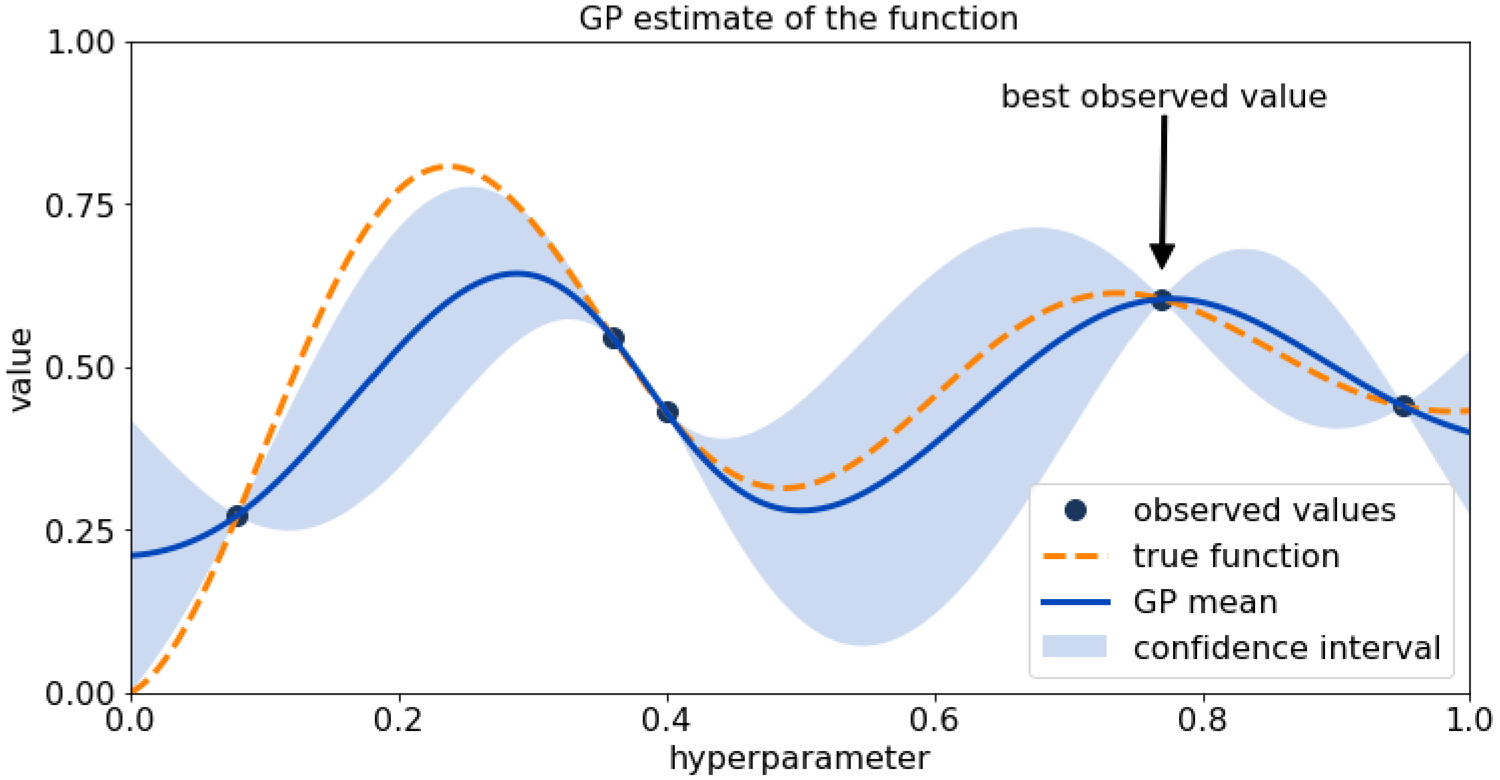


Image credit: MLConf

Step 2

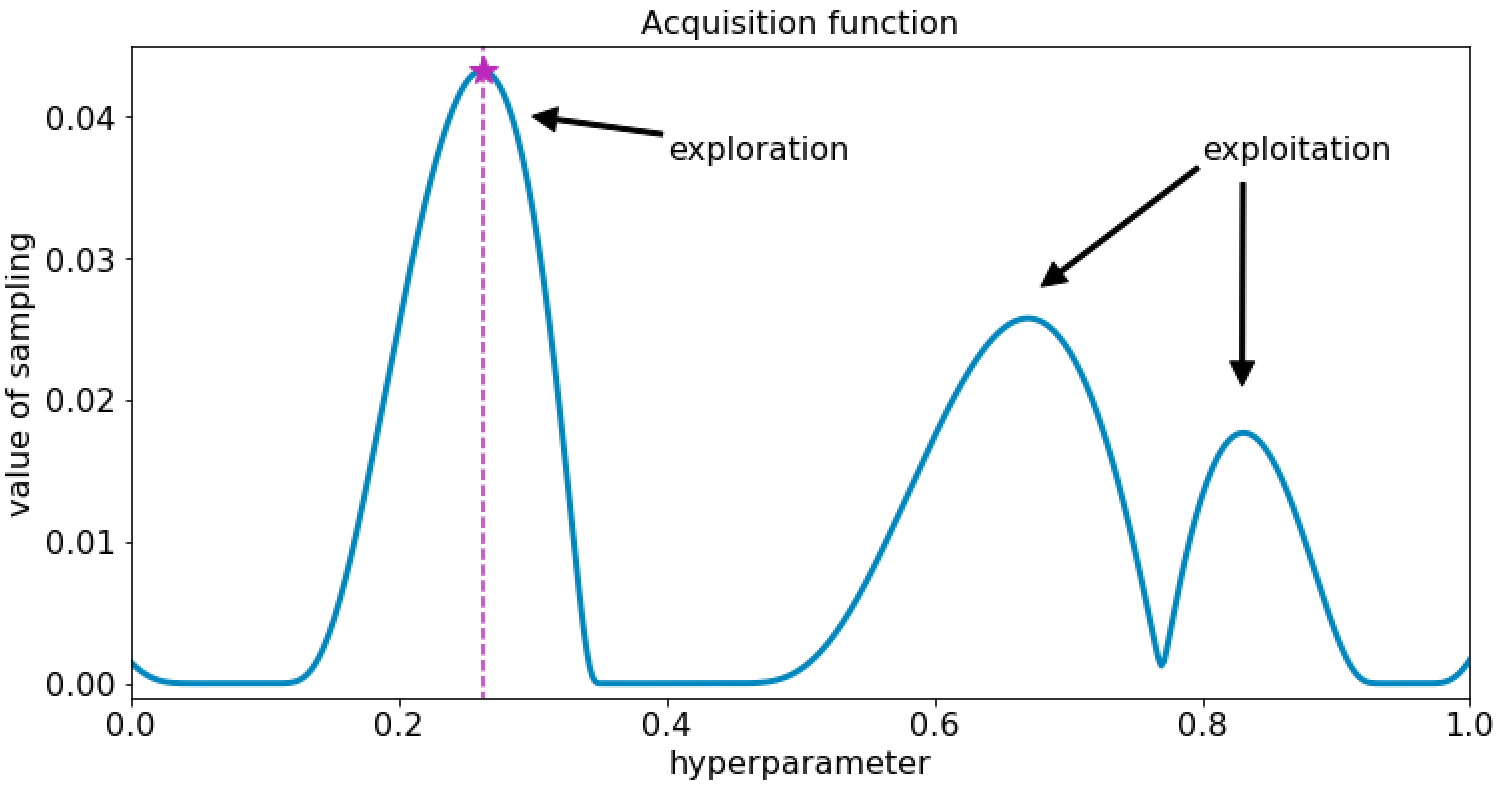
Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods



