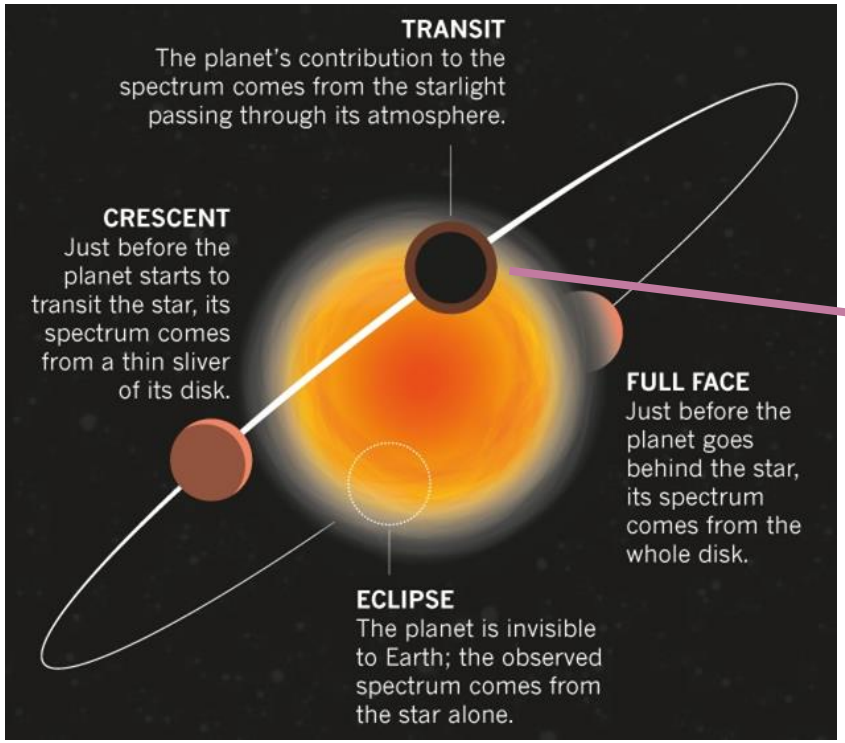
Hands-on Session: Neural Networks

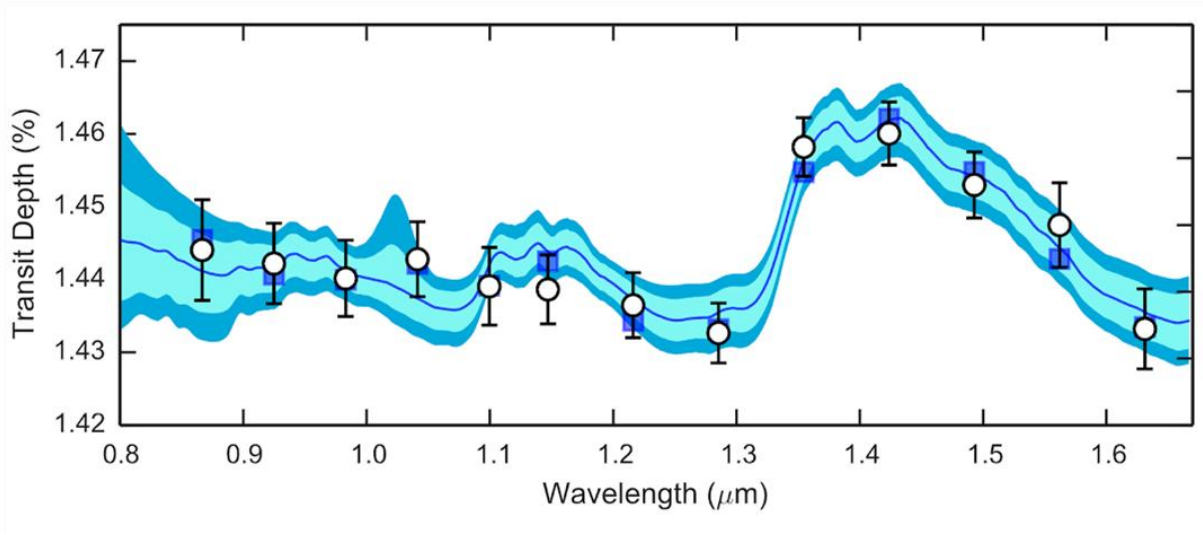
Francisco Ardevol Martinez and Till Käufer

CHAMELEON School 1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 860470.

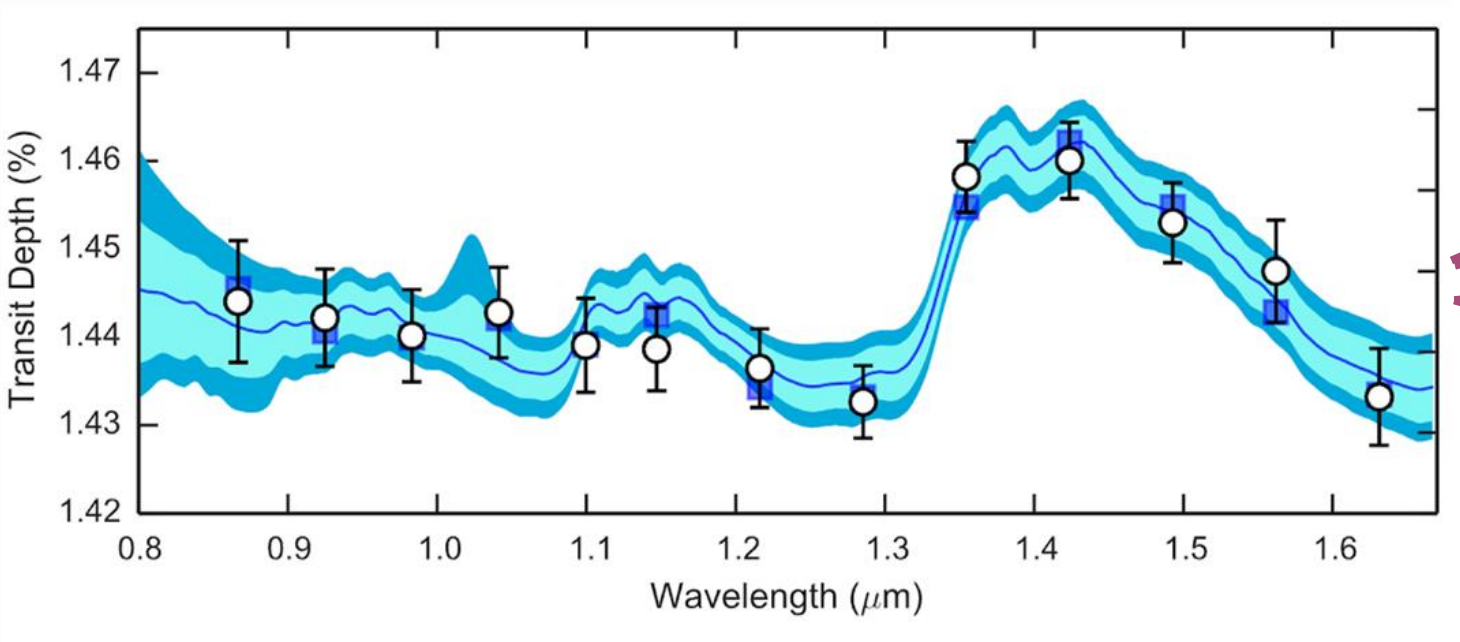
Extracting exoplanet parameters using spectra





