
MULTIDIMENSIONAL PARAMETER ESTIMATION WITH DEEP LEARNING

M.Sc. Steffen Schieler



OUTLINE

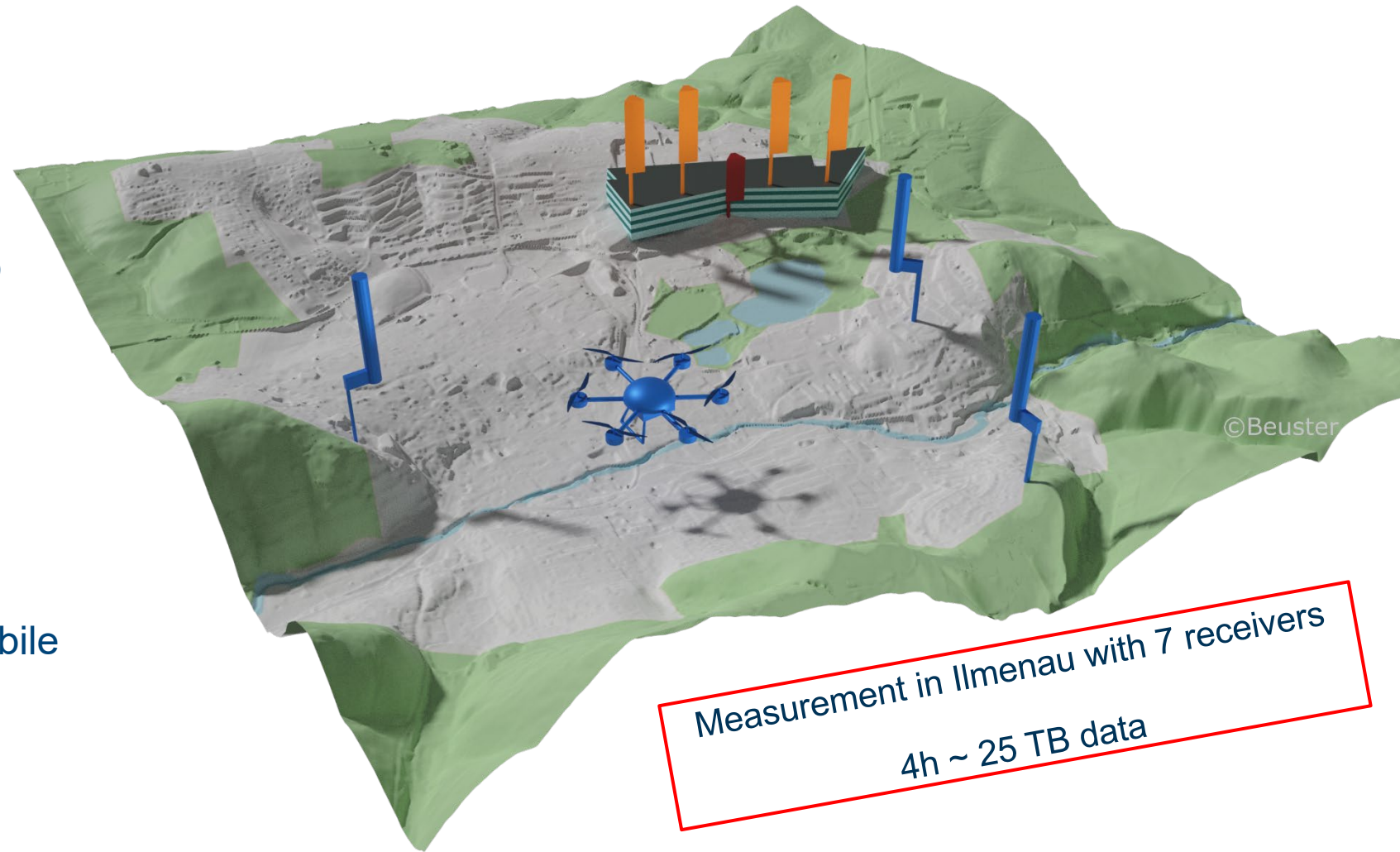
Multidimensional Parameter Estimation with Deep Learning

