Convolutional Neural Network and Soprano: computing numerically expensive models in the blink of an eye.

Damien Bégué

Bar Ilan University

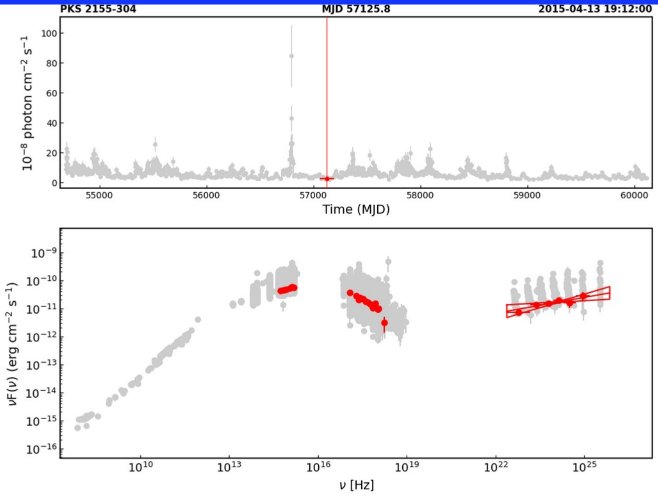
With

Narek Sahakyan, Hüsne Dereli-Bégué, Sargis Gasparyan, Asaf Pe’er

Paris, 21st-23rd of February 2024

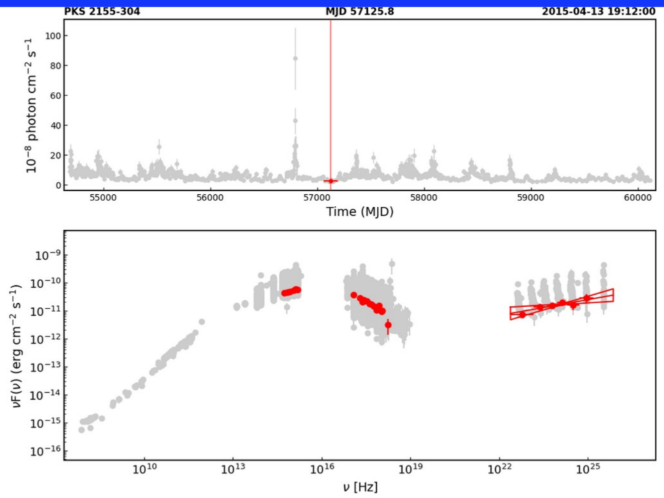
Bar-Ilan University
אוניברסיטת בר־אילן

Goal



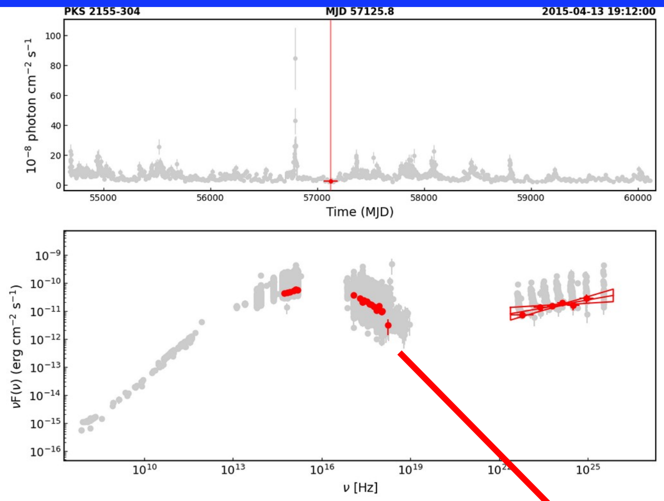
PKS 2155-304
MJD 57125.8
2015-04-13 19:12:00
10^{-8} photon cm^{-2} s^{-1}
Time (MJD)
$\nu F(\nu)$ (erg cm^{-2} s^{-1})
ν [Hz]

Goal



Your favorite model
with many parameters taking
time to compute.

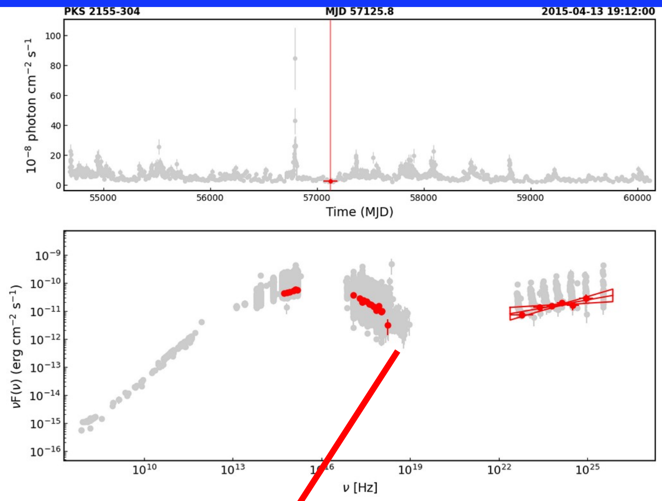
Goal



Your favorite model
with many parameters taking
time to compute.

Parameter posterior
distribution

Goal

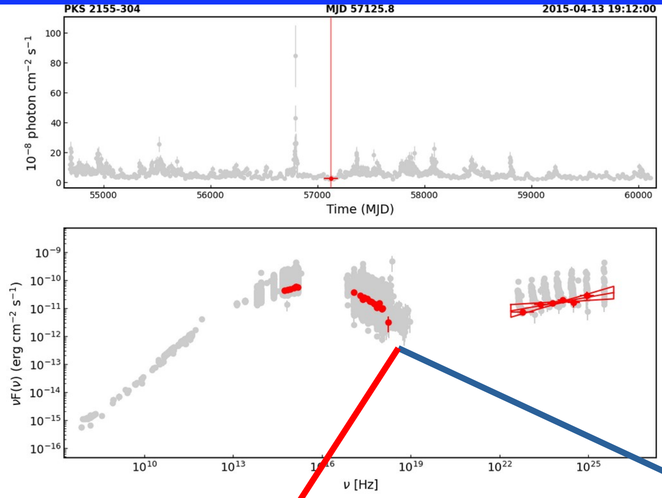


Your favorite model with many parameters taking time to compute.

Detailed data treatment

Parameter posterior distribution

Goal



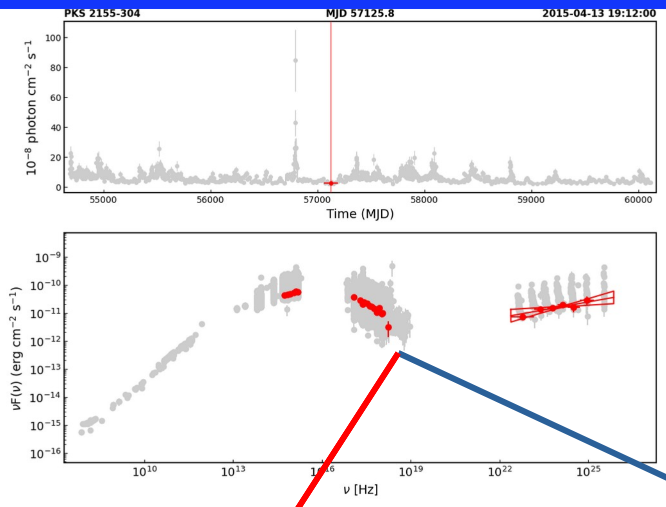
Your favorite model
with many parameters taking
time to compute.

Detailed data
treatment

Parameter posterior
distribution

PPC

Goal



Your favorite model
with many parameters taking
time to compute.

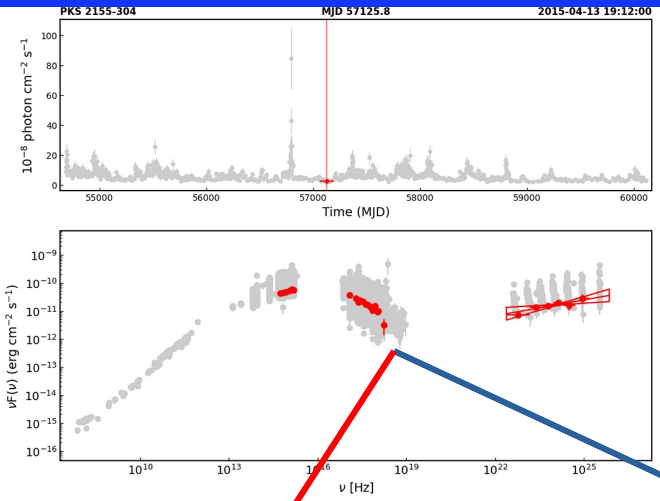
Detailed data treatment

Parameter posterior distribution

PPC

Model comparison

Goal



Your favorite model with many parameters taking time to compute.

Detailed data treatment

Parameter posterior distribution

PPC

Model comparison

Disclamer !

1. My talk is not about physics (although I would be happy to discuss it separately).

Disclamer !

1. My talk is not about physics (although I would be happy to discuss it separately).

2. My talk is not about data/observations.

Disclamer !

1. My talk is not about physics (although I would be happy to discuss it separately).
2. My talk is not about data/observations.
3. My talk is not about numerical modeling (my favorite part of the problem).

Disclamer !

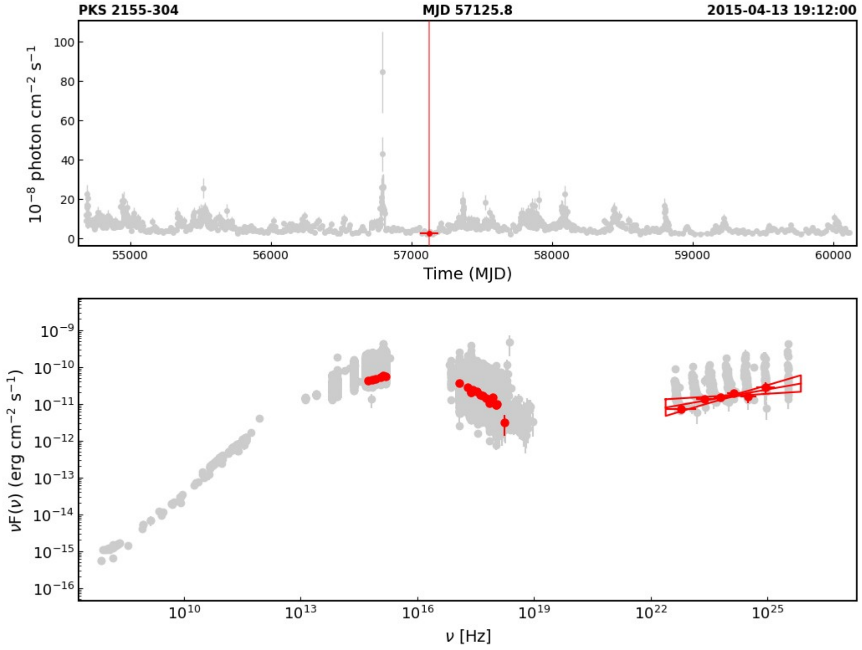
1. My talk is not about physics (although I would be happy to discuss it separately).

2. My talk is not about data/observations.

3. My talk is not about numerical modeling (my favorite part of the problem).

4. My talk is about what happens at the intersection of
physics/numerical modeling and data.

Goal



Your favorite model
with many parameters taking
time to compute.

Strategies

1) Give up.
   - Keep throwing a single line throught the data, one parameter set explains the data, no uncertainty, no inference, no data exploitation, no model exploration.

Strategies

1) Give up.
   - Keep throwing a single line throught the data, one parameter set explains the data, no uncertainty, no inference, no data exploitation, no model exploration.
2) Put the model in a fitting engine and hope.
   - After waiting for a very long time, get an answer. If it works good, but if not, it is too expensive to redo it. Even though, limited to a couple of dataset.

Strategies

1) Give up.
   - Keep throwing a single line throught the data, one parameter set explains the data, no uncertainty, no inference, no data exploitation, no model exploration.
2) Put the model in a fitting engine and hope.
   - After waiting for a very long time, get an answer. If it works good, but if not, it is too expensive to redo it. Even though, limited to a couple of dataset.
3) Table model.
   - Interpolate between sampled parameter sets.

Strategies

1) Give up.
   - Keep throwing a single line throught the data, one parameter set explains the data, no uncertainty, no inference, no data exploitation, no model exploration.
2) Put the model in a fitting engine and hope.
   - After waiting for a very long time, get an answer. If it works good, but if not, it is too expensive to redo it. Even though, limited to a couple of dataset.
3) Table model.
   - Interpolate between sampled parameter sets.
4) Machine learning approach.

Strategies

1) Give up.
    - Keep throwing a single line throught the data, one parameter set explains the data, no uncertainty, no inference, no data exploitation, no model exploration.

2) Put the model in a fitting engine and hope.
    - After waiting for a very long time, get an answer. If it works good, but if not, it is too expensive to redo it. Even though, limited to a couple of dataset.

3) Table model.
    - Interpolate between sampled parameter sets.

4) Machine learning approach.

The model needs to be computed many times

Strategies

1) Give up.
   - Keep throwing a single line throught the data, one parameter set explains the data, no uncertainty, no inference, no data exploitation, no model exploration.
2) Put the model in a fitting engine and hope.
   - After waiting for a very long time, get an answer. If it works good, but if not, it is too expensive to redo it. Even though, limited to a couple of dataset.
3) Table model.
   - Interpolate between sampled parameter sets.
4) Machine learning approach.

The model needs to be computed many times

But that computation needs to be done only once !

The model needs to be computed many times ...

How many times ?

The model needs to be computed many times ...

How many times ?

- For a bayesian fit: ~10^3 - 10^5 times. Non re-usable !?

The model needs to be computed many times ...

How many times ?

- For a bayesian fit: ~10^3 - 10^5 times. Non re-usable !?
- Table model: Strongly dependent on the number of parameters. SSC: 10^6-10^7 times.

The model needs to be computed many times ...

How many times ?

- For a bayesian fit: ~10^3 - 10^5 times. Non re-usable !?
- Table model: Strongly dependent on the number of parameters. SSC: 10^6-10^7 times.
  - Requires equidistant sampling = curse of dimensionality.
  - Unfailable numeric code.

The model needs to be computed many times ...

How many times ?

- For a bayesian fit: ~10^3 - 10^5 times. Non re-usable !?
- Table model: Strongly dependent on the number of parameters. SSC: 10^6-10^7 times.
  - Requires equidistant sampling = curse of dimensionality.
  - Unfailable numeric code.
- Machine learning: 10^5-10^6 depending on the number of parameters and the model complexity.

The model needs to be computed many times ...

How many times ?