Step 3

Image credit: MLConf

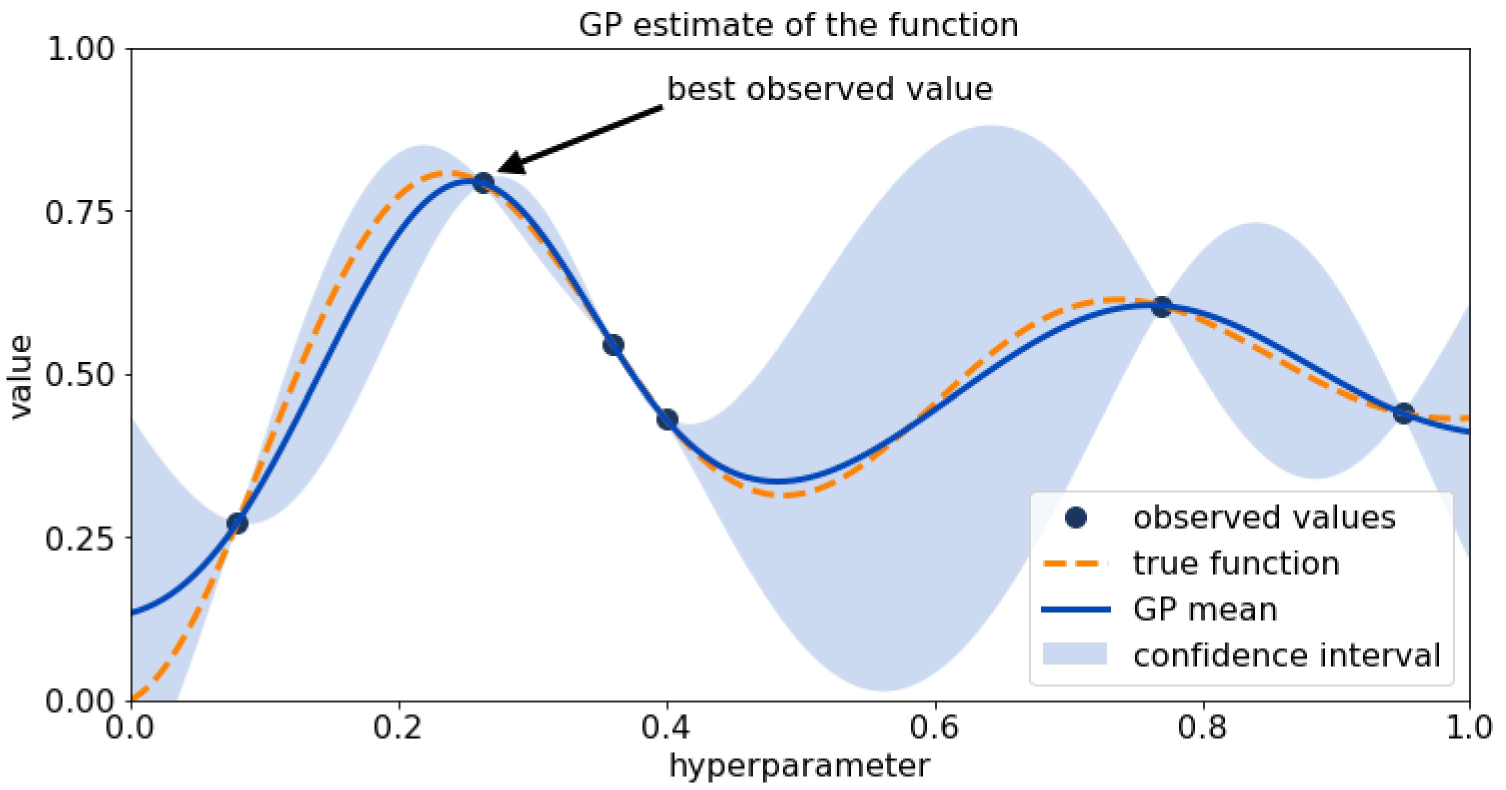
Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods



