HW2

April 1, 2025

1 Attempt to forecast the price of MSFT by analyzing the prices of multiple stocks, including MSFT, over several consecutive days leading up to the target day.

N.B. Different setup from HW1

[12]:
```
from torch.utils.data import DataLoader,Dataset

class StockDataset(Dataset):
    def __init__(self,X,Y,days):
        self.X = X
        self.Y = Y.reshape(-1)
        self.days = days # days ahead for prediction

    def __len__(self):
        return (len(self.Y)-self.days)

    def __getitem__(self,index):
        x=self.X[:,index:index+self.days]
        y=self.Y[index+self.days]
        return x,y
```

[13]:
```
# !pip install pandas
# !pip install yfinance
import pandas as pd
import yfinance as yf
import numpy as np
from numpy import exp, sum, log, log10


def get_price(tick,start='2020-01-01',end=None):
    return yf.Ticker(tick).history(start=start,end=end)['Close']

def get_prices(tickers,start='2020-01-01',end=None):
    df=pd.DataFrame()
    for s in tickers:
        df[s]=get_price(s,start,end)
    return df
```

```
feature_stocks=['tsla','meta','nvda','amzn','nflx','gbtc','gdx','intc','dal','c','goog','aapl'
predict_stock='msft'

# getting data
start_date='2020-01-01'

allX=get_prices(feature_stocks,start=start_date)
ally=get_prices([predict_stock],start=start_date)
```

[14]:
```
import torch.utils.data as data
import torch

stockData = StockDataset(allX.to_numpy().transpose().astype(np.float32),ally.
 ↪to_numpy().astype(np.float32),days=5)
train_set_size = int(len(stockData)*0.7)
valid_set_size = int(len(stockData)*0.2)
test_set_size = len(stockData)-train_set_size-valid_set_size

train_set, valid_set, test_set = data.
 ↪random_split(stockData,[train_set_size,valid_set_size,test_set_size],\
                                                generator=torch.Generator().
 ↪manual_seed(42))

batch_size = train_set_size # use entire dataset as batch
train_dataloader = DataLoader(train_set,batch_size=batch_size,shuffle=True)  #␣
 ↪input:(20,5), label:1
valid_dataloader = DataLoader(valid_set,batch_size=batch_size,shuffle=False)
test_dataloader = DataLoader(test_set,batch_size=batch_size,shuffle=False)
```

2 1. Build a simple MLP to forecast MSFT price using PyTorch Lightning.

You have total freedom of your MLP. But your MLP should take the last five day ($5 \times 20 = 100$) prices as input and you have to add dropout into your network.

2.1 1a. Create a subclass of pytorch_lightning.LightningModule. It should include ___init___, training_step, validation_step, configure_optimizers in the class. (6 points)

[15]:
```
import torch
from torch import nn
import pytorch_lightning as pl
from torch.utils.data import Dataset, DataLoader, random_split
import numpy as np
import pandas as pd
```

```
import yfinance as yf


window_size = 5  # Using 5 days of historical data



# Create sequences of 5 days × 20 features = 100 inputs
def create_sequences(features, target, window_size):
    X, y = [], []
    for i in range(window_size, len(features)):
        X.append(features.iloc[i-window_size:i].values.flatten())
        y.append(target.iloc[i])
    return torch.FloatTensor(X), torch.FloatTensor(y)

X, y = create_sequences(allX, ally, window_size)

# Split dataset
dataset = torch.utils.data.TensorDataset(X, y)
train_size = int(0.8 * len(dataset))
val_size = len(dataset) - train_size
train_set, val_set = random_split(dataset, [train_size, val_size])

# PyTorch Lightning Module
class StockPredictor(pl.LightningModule):
    def __init__(self, input_size=100, dropout=0.2):
        super().__init__()
        self.mlp = nn.Sequential(
            nn.Linear(input_size, 256),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(64, 1)
        )
        self.loss_fn = nn.MSELoss()

    def forward(self, x):
        return self.mlp(x).squeeze()

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
```

```
        loss = self.loss_fn(y_pred, y)
        self.log('train_loss', loss, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('val_loss', loss, prog_bar=True)
        return loss

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=1e-3)

# Initialize model and trainer
model = StockPredictor(input_size=window_size*len(feature_stocks))
trainer = pl.Trainer(max_epochs=50, accelerator='auto', deterministic=True)

# Create data loaders
train_loader = DataLoader(train_set, batch_size=32, shuffle=True)
val_loader = DataLoader(val_set, batch_size=32)

# Train the model
trainer.fit(model, train_loader, val_loader)
```

```
/tmp/ipykernel_1218/1162869641.py:20: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  return torch.FloatTensor(X), torch.FloatTensor(y)
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | mlp     | Sequential | 67.1 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
67.1 K    Trainable params
0         Non-trainable params
67.1 K    Total params
0.268     Total estimated model params size (MB)
12        Modules in train mode
0         Modules in eval mode
```

```
Sanity Checking: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([32, 1])) that is different to the input size (torch.Size([32])).
This will likely lead to incorrect results due to broadcasting. Please ensure
they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(33) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.

Training: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([27, 1])) that is different to the input size (torch.Size([27])).
This will likely lead to incorrect results due to broadcasting. Please ensure
they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Validation: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([7, 1])) that is different to the input size (torch.Size([7])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
`Trainer.fit` stopped: `max_epochs=50` reached.
```