- Introduction
 - Wireless propagation
 - 1-D parameter estimation
- Evolution from 1-D to N-D
- Preliminary results
- Summary and outlook

Introduction

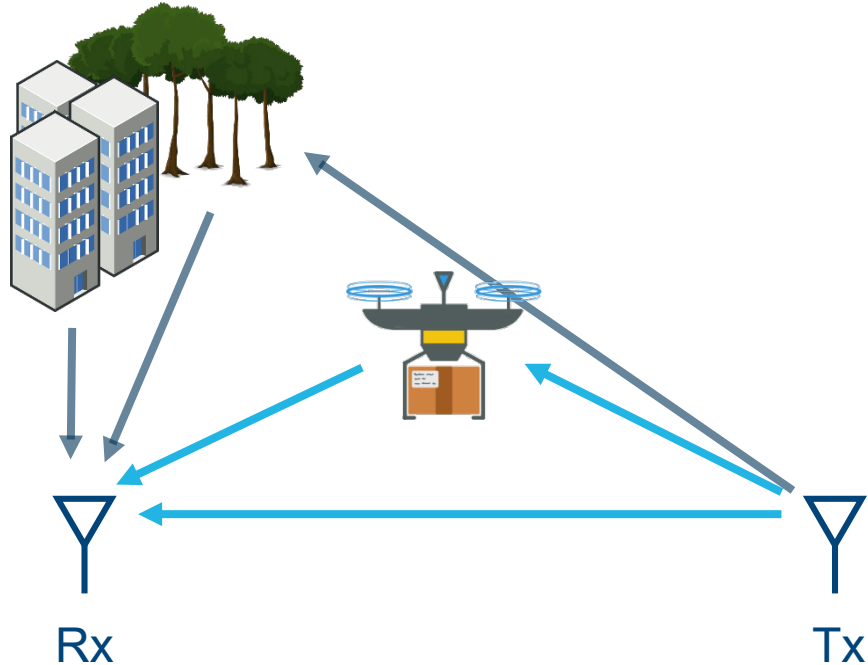
Drone-Shield

- Use existing wireless communication infrastructure to perform radar detection
- Similar to passive radar
- Uses OFDM(A) waveform
 - LTE, 5G, WiFi 802.11
- E.g. to detect UAVs with an mobile communication network