Step 4

Image credit: MLConf

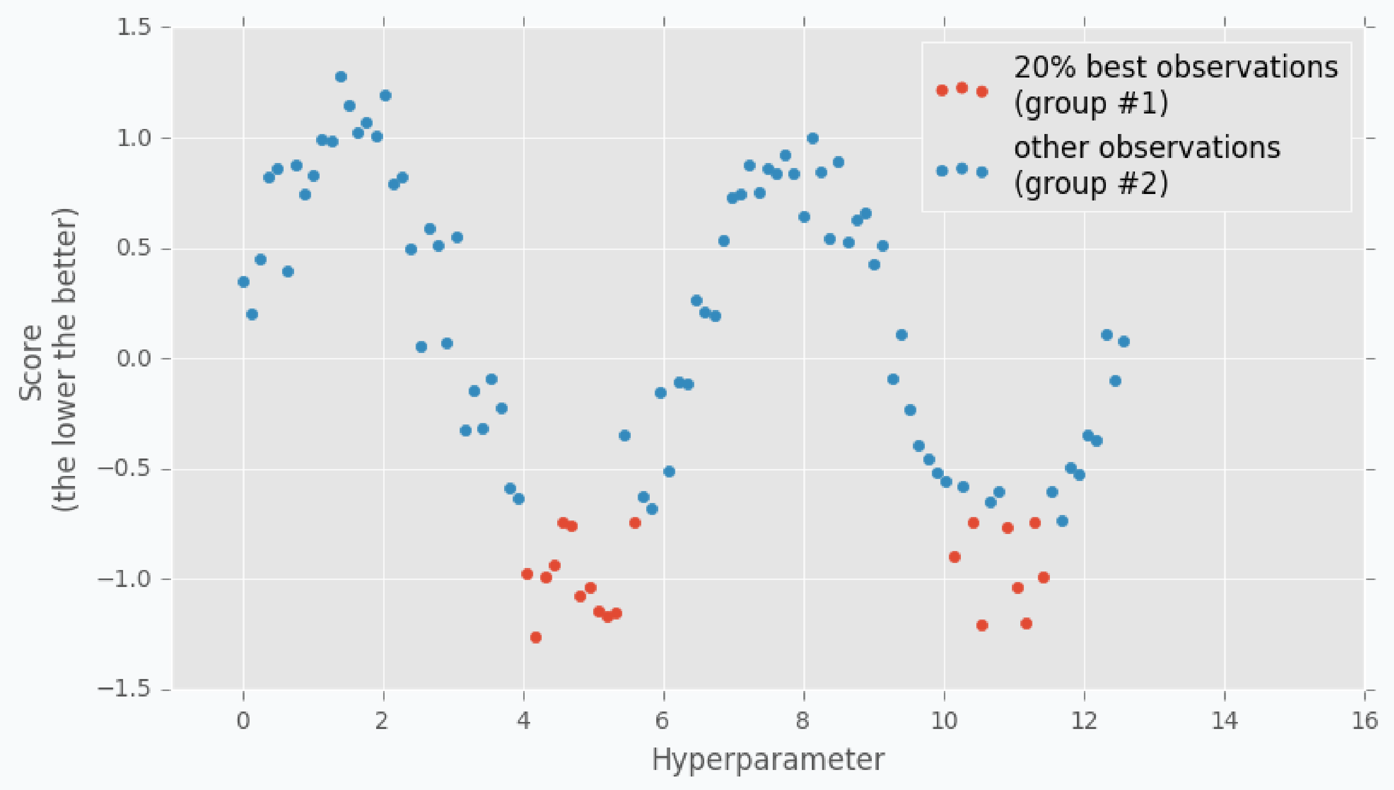
Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods