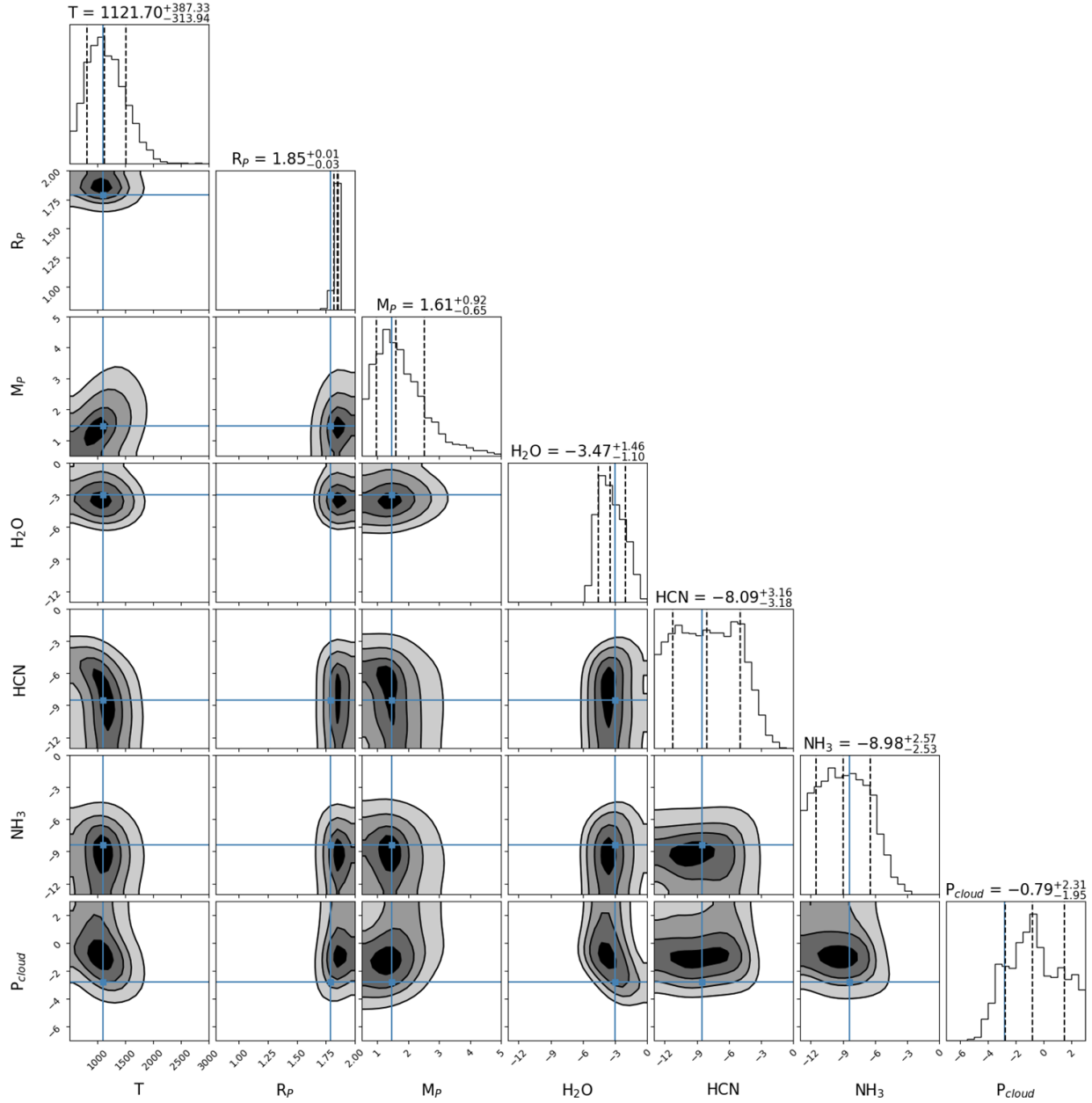
Retrieval

- Estimating atmospheric parameters from observations ('inverse modeling').
- Traditionally (nested sampling) requires 10^4-10^6 forward model computations.

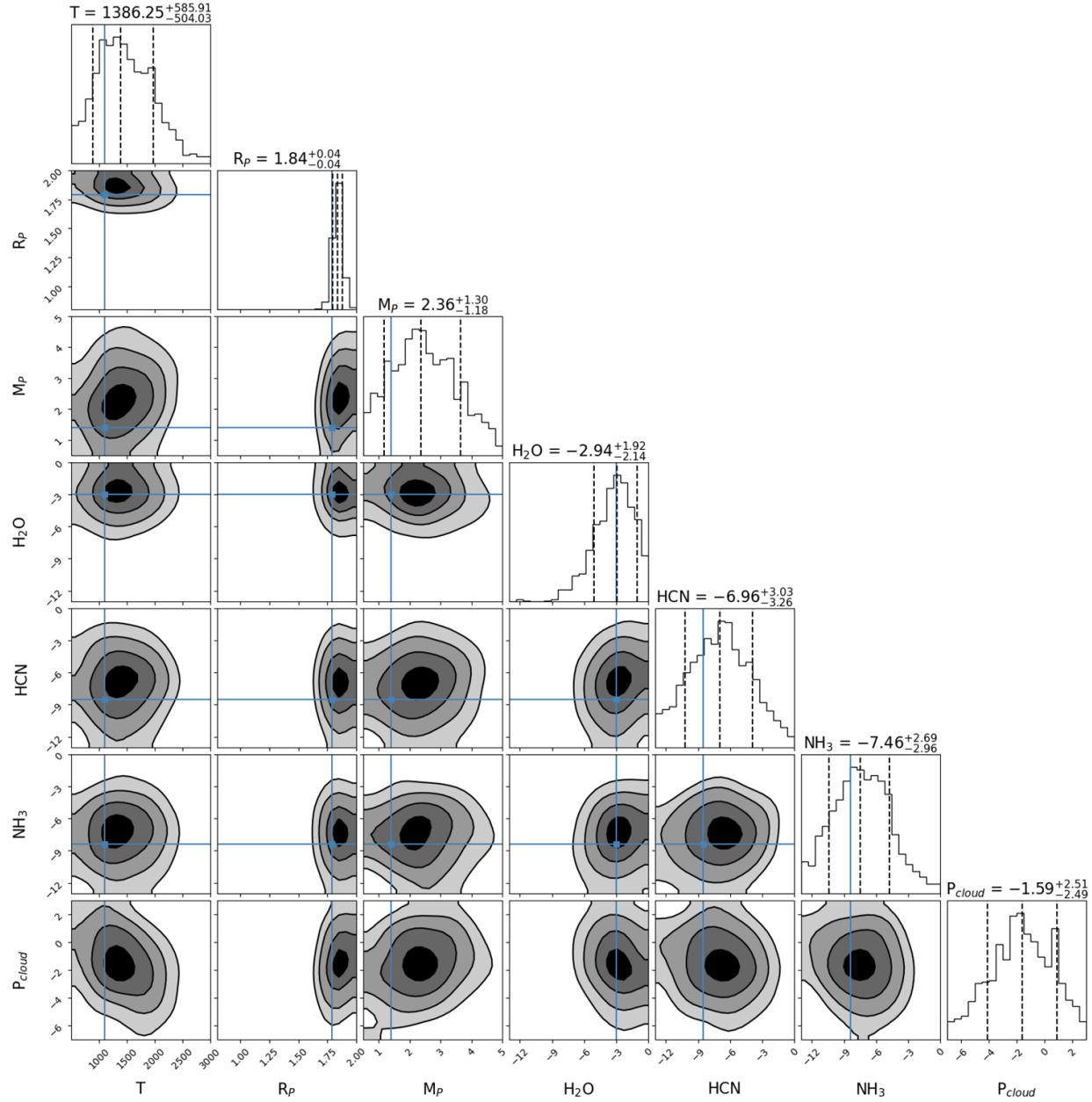


- T
- $\log(H_2O)$
- $\log(HCN)$
- $\log(NH_3)$
- κ

Nested Sampling (ARCiS)