2.2 1b. Create a subclass of pytorch_lightning.LightningDataModule. It should include ___init___, train_dataloader, and val_dataloader in the class. (4 points)

[16]:
```python
import pytorch_lightning as pl
from torch.utils.data import DataLoader, TensorDataset, random_split
import torch
import numpy as np
import pandas as pd
import yfinance as yf

class StockDataModule(pl.LightningDataModule):
    def __init__(self, feature_stocks, predict_stock='msft',
                 window_size=5, batch_size=32, start_date='2020-01-01'):
        super().__init__()
        self.feature_stocks = feature_stocks
        self.predict_stock = predict_stock
        self.window_size = window_size
        self.batch_size = batch_size
        self.start_date = start_date

        # Will be initialized in setup
        self.train_dataset = None
        self.val_dataset = None

    def setup(self, stage=None):
        # Download and align data
        all_data = yf.download(
            self.feature_stocks + [self.predict_stock],
```

```
            start=self.start_date
        )['Close'].dropna()

        features = all_data[self.feature_stocks]
        target = all_data[self.predict_stock]

        # Create sequences (5 days × 20 stocks = 100 features)
        X, y = [], []
        for i in range(self.window_size, len(features)):
            X.append(features.iloc[i-self.window_size:i].values.flatten())
            y.append(target.iloc[i])

        X = torch.tensor(np.array(X), dtype=torch.float32)
        y = torch.tensor(np.array(y), dtype=torch.float32)

        # Create dataset and split
        dataset = TensorDataset(X, y)
        train_size = int(0.8 * len(dataset))
        val_size = len(dataset) - train_size
        self.train_dataset, self.val_dataset = random_split(dataset,␣
↪[train_size, val_size])

    def train_dataloader(self):
        return DataLoader(
            self.train_dataset,
            batch_size=self.batch_size,
            shuffle=True,
            num_workers=4
        )

    def val_dataloader(self):
        return DataLoader(
            self.val_dataset,
            batch_size=self.batch_size,
            num_workers=4
        )
```

2.3 1c. Complete the rest of the code and train the model with 70% of the data. You should set aside 15% of the data each for validation and testing. Show the training and validation MSE (5 points)