Step 1

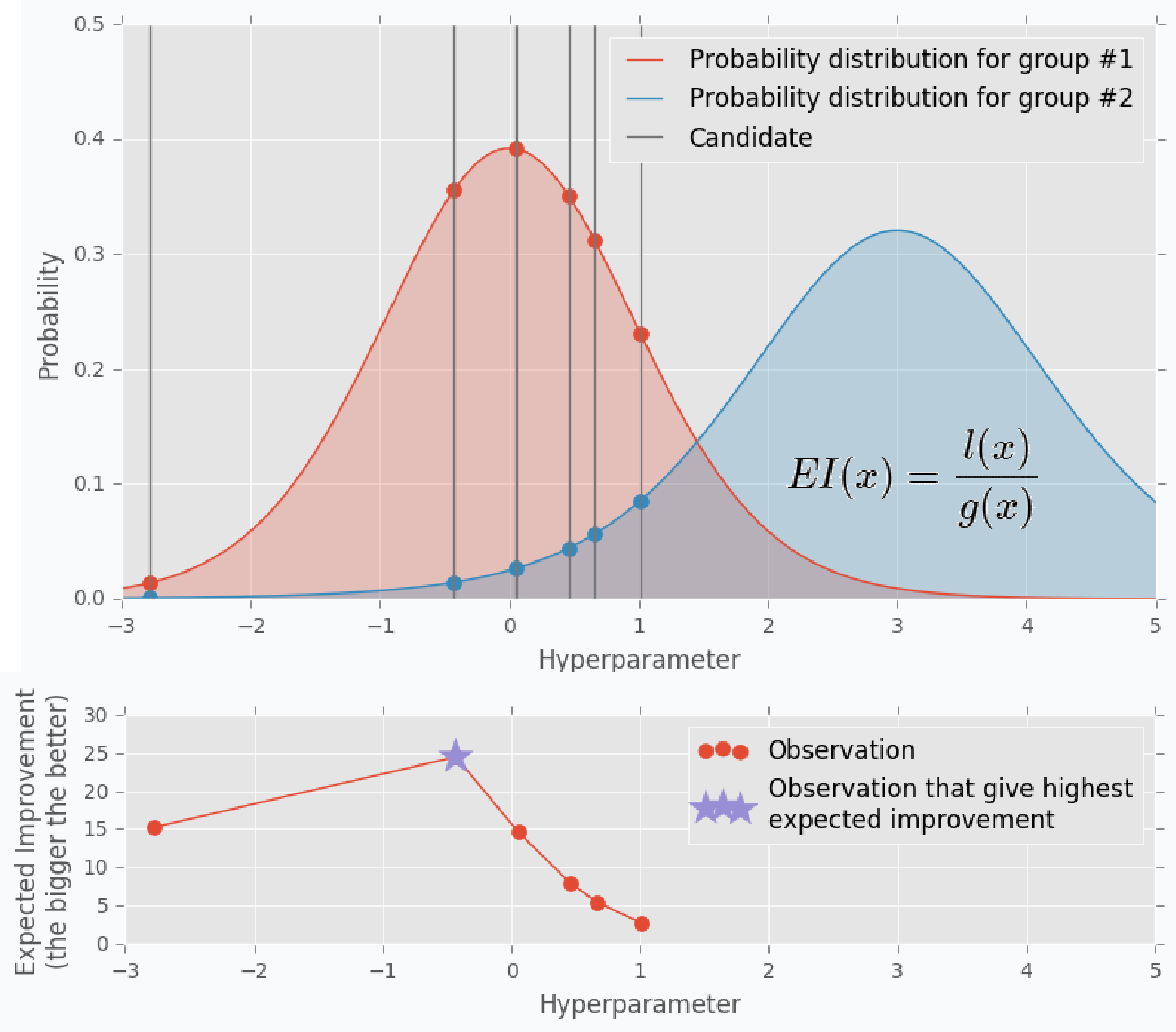
Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning methods



Grid Search

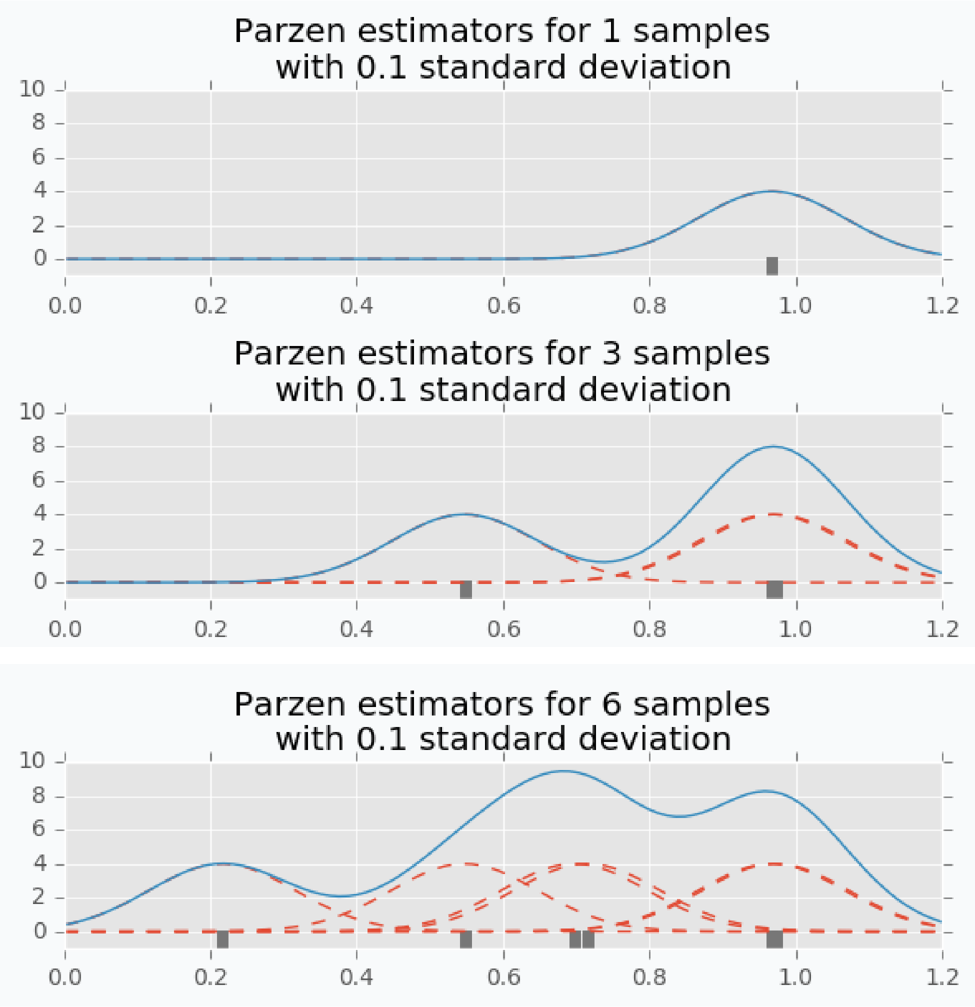
Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Step 2

