# New neuronal Net API

## Class: Model

Main-class for all highlevel-functions:

### Variablen:

#### Network (list)

### Funktionen:

#### Train

#### Full\_forward\_propagation

Input: input (Arrray with x datapoints)

Goes throw the layers and uses the forward\_propagation-function of the spezic Layer

#### Full\_backpropagation

Goes backward throw the layers and calculates the change for each indiviuall Layer

! must use the Optimizer to update the weights and biases

#### Predict

Uses full forward propagation to predict single or list of inputs

#### Evaluate

Unses a sample of data to evaluate accuracy and loss

## Class: Dense (Layer)

Class for specific Layer in Network

### Variable:

#### Weigths

##### Biases

### Funktionen:

#### Forward\_propagation

Calculates forward throw the layer

Input: Vector (from input or layer before)

Output: Vector

#### Backpropagation

calculates the delta (change) for the weights and biases

Input: Vector (Delta from Layer after)

Output: Vector (Delta (for Layer before), Matrix (calculated change for weights and biases)

#### Change\_weights

Changes the weights and biases

Input: Matrix (changes for weights and biases)

Output: None

## Class: AdaGrad (Optimizer)

Specifices Functions for the backpropagation

### Variables:

Learningrate

Alpha (sum of change of all weights and bias squared)

Momentum (specific value for momentum)

### Functions:

#### Calculate\_Delta

Calculates the Delta (change) for weights and biases

Input: target (true y), pred (predicted y)

Output: None

#### Change\_weights

Acually changes the weights an biases; Not in calculate\_delta because of minibatch\_training