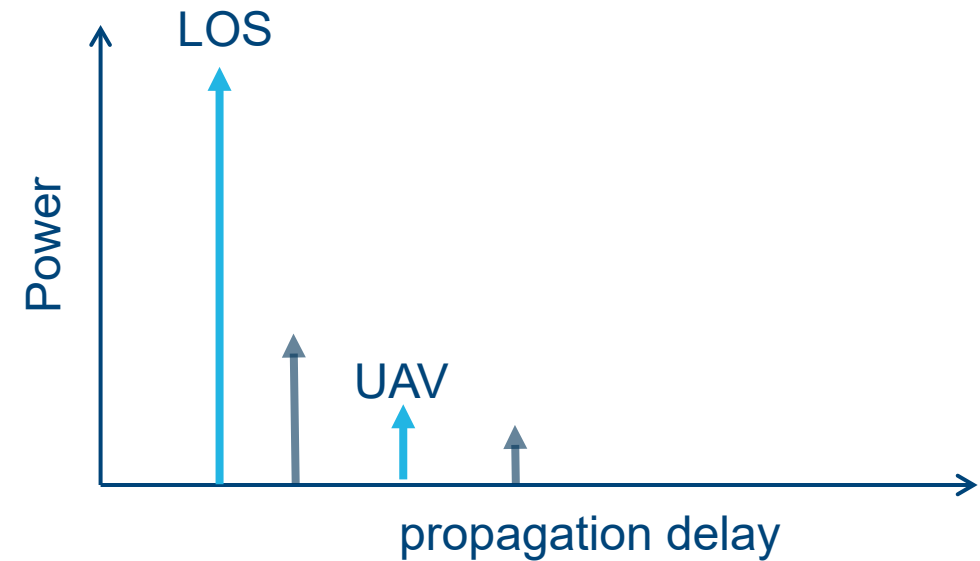


INTRODUCTION

Wireless Propagation and Parameter Estimation



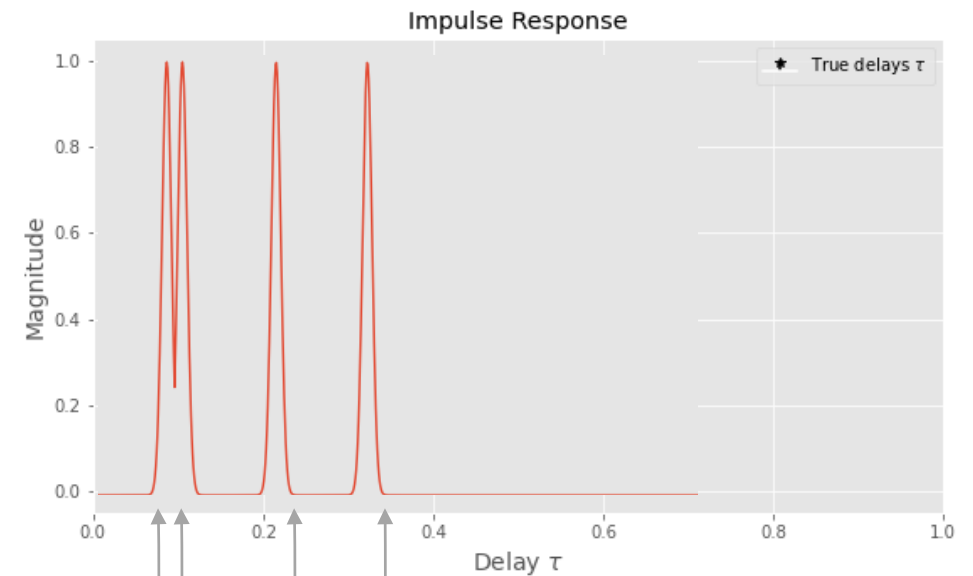
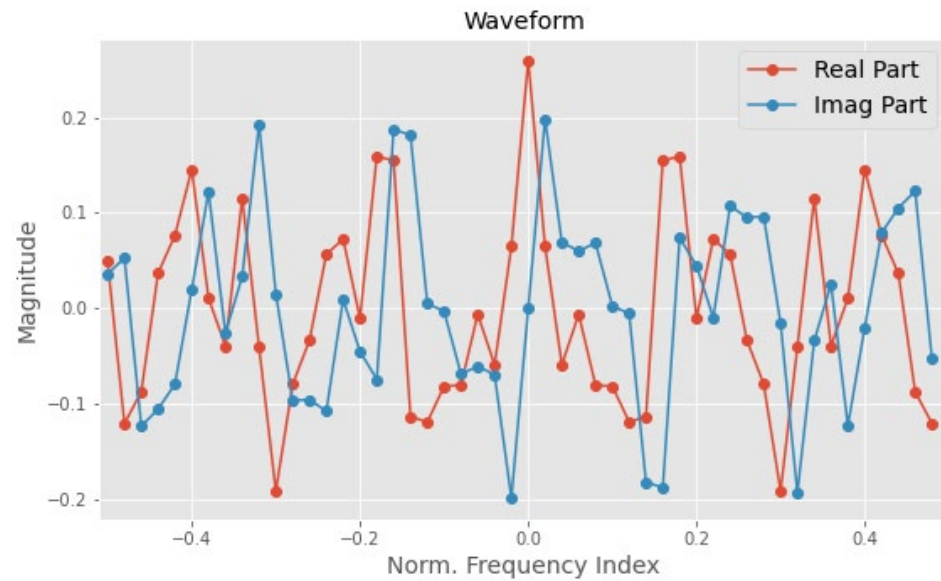
Radar localization