Hyperparameter tuning methods



Grid Search

Random Search

Bayesian Optimization

Tree-structured Parzan Estimators (TPE)

Hyperparameter tuning with hyperopt

```
from hyperopt import fmin, tpe, hp
import matplotlib.pyplot as plt

def f(x):
    return x**2 - x + 1

space = hp.uniform('x', -5, 5)

best = fmin(
    fn=f,  # "Loss" function to minimize
    space=space,  # Hyperparameter space
    algo=tpe.suggest,  # Tree-structured Parzen Estimator (TPE)
    max_evals=1000  # Perform 1000 trials
)

print(best)
```

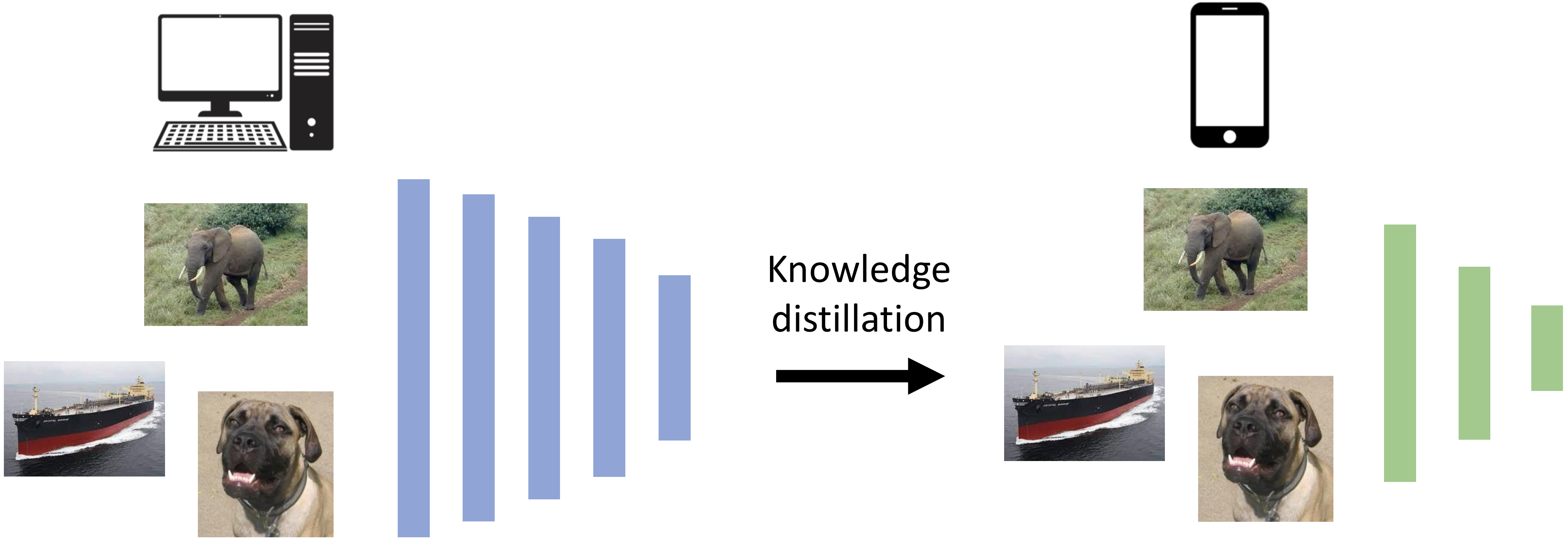
Define an objective function

Define a search space

Minimize the objective function over the space

Part 2: Knowledge Distillation

Motivation

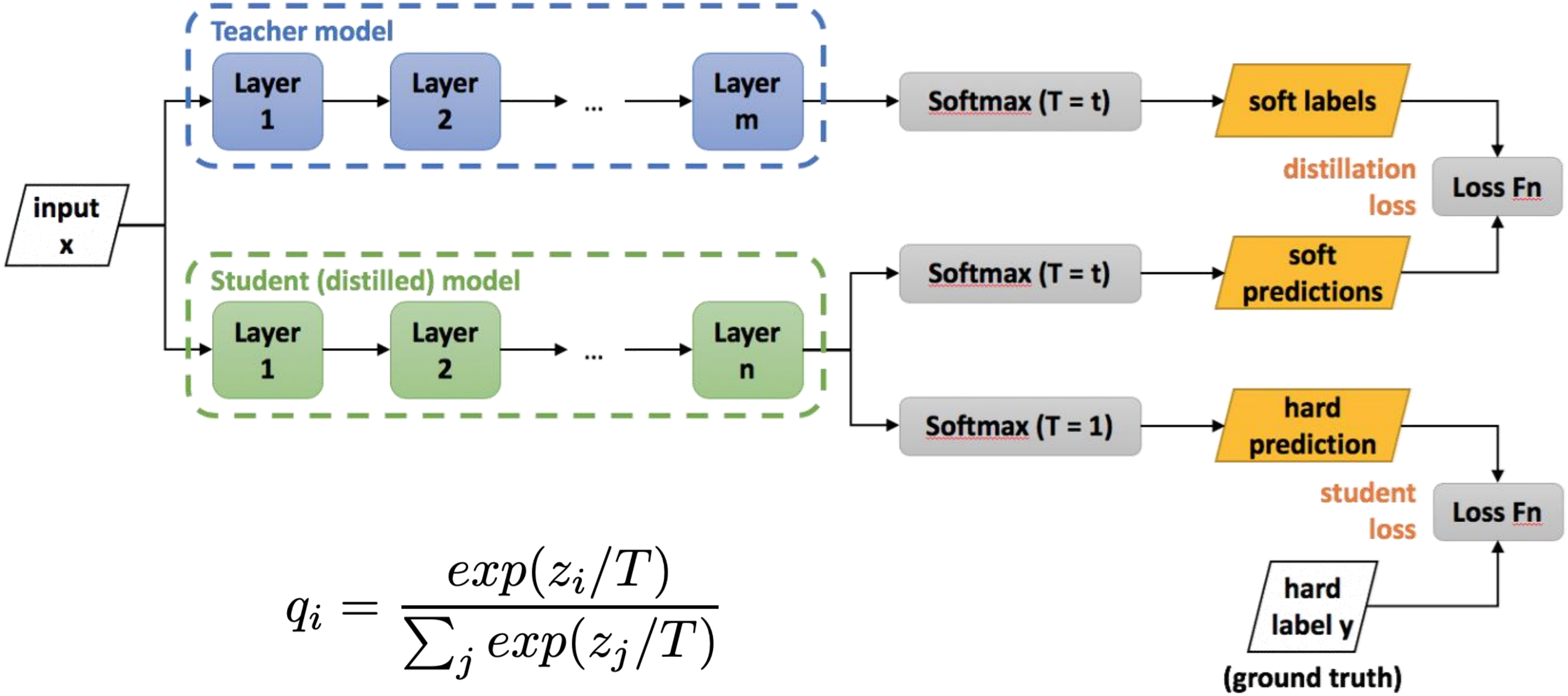


Accuracy = 93%

Accuracy = 90%

Objective: Compress large model into small/compact model with minimal performance loss

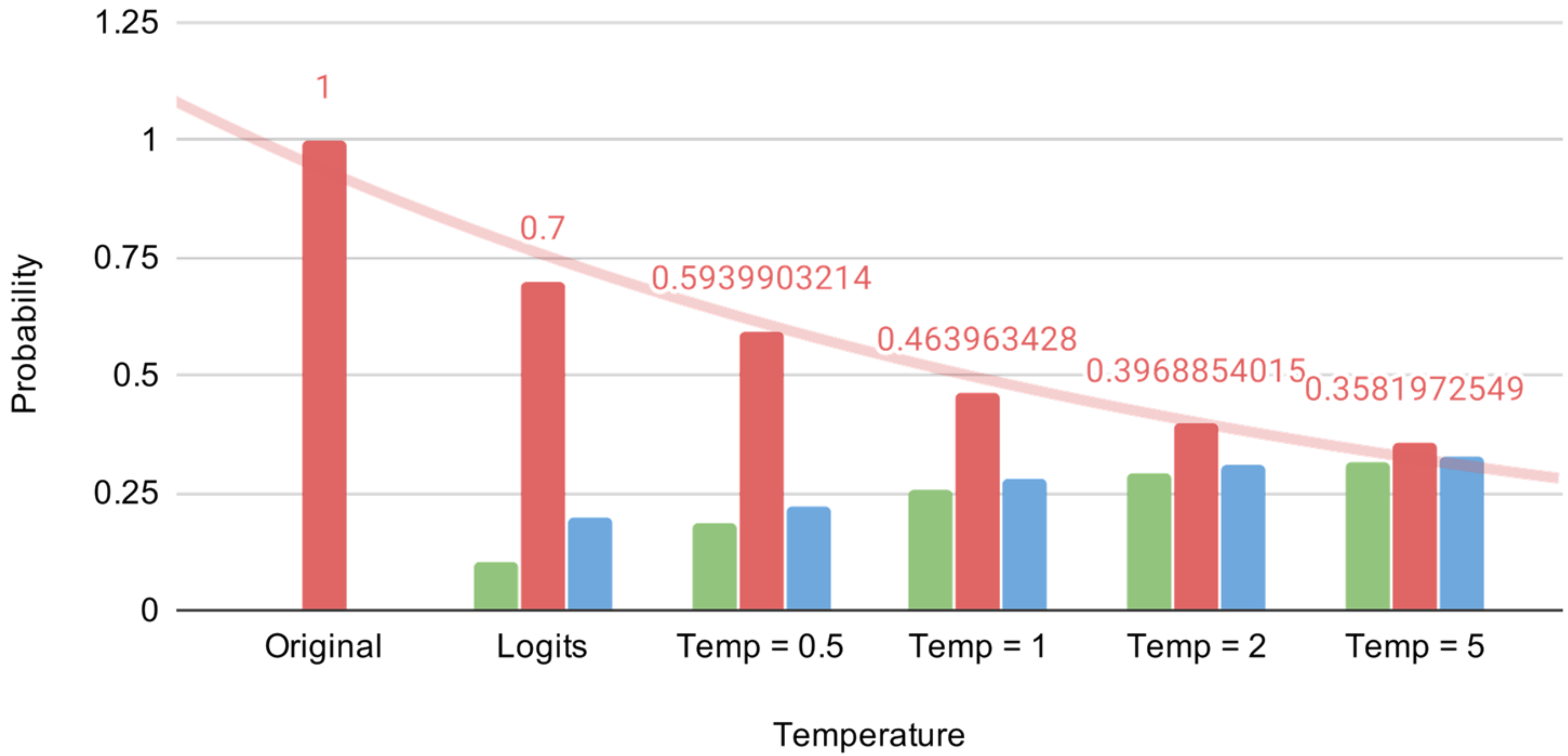
Distillation Training (Hinton et al)



$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Soft prediction (soft target)