[17]:
```
import torch
from torch import nn
import pytorch_lightning as pl
from torch.utils.data import Dataset, DataLoader, TensorDataset, random_split
import numpy as np
import pandas as pd
```

```
import yfinance as yf
from sklearn.preprocessing import StandardScaler

# 1. Enhanced Data Module with 70-15-15 split
class StockDataModule(pl.LightningDataModule):
    def __init__(self, feature_stocks, predict_stock='msft',
                 window_size=5, batch_size=32, start_date='2020-01-01',␣
 ↪end_date = None):
        super().__init__()
        self.feature_stocks = [t.upper() for t in feature_stocks]
        self.predict_stock = predict_stock.upper()
        self.window_size = window_size
        self.batch_size = batch_size
        self.start_date = start_date
        self.end_date = end_date
        self.scaler = StandardScaler()

        # Initialize datasets
        self.train_dataset = None
        self.val_dataset = None
        self.test_dataset = None

    def setup(self, stage=None):
        # Get data with time-series split
        all_data = yf.download(self.feature_stocks + [self.predict_stock],␣
 ↪start=self.start_date,
                               end=self.end_date)['Close'].dropna()

        # 1. Normalize features
        feature_values = all_data[self.feature_stocks].values
        scaled_features = self.scaler.fit_transform(feature_values)

        # 2. Time-series split (NO RANDOMIZATION)
        total_samples = len(scaled_features) - self.window_size
        train_end = int(0.7 * total_samples)
        val_end = train_end + int(0.15 * total_samples)

        # Create sequences
        X, y = [], []
        for i in range(self.window_size, len(scaled_features)):
            X.append(scaled_features[i-self.window_size:i].flatten())
            y.append(all_data[self.predict_stock].iloc[i])  # Don't scale target

        # 3. Split without shuffling
        X_train, y_train = X[:train_end], y[:train_end]
        X_val, y_val = X[train_end:val_end], y[train_end:val_end]
        X_test, y_test = X[val_end:], y[val_end:]
```

```
        # Convert to tensors
        self.train_dataset = TensorDataset(torch.FloatTensor(X_train), torch.
 ↪FloatTensor(y_train))
        self.val_dataset = TensorDataset(torch.FloatTensor(X_val), torch.
 ↪FloatTensor(y_val))
        self.test_dataset = TensorDataset(torch.FloatTensor(X_test), torch.
 ↪FloatTensor(y_test))

    def train_dataloader(self):
        return DataLoader(
            self.train_dataset,
            batch_size=self.batch_size,
            shuffle=True,
            num_workers=4
        )

    def val_dataloader(self):
        return DataLoader(
            self.val_dataset,
            batch_size=self.batch_size,
            num_workers=4
        )

    def test_dataloader(self):
        return DataLoader(
            self.test_dataset,
            batch_size=self.batch_size,
            num_workers=4
        )

# 2. Model Module with Metrics Tracking
class StockPredictor(pl.LightningModule):
    def __init__(self, input_size=100, dropout=0.2):
        super().__init__()
        self.mlp = nn.Sequential(
            nn.Linear(input_size, 256),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(64, 1)
        )
```

```
        self.loss_fn = nn.MSELoss()

    def forward(self, x):
        return self.mlp(x).squeeze()

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('train_mse', loss, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('val_mse', loss, prog_bar=True)
        return loss

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=1e-5)

# 3. Training and Evaluation
if __name__ == "__main__":
    # Configuration
    feature_stocks = [
        'tsla', 'meta', 'nvda', 'amzn', 'nflx', 'gbtc', 'gdx', 'intc', 'dal',
        'c', 'goog', 'aapl', 'msft', 'ibm', 'dis', 'bac', 'gs', 'jpm', 'xom',␣
 ↪'cvx'
    ]

    # Initialize components
    datamodule = StockDataModule(
        feature_stocks=feature_stocks,
        predict_stock='msft',
        window_size=5,
        batch_size=32
    )

    model = StockPredictor(input_size=5*len(feature_stocks))

    trainer = pl.Trainer(
        max_epochs=100,
        accelerator='auto',
        deterministic=True,
        enable_progress_bar=True
    )
```

```python
    # Train the model
    trainer.fit(model, datamodule=datamodule)

    # Evaluate final performance
    def print_final_metrics(trainer):
        print("\nFinal Metrics:")
        print(f"Training MSE: {trainer.callback_metrics['train_mse'].item():.
↪4f}")
        print(f"Validation MSE: {trainer.callback_metrics['val_mse'].item():.
↪4f}")

    print_final_metrics(trainer)
```

```
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-------------------------------------------
0 | mlp     | Sequential | 67.1 K | train
1 | loss_fn | MSELoss    | 0      | train
-------------------------------------------
67.1 K    Trainable params
0         Non-trainable params
67.1 K    Total params
0.268     Total estimated model params size (MB)
12        Modules in train mode
0         Modules in eval mode

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(29) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.

Training: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
`Trainer.fit` stopped: `max_epochs=100` reached.


Final Metrics:
Training MSE: 22055.2852
Validation MSE: 128602.5078
```

3 2. Construct a 1-D CNN to forecast MSFT stock price. You are free to use any design, but your network must consist of at least one convolutional layer and one dropout layer. You can also extend the duration leading up to the target day by modifying the "days" argument in the StockDataset. But "days" should not be larger than 32. (10 points)