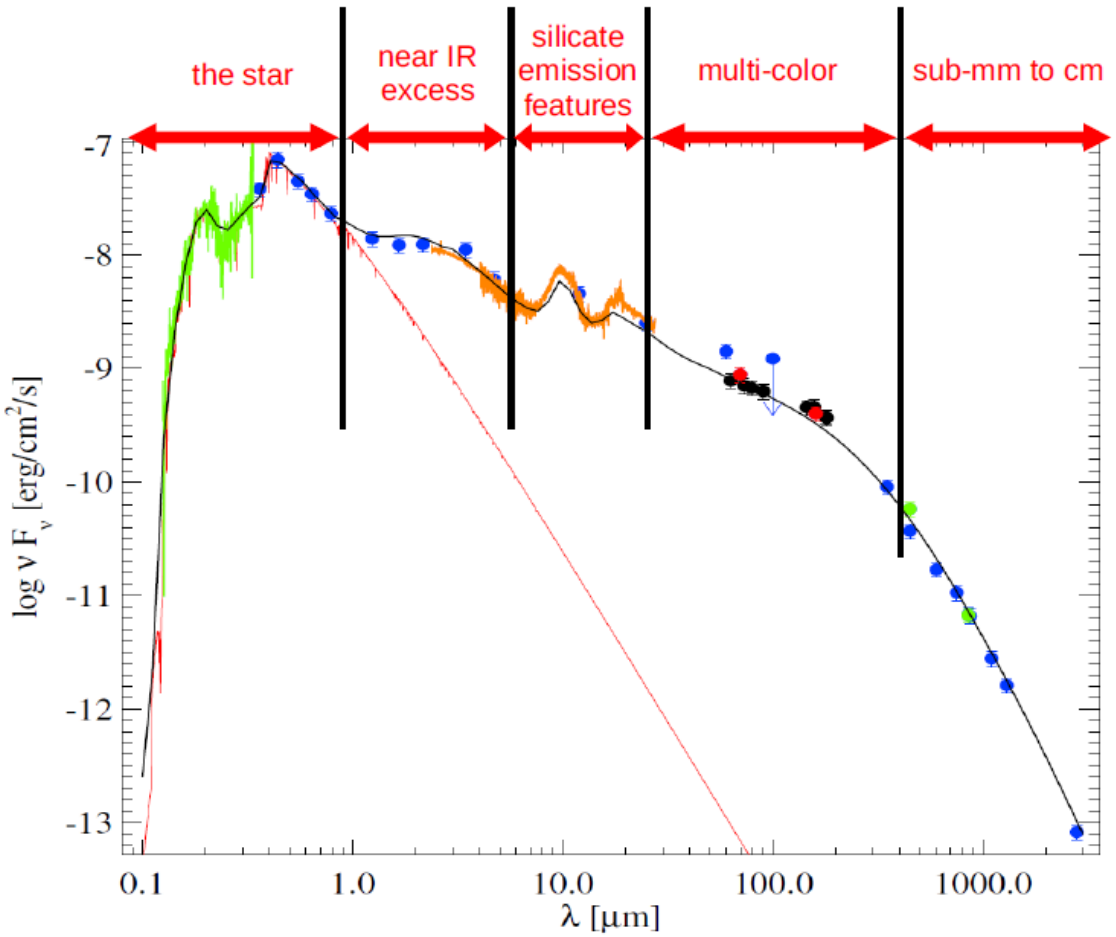
Neural Network



Emulating SED modelling of protoplanetary disks

Mstar [Msun] =	0.7500
Teff [K] =	4000.00
Lstar [Lsun] =	0.2420
Rstar [Rsun] =	1.0888
log(g) =	4.2389
⋮	
---------- inner zone ----------	
gas mass [Msun] =	7.7168E-08
dust mass [Msun] =	7.7168E-10
inner radius [AU] =	0.0716
outer radius [AU] =	4.0096
col.dens.power-index =	-0.7709
scale height @ 1AU [AU] =	0.0340
flaring index =	1.4653
minimum dust radius [mic] =	0.10904
maximum dust radius [mic] =	6648.1
size-dist. power-index =	3.9935
fPAH =	n.a.
---------- outer zone ----------	
disk gas mass [Msun] =	8.8867E-03
disk dust mass [Msun] =	8.8867E-05
inner radius [AU] =	4.0096
tapering-off radius [AU] =	40.5000
outer radius [AU] =	220.0000
col.dens.power-index =	1.5000
tapering-off power-index =	0.7600
scale height @ 100AU [AU] =	7.0647
flaring index =	1.2756
minimum dust radius [mic] =	0.10904
maximum dust radius [mic] =	6648.1
size-dist. power-index =	3.9935
fPAH =	n.a.

Modelling

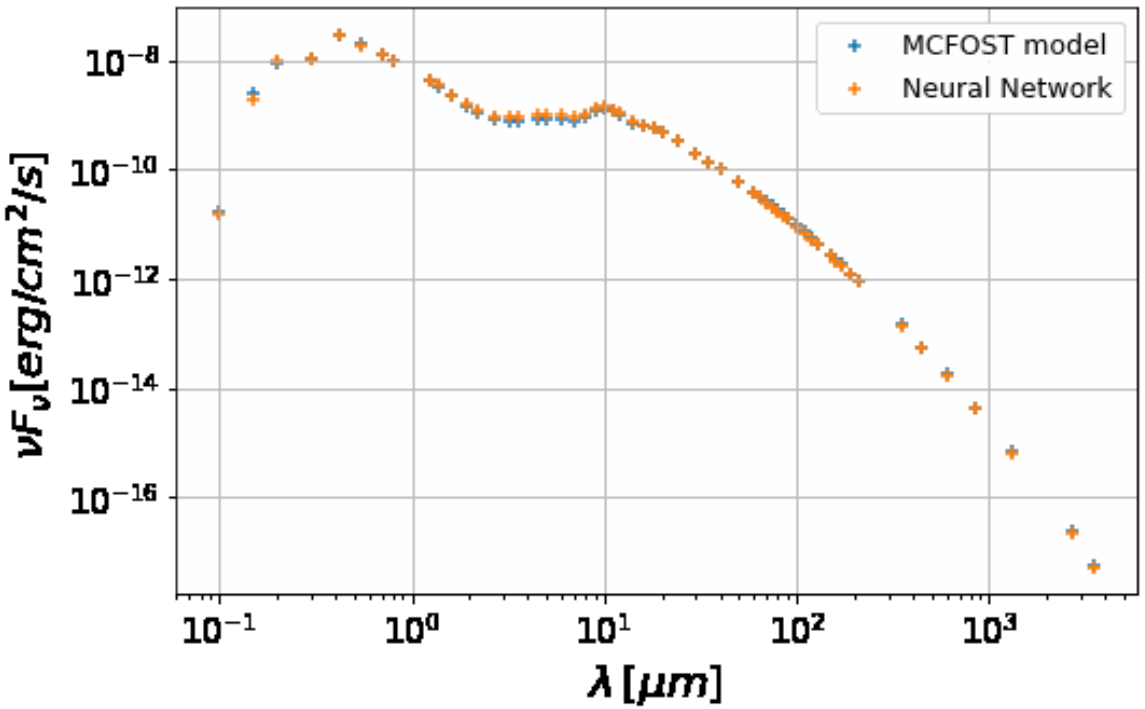


Tilling et al. 2012, adopted by Woitke 2015

Table 1: Parameters for the model grid and values assumed.

stellar parameter		
$M_\star$	stellar mass $[M_\odot]$	0.5, 1.0, 1.5, 2.0, 2.5
age	age [Myr]	1, 3, 10, 100
L_*	dependent parameter	from evolutionary tracks
$T_{\rm eff}$	dependent parameter	from evolutionary tracks
$f_{\rm UV}$	excess UV $f_{\rm UV} = L_{UV}/L_\star$	0.001, 0.1
disc parameter		
$M_{\rm d}$	disc dust mass $[M_\odot]$	10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}
$\rho_{\rm d}/\rho_{\rm g}$	dust/gas mass ratio δ	0.001, 0.01, 0.1, 1, 10
$R_{\rm in}$	inner disc radius $[R_{\rm subli}]$	1, 10, 100
$R_{\rm out}$	outer disc radius [AU]	100, 300, 500
ϵ	column density $N_{\rm H}(r) \propto r^{-\epsilon}$	0.5, 1.0, 1.5
β	flaring $H(r) = H_0(\frac{r}{r_0})^\beta$	0.8, 1.0, 1.2
r_0	reference radius [AU]	100
H_0	scale height at r_0 [AU]	10
$\chi_{\rm ISM}$	strength of incident ISM UV	1
$\zeta_{\rm CR}$	cosmic ray H_2 ionization rate $[{\rm s}^{-1}]$	5×10^{-17}
$f_{\rm PAH}$	abundance of PAHs relative to ISM	0
α	viscosity parameter	0
dust parameter		
s	settling $H(r,a) \propto H(r)\,(\frac{a}{0.05\,\mu{\rm m}})^{-s/2}$	0, 0.5
$a_{\rm min}$	minimum grain size $[\mu m]$	0.05, 1
$a_{\rm max}$	maximum grain size $[\mu m]$	1000
$\rho_{\rm gr}$	grain material density $[{\rm g/cm}^3]$	3.5
radiative transfer parameter		
i	inclination	$0^o, 41.41^o, 60^o, 75.52^o$, 90^o (edge-on)
$v_{\rm turb}$	turbulent line width [km/s]	0.15