Soft targets and Temperature



Soft targets

1. Generalize the model
2. Act as regularizers

Image credit:
Toward data science

Example using MNIST dataset: Teacher Network

```
class teacher_net(nn.Module):
    def __init__(self, dropout=0.5):
        super(teacher_net, self).__init__()
        self.linear_1 = nn.Linear(784, 1200)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=dropout)
        self.linear_2 = nn.Linear(1200, 1200)
        self.dropout = nn.Dropout(p=dropout)
        self.linear_3 = nn.Linear(1200, 10)

    def forward(self, input):
        scores = self.linear_1(input)
        scores = self.relu(scores)
        scores = self.linear_2(scores)
        scores = self.relu(scores)
        scores = self.dropout(scores)
        scores = self.linear_3(scores)
        return scores
```

Define network architecture

Define forward operation

Example using MNIST dataset: Teacher Network

```
teacher_model = teacher_net().to(device)
loss_fn = nn.CrossEntropyLoss()
optimizer = Adam(teacher_model.parameters(), lr=lr)
epoch = 20

for epoch in range(epochs):

    for features, labels in tqdm(train_loader):

        scores = teacher_model(features)
        loss = loss_fn(scores, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        train_loss.append(loss.item())
```

Define loss and optimizer

Train the network up until target performance

Example using MNIST dataset: Student Network

```
class student_net(nn.Module):
    def __init__(self):
        super(student_net, self).__init__()
        self.linear_1 = nn.Linear(784, 50)
        self.relu = nn.ReLU()
        self.linear_2 = nn.Linear(50, 10)

    def forward(self, input):
        scores = self.linear_1(input)
        scores = self.relu(scores)
        scores = self.linear_2(scores)
        return scores
```

Define network architecture
(More compact than teacher)

Define forward operation

Example using MNIST dataset: Distillation Training (soft target only)

```
softmax_op = nn.Softmax(dim=1)
mseloss_fn = nn.MSELoss()

def my_loss(scores, targets, T):
    soft_pred = softmax_op(scores / T)
    soft_targets = softmax_op(targets / T)
    loss = mseloss_fn(soft_pred, soft_targets)
    return loss
```