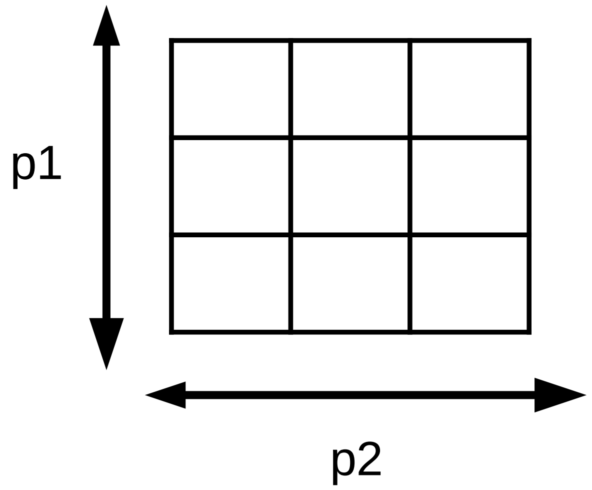
- For a bayesian fit: ~10^3 - 10^5 times. Non re-usable !?

- Table model: Strongly dependent on the number of parameters. SSC: 10^6-10^7 times.
  - Requires equidistant sampling = curse of dimensionality.
  - Unfailable numeric code.

- Machine learning: 10^5-10^6 depending on the number of parameters and the model complexity.
  - Smaller impact of curse of dimensionality,
  - Sample can be non-equidistant.

So the model needs to be computed many times. But for which parameter combinations?

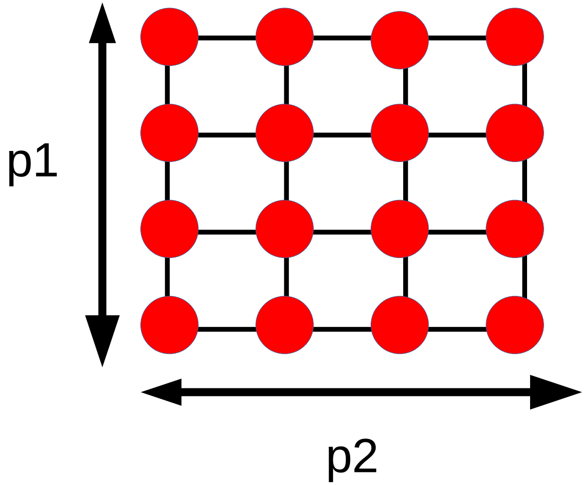
<ins>“Square parameter space”</ins>



So the model needs to be computed many times. But for which parameter combinations?

<u>"Square parameter space"</u>
<u>Table model/parameter scan:</u>

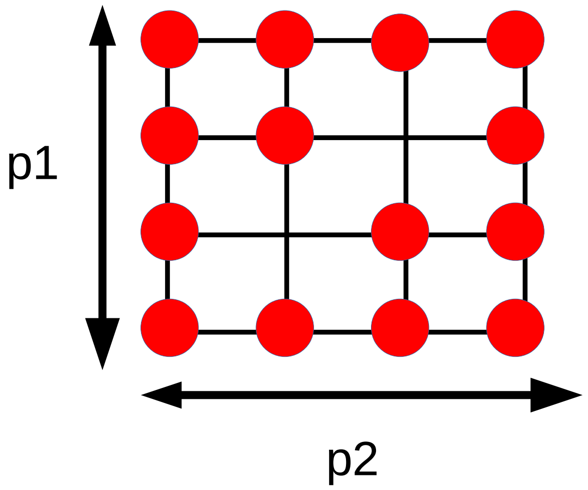
$$N = N_{p_1} * N_{p_2} * N_{p_3} \dots$$



So the model needs to be computed many times. But for which parameter combinations?

<u>“Square parameter space”</u>
<u>Table model/parameter scan:</u>

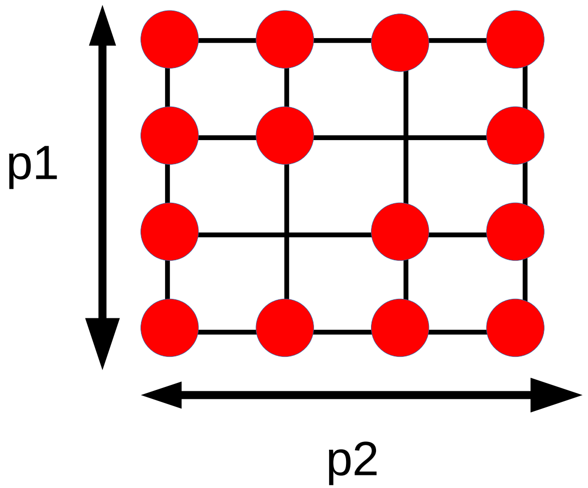
$$N = N_{p_1} * N_{p_2} * N_{p_3} \ldots$$



So the model needs to be computed many times. But for which parameter combinations?

“Square parameter space”
Table model/parameter scan:

$$N = N_{p_1} * N_{p_2} * N_{p_3} \ldots$$

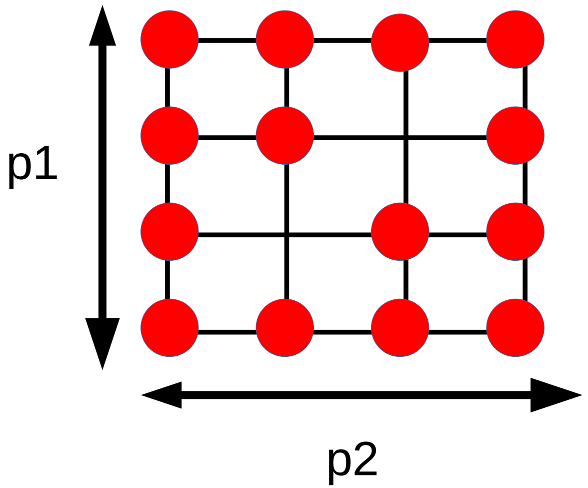


Machine learning:

So the model needs to be computed many times. But for which parameter combinations?

<u>"Square parameter space"</u>
<u>Table model/parameter scan:</u>

$$N = N_{p_1} * N_{p_2} * N_{p_3} \ldots$$



<u>Machine learning:</u>
- equal spacing is NOT required

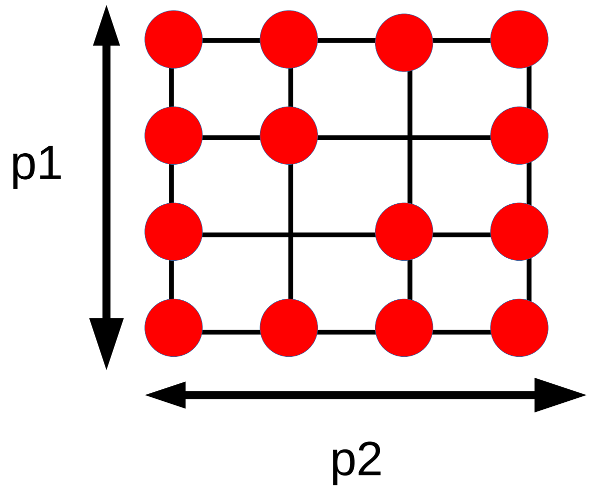
→ Latin hypercube sampling

<u>Viana (2016): A tutorial on Latin hypercube design of experiments</u>

So the model needs to be computed many times. But for which parameter combinations?

<u>"Square parameter space"</u>
<u>Table model/parameter scan:</u>

$$N = N_{p_1} * N_{p_2} * N_{p_3} \dots$$



<u>Machine learning:</u>

- equal spacing is NOT required

→ Latin hypercube sampling

<u>Viana (2016): A tutorial on Latin hypercube design of experiments</u>

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

You have to know in advance:
1) required ressources (memory, number of cores),

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

You have to know in advance:
1) required ressources (memory, number of cores),
2) an estimate of total compute time,

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

You have to know in advance:
1) required ressources (memory, number of cores),
2) an estimate of total compute time,
3) an estimate of compute time per job/model estimation,

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

You have to know in advance:
1) required ressources (memory, number of cores),
2) an estimate of total compute time,
3) an estimate of compute time per job/model estimation,
4) track a performance status for each job,

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

You have to know in advance:
1) required ressources (memory, number of cores),
2) an estimate of total compute time,
3) an estimate of compute time per job/model estimation,
4) track a performance status for each job,
5) track compute status (slurm interuption).

Parameter sampling done, computation of the corresponding models

Embarrassingly parallel problem: read a parameter set based on a simulation index.

1) slurm (< 50k) or slurm array.

2) distribution layer + slurm (ronswanson: https://github.com/grburgess/ronswanson).

3) FNC (few millions jobs).

You have to know in advance:
1) required ressources (memory, number of cores),
2) an estimate of total compute time,
3) an estimate of compute time per job/model estimation,
4) track a performance status for each job,
5) track compute status (slurm interuption).

Run successive job fractions: 0.1%, 1%, Maybe 10%.

Run successive job fractions

Run successive job fractions: 0.1%, 1%, maybe 10%

1) evaluate the performances of the parameter space:

Run successive job fractions

Run successive job fractions: 0.1%, 1%, maybe 10%

1) evaluate the performances of the parameter space:

- It is still time to change it (parameter bounds)

Run successive job fractions

Run successive job fractions: 0.1%, 1%, maybe 10%

1) evaluate the performances of the parameter space:
- It is still time to change it (parameter bounds)

2) understand where the numerical bottlenecks are and why.
- evaluate average compute time,
- total compute time,
- identify code failures and reasons.

Run successive job fractions

Run successive job fractions: 0.1%, 1%, maybe 10%

1) evaluate the performances of the parameter space:
   - It is still time to change it (parameter bounds)

2) understand where the numerical bottlenecks are and why.
   - evaluate average compute time,
   - total compute time,
   - identify code failures and reasons.

3) test the neural network with smaller training set
   - Smaller sample, faster training,

FNC is the synthesis of decades of experience with jobs schedulers

PBS + Grid Engine + Accelerator

ALTAIR

Scalable
- From 1 job to 20 million jobs in queue

Small, Quick
- Small memory footprint
- Speed: clocked up to 70k+ tasks/second

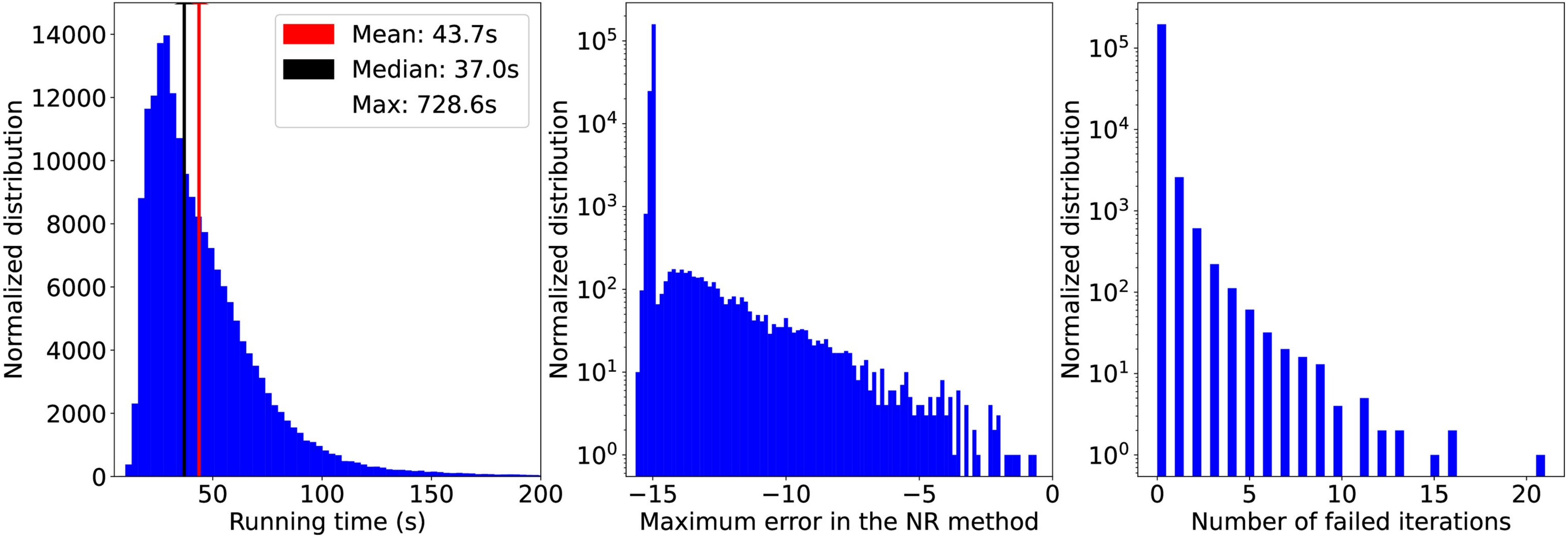
Feature Rich
- Full-cycle scheduler
- Cost-driven job placement
- Workload analysis (via simulation)
- Prediction of job duration + job size
- PMIx support (evolution of MPI)
- SAGA (storage-aware scheduling)
- Rapid Scaling in cloud

Contact: casotto@altair.com

Use of monitoring data

Metadata for SSC model:

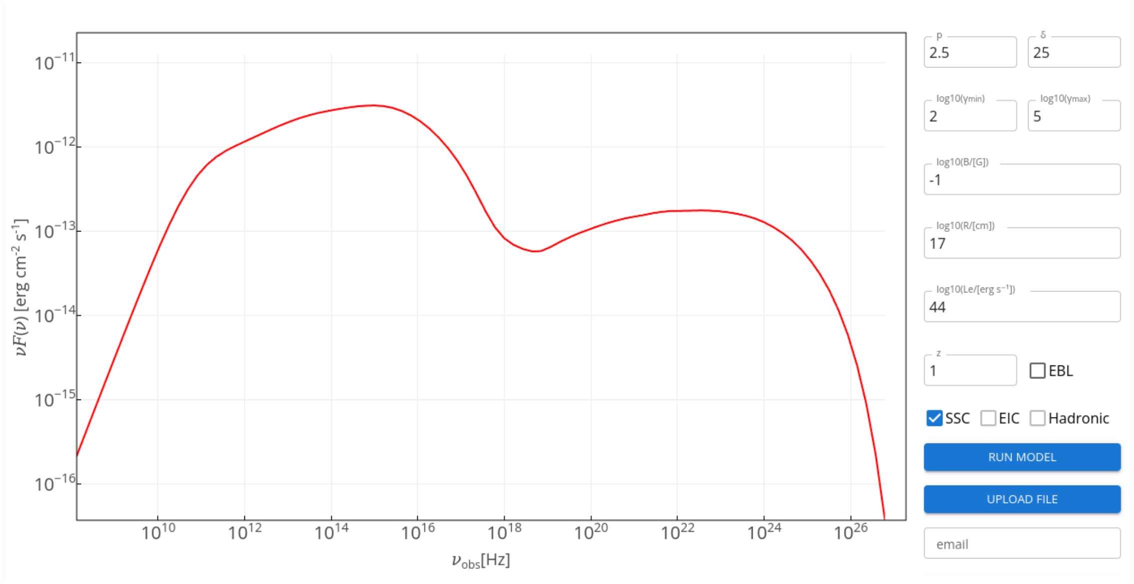
$2x10^5$ spectra



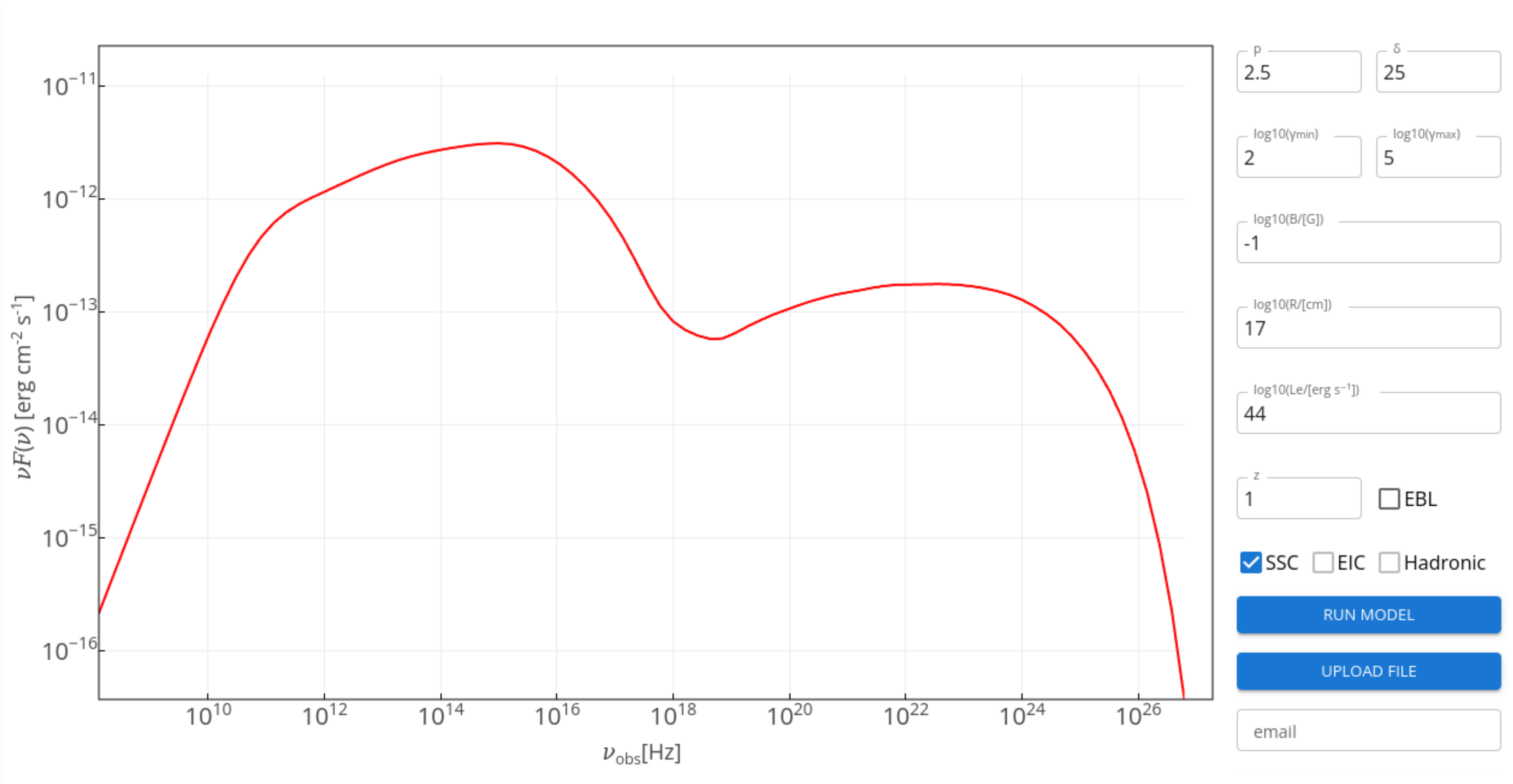
Building the neural network

P1
P2
P3
P4
P5
P6
...

f(P1, P2, P3 ...)