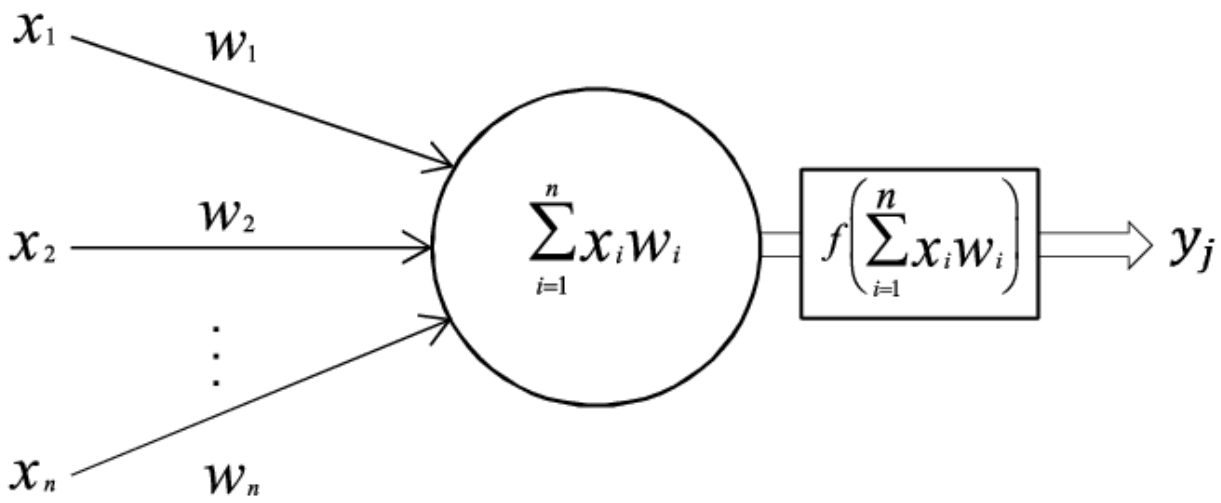
Machine Learning



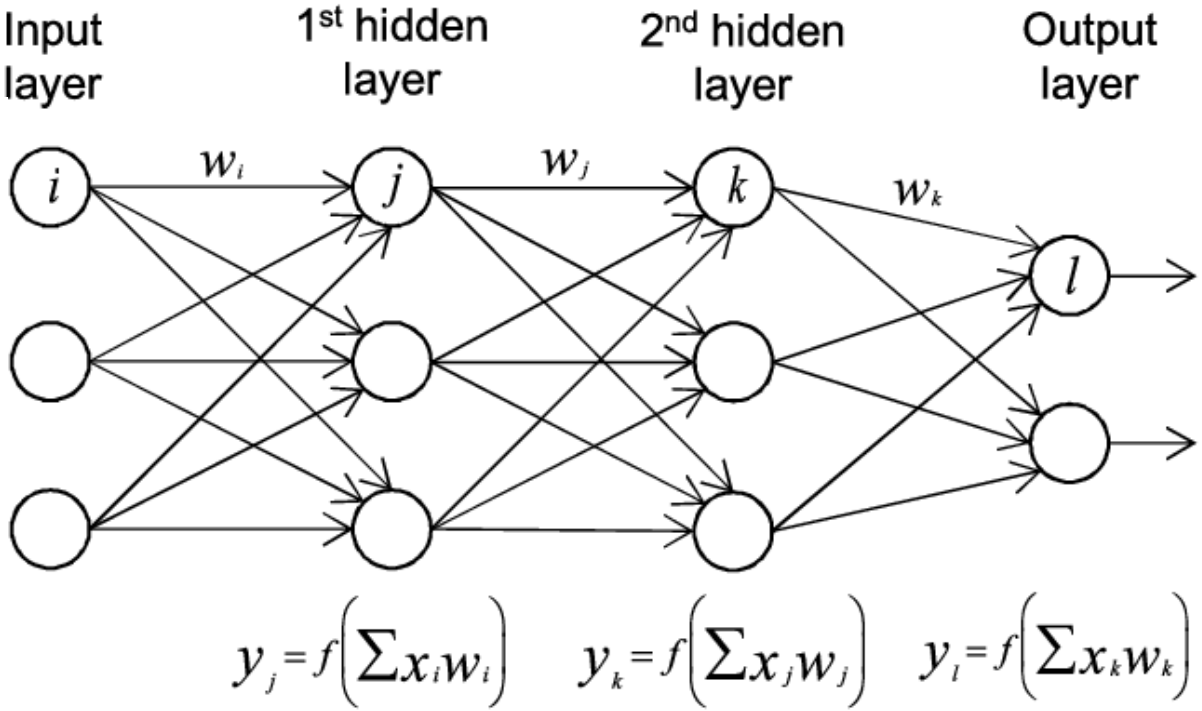
Kamp et al. 2011

Theory in a nutshell

(a)



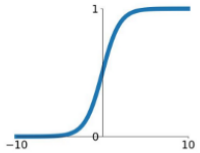
(b)



Activation Functions

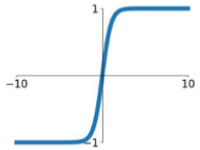
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



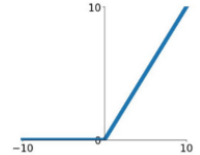
tanh

$\tanh(x)$



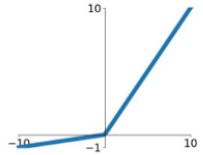
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

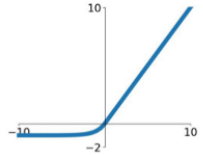


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



General Process

Training Data
Input and Output

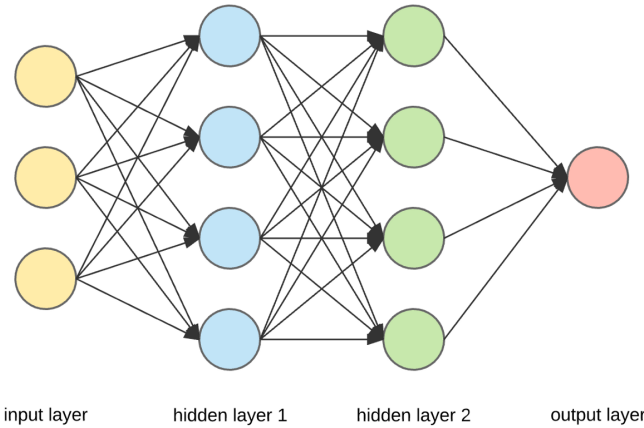
Scaling

Training

Machine learning

Input

Output



Test Data
Input and Output

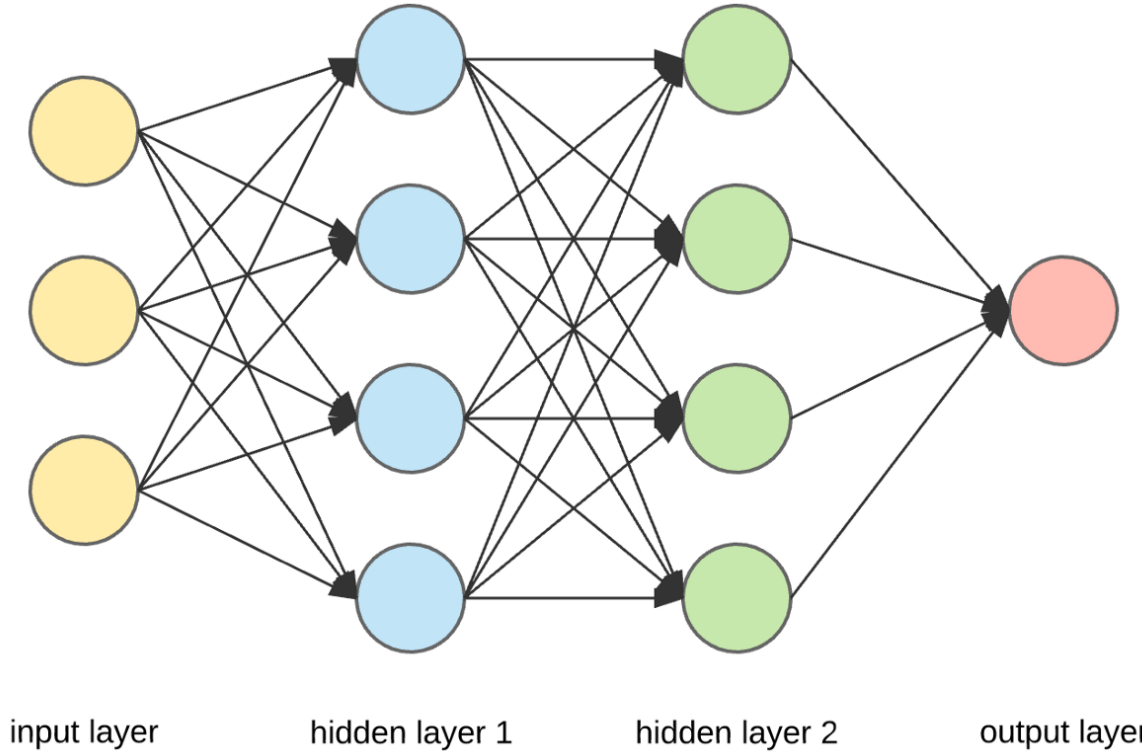
Scaling

Predcition

Inverse Scaling

Comparison

How to build a NN




```
from tensorflow.keras.layers import Dense, Input, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD, Adam
```