QUICK RECAP

1D parameter estimation

Estimate a pseudo PDF for the parameters space from a channel transfer function

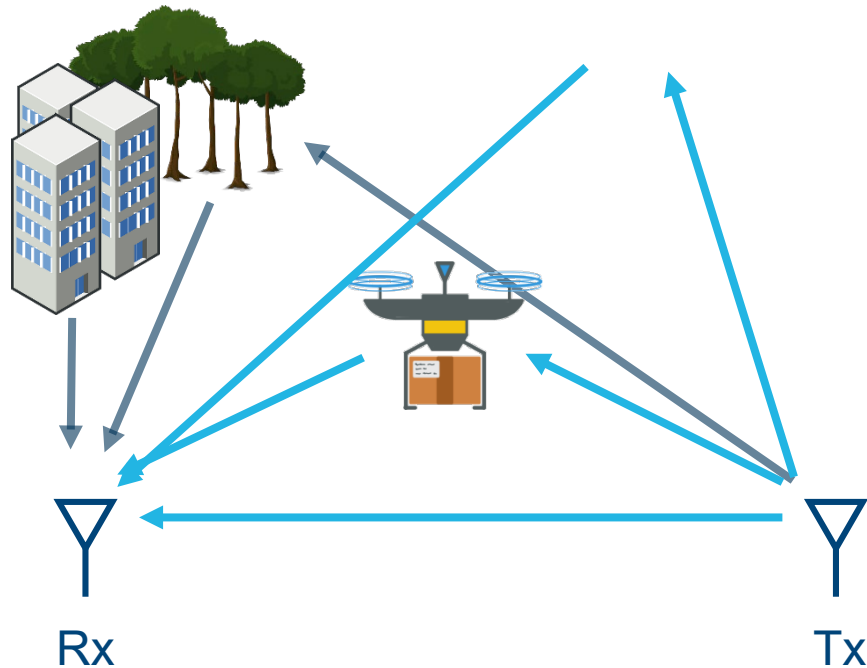


$$\sum_i a_i e^{-j2\pi f \tau_i}$$

Band-unlimited Impulse response

MOTIVATION

Multidimensional Parameter Estimation



- Moving objects cause Doppler shifts in the received signal
 - Each target has a characteristic delay and Doppler
 - Estimation results now has two dimensions (delay, Doppler)
- Why to consider multiple dimensions (e.g. Doppler)?
 - Doppler helps separating static clutter and moving targets
 - Easy filtering and removal of static clutter