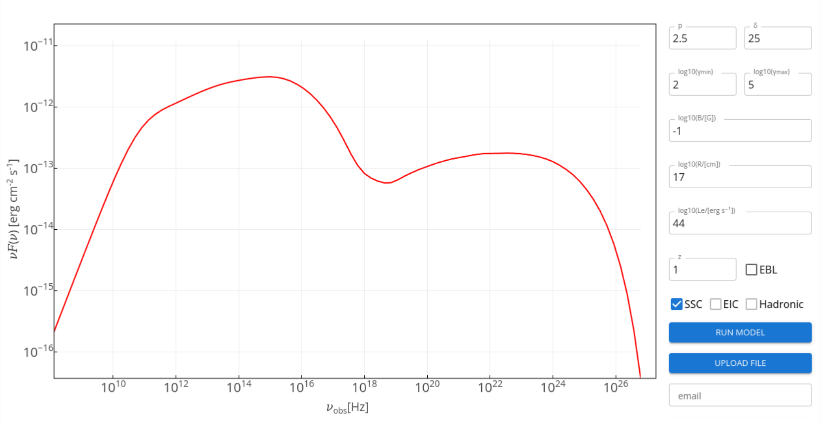


Data preprocessing

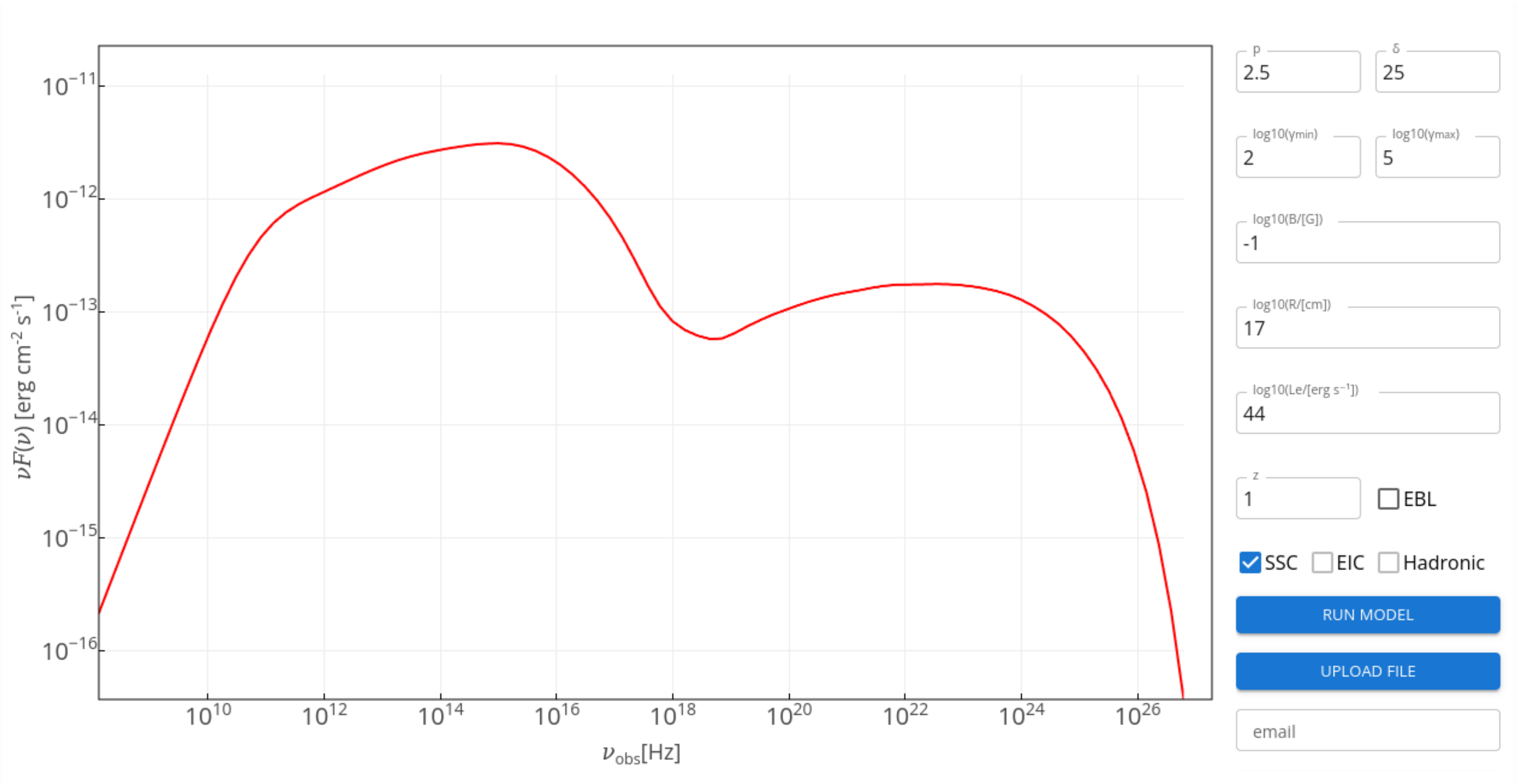


Data preprocessing

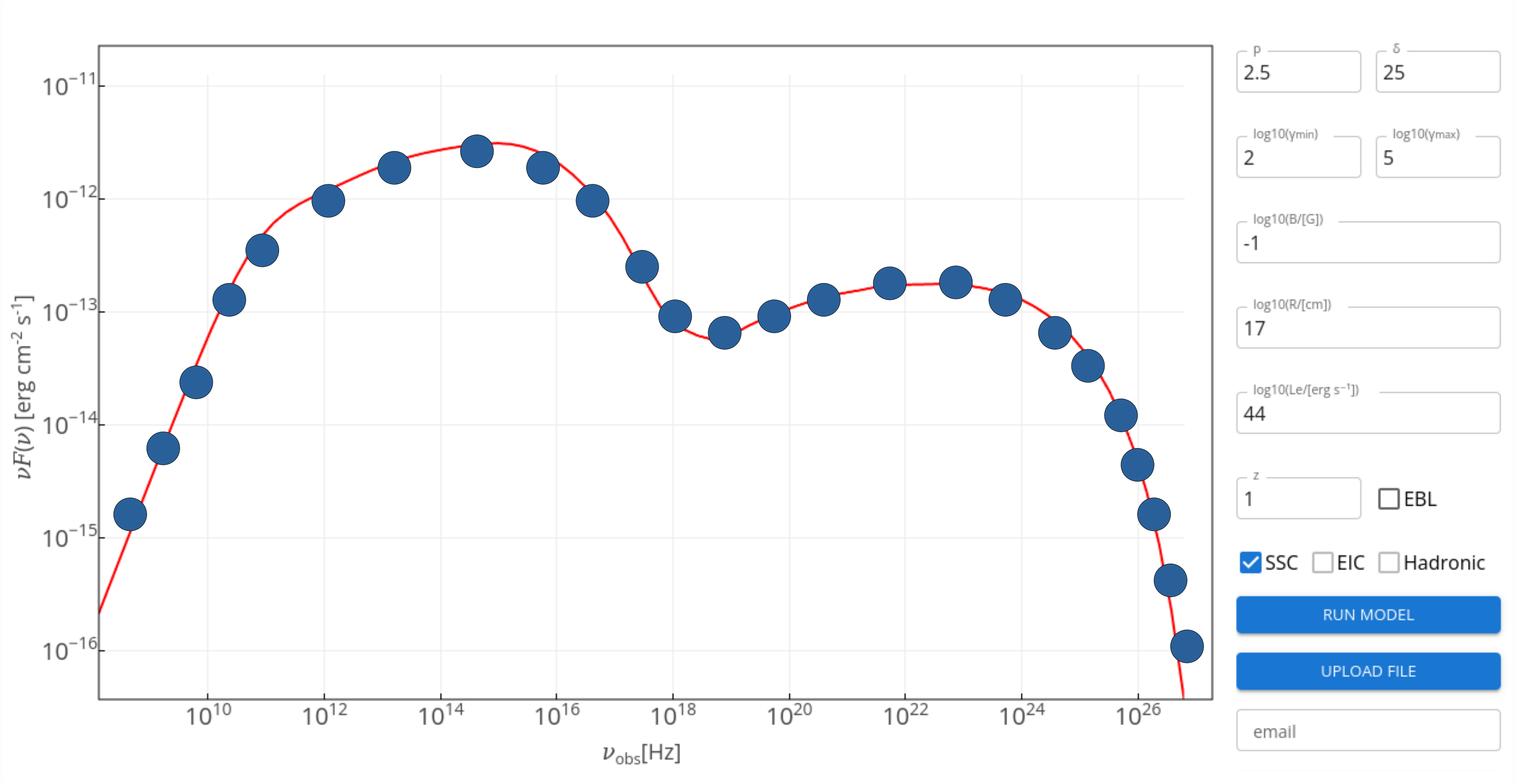
Step 1: log the data



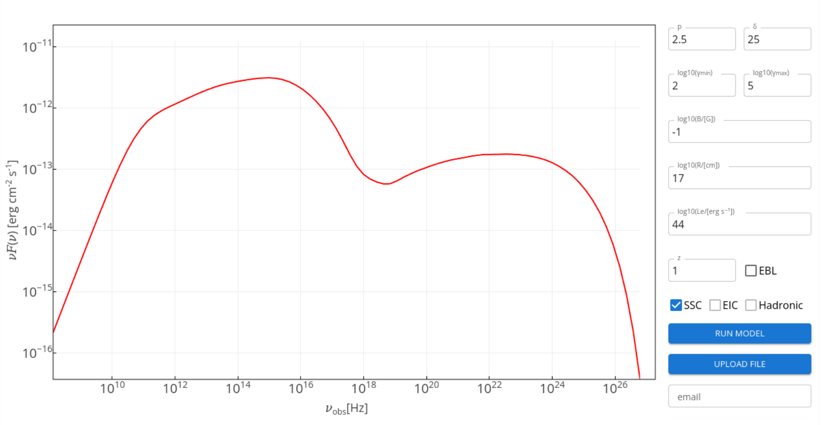
Data preprocessing



Data preprocessing



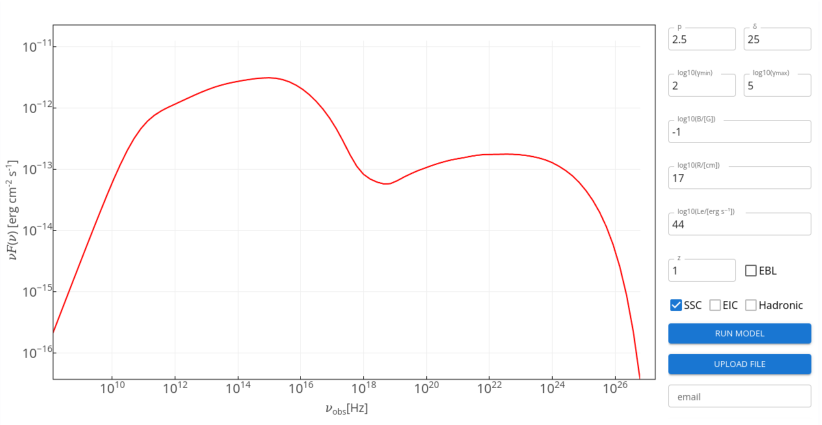
Data preprocessing



Step 1: log the data

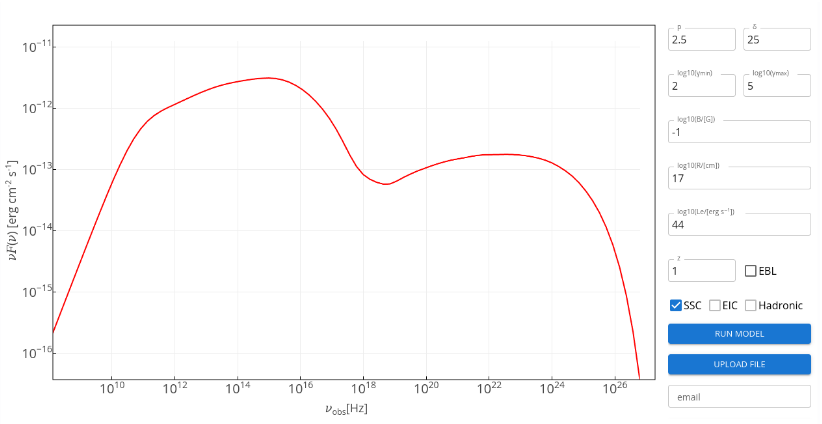
Step 2: remove the mean

Data preprocessing



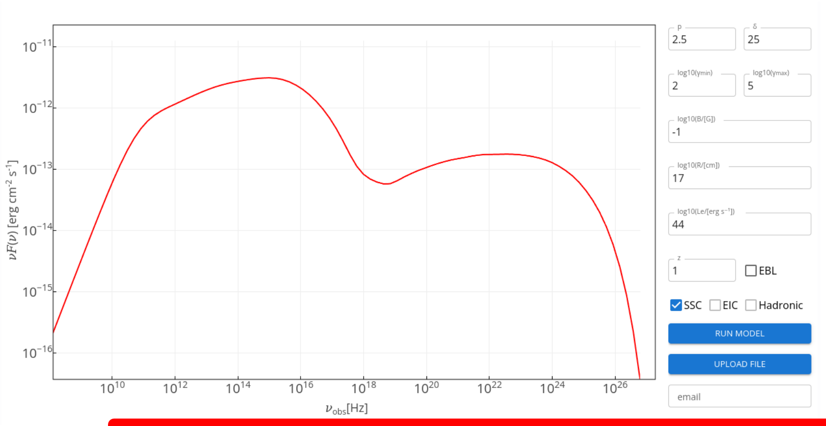
Step 1: log the data
Step 2: remove the mean
(Step 3: detrend)

Data preprocessing



Step 1: log the data
Step 2: remove the mean
(Step 3: detrend)
Step 4: normalise amplitude to]-1,1[

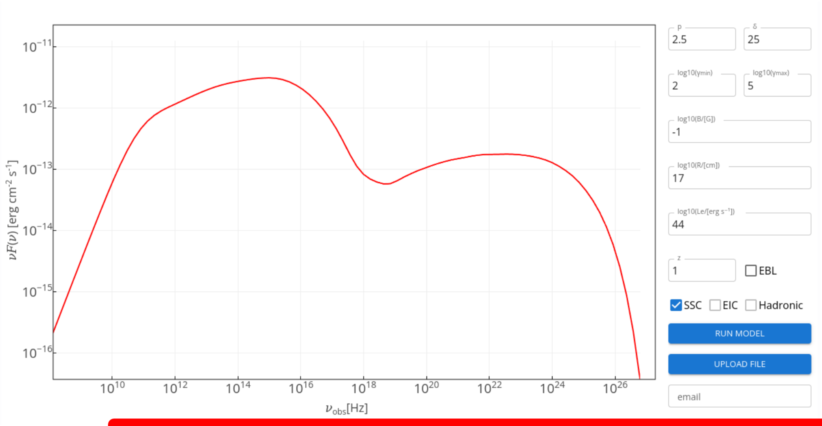
Data preprocessing



Step 1: log the data
Step 2: remove the mean
(Step 3: detrend)
Step 4: normalise amplitude to]-1,1[

Steps 2, 3 and 4 must be done for all data, not per energy bin.

Data preprocessing

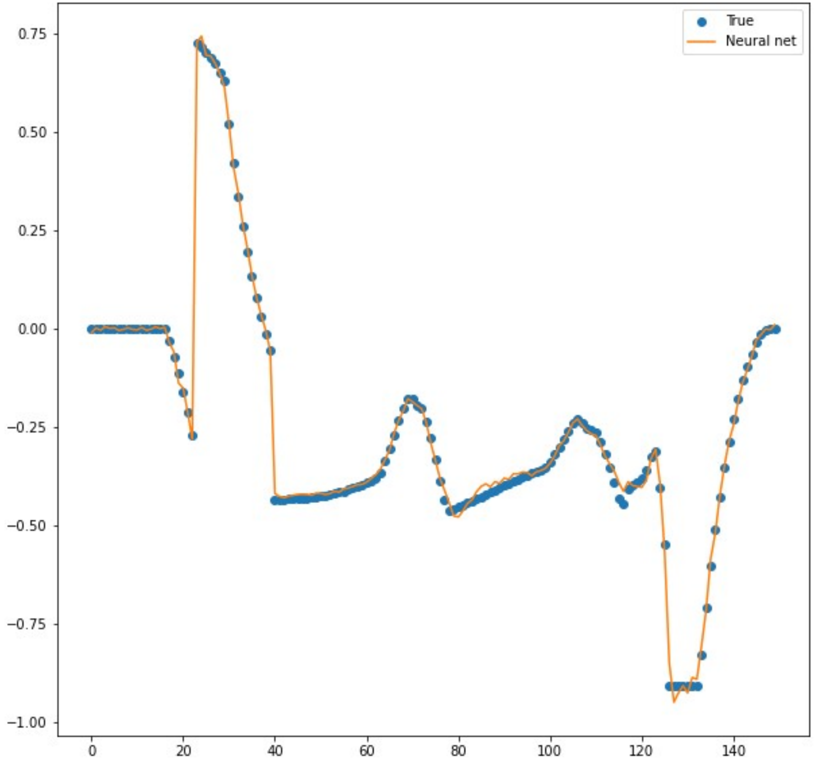


Step 1: log the data
Step 2: remove the mean
(Step 3: detrend)
Step 4: normalise amplitude to]-1,1[

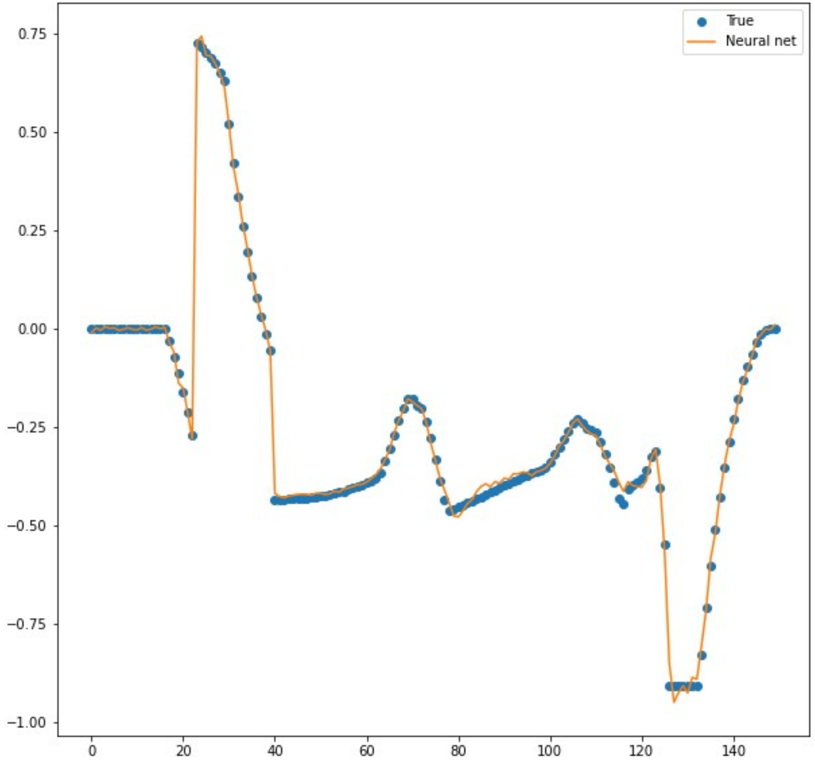
Steps 2, 3 and 4 must be done for all data, not per energy bin.

Any different treatment is producing oscillations in the resulting spectrum

Data preprocessing

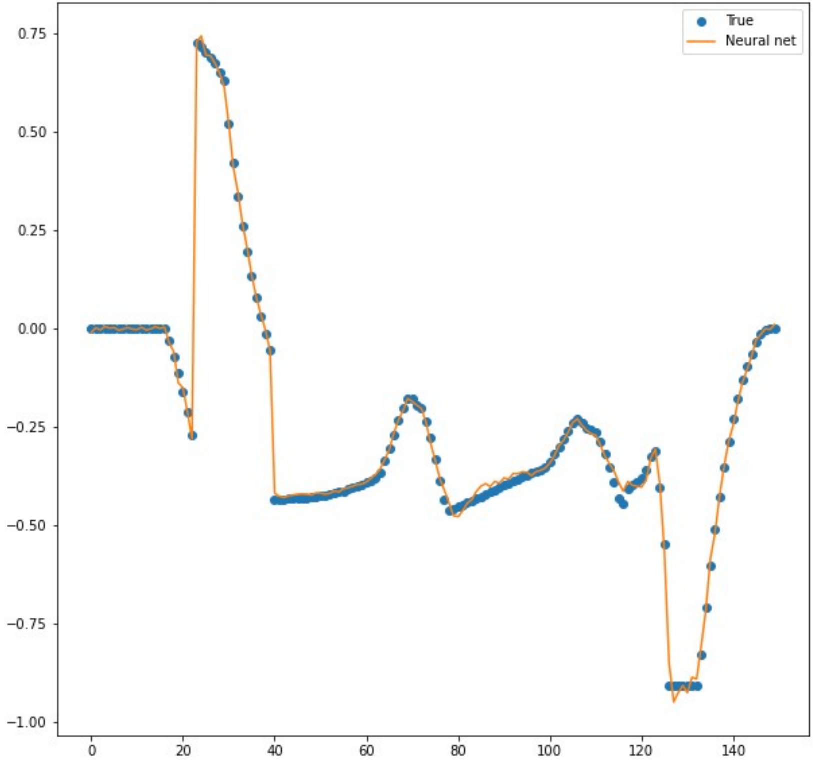


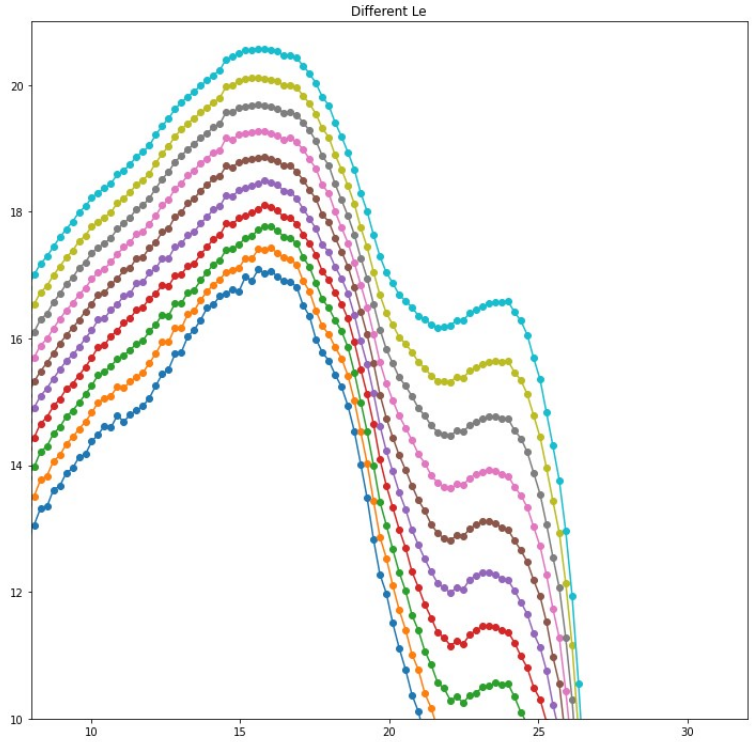
Data preprocessing



The resulting spectrum is NOT smooth !

Data preprocessing





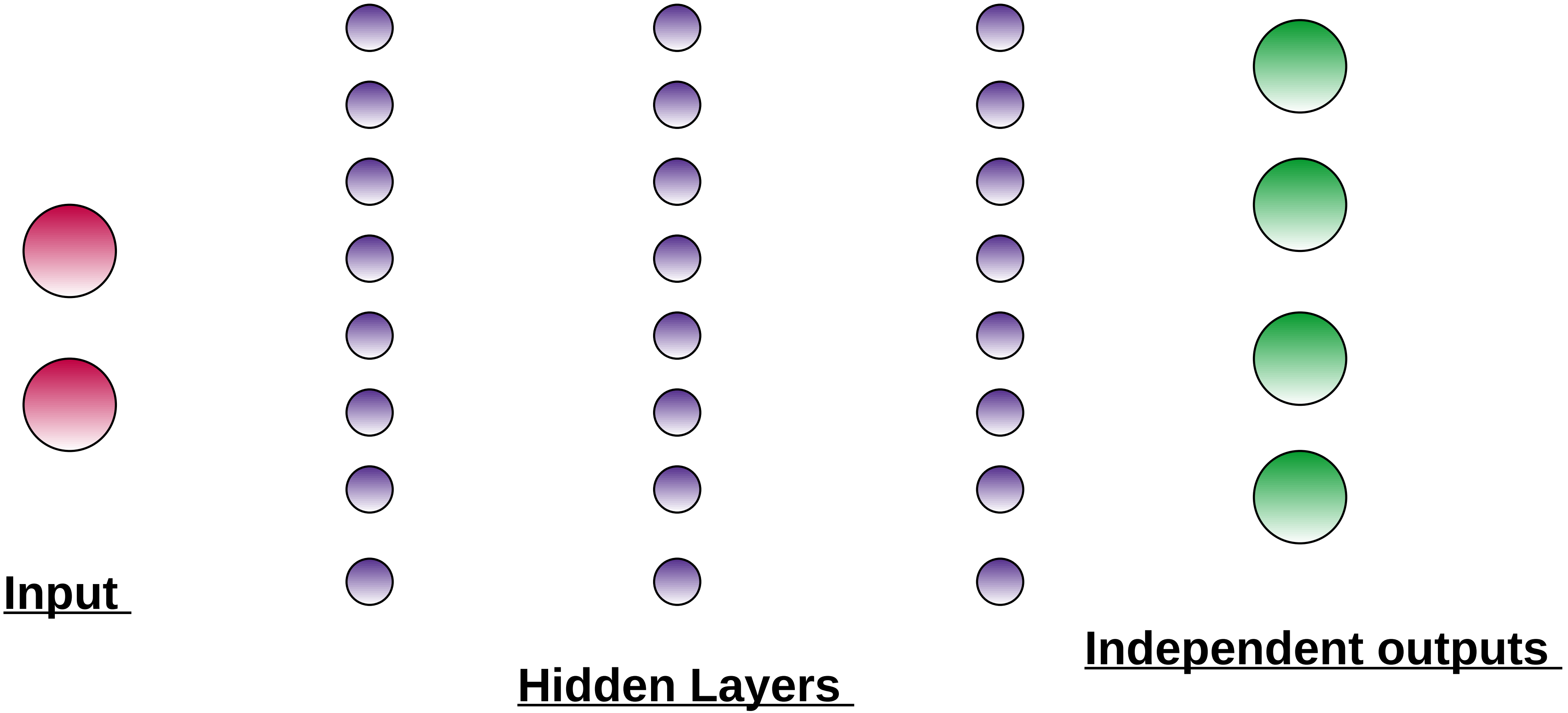
The resulting spectrum is NOT smooth !