#Input layer

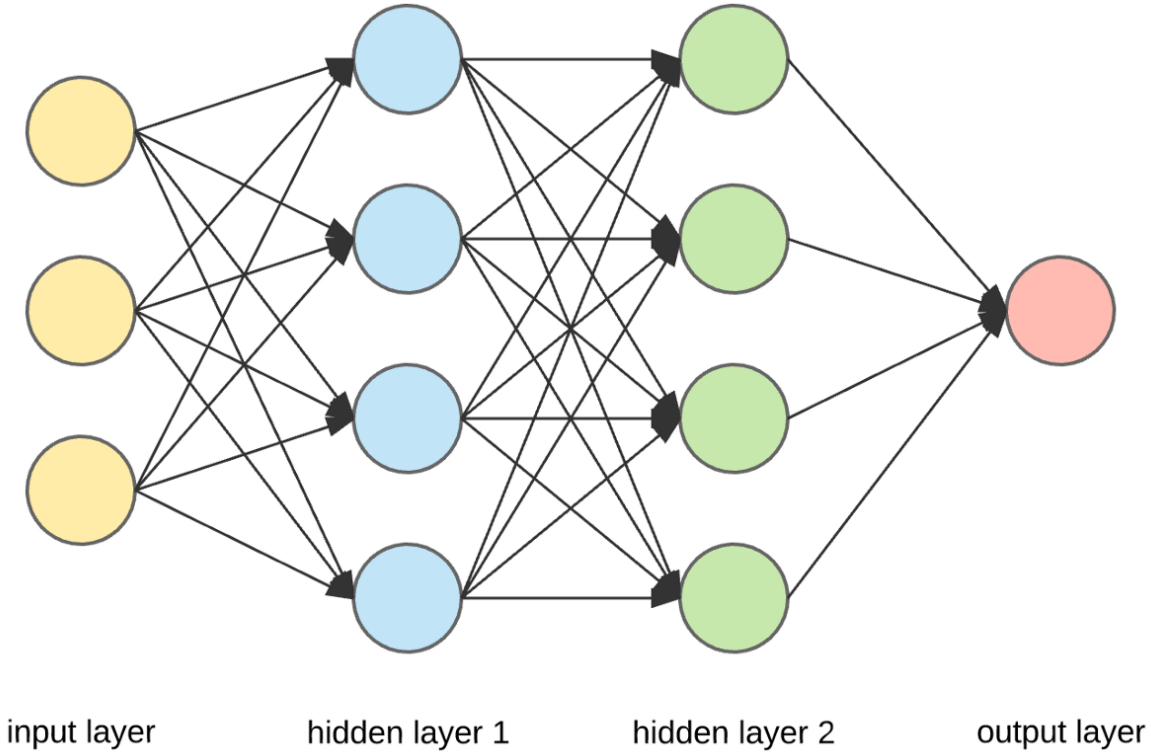
#Hidden Layers

Output Layer

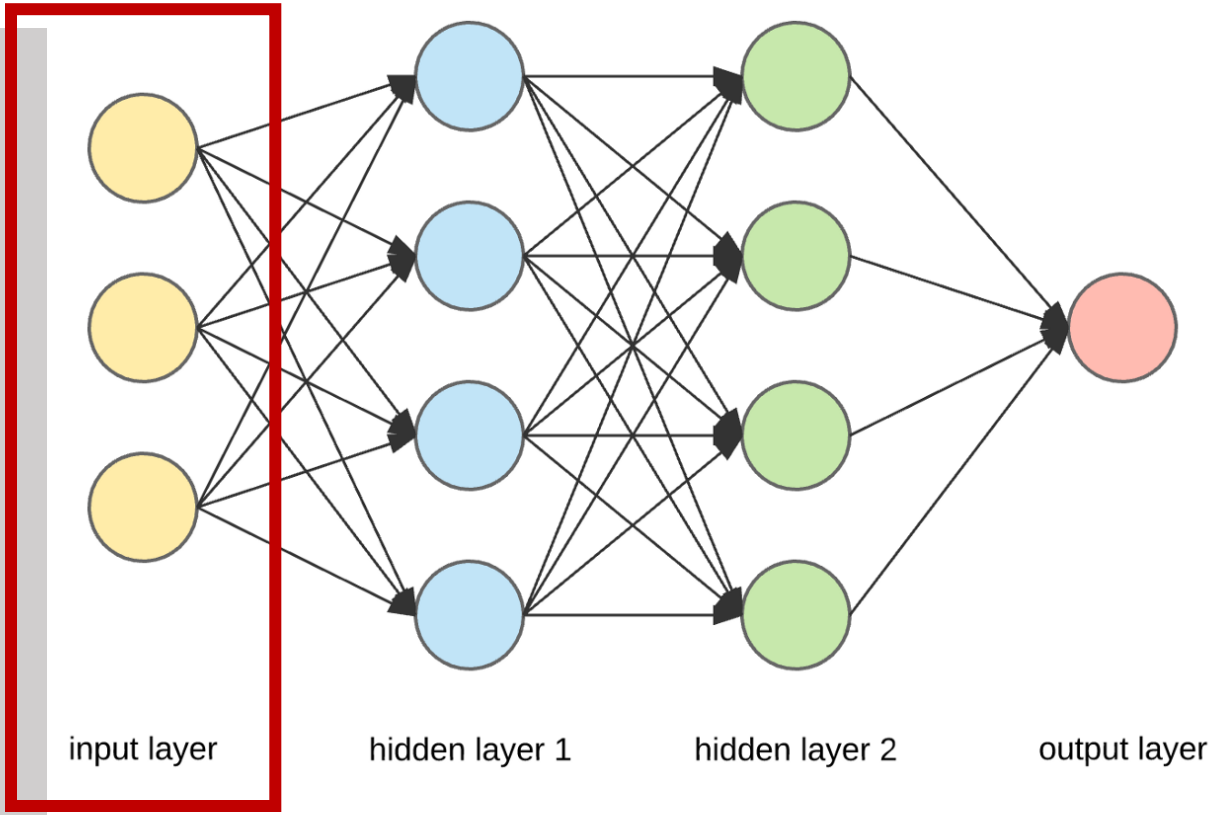
#Build Model

Compiling model

#Print it

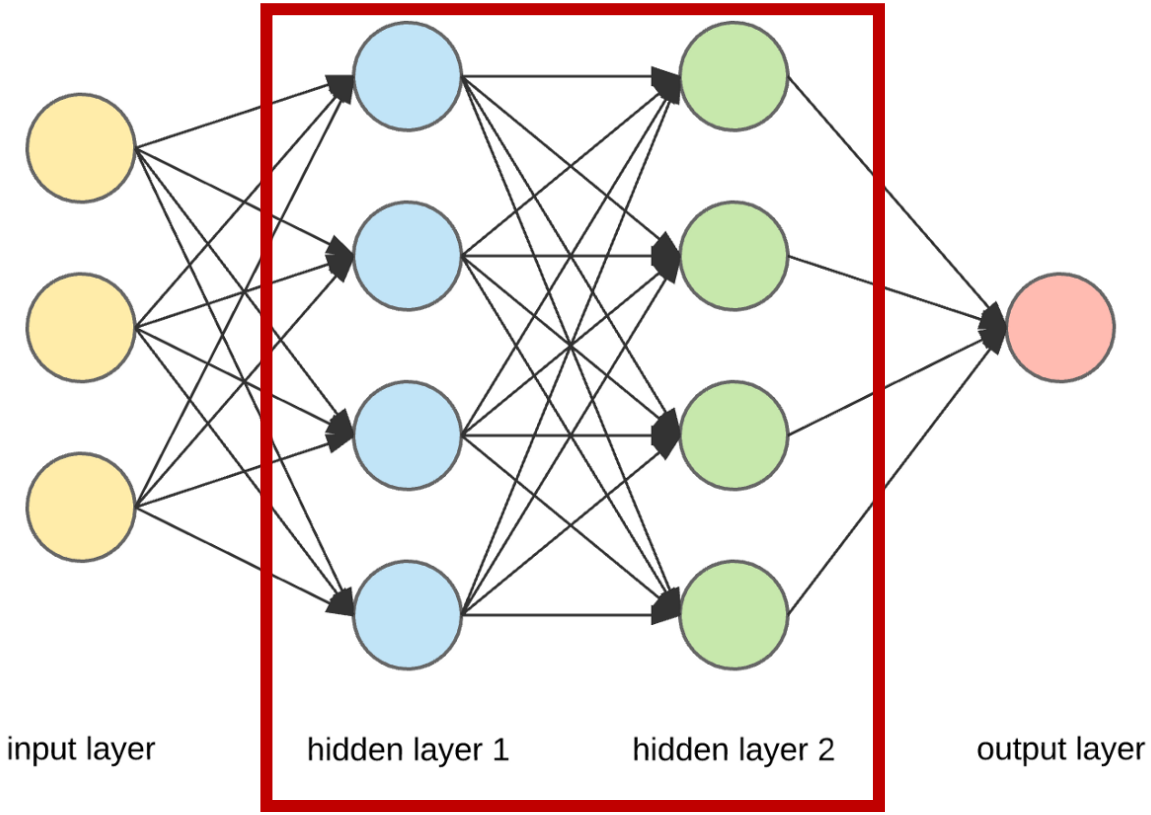


```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers


# Output Layer


#Build Model


# Compiling model


#Print it
```



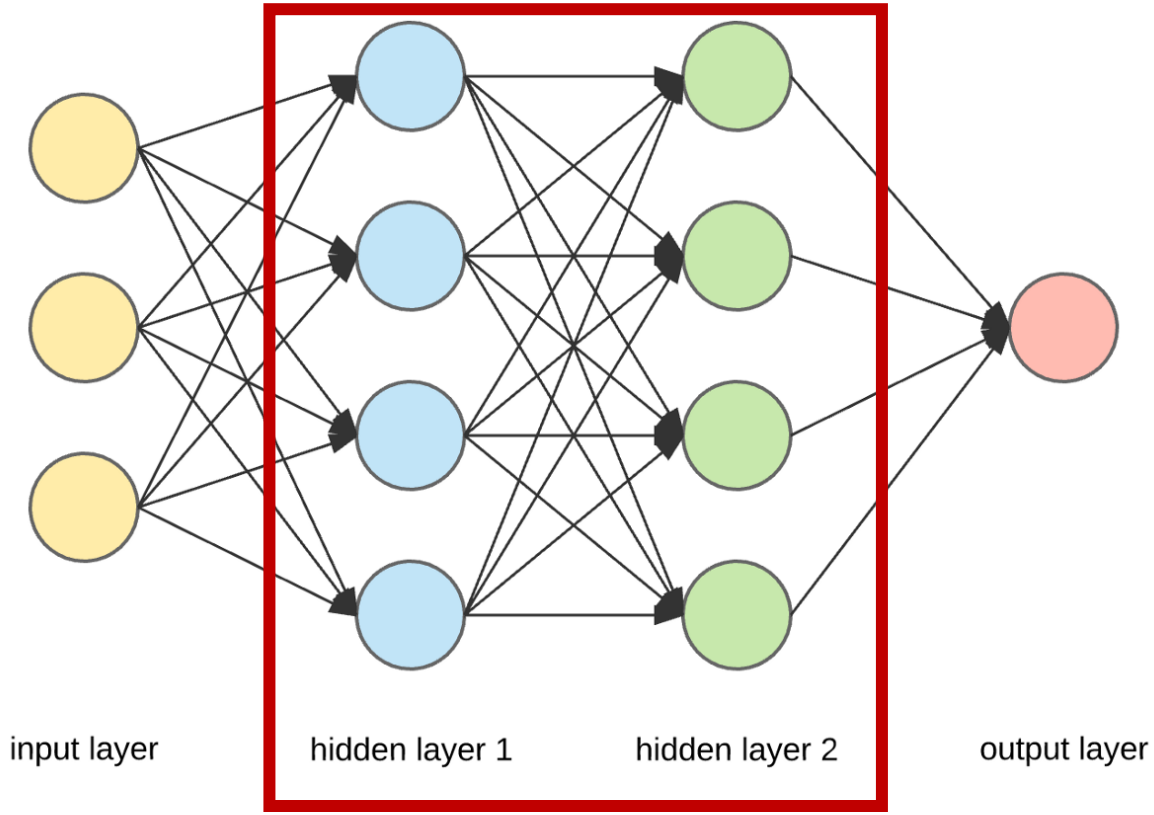
https://keras.io/api/layers/core_layers/input/

```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)

# Output Layer

#Build Model

# Compiling model

#Print it
```