[18]:
```python
import torch
from torch import nn
import pytorch_lightning as pl
from torch.utils.data import Dataset, DataLoader, TensorDataset
import numpy as np
import pandas as pd
import yfinance as yf
from sklearn.preprocessing import StandardScaler

# Custom Dataset with adjustable window size
class StockDataset(Dataset):
    def __init__(self, features, target, days=30):
        self.days = days
        self.X, self.y = self.create_sequences(features, target)

    def create_sequences(self, features, target):
        X, y = [], []
        for i in range(self.days, len(features)):
            seq_features = features[i-self.days:i]
            seq_target = target[i]
            X.append(seq_features)
            y.append(seq_target)
        return torch.FloatTensor(np.array(X)), torch.FloatTensor(np.array(y))

    def __len__(self):
        return len(self.X)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]

class CNNStockPredictor(pl.LightningModule):
```

```python
    def __init__(self, input_channels=20, days=30, dropout=0.3, lr=1e-3):
        super().__init__()
        self.save_hyperparameters()

        # 1D CNN architecture
        self.cnn = nn.Sequential(
            nn.Conv1d(input_channels, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2),

            nn.Conv1d(64, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Dropout(dropout),

            nn.Flatten(),

            nn.Linear(32 * (days//2), 128),
            nn.ReLU(),
            nn.Dropout(dropout),

            nn.Linear(128, 1)
        )

        self.loss_fn = nn.MSELoss()

    def forward(self, x):
        # Input shape: [batch, features, days]
        x = x.permute(0, 2, 1)  # [batch, days, features] -> [batch, features,␣
↪days]
        return self.cnn(x).squeeze()

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('train_mse', loss, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('val_mse', loss, prog_bar=True)
        return loss

    def test_step(self, batch, batch_idx):
        x, y = batch
```

```
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('test_mse', loss)
        return loss

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.hparams.lr)

class StockDataModule(pl.LightningDataModule):
    def __init__(self, feature_stocks, predict_stock='MSFT',
                 days=30, batch_size=32, start_date='2020-01-01'):
        super().__init__()
        self.feature_stocks = [t.upper() for t in feature_stocks]
        self.predict_stock = predict_stock.upper()
        self.days = days
        self.batch_size = batch_size
        self.start_date = start_date
        self.scaler = StandardScaler()

    def setup(self, stage=None):
        # Download and process data
        data = yf.download(
            self.feature_stocks + [self.predict_stock],
            start=self.start_date
        )['Close'].dropna()

        # Handle symbol changes
        if 'META' not in data.columns and 'FB' in data.columns:
            data = data.rename(columns={'FB': 'META'})

        features = data[self.feature_stocks].values
        target = data[self.predict_stock].values

        # Normalize features
        features = self.scaler.fit_transform(features)

        # Create sequences
        dataset = StockDataset(features, target, self.days)

        # Time-series split (70-15-15)
        n = len(dataset)
        self.train_dataset = TensorDataset(dataset.X[:int(0.7*n)], dataset.y[:
 ↪int(0.7*n)])
        self.val_dataset = TensorDataset(dataset.X[int(0.7*n):int(0.85*n)],
                                         dataset.y[int(0.7*n):int(0.85*n)])
        self.test_dataset = TensorDataset(dataset.X[int(0.85*n):], dataset.
 ↪y[int(0.85*n):])
```

```
    def train_dataloader(self):
        return DataLoader(self.train_dataset, batch_size=self.batch_size,␣
 ↪shuffle=False)

    def val_dataloader(self):
        return DataLoader(self.val_dataset, batch_size=self.batch_size)

    def test_dataloader(self):
        return DataLoader(self.test_dataset, batch_size=self.batch_size)

# Training configuration
if __name__ == "__main__":
    feature_stocks = ['TSLA', 'META', 'NVDA', 'AMZN', 'NFLX', 'INTC', 'DAL',
                      'C', 'GOOG', 'AAPL', 'MSFT', 'IBM', 'DIS', 'BAC', 'GS',
                      'JPM', 'XOM', 'CVX', 'SPY', 'QQQ']

    datamodule = StockDataModule(
        feature_stocks=feature_stocks,
        predict_stock='MSFT',
        days=30,  # Using 30-day window (within 32 limit)
        batch_size=64
    )

    model = CNNStockPredictor(
        input_channels=len(feature_stocks),
        days=30,
        dropout=0.3,
        lr=1e-4
    )

    trainer = pl.Trainer(
        max_epochs=100,
        accelerator='auto',
        callbacks=[
            pl.callbacks.EarlyStopping(monitor='val_mse', patience=15),
            pl.callbacks.ModelCheckpoint(monitor='val_mse')
        ],
        enable_progress_bar=True,
        precision='16-mixed'
    )

    trainer.fit(model, datamodule=datamodule)
    trainer.test(datamodule=datamodule)
```

```
Using 16bit Automatic Mixed Precision (AMP)
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
```

```
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 71.8 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
71.8 K    Trainable params
0         Non-trainable params
71.8 K    Total params
0.287     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.

Training: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
`Trainer.fit` stopped: `max_epochs=100` reached.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/checkpoint_connector.py:149:
`.test(ckpt_path=None)` was called without a model. The best model of the
previous `fit` call will be used. You can pass `.test(ckpt_path='best')` to use
the best model or `.test(ckpt_path='last')` to use the last model. If you pass a
value, this warning will be silenced.
[**********100%**********]  20 of 20 completed
Restoring states from the checkpoint path at /home/barrytan/MSFT-
Prediction/HW2/lightning_logs/version_47/checkpoints/epoch=99-step=1500.ckpt
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Loaded model weights from the checkpoint at /home/barrytan/MSFT-
Prediction/HW2/lightning_logs/version_47/checkpoints/epoch=99-step=1500.ckpt
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'test_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
Testing: |          | 0/? [00:00<?, ?it/s]
```