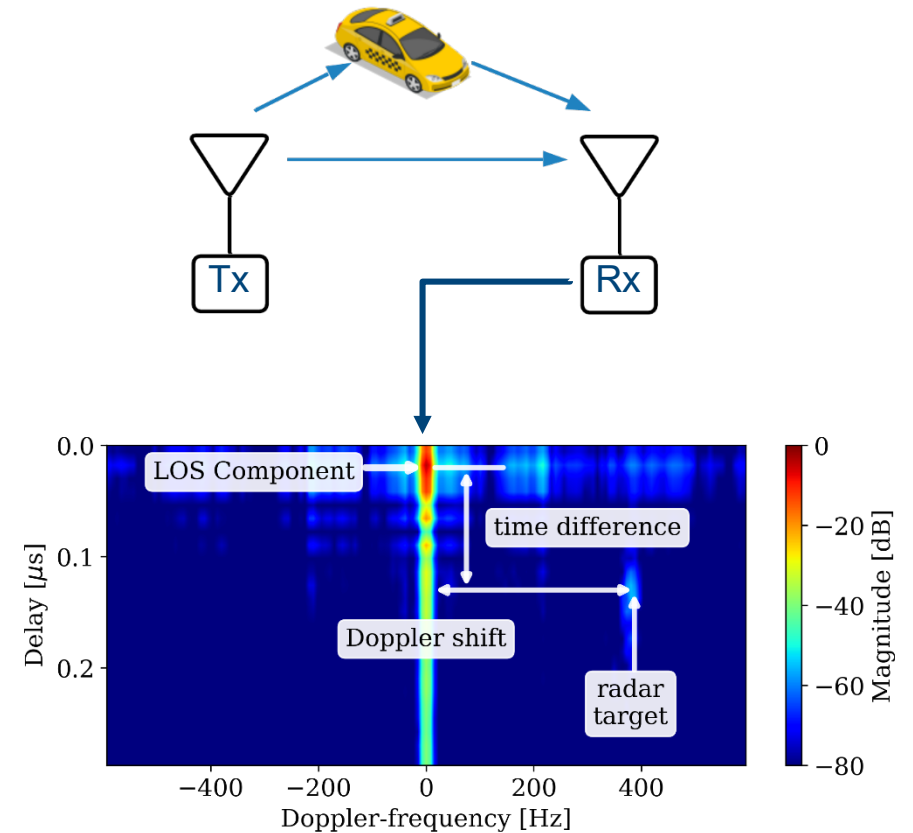
$$y = \sum_i a_i e^{-j2\pi f \tau_i}$$

More dimensions possible!
e.g. spatial, polarimetric

PRACTICLE EXAMPLE

From measurements

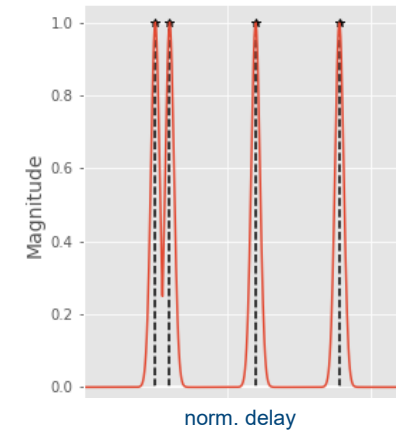
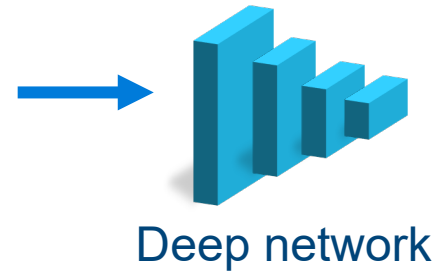
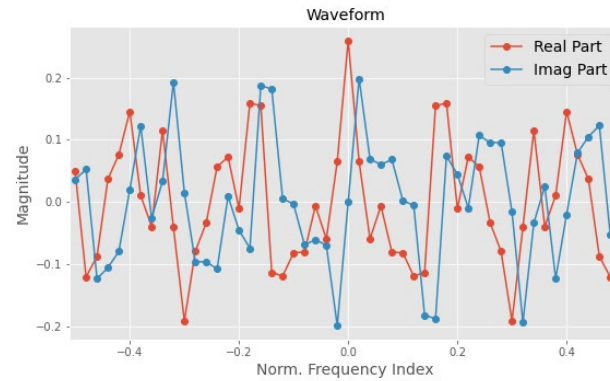
- Use standard OFDM receiver processing to estimate the wireless channel
 - Target can be detected
 - Target can be separated from static clutter
- Also works for UAVs (hopefully, very likely!)



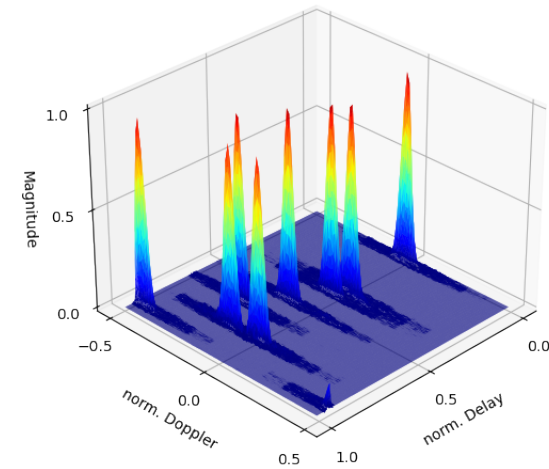
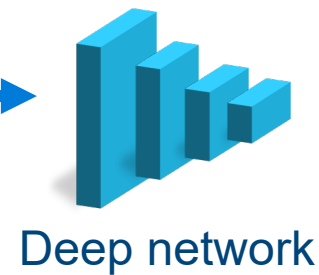
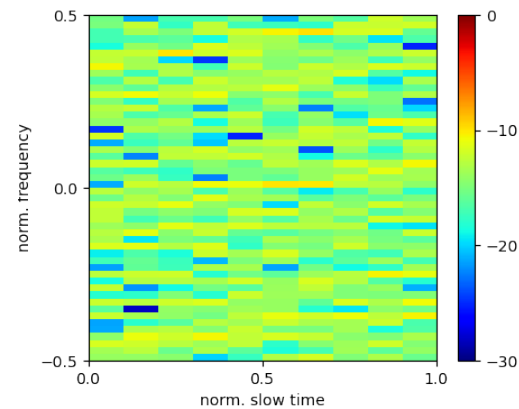
EVOLUTION

