First we have to download and install the pip package

```
!pip3 install -U ncps pytorch-lightning
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ncps
  Downloading ncps-0.0.7-py3-none-any.whl (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.8/44.8 KB 1.2 MB/s eta 0:00:00
Collecting pytorch-lightning
  Downloading pytorch_lightning-1.9.1-py3-none-any.whl (825 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 825.8/825.8 KB 13.8 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from ncps) (23.0)
Requirement already satisfied: future in /usr/local/lib/python3.8/dist-packages (from ncps) (0.16.0)
Requirement already satisfied: torch>=1.10.0 in /usr/local/lib/python3.8/dist-packages (from pytorch-lightning) (1.13.1+cu11
Requirement already satisfied: fsspec[http]>2021.06.0 in /usr/local/lib/python3.8/dist-packages (from pytorch-lightning) (20
Collecting torchmetrics>=0.7.0
  Downloading torchmetrics-0.11.1-py3-none-any.whl (517 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 517.2/517.2 KB 19.5 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.8/dist-packages (from pytorch-lightning) (4.64.1)
Requirement already satisfied: numpy>=1.17.2 in /usr/local/lib/python3.8/dist-packages (from pytorch-lightning) (1.21.6)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.8/dist-packages (from pytorch-lightning) (6.0)
Collecting lightning-utilities>=0.6.0.post0
  Downloading lightning_utilities-0.6.0.post0-py3-none-any.whl (18 kB)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.8/dist-packages (from pytorch-lightning) (
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.8/dist-packages (from fsspec[http]>2021.
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from fsspec[http]>2021.06.0->pytorch-ligh
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fs
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0,
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fss
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->fsspec[http]>202
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->fsspec[http]>
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->fsspec[http]>2021
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->fsspec[http]>2021.06.0
Installing collected packages: ncps, lightning-utilities, torchmetrics, pytorch-lightning
Successfully installed lightning-utilities-0.6.0.post0 ncps-0.0.7 pytorch-lightning-1.9.1 torchmetrics-0.11.1
```

```
import numpy as np
import torch.nn as nn
from ncps.wirings import AutoNCP
from ncps.torch import LTC
import pytorch_lightning as pl
import torch
import torch.utils.data as data
```

For the training we will use Pytorch-Lightning, thus we have to define our learner module.

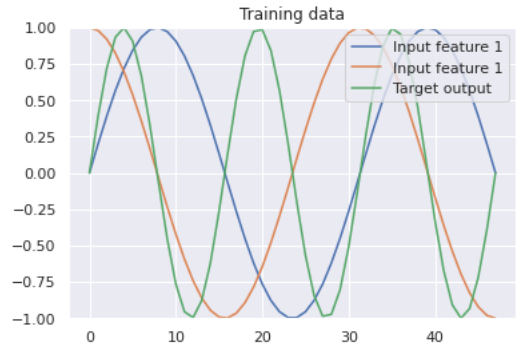
```
# LightningModule for training a RNNSequence module
class SequenceLearner(pl.LightningModule):
    def __init__(self, model, lr=0.005):
        super().__init__()
        self.model = model
        self.lr = lr

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat, _ = self.model.forward(x)
        y_hat = y_hat.view_as(y)
        loss = nn.MSELoss()(y_hat, y)
        self.log("train_loss", loss, prog_bar=True)
        return {"loss": loss}

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_hat, _ = self.model.forward(x)
        y_hat = y_hat.view_as(y)
        loss = nn.MSELoss()(y_hat, y)

        self.log("val_loss", loss, prog_bar=True)
        return loss

    def test_step(self, batch, batch_idx):
```

```
        # Here we just reuse the validation_step for testing
        return self.validation_step(batch, batch_idx)

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

Next we define some toy dataset and create the corresponding DataLoaders

```
import matplotlib.pyplot as plt
import seaborn as sns

in_features = 2
out_features = 1
N = 48  # Length of the time-series
# Input feature is a sine and a cosine wave
data_x = np.stack(
    [np.sin(np.linspace(0, 3 * np.pi, N)), np.cos(np.linspace(0, 3 * np.pi, N))], axis=1
)
data_x = np.expand_dims(data_x, axis=0).astype(np.float32)  # Add batch dimension
# Target output is a sine with double the frequency of the input signal
data_y = np.sin(np.linspace(0, 6 * np.pi, N)).reshape([1, N, 1]).astype(np.float32)
data_x = torch.Tensor(data_x)
data_y = torch.Tensor(data_y)
print("data_x.size: ", str(data_x.size()))
print("data_y.size: ", str(data_y.size()))
dataloader = data.DataLoader(
    data.TensorDataset(data_x, data_y), batch_size=1, shuffle=True, num_workers=4
)
# Let's visualize the training data
sns.set()
plt.figure(figsize=(6, 4))
plt.plot(data_x[0, :, 0], label="Input feature 1")
plt.plot(data_x[0, :, 1], label="Input feature 1")
plt.plot(data_y[0, :, 0], label="Target output")
plt.ylim((-1, 1))
plt.title("Training data")
plt.legend(loc="upper right")
plt.show()
```

