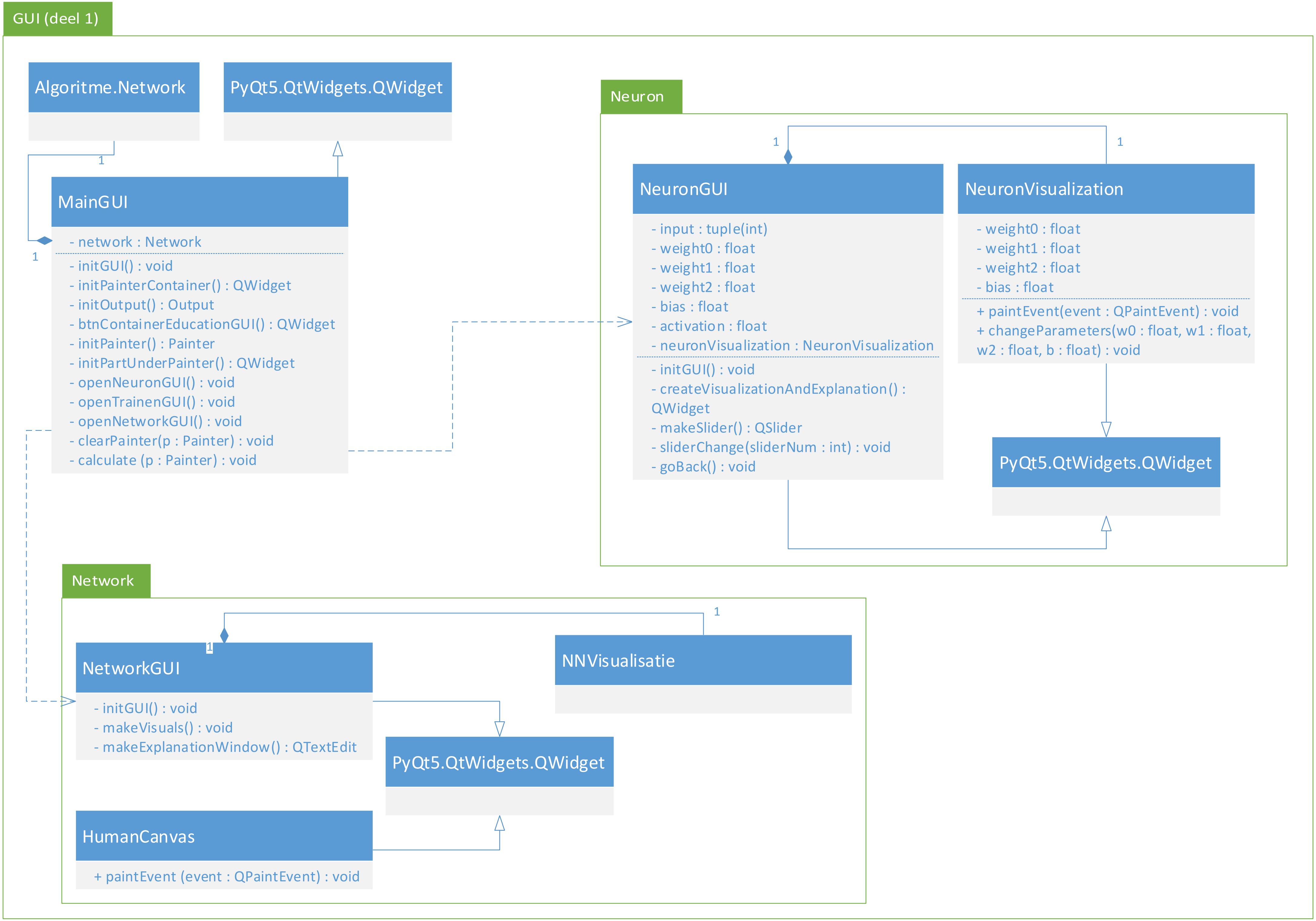


Algoritme.Network

PyQt5.QtWidgets.QWidget

MainGUI

- network : Network

- initGUI() : void
- initPainterContainer() : QWidget
- initOutput() : Output
- btnContainerEducationGUI() : QWidget
- initPainter() : Painter
- initPartUnderPainter() : QWidget
- openNeuronGUI() : void
- openTrainenGUI() : void
- openNetworkGUI() : void
- clearPainter(p : Painter) : void
- calculate (p : Painter) : void

NeuronGUI

- input : tuple(int)
- weight0 : float
- weight1 : float
- weight2 : float
- bias : float
- activation : float
- neuronVisualization : NeuronVisualization

- initGUI() : void
- createVisualizationAndExplanation() : QWidget
- makeSlider() : QSlider
- sliderChange(sliderNum : int) : void
- goBack() : void

NeuronVisualization

- weight0 : float
- weight1 : float
- weight2 : float
- bias : float

+ paintEvent(event : QPaintEvent) : void
+ changeParameters(w0 : float, w1 : float, w2 : float, b : float) : void

PyQt5.QtWidgets.QWidget

NetworkGUI