Moving from 1-D to 2-D to N-D

1D Task



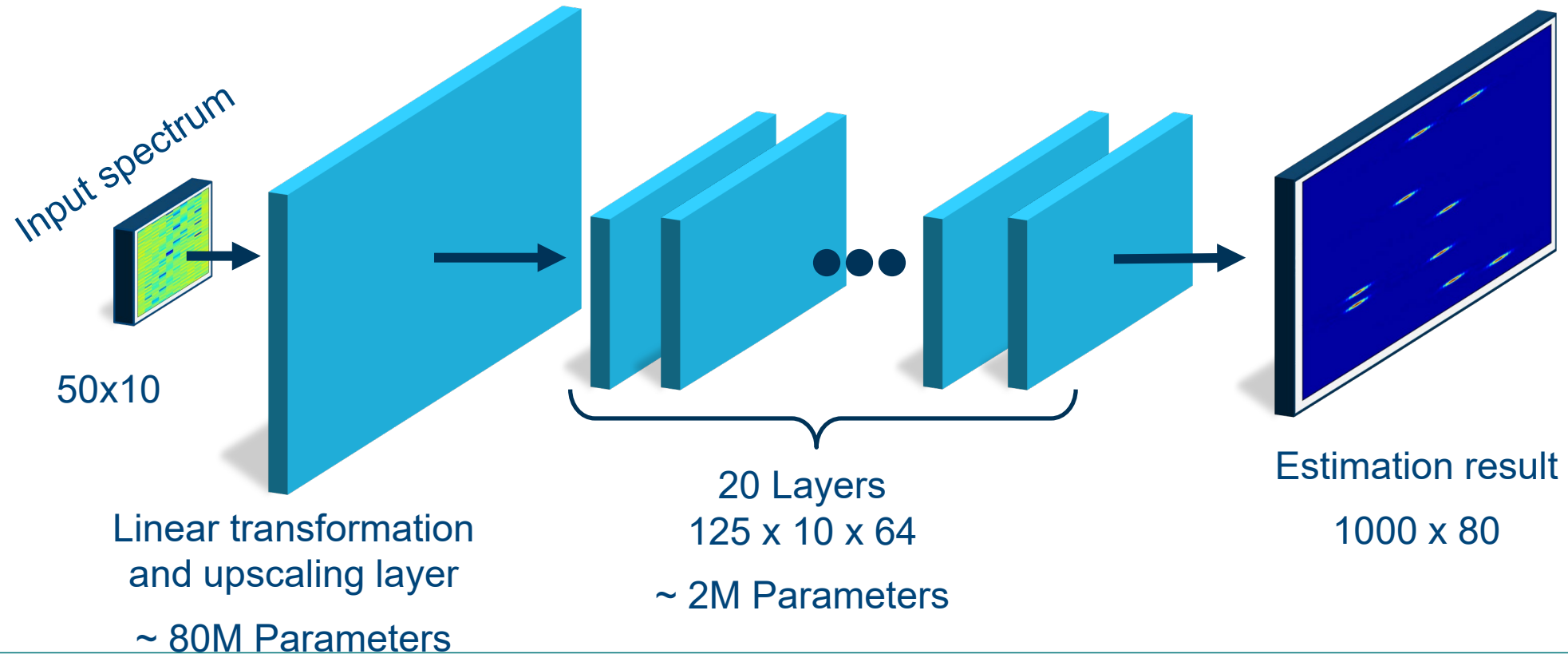
2-D example



NETWORK ARCHITECTURE

Information

Summary: ~ 82,000,000 (trainable) parameters



TRAINING

Information

- The network is trained on 10,000,000 different synthetic examples
- The network is trained for 100 epochs, during each epoch
 - Each sample is passed through the network once
 - Synthetic time-variant channel transfer functions (input) are calculated on the fly
 - Different noise realizations are added
- Training takes ~ 1 days on 8x Nvidia A100 GPUs (training time ~ # GPUs)
- Model is saved and used for testing/evaluation