The neural network

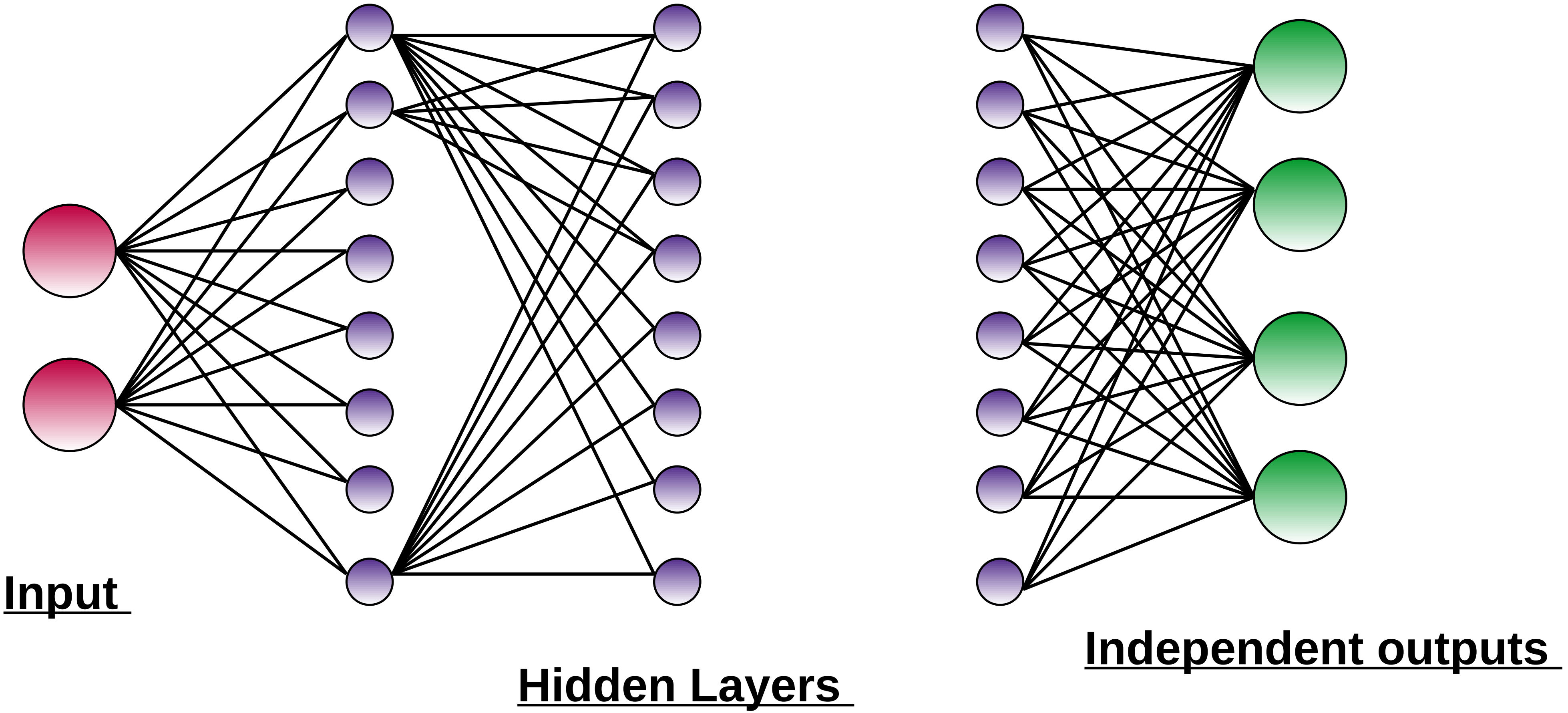
Input

Independent outputs

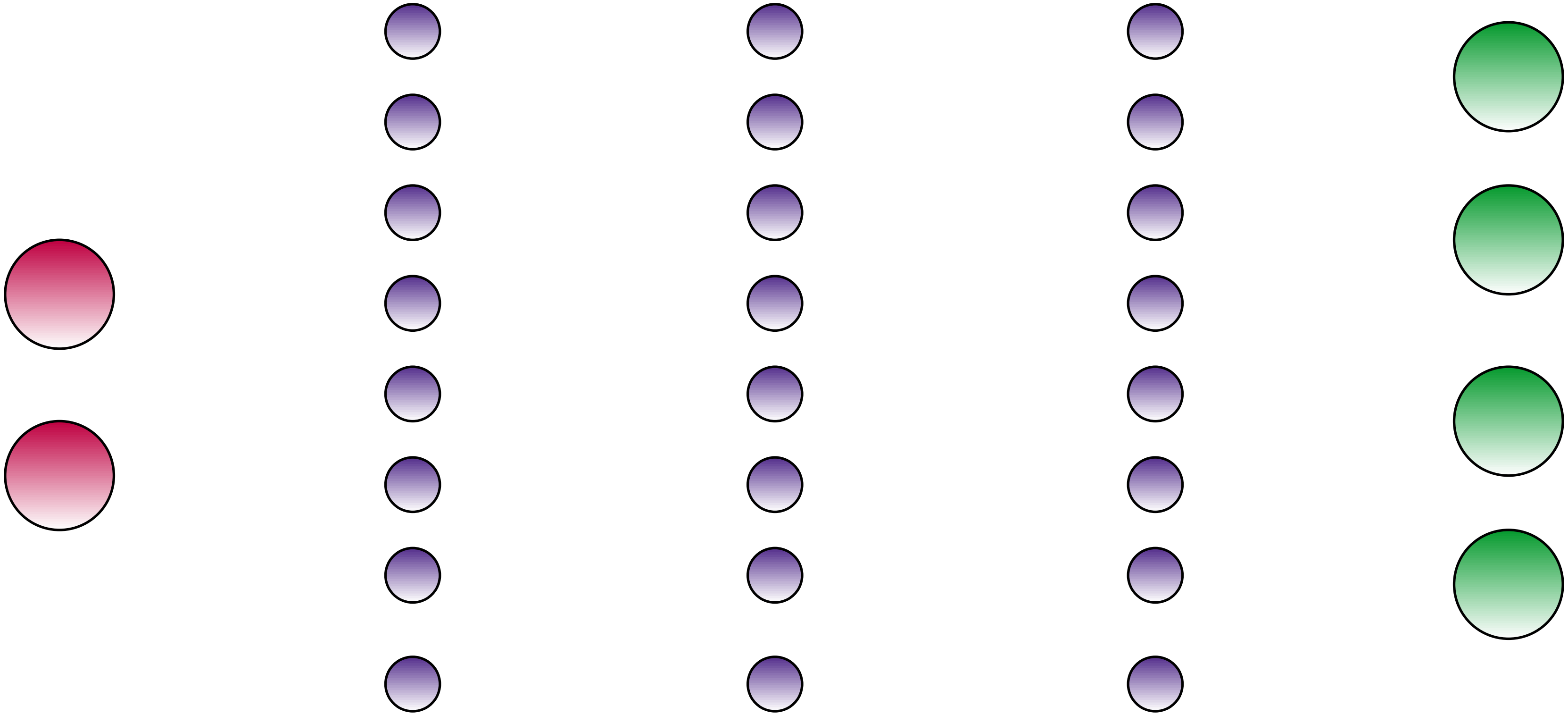
The neural network



The neural network

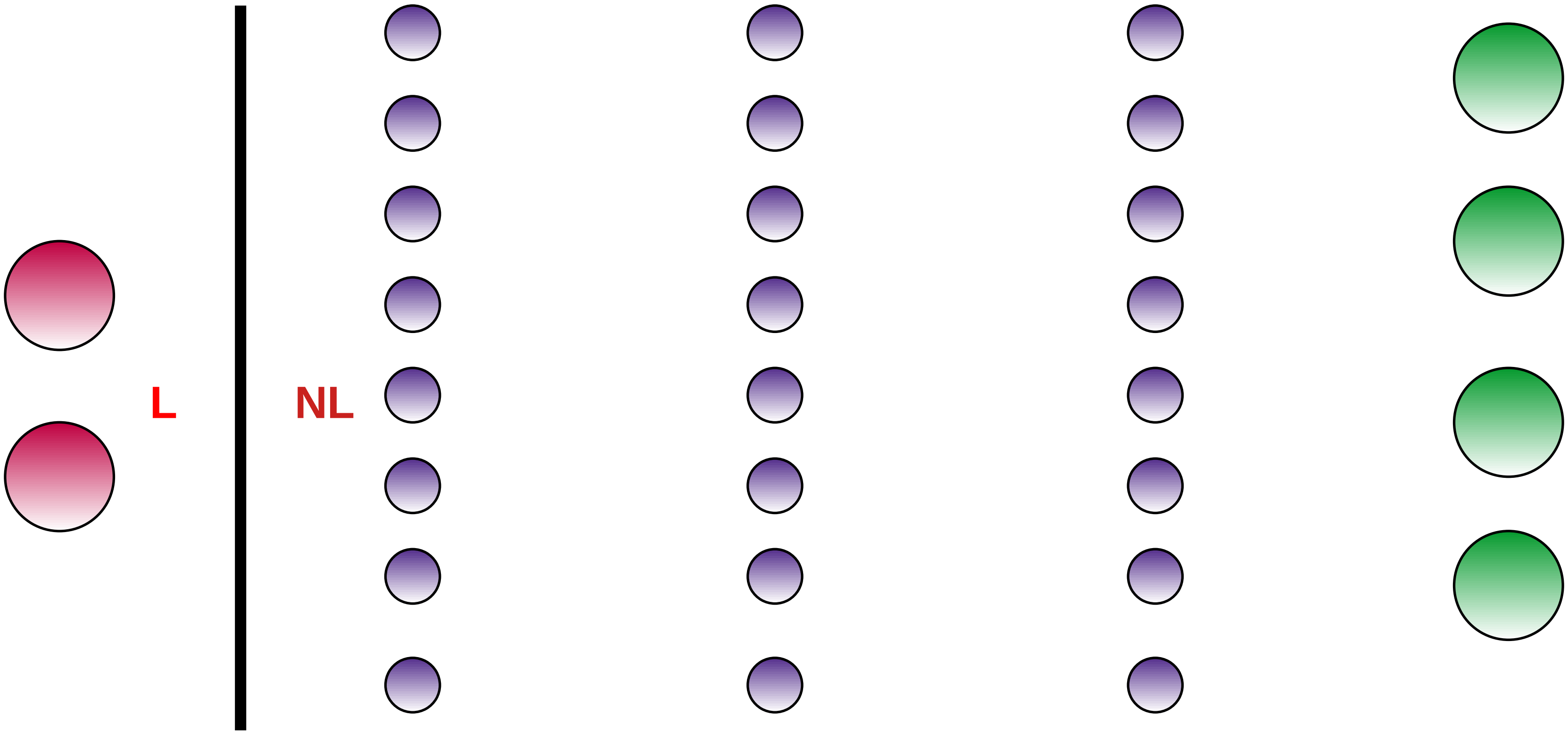


The neural network



i

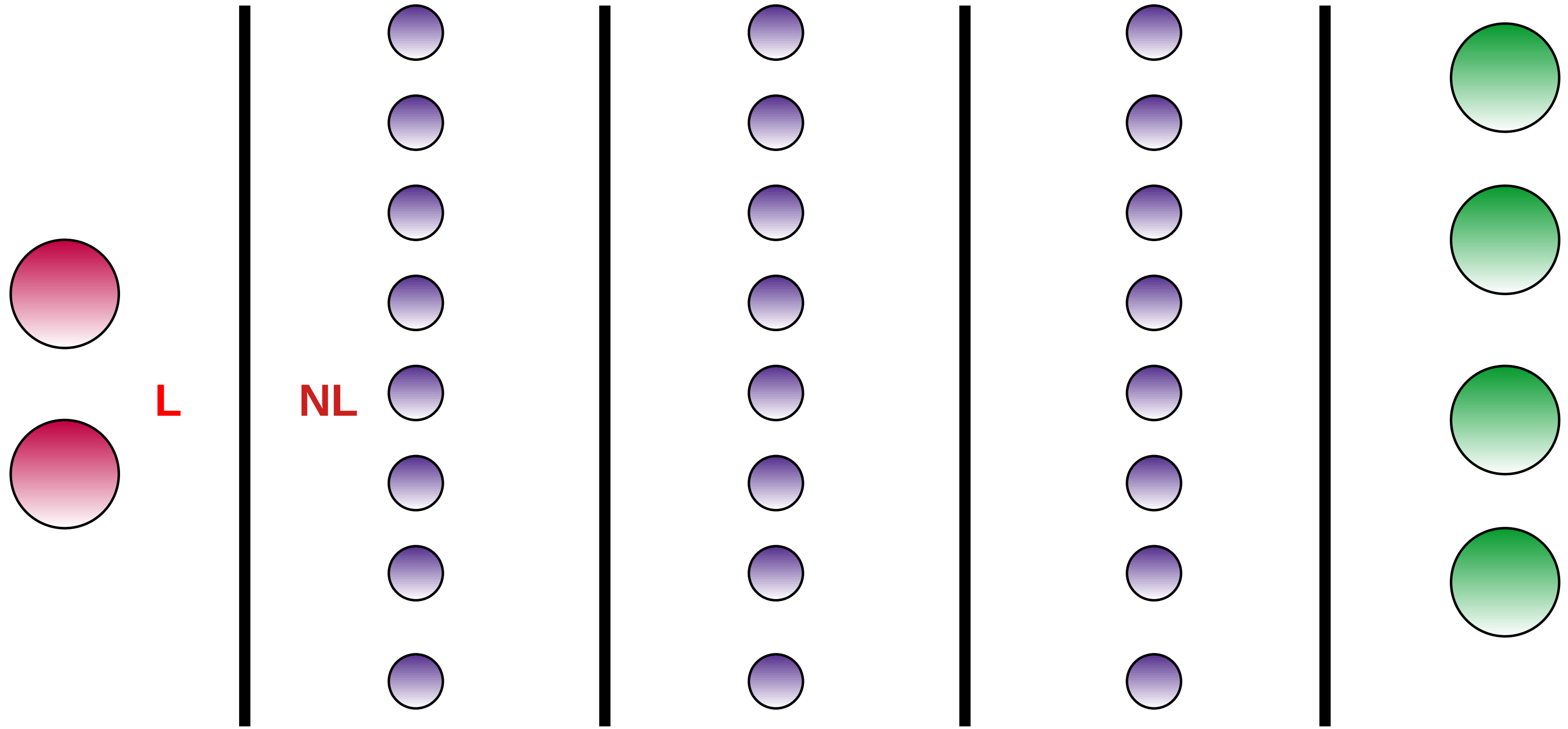
The neural network



i

$l1 = NL(L(i))$

The neural network



i   l1 = NL(L(i))   l2 = NL(L(l1))   l3 = NL(L(l2))   o = NL(L(l3))

Construction of a neural network

To be specified:

1)The number of layers

- Too many: strong risk of over fitting
- Too few: results are not accurate

Construction of a neural network

To be specified:

1)The number of layers

- Too many: strong risk of over fitting
- Too few: results are not accurate

2)The type of layers:

- Fully connected (linear)

Construction of a neural network

To be specified:

1) The number of layers
   - Too many: strong risk of over fitting
   - Too few: results are not accurate
2) The type of layers:
   - Fully connected (linear)

The size of the linear layers:
- Too large: risk of over fitting
- Too small: results are not accurate

Construction of a neural network

To be specified:

1) The number of layers
   - Too many: strong risk of over fitting
   - Too few: results are not accurate
2) The type of layers:
   - Fully connected (linear)

   - Convolutional layers (filter)

The size of the linear layers:
- Too large: risk of over fitting
- Too small: results are not accurate

Construction of a neural network

To be specified:

1) The number of layers
   - Too many: strong risk of over fitting
   - Too few: results are not accurate

2) The type of layers:
   - Fully connected (linear)
   - Convolutional layers (filter)

The size of the linear layers:
- Too large: risk of over fitting
- Too small: results are not accurate

Many options: padding, offset, size ...