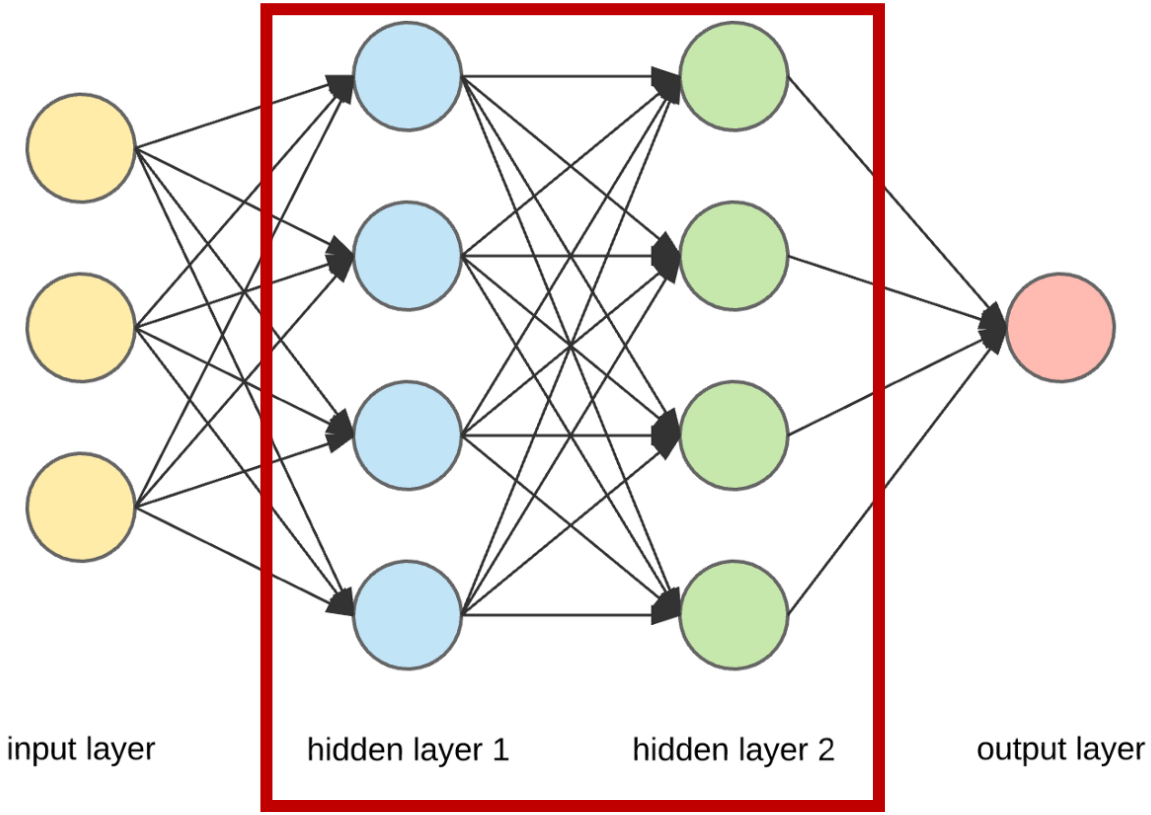


https://valueml.com/activation-functions-in-keras/
https://keras.io/api/layers/core_layers/dense/

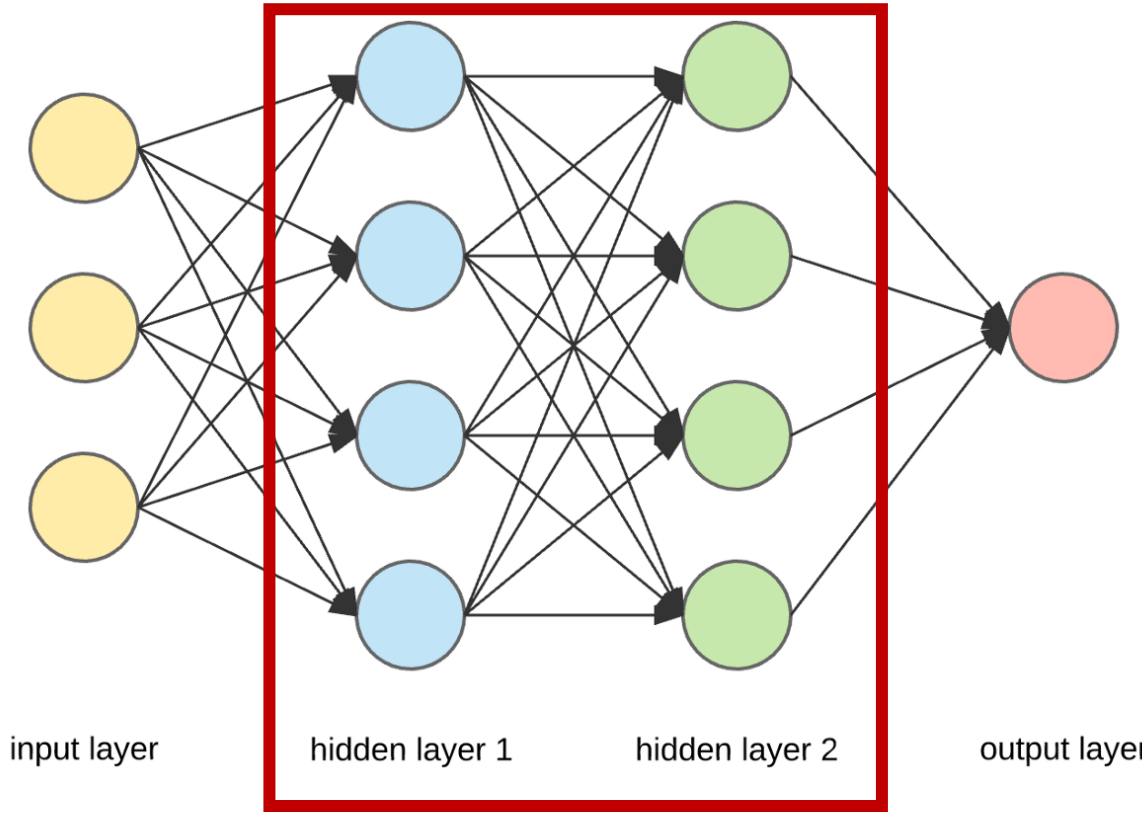
```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)

# Output Layer

#Build Model

# Compiling model

#Print it
```



```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)

# Output Layer

#Build Model

# Compiling model

#Print it
```

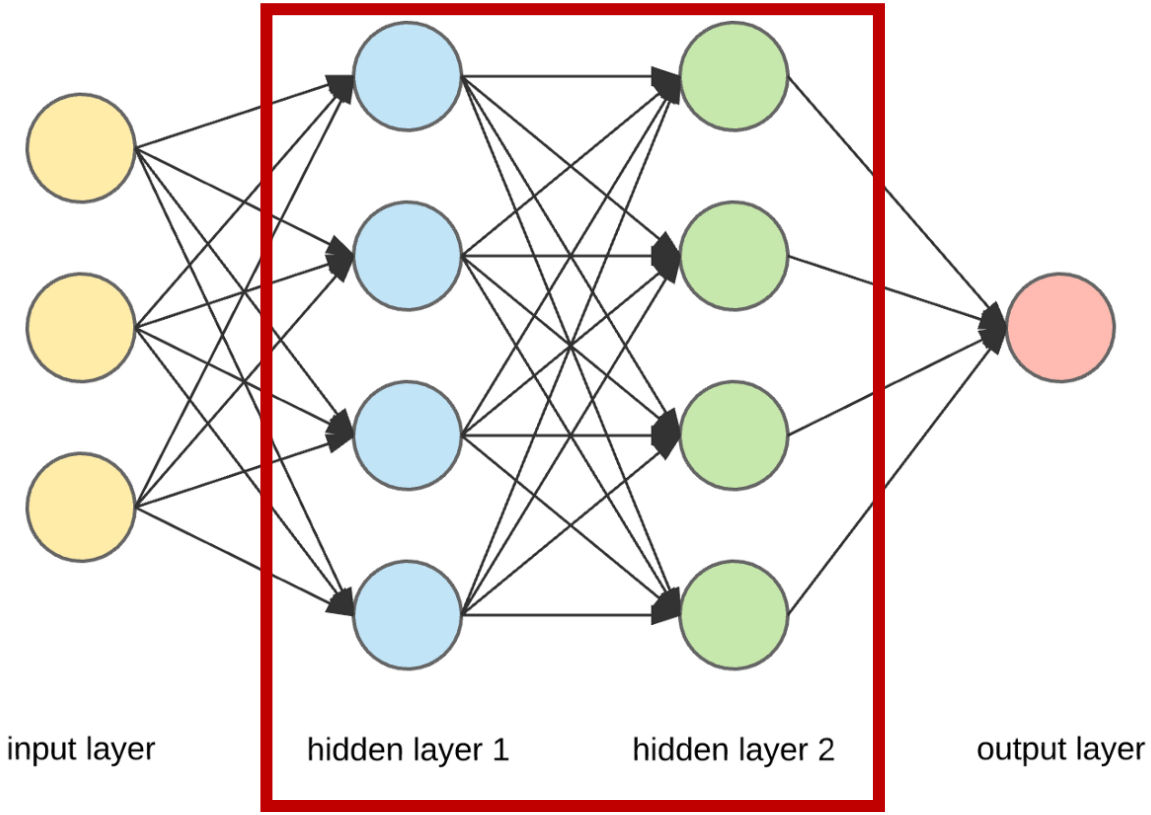


```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.05)(model)

# Output Layer

#Build Model

# Compiling model

#Print it
```