Test metric	DataLoader 0
test_mse	4619.501953125

4 3. Please try to enhance the performance of the previously created MLP or CNN by applying hyperparameter tuning. You can use tools such as W&B hyperparameter sweep, SMAP, Optuna, or similar packages to achieve this. You need to optimize at least two parameters, with the dropout rate being one of them. (5 points)

[19]:
```python
import optuna
from optuna.integration import PyTorchLightningPruningCallback
from optuna.pruners import MedianPruner
from pytorch_lightning.callbacks import EarlyStopping
import torch
from torch import nn
import pytorch_lightning as pl

# 1. Modified CNN Model with Tunable Parameters
class TunedCNNStockPredictor(pl.LightningModule):
    def __init__(self, input_channels=20, days=30,
```

```
                dropout=0.3, lr=1e-3, num_filters=64):
        super().__init__()
        self.save_hyperparameters()

        self.cnn = nn.Sequential(
            nn.Conv1d(input_channels, num_filters, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2),

            nn.Conv1d(num_filters, num_filters//2, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Dropout(dropout),

            nn.Flatten(),

            nn.Linear((num_filters//2) * (days//2), 128),
            nn.ReLU(),
            nn.Dropout(dropout),

            nn.Linear(128, 1)
        )

        self.loss_fn = nn.MSELoss()

    def forward(self, x):
        x = x.permute(0, 2, 1)
        return self.cnn(x).squeeze()

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('train_mse', loss, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('val_mse', loss, prog_bar=True)
        return loss

    def test_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self(x)
        loss = self.loss_fn(y_pred, y)
        self.log('test_mse', loss, prog_bar=True)
```

```
        return loss

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.hparams.lr)

# 2. Optuna Optimization Setup
def objective(trial):
    params = {
        'dropout': trial.suggest_float('dropout', 0.1, 0.4),
        'lr': trial.suggest_float('lr', 1e-4, 1e-3, log=True),
        'num_filters': trial.suggest_categorical('num_filters', [32, 64, 128]) ␣
 ↪# Changed to num_filters
    }

    model = TunedCNNStockPredictor(
        input_channels=len(feature_stocks),
        days=30,
        **params
    )

    # Simplified trainer without problematic callbacks
    trainer = pl.Trainer(
        max_epochs=50,
        accelerator='auto',
        enable_progress_bar=False,
        callbacks=[EarlyStopping(monitor='val_mse', patience=5)]
    )

    trainer.fit(model, datamodule=datamodule)

    # Manual pruning logic
    val_mse = trainer.callback_metrics["val_mse"].item()
    trial.report(val_mse, step=trainer.current_epoch)

    if trial.should_prune():
        raise optuna.TrialPruned()

    return val_mse

# 3. Run Hyperparameter Search
if __name__ == "__main__":
    # Initialize study with median pruning
    study = optuna.create_study(
        direction='minimize',
        pruner=MedianPruner(n_startup_trials=5, n_warmup_steps=10)
    )
```

```python
    # Run optimization
    study.optimize(objective, n_trials=20, timeout=3600)

    # Output results
    print("\nBest trial:")
    trial = study.best_trial
    print(f"Validation MSE: {trial.value:.4f}")
    print("Optimized parameters:")
    for key, value in trial.params.items():
        print(f"  {key}: {value}")

    # Final training with best parameters
    best_model = TunedCNNStockPredictor(
        input_channels=len(feature_stocks),
        days=30,
        **trial.params
    )

    final_trainer = pl.Trainer(
        max_epochs=100,
        accelerator='auto',
        callbacks=[
            EarlyStopping(monitor='val_mse', patience=15),
            pl.callbacks.ModelCheckpoint(monitor='val_mse')
        ]
    )

    final_trainer.fit(best_model, datamodule=datamodule)
    final_trainer.test(datamodule=datamodule)
```

```
[I 2025-04-01 17:13:15,079] A new study created in memory with name: no-
name-003eb64c-feb6-4f6e-b397-039c75dfb2a7
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-------------------------------------------
0 | cnn     | Sequential | 34.5 K | train
1 | loss_fn | MSELoss    | 0      | train
-------------------------------------------
34.5 K    Trainable params
0         Non-trainable params
34.5 K    Total params
```

```
0.138     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:17,916] Trial 0 finished with value: 20072.537109375 and
parameters: {'dropout': 0.1763860725518809, 'lr': 0.0007115450599976062,
'num_filters': 32}. Best is trial 0 with value: 20072.537109375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
```

```
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:20,620] Trial 1 finished with value: 19521.658203125 and