Construction of a neural network

To be specified:

1) The number of layers
   - Too many: strong risk of over fitting
   - Too few: results are not accurate
2) The type of layers:
   - Fully connected (linear)

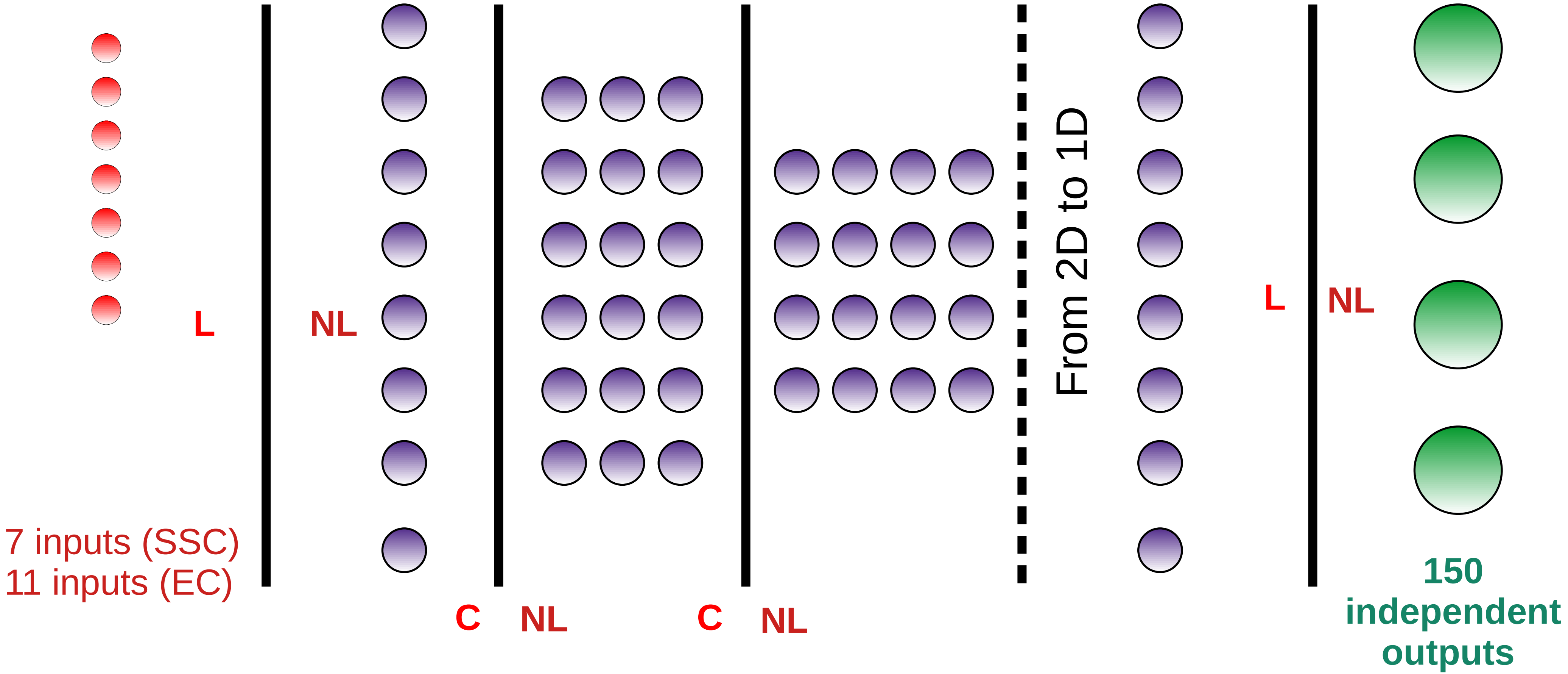
The size of the linear layers:
- Too large: risk of over fitting
- Too small: results are not accurate

   - Convolutional layers (filter)

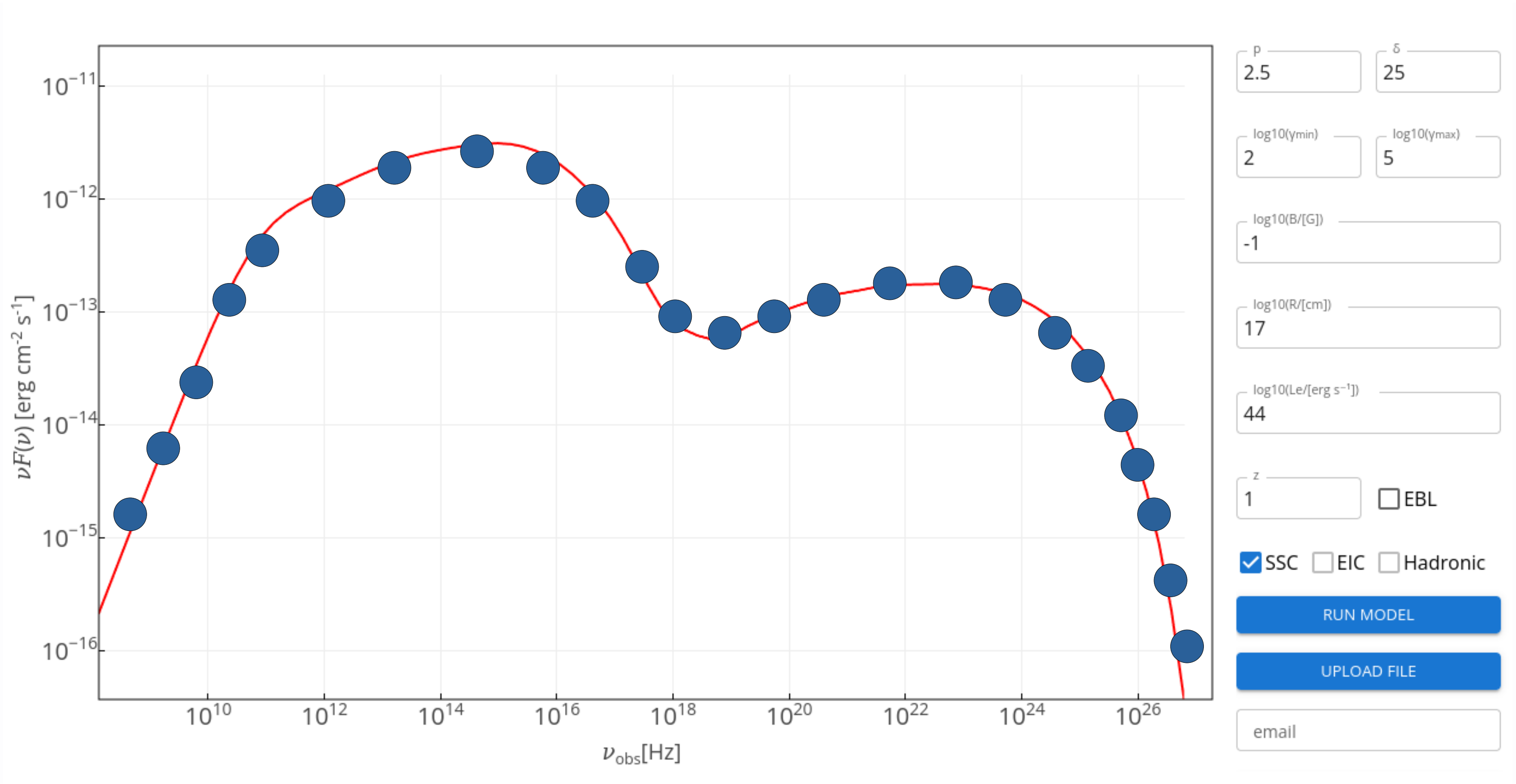
Many options: padding, offset, size ...

3) The type of activation layer (non-linearity):
   - ReLu
   - atan

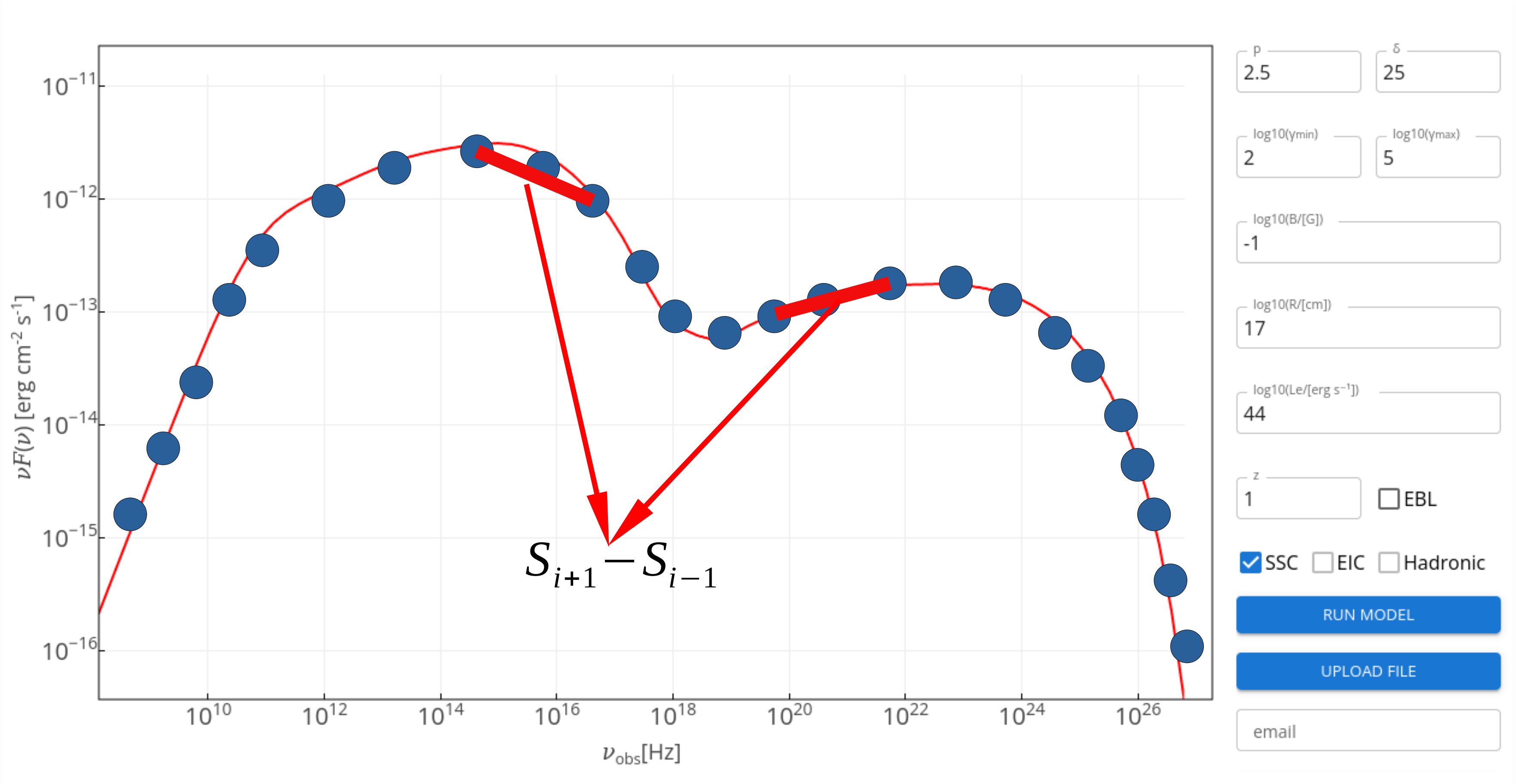
The neural network we are using



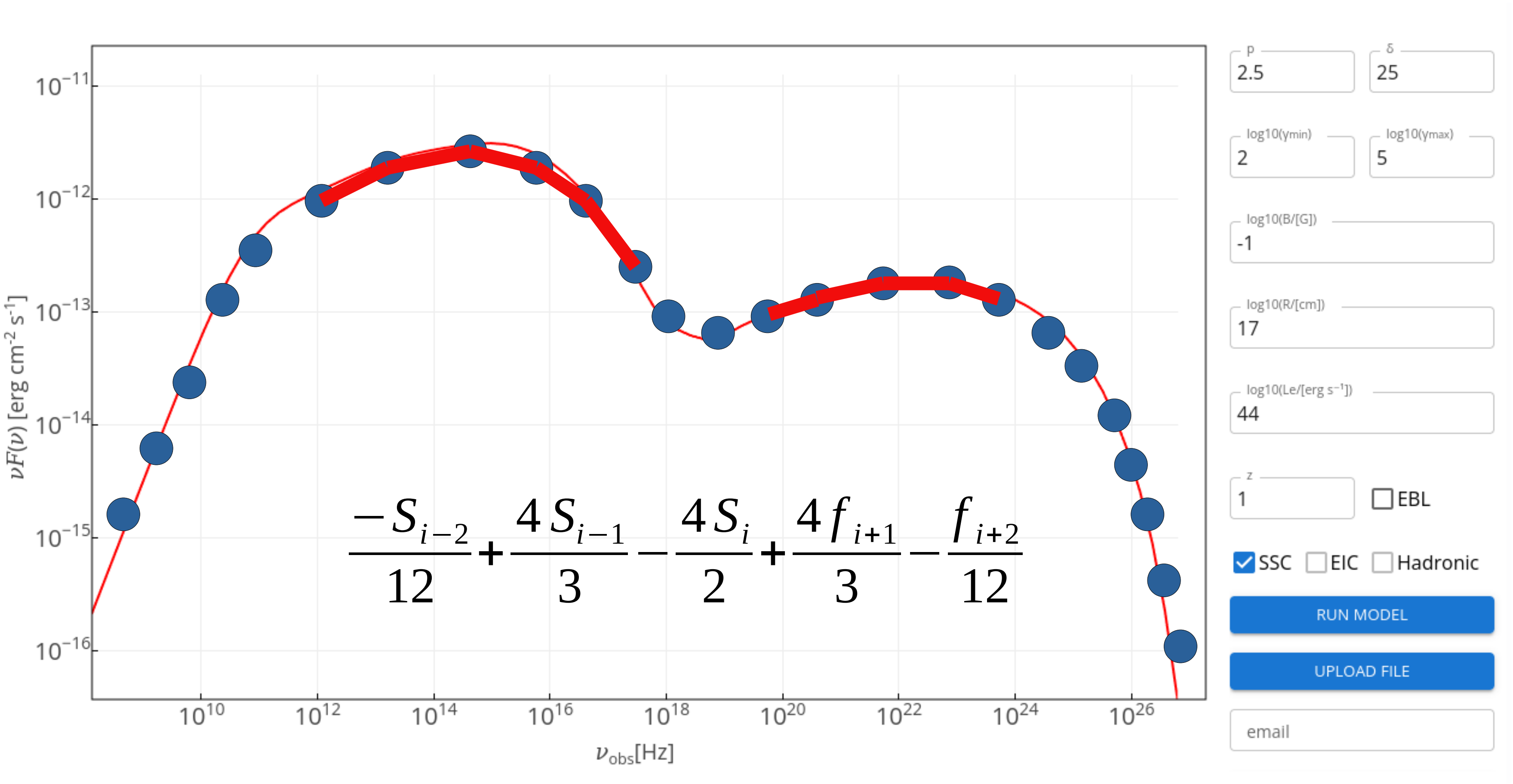
How to produce dependent ouputs?