TRAINING

Process on our GPU infrastructure

Code Development

(using Jupyter Notebooks)

```
[3]: system = deepest.training.Superres2dSystem(
    dataDim=(50, 10),
    nFilters=64,
    nLayers=10,
    innerDim=(125, 10),
    kernelDeconv=25,
    upsampling=8,
    activation=nn.Mish(),
    batch_size=batchSize,
)
if DEBUG:
    from torchinfo import summary
    print(summary(system.float(), (4, 50, 10, 2)))
system = system.double()
```

Layer (type:depth-idx)	Output Shape	Param #
Superres2dSystem	--	--
└─ Superres2dNet: 1	--	--
└─ TransformLearned: 2-1	[4, 80000, 1, 1, 1]	--
└─ Conv3d: 3-1	[4, 80000, 1, 1, 1]	80,000,000
└─ Sequential: 2-2	[4, 64, 125, 10]	--
└─ DoubleConv2d: 3-2	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-3	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-4	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-5	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-6	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-7	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-8	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-9	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-10	[4, 64, 125, 10]	73,984
└─ DoubleConv2d: 3-11	[4, 64, 125, 10]	73,984
└─ TransposedConv2d: 2-3	[4, 1, 1000, 80]	--
└─ ConvTranspose2d: 3-12	[4, 1, 1000, 80]	40,000

Total params: 80,779,840
Trainable params: 80,779,840
Non-trainable params: 0
Total mult-adds (G): 16.81

GPU Batchjob

```
1 #!/bin/csh
2 #BSUB -q BatchGPU
3 #BSUB -R "a100 span[hosts=1] select[h
4 #BSUB -eo ./logs/batchgpu/%J.log
5 #BSUB -oo ./logs/batchgpu/%J.err
6 #BSUB -J deepest
7 #BSUB -L /bin/csh
8 #BSUB -n 30
9 #cd /usr/scratch4/stsc1402/Learning/t
10 pwd
11 module purge
12 module load intel/v2020 cuda/v11.2
13 date
14 nvidia-smi
15 conda activate a100
16 jupyter nbconvert --to script trainin
17 mv training-superres2d.py training-$L
18 chmod +x training-$LSB_JOBID.py
19 setenv WANDB_MODE dryrun
20 setenv WANDB_DIR /usr/scratch4/stsc14
21 setenv WANDB_SILENT true
22 setenv WANDB_TAGS deepest
23 setenv NCCL_P2P_DISABLE 1 # disables
```



```
g/deepest :> batch.8gpu < train.lsf
```

Model Evaluation

(using Jupyter Notebooks)

```
[1]: import deepest
from deepest.training import *
from torchinfo import summary
import importlib

[10]: inferer = deepest.InferenceSuperres2d("2021.10.05.2040.model")

# the Datasets
dataset = deepest.TrainingData(
    name="testset2d",
    nDim=2,
    dataBin=(50, 10),
    resultBin=(1000, 80),
    noiseVar=(1e-12, 1),
    gaussianStd=(1/(3*50), 1/(3*10)),
    maskProbability=(0,1),
)

# the corresponding DataLoaders
dataLoader = torch.utils.data.DataLoader(
    dataset,
    batch_size=128,
    num_workers=1,
    worker_init_fn=lambda worker_id: np.random.seed(
        worker_id
    ),
    shuffle=True,
)

result = inferer.forward(next(iter(dataLoader)))

print("Successful inference. Result shape is {}".format(result.shape))
{'dataDim': (50, 10), 'nFilters': 64, 'nLayers': 10, 'innerDim': (125, 10)}
WARNING:root: Logger KroneckerModel is not activated yet.
WARNING:root: Logger ScaledIdentity is not activated yet.
WARNING:root: Logger ScaledIdentity is not activated yet.
WARNING:root: Logger Kronecker is not activated yet.
Successfully loaded net from 2021.10.05.2040.model
Successful inference. Result shape is (128, 1000, 80)
```

TRAINING

Statistics on HPC Utilization

Live Demo (lets hope this works)

www.wandb.ai

TRAINING

Progress Visualization