parameters: {'dropout': 0.18462152698220702, 'lr': 0.0003111966996560656,
'num_filters': 128}. Best is trial 1 with value: 19521.658203125.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 34.5 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
34.5 K    Trainable params
0         Non-trainable params
34.5 K    Total params
0.138     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
```

```
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:23,731] Trial 2 finished with value: 12771.8701171875 and
parameters: {'dropout': 0.21229509335943134, 'lr': 0.0008665387431284597,
'num_filters': 32}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
```

```
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:26,788] Trial 3 finished with value: 14644.9140625 and
parameters: {'dropout': 0.1658920605233401, 'lr': 0.000257788653400221,
'num_filters': 128}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 34.5 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
34.5 K    Trainable params
0         Non-trainable params
34.5 K    Total params
0.138     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
```

```
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
`Trainer.fit` stopped: `max_epochs=50` reached.
[I 2025-04-01 17:13:30,860] Trial 4 finished with value: 72284.8125 and
parameters: {'dropout': 0.23589072257333973, 'lr': 0.0001268963859098027,
'num_filters': 32}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
[I 2025-04-01 17:13:34,749] Trial 5 finished with value: 15821.111328125 and
parameters: {'dropout': 0.2279891104288543, 'lr': 0.0003218032809467701,
'num_filters': 128}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 34.5 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
34.5 K    Trainable params
0         Non-trainable params
34.5 K    Total params
0.138     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
`Trainer.fit` stopped: `max_epochs=50` reached.
[I 2025-04-01 17:13:39,733] Trial 6 finished with value: 32548.904296875 and
parameters: {'dropout': 0.1140471760960185, 'lr': 0.00013331351721144644,
'num_filters': 32}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
```

```
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 71.8 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
71.8 K    Trainable params
0         Non-trainable params
71.8 K    Total params
0.287     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:42,202] Trial 7 finished with value: 16382.23828125 and
parameters: {'dropout': 0.38155917939826356, 'lr': 0.0009439363493505601,
'num_filters': 64}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
```

```
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 71.8 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
71.8 K    Trainable params
0         Non-trainable params
71.8 K    Total params
0.287     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
`Trainer.fit` stopped: `max_epochs=50` reached.
[I 2025-04-01 17:13:47,202] Trial 8 finished with value: 36231.69921875 and
parameters: {'dropout': 0.2053861535372876, 'lr': 0.00011505252915913126,
'num_filters': 64}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
```

```
0 | cnn     | Sequential | 71.8 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
71.8 K    Trainable params
0         Non-trainable params
71.8 K    Total params
0.287     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:49,701] Trial 9 finished with value: 15709.658203125 and
parameters: {'dropout': 0.17958169457885897, 'lr': 0.0009239217875602605,
'num_filters': 64}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 34.5 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
34.5 K    Trainable params
0         Non-trainable params
```

```
34.5 K    Total params
0.138     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:52,543] Trial 10 pruned.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
```

```
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:13:55,716] Trial 11 finished with value: 16887.115234375 and
parameters: {'dropout': 0.10861351842498115, 'lr': 0.00024249501720367592,
'num_filters': 128}. Best is trial 2 with value: 12771.8701171875.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
```

```
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
`Trainer.fit` stopped: `max_epochs=50` reached.
[I 2025-04-01 17:14:00,302] Trial 12 finished with value: 11565.52734375 and
parameters: {'dropout': 0.3061963320837304, 'lr': 0.00020168694191495125,
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 34.5 K | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
34.5 K    Trainable params
0         Non-trainable params
34.5 K    Total params
0.138     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
```

```
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