Define soft target loss function with temperature

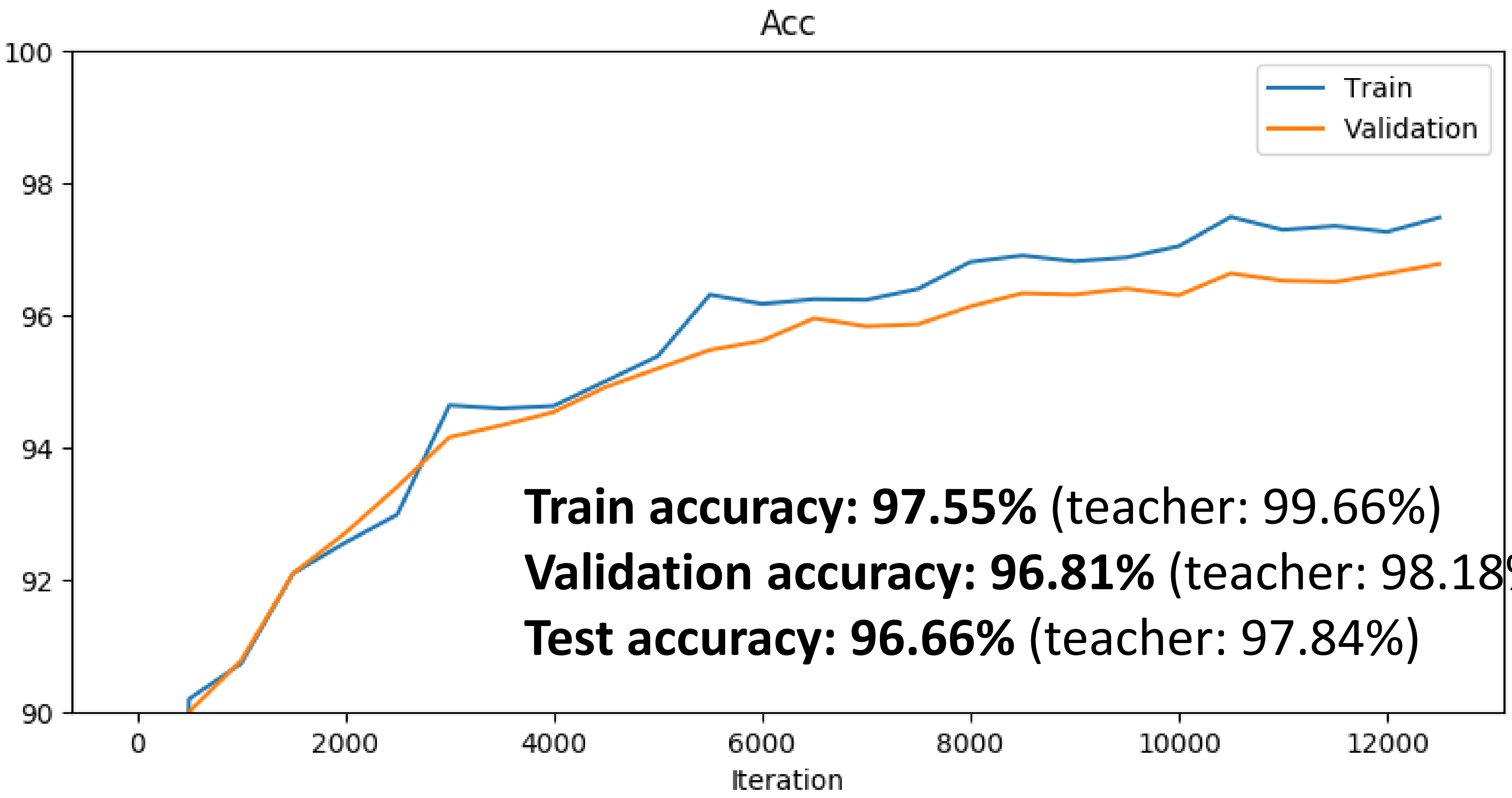
```
student_model = student_net().to(device)

lr = 5e-3
epochs = 5
temp = 5

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

for epoch in range(epochs):

    for features, labels in tqdm(train_loader):

        scores = student_model(features)
        targets = teacher_model(features)
        loss = my_loss(scores, targets, T = temp)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Define hyperparameters

Define optimizer

Train the student network using soft target from teacher network

Example using MNIST dataset: Distillation Training (soft target only)

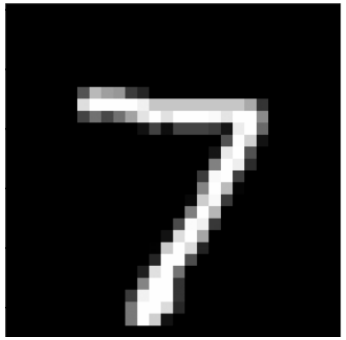


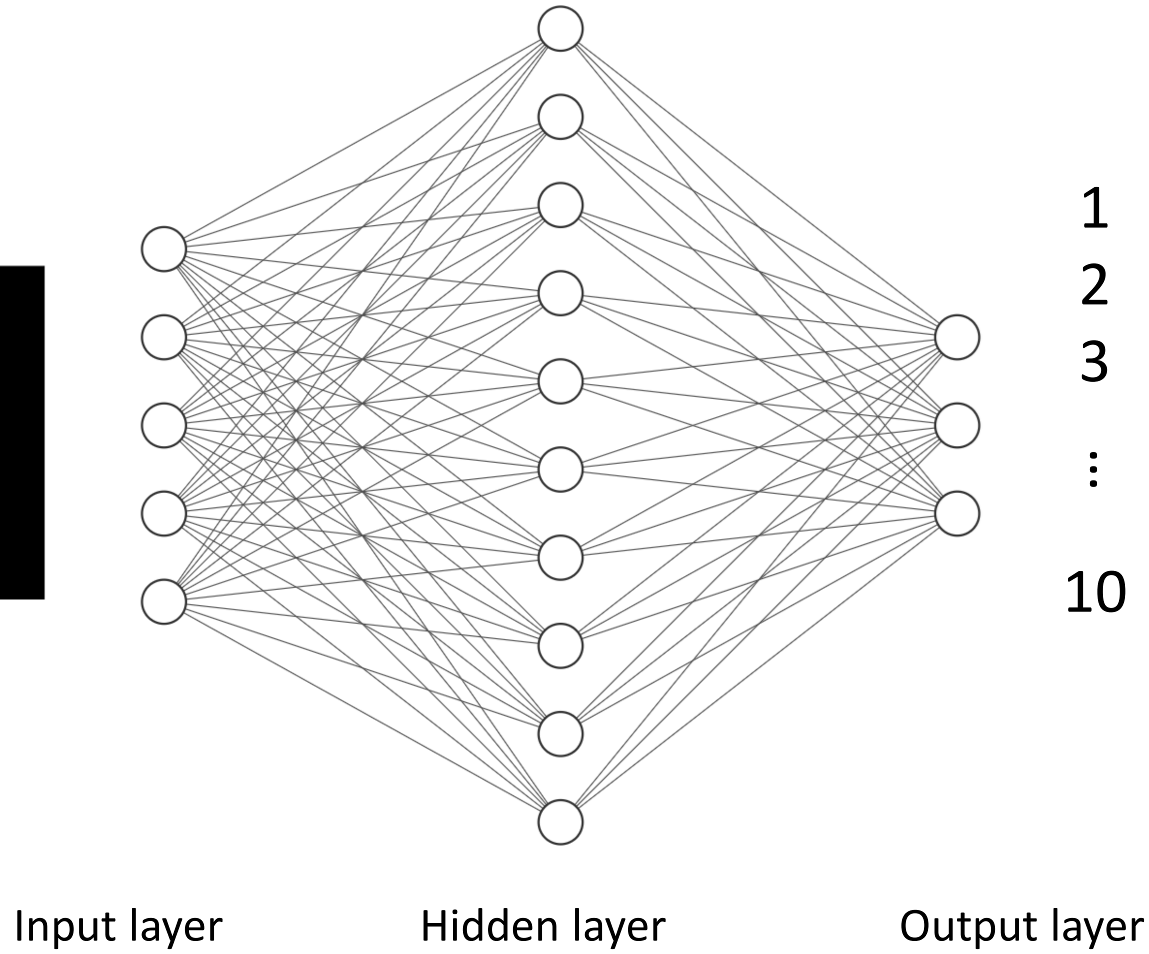
Train accuracy: 97.55% (teacher: 99.66%)
Validation accuracy: 96.81% (teacher: 98.18%)
Test accuracy: 96.66% (teacher: 97.84%)

Lab Assignment:

Optimize neural network using hyperopt (MNIST dataset)

Revisiting MNIST Classification from Lab 2





Use HyperOpt Python package (or your preferred package) to meta-optimize your deep neural network

You are free to choose
- Hyperparameter space
- Search method

Evaluation – Upload:
1. Original accuracy w.o HP tuning
2. New accuracy w HP tuning