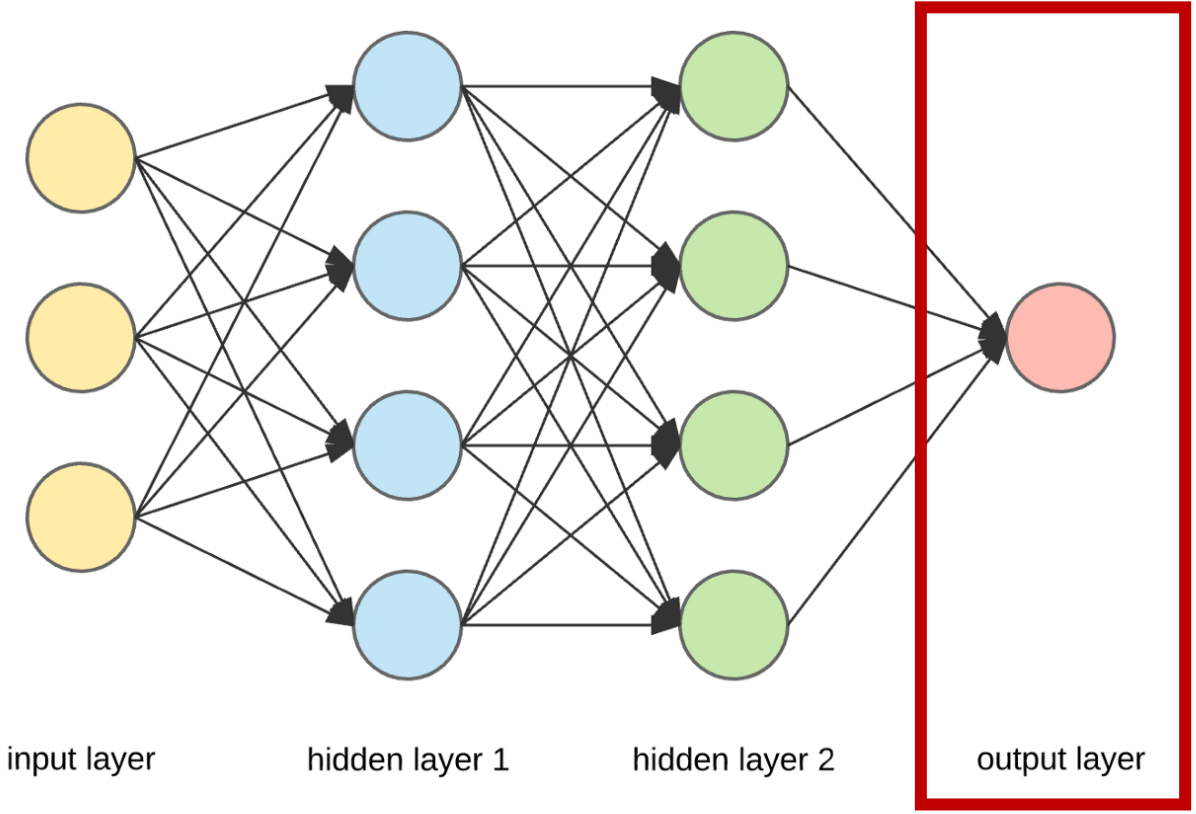


https://keras.io/api/layers/regularization_layers/dropout/

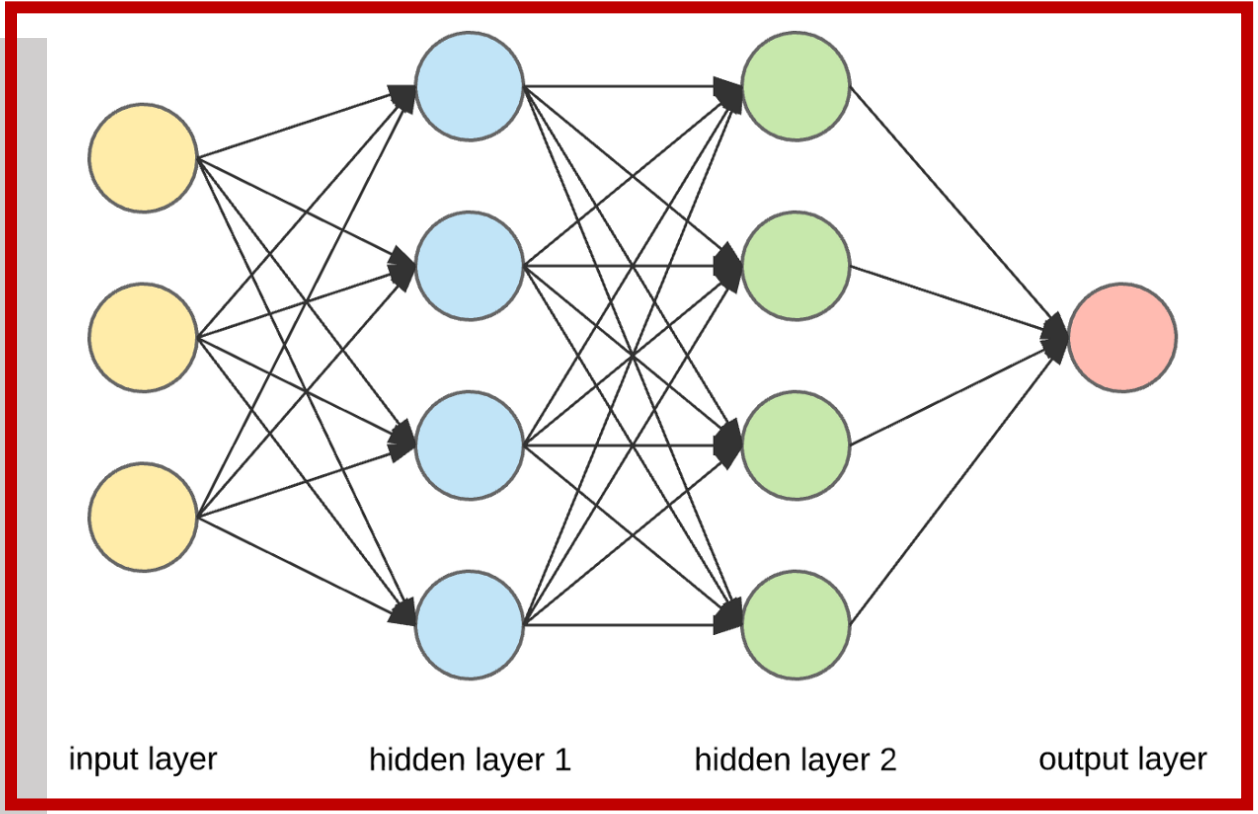
```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.05)(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.1)(model)

# Output Layer


#Build Model


# Compiling model


#Print it
```



```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.05)(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.1)(model)

# Output Layer
output = Dense(50, activation="linear")(model)

#Build Model


# Compiling model


#Print it
```