`Trainer.fit` stopped: `max_epochs=50` reached.
[I 2025-04-01 17:14:04,574] Trial 13 pruned.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
```

```
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:14:07,348] Trial 14 finished with value: 11792.76171875 and
parameters: {'dropout': 0.2905496071269708, 'lr': 0.0004474896488552141,
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:14:09,192] Trial 15 finished with value: 13658.51953125 and
parameters: {'dropout': 0.2906407623574994, 'lr': 0.0004566508894961526,
```

```
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:14:12,055] Trial 16 finished with value: 13711.0546875 and
parameters: {'dropout': 0.35613773831333184, 'lr': 0.0004584604378292357,
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
```

```
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:14:15,222] Trial 17 finished with value: 16121.7490234375 and
parameters: {'dropout': 0.3303074338916467, 'lr': 0.00016927607436587833,
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
```

```
---------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
---------------------------------------------
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:14:18,924] Trial 18 finished with value: 14370.8583984375 and
parameters: {'dropout': 0.2626587294910784, 'lr': 0.0005598423329231155,
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
You are using the plain ModelCheckpoint callback. Consider using
LitModelCheckpoint which with seamless uploading to Model registry.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
---------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
---------------------------------------------
155 K     Trainable params
```

```
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
[I 2025-04-01 17:14:21,361] Trial 19 finished with value: 13915.3642578125 and
parameters: {'dropout': 0.2651381177462964, 'lr': 0.00035114066239636286,
'num_filters': 128}. Best is trial 12 with value: 11565.52734375.
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs


Best trial:
Validation MSE: 11565.5273
Optimized parameters:
  dropout: 0.3061963320837304
  lr: 0.00020168694191495125
  num_filters: 128

[**********100%**********]  20 of 20 completed
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params | Mode
-----------------------------------------------
0 | cnn     | Sequential | 155 K  | train
1 | loss_fn | MSELoss    | 0      | train
-----------------------------------------------
```

```
155 K     Trainable params
0         Non-trainable params
155 K     Total params
0.622     Total estimated model params size (MB)
13        Modules in train mode
0         Modules in eval mode

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'val_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=15`
in the `DataLoader` to improve performance.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/loops/fit_loop.py:310: The number of training batches
(15) is smaller than the logging interval Trainer(log_every_n_steps=50). Set a
lower value for log_every_n_steps if you want to see logs for the training
epoch.

Training: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/torch/nn/modules/loss.py:610: UserWarning: Using a target size
(torch.Size([1])) that is different to the input size (torch.Size([])). This
will likely lead to incorrect results due to broadcasting. Please ensure they
have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
```

```
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
Validation: |          | 0/? [00:00<?, ?it/s]
`Trainer.fit` stopped: `max_epochs=100` reached.
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/checkpoint_connector.py:149:
`.test(ckpt_path=None)` was called without a model. The best model of the
previous `fit` call will be used. You can pass `.test(ckpt_path='best')` to use
the best model or `.test(ckpt_path='last')` to use the last model. If you pass a
value, this warning will be silenced.
[*********************100%***********************]  20 of 20 completed
Restoring states from the checkpoint path at /home/barrytan/MSFT-
Prediction/HW2/lightning_logs/version_68/checkpoints/epoch=89-step=1350.ckpt
```

```
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Loaded model weights from the checkpoint at /home/barrytan/MSFT-
Prediction/HW2/lightning_logs/version_68/checkpoints/epoch=89-step=1350.ckpt
/home/barrytan/miniconda3/envs/ANN/lib/python3.12/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:425: The
'test_dataloader' does not have many workers which may be a bottleneck. Consider
increasing the value of the `num_workers` argument` to `num_workers=15` in the
`DataLoader` to improve performance.
Testing: |          | 0/? [00:00<?, ?it/s]
```

Test metric	DataLoader 0
test_mse	2643.39794921875