How to produce dependent ouputs?

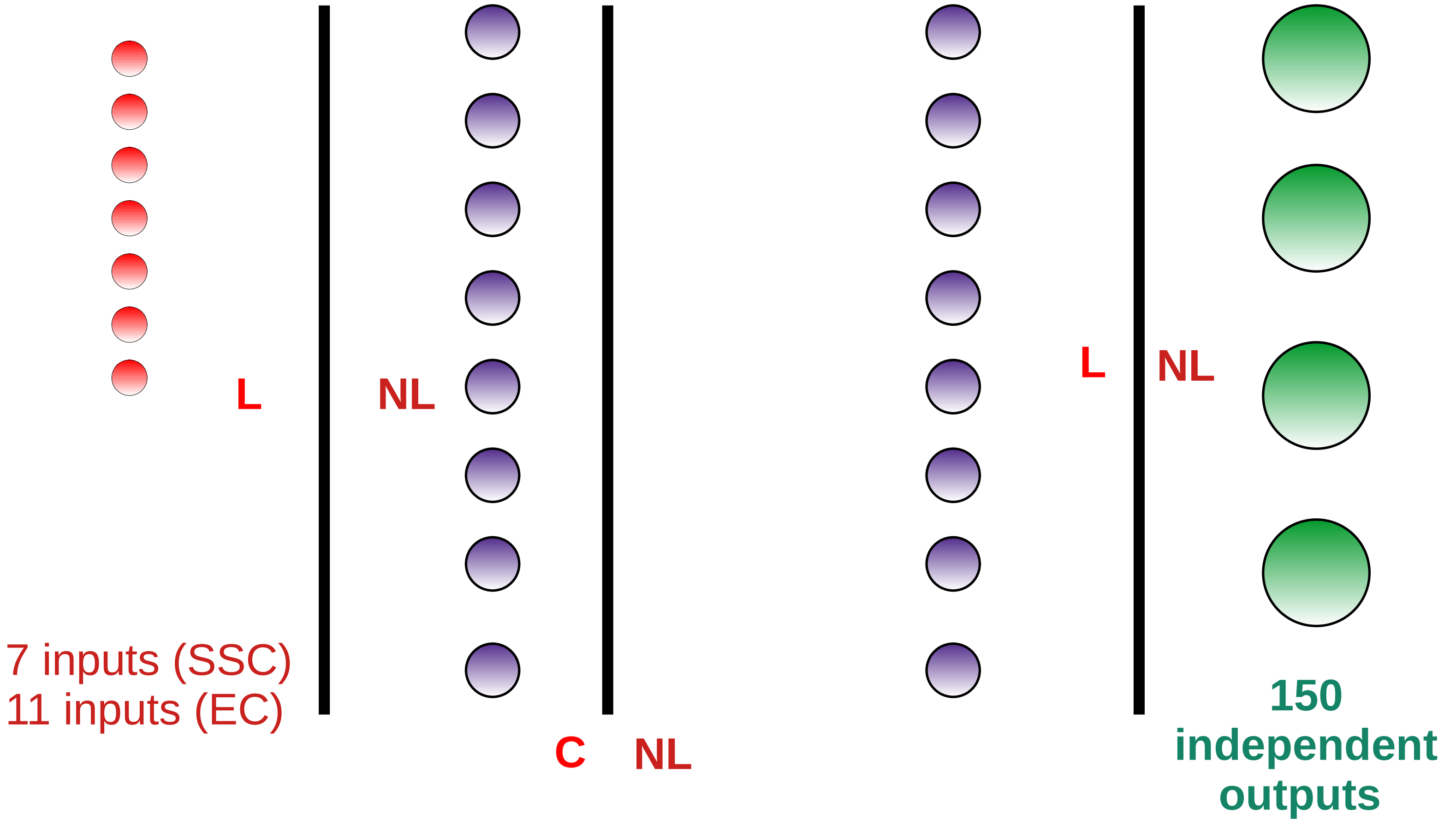


How to produce dependent ouputs?

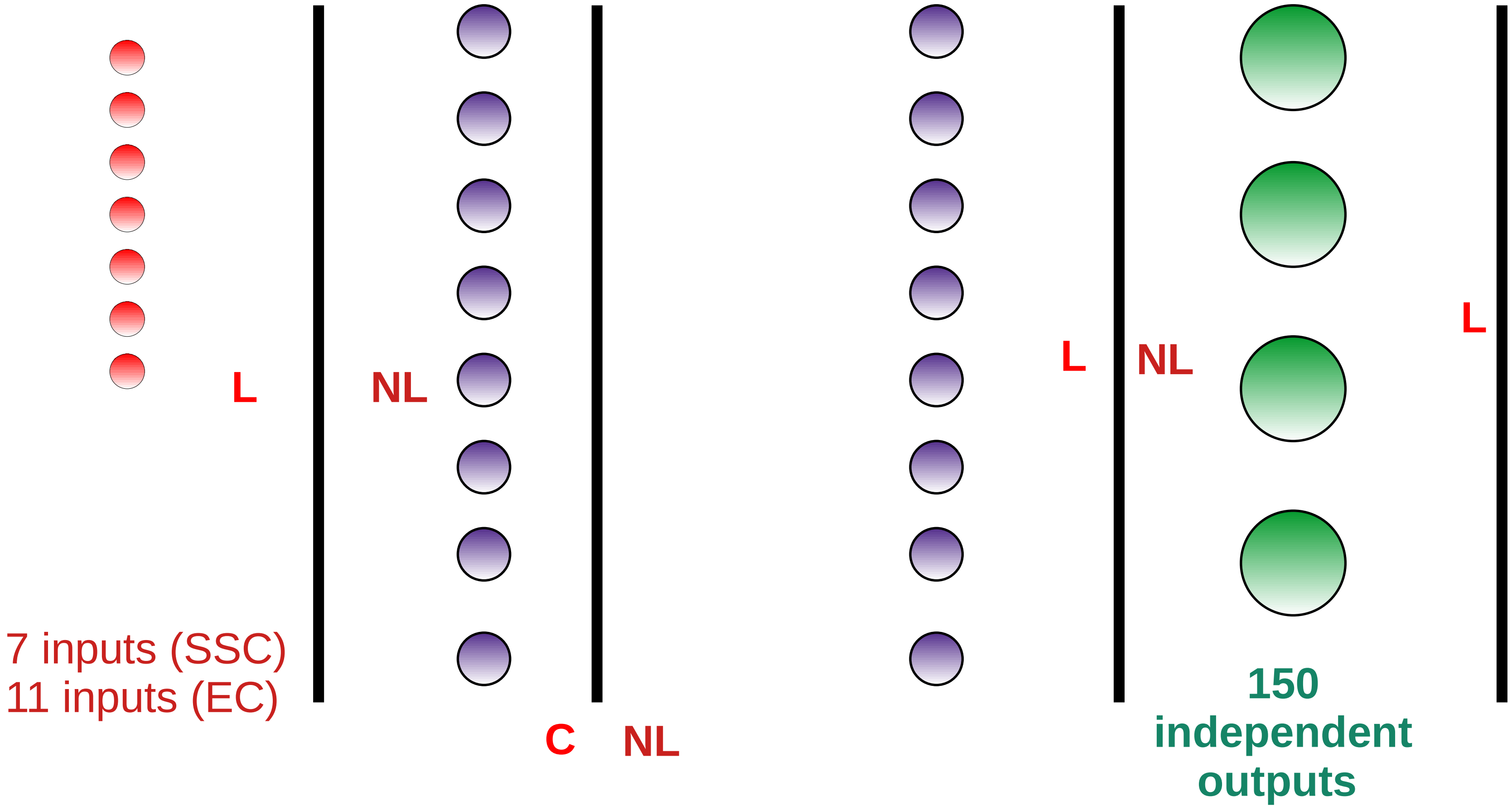


$$\frac{-S_{i-2}}{12}+\frac{4\,S_{i-1}}{3}-\frac{4\,S_i}{2}+\frac{4\,f_{i+1}}{3}-\frac{f_{i+2}}{12}$$

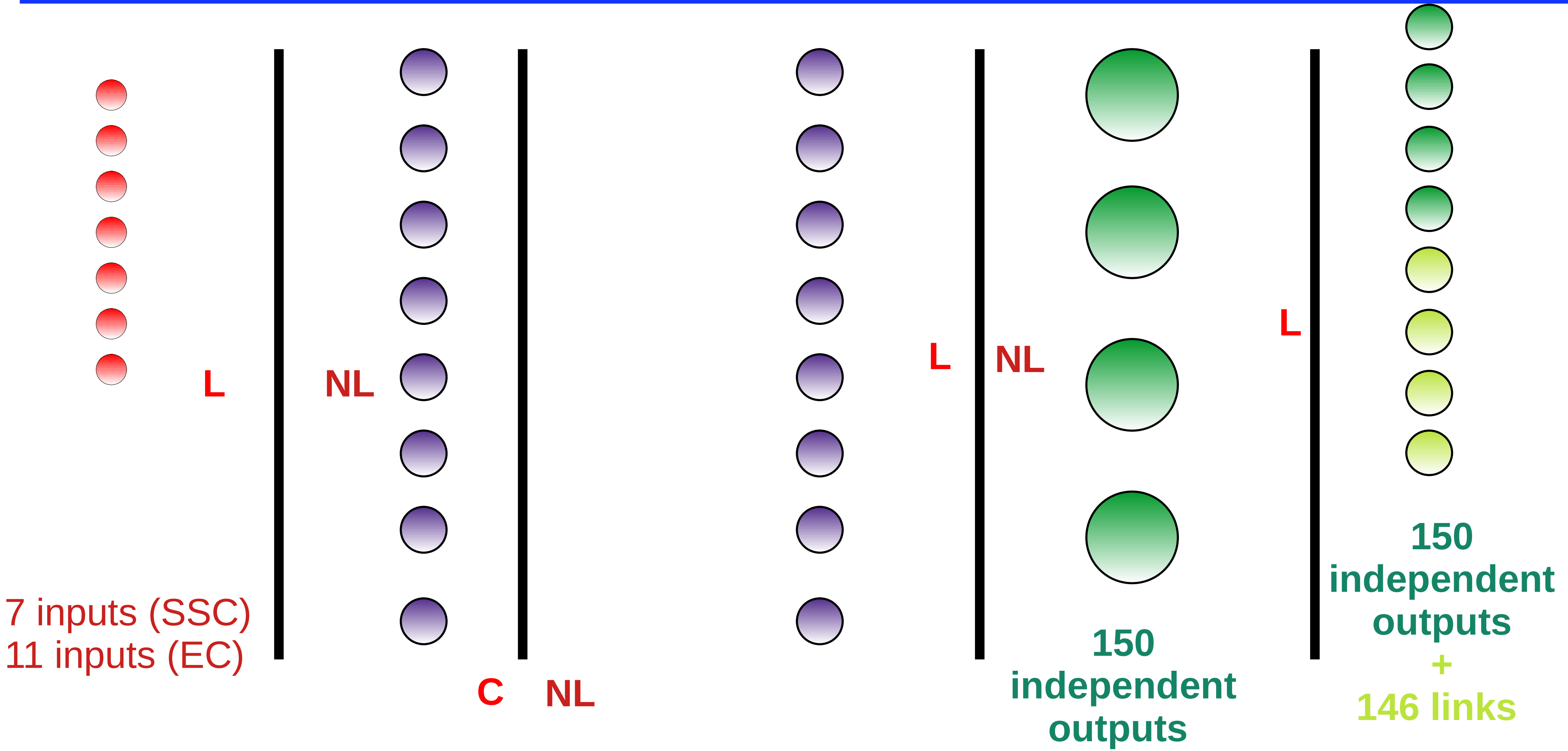
The neural network we are using



The neural network we are using



The neural network we are using



Model implementation and training

Model implementation and training

Model implemented in Pytorch: easy and straightforward.

Model implementation and training

Model implemented in Pytorch: easy and straightforward.

Several trainings:
1)different batch size
2)different training sets
3)different fractions between training set/validation set.

Model implementation and training

Model implemented in Pytorch: easy and straightforward.

Several trainings:
1)different batch size
2)different training sets
3)different fractions between training set/validation set.

Best model chosen using final metrics (R^2,MSE, ME)

Model implementation and training

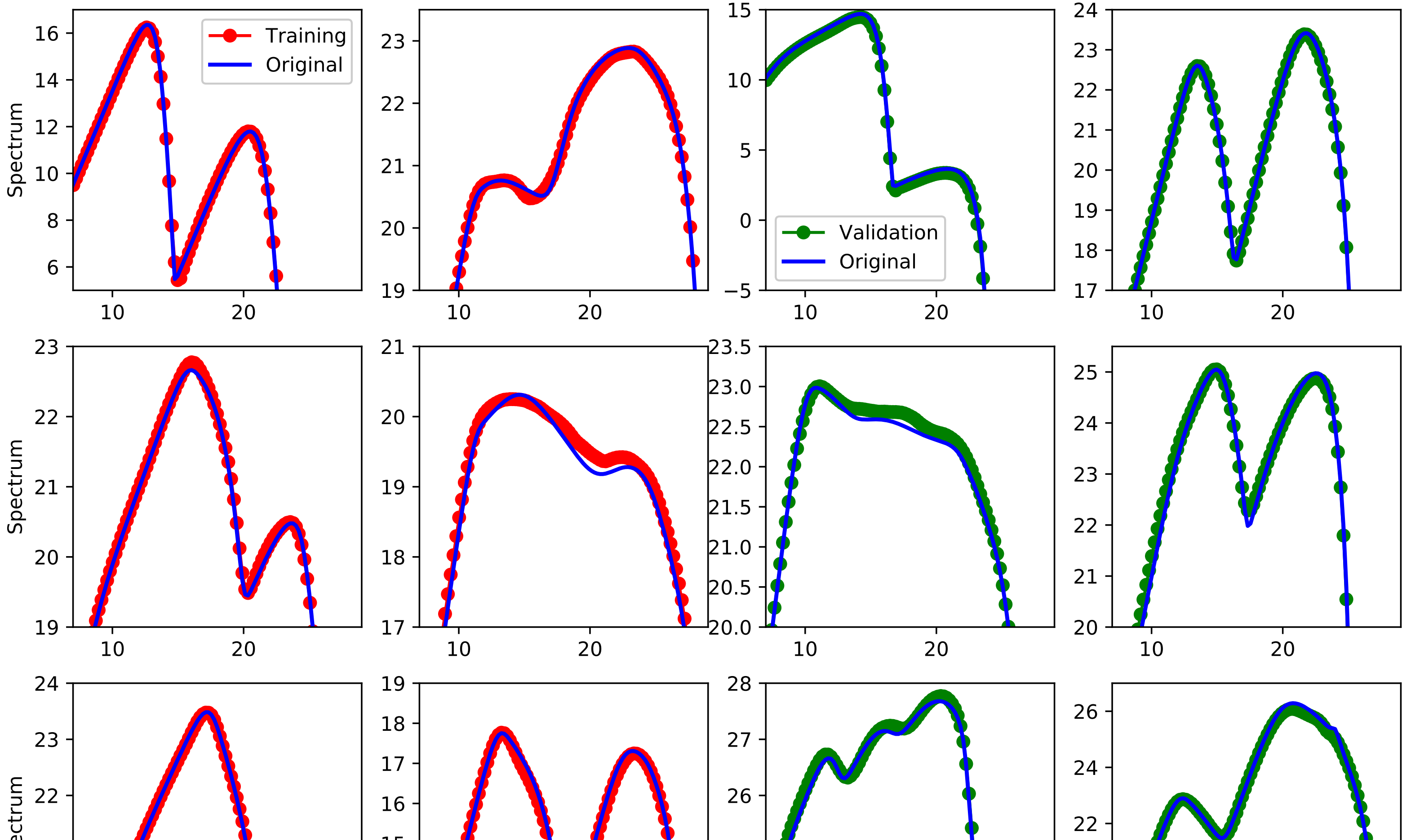
Model implemented in Pytorch: easy and straightforward.

Several trainings:
1)different batch size
2)different training sets
3)different fractions between training set/validation set.

Best model chosen using final metrics (R^2,MSE, ME)

Large compute load. Done on 4 A100 GPUs,
2 training setup at a time per GPUs

Final result



Summary

We **created** and **trained** a convolutional neural network which computes the spectrum from an input parameter set.

Summary

We **created** and **trained** a convolutional neural network which computes the spectrum from an input parameter set.

It required to:

1) Efficiently sampled the parameter space.

Summary

We **created** and **trained** a convolutional neural network which computes the spectrum from an input parameter set.

It required to:

1) Efficiently sampled the parameter space.
2) Compute a large number of spectra.

Summary

We **created** and **trained** a convolutional neural network which computes the spectrum from an input parameter set.

It required to:

1) Efficiently sampled the parameter space.
2) Compute a large number of spectra.
3) Solve the oscillation problem in the resulting spectra, by training on the derivative.

Summary

We **created** and **trained** a convolutional neural network which computes the spectrum from an input parameter set.

It required to:

1) Efficiently sampled the parameter space.
2) Compute a large number of spectra.
3) Solve the oscillation problem in the resulting spectra, by training on the derivative.

As of today, we have two models: synchrotron self-Compton and external Compton.

So much effort .... for what ?

We have a black box which compute a spectrum from a parameter set in ~1 ms.

So much effort .... for what ?

We have a black box which compute a spectrum from a parameter set in ~1 ms.

Put this black box in your favorite fitting engine and have fun.

So much effort .... for what ?

We have a black box which compute a spectrum from a parameter set in ~1 ms.

Put this black box in your favorite fitting engine and have fun.

Additional possibilities: restrictive models

So much effort .... for what ?

We have a black box which compute a spectrum from a parameter set in ~1 ms.

Put this black box in your favorite fitting engine and have fun.

Additional possibilities: restrictive models

- Link parameters $t_{var} = \dfrac{R}{c\gamma}$

So much effort .... for what ?

We have a black box which compute a spectrum from a parameter set in ~1 ms.

Put this black box in your favorite fitting engine and have fun.

Additional possibilities: restrictive models

- Link parameters $t_{var} = \frac{R}{c\gamma}$
- Set parameters

Perspectives

1) Improvement of the SSC model (larger parameter set).

Perspectives

1) Improvement of the SSC model (larger parameter set).

2) More flexible EC model (for the external field).

Perspectives

1) Improvement of the SSC model (larger parameter set).

2) More flexible EC model (for the external field).

3) Hadronic models in association with neutrino production (with and without external field).

Perspectives

1) Improvement of the SSC model (larger parameter set).

2) More flexible EC model (for the external field).

3) Hadronic models in association with neutrino production (with and without external field).

4) Time dependent models for flares.

Perspectives

1) Improvement of the SSC model (larger parameter set).

2) More flexible EC model (for the external field).

3) Hadronic models in association with neutrino production (with and without external field).

4) Time dependent models for flares.

5) New type of CNN to derive the posterior distribution directly from the data (remove entirely the fit procedure).

Perspectives

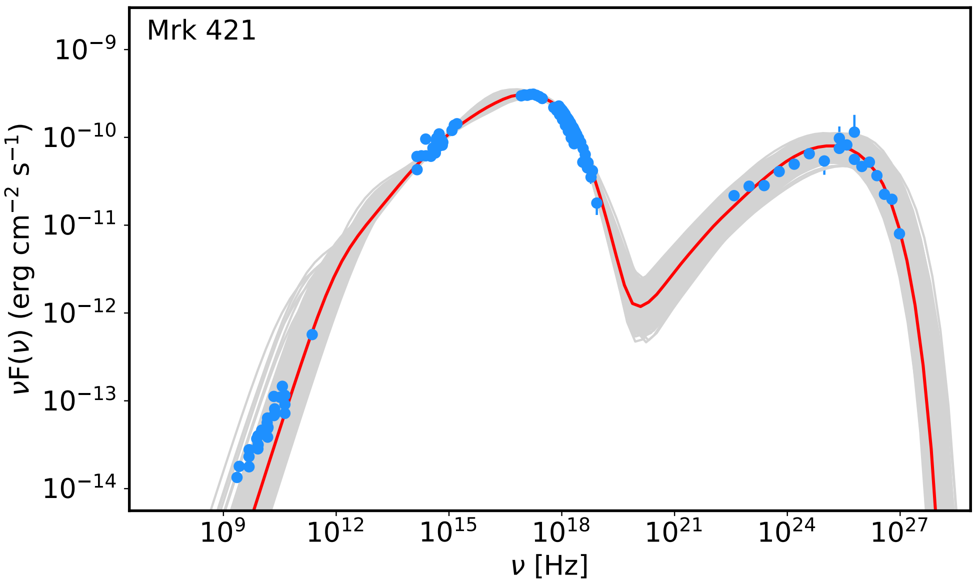
1) Improvement of the SSC model (larger parameter set).

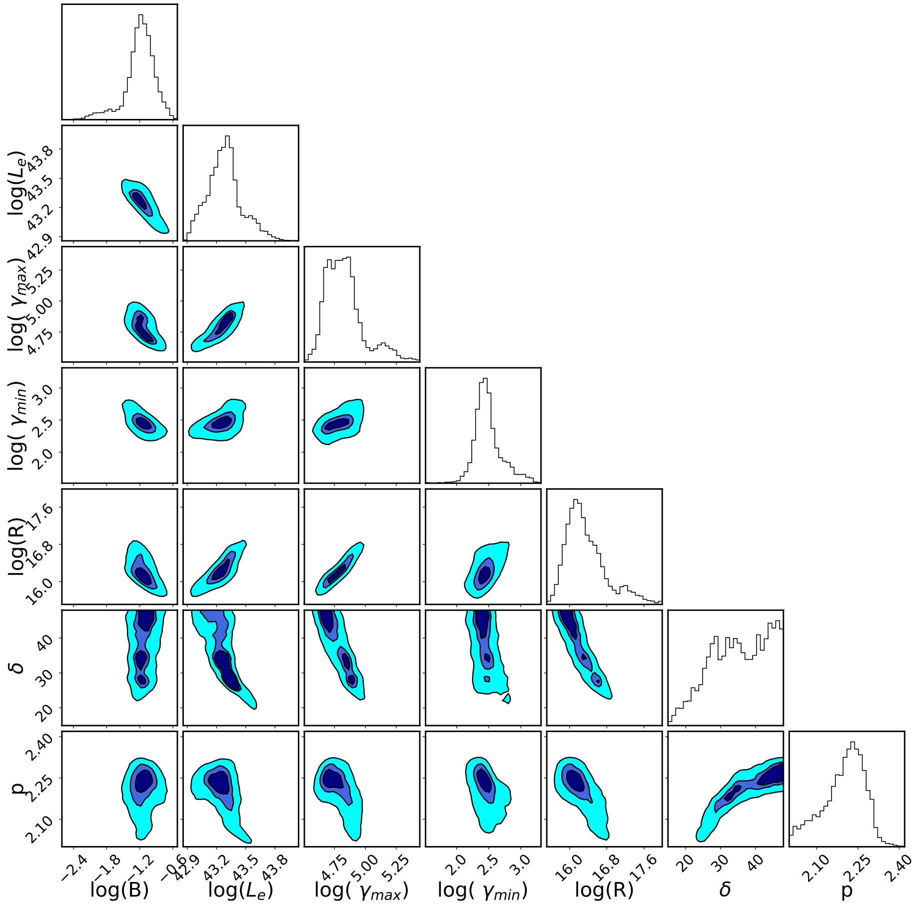
2) More flexible EC model (for the external field).

3) Hadronic models in association with neutrino production (with and without external field).

4) Time dependent models for flares.

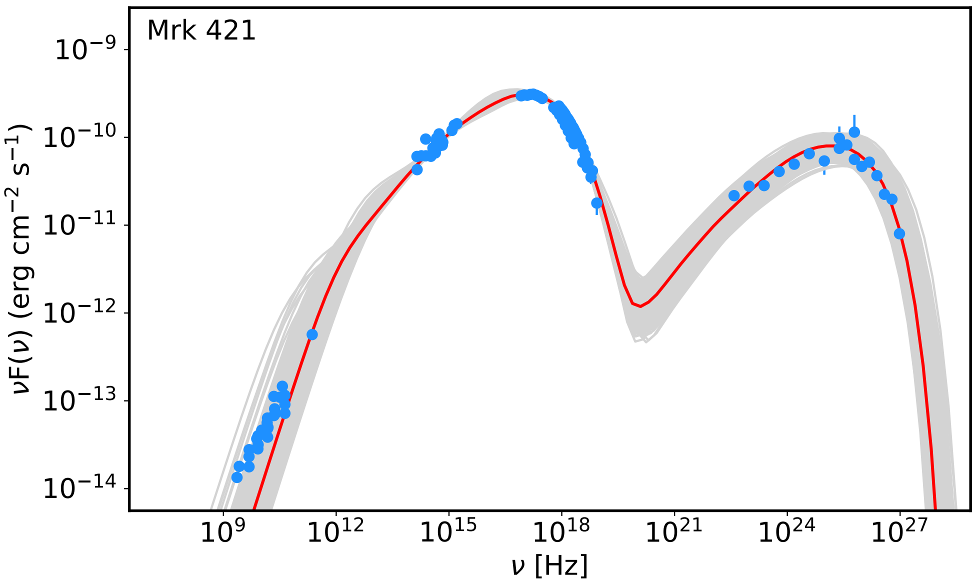
5) New type of CNN to derive the posterior distribution directly from the data (remove entirely the fit procedure).
   - Requires many fit results to train the network.... will be feasible when we will have done many many fits.

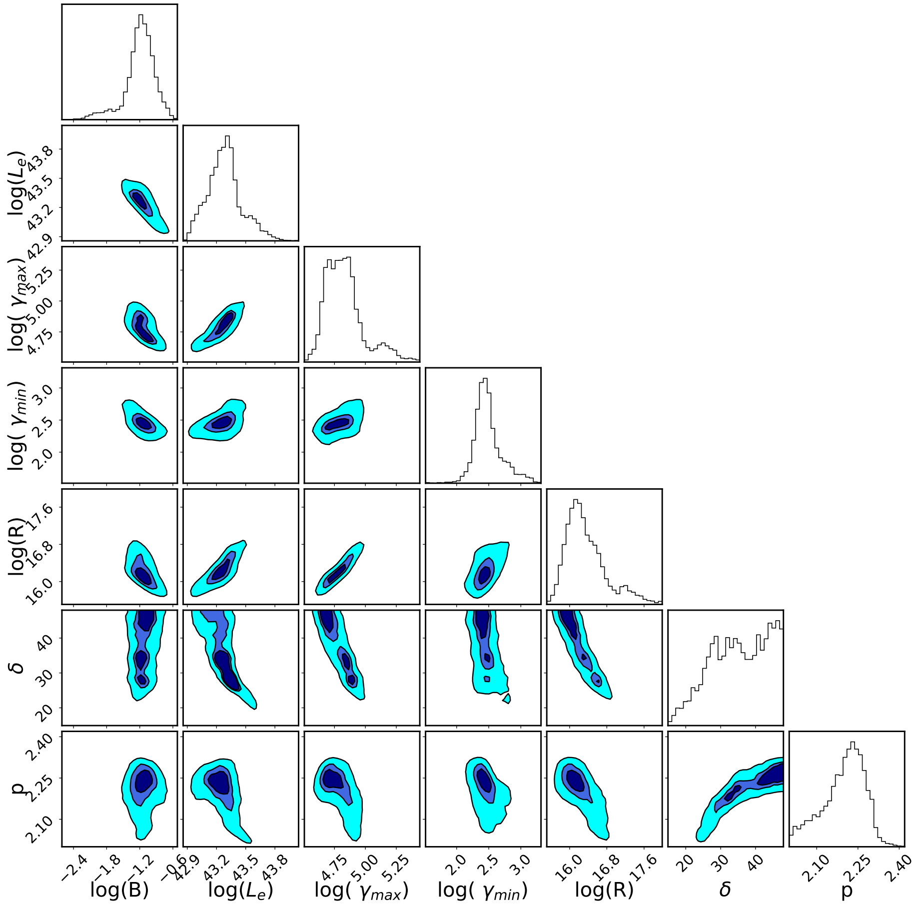




See Husne's and Narek's talks

Thank you





See Husne’s and Narek’s talks