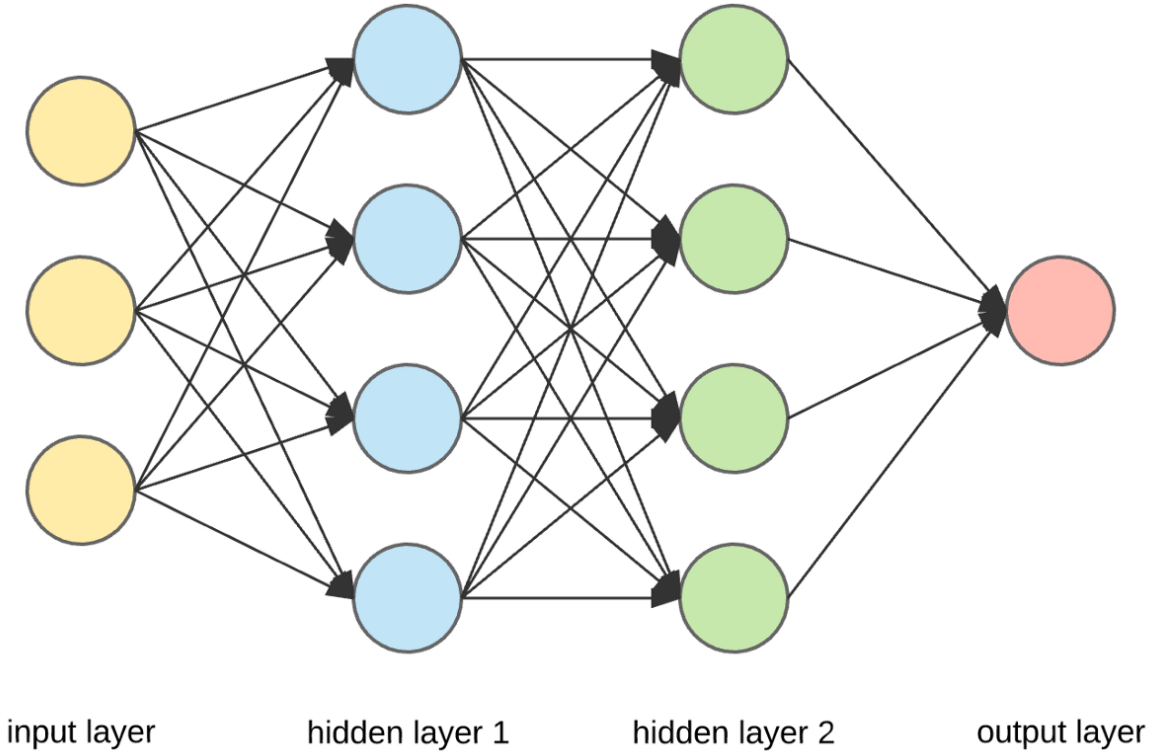


```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.05)(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.1)(model)

# Output Layer
output = Dense(50, activation="linear")(model)

#Build Model
model = Model(input_layer, output)

# Compiling model



#Print it
```



https://keras.io/api/models/model/

```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.05)(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.1)(model)

# Output Layer
output = Dense(50, activation="linear")(model)

#Build Model
model = Model(input_layer, output)

# Compiling model
model.compile(loss="mean_squared_error",
optimizer=SGD(lr=0.01))

#Print it
```

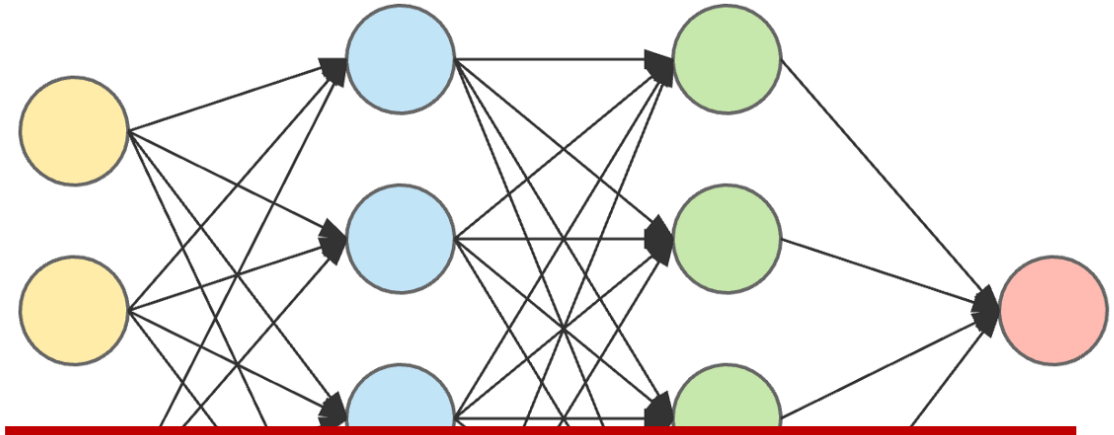


How to improve the parameters?

https://keras.io/api/losses/

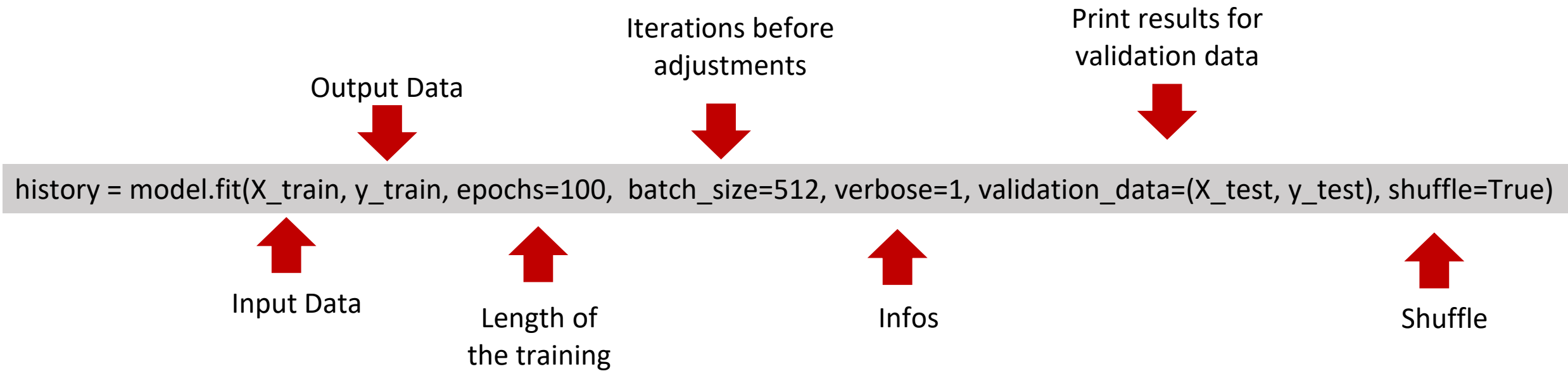
https://keras.io/api/optimizers/

```
#Input layer
input_layer = Input(shape=(10,))

#Hidden Layers
model = Dense(16, activation='relu')(input_layer)
model = Dense(32, activation='relu')(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.05)(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.1)(model)

# Output Layer
output = Dense(50, activation="linear")(model)

#Build Model
model = Model(input_layer, output)

# Compiling model
model.compile(loss="mean_squared_error",
optimizer=SGD(lr=0.01))

#Print it
model.summary()
```



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10)]              0
_________________________________________________________________
dense (Dense)                (None, 16)                176
_________________________________________________________________
dense_1 (Dense)              (None, 32)                544
_________________________________________________________________
dense_2 (Dense)              (None, 64)                2112
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 50)                1650
=================================================================
Total params: 6,562
Trainable params: 6,562
Non-trainable params: 0
_________________________________________________________________
```

Training



https://keras.rstudio.com/reference/fit.html

Training

```
Epoch 1/100
301/301 [==============================] - 1s 5ms/step - loss: 0.9901 - val_loss: 0.9745
Epoch 2/100
301/301 [==============================] - 1s 4ms/step - loss: 0.9432 - val_loss: 0.8912
Epoch 3/100
301/301 [==============================] - 1s 4ms/step - loss: 0.7947 - val_loss: 0.6874
Epoch 4/100
301/301 [==============================] - 1s 4ms/step - loss: 0.5938 - val_loss: 0.5011
Epoch 5/100
301/301 [==============================] - 1s 4ms/step - loss: 0.4306 - val_loss: 0.3768
Epoch 6/100
301/301 [==============================] - 1s 4ms/step - loss: 0.3503 - val_loss: 0.3289
Epoch 7/100
301/301 [==============================] - 1s 4ms/step - loss: 0.3122 - val_loss: 0.2948
Epoch 8/100
301/301 [==============================] - 1s 4ms/step - loss: 0.2762 - val_loss: 0.2573
Epoch 9/100
301/301 [==============================] - 1s 4ms/step - loss: 0.2396 - val_loss: 0.2236
Epoch 10/100
301/301 [==============================] - 1s 4ms/step - loss: 0.2104 - val_loss: 0.1988
Epoch 11/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1892 - val_loss: 0.1808
Epoch 12/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1733 - val_loss: 0.1665
Epoch 13/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1600 - val_loss: 0.1538
Epoch 14/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1476 - val_loss: 0.1416
Epoch 15/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1356 - val_loss: 0.1297
Epoch 16/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1242 - val_loss: 0.1188
Epoch 17/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1139 - val_loss: 0.1092
Epoch 18/100
301/301 [==============================] - 1s 4ms/step - loss: 0.1051 - val_loss: 0.1011
```

Other useful functions

Prediction

```
y_predict = model.predict(X_test)
```

Evaluation

```
from sklearn.metrics import mean_squared_error

mean_squared_error(y_true, y_predict)
```

https://www.tutorialspoint.com/keras/keras_model_evaluation_and_prediction.htm

https://scikit-learn.org/stable/modules/model_evaluation.html#

Organization

- Everyone runs an own program
- 2 Breakout rooms for both problems
  - Discussing problems
  - Sharing workload

Go into a room for your task, if there are more than 5 people have a look at the other room.

Time
13:30h - 14:50h GMT: Group work
14:50h - 15:00h GMT: Coffee break
15:00h - 15:30h +X GMT: Presentations