```
data_x.size:  torch.Size([1, 48, 2])
data_y.size:  torch.Size([1, 48, 1])
/usr/local/lib/python3.8/dist-packages/torch/utils/data/dataloader.py:554: UserWarning: This DataLoader will create 4 worker
  warnings.warn(_create_warning_msg(
```

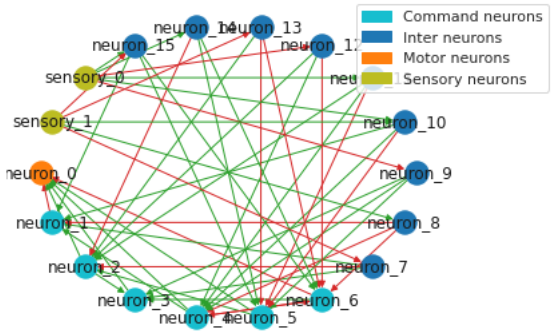


Here we can finally create a LTCCell and make use of the predefined sparse wiring structures of the keras-ncp package. For simplicity we will just define a fully-connected RNN

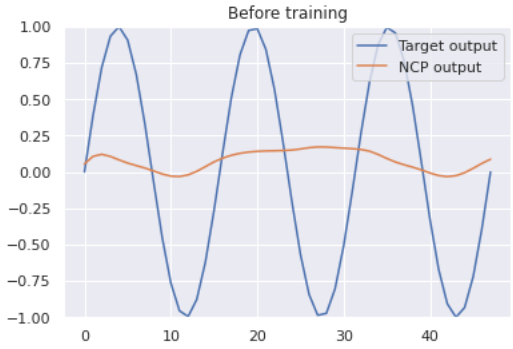
```
wiring = AutoNCP(16, out_features)  # 16 units, 1 motor neuron

ltc_model = LTC(in_features, wiring, batch_first=True)
learn = SequenceLearner(ltc_model, lr=0.01)
trainer = pl.Trainer(
    logger=pl.loggers.CSVLogger("log"),
    max_epochs=400,
    gradient_clip_val=1,  # Clip gradient to stabilize training
    gpus=0,
)
```

```
/usr/local/lib/python3.8/dist-packages/pytorch_lightning/trainer/connectors/accelerator_connector.py:466: LightningDeprecati
  rank_zero_deprecation(
/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
alloc!
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
```

```
sns.set_style("white")
plt.figure(figsize=(6, 4))
legend_handles = wiring.draw_graph(draw_labels=True,  neuron_colors={"command": "tab:cyan"})
plt.legend(handles=legend_handles, loc="upper center", bbox_to_anchor=(1, 1))
sns.despine(left=True, bottom=True)
plt.tight_layout()
plt.show()
```




```
sns.set()
with torch.no_grad():
  prediction = ltc_model(data_x)[0].numpy()
plt.figure(figsize=(6, 4))
plt.plot(data_y[0, :, 0], label="Target output")
plt.plot(prediction[0, :, 0], label="NCP output")
plt.ylim((-1, 1))
plt.title("Before training")
plt.legend(loc="upper right")
plt.show()
```



... and train our network

```
trainer.fit(learn, dataloader)
```

```
/usr/local/lib/python3.8/dist-packages/pytorch_lightning/trainer/configuration_validator.py:108: PossibleUserWarning: You de
  rank_zero_warn(
WARNING:lightning_fabric.loggers.csv_logs:Missing logger folder: log/lightning_logs
INFO:pytorch_lightning.callbacks.model_summary:
  | Name  | Type | Params
-------------------------------
0 | model | LTC  | 1.5 K
-------------------------------
1.2 K     Trainable params
```

```
results = trainer.test(learn, dataloader)
```

```
/usr/local/lib/python3.8/dist-packages/pytorch_lightning/trainer/connectors/data_connector.py:488: PossibleUserWarning: Your
  rank_zero_warn(
```

Testing DataLoader 0: 100% 1/1 [00:00<00:00, 12.98it/s]

Test metric	DataLoader 0
val_loss	0.0017994643421843648

```
sns.set()
with torch.no_grad():
  prediction = ltc_model(data_x)[0].numpy()
plt.figure(figsize=(6, 4))
plt.plot(data_y[0, :, 0], label="Target output")
plt.plot(prediction[0, :, 0], label="NCP output")
plt.ylim((-1, 1))
plt.title("After training")
plt.legend(loc="upper right")
plt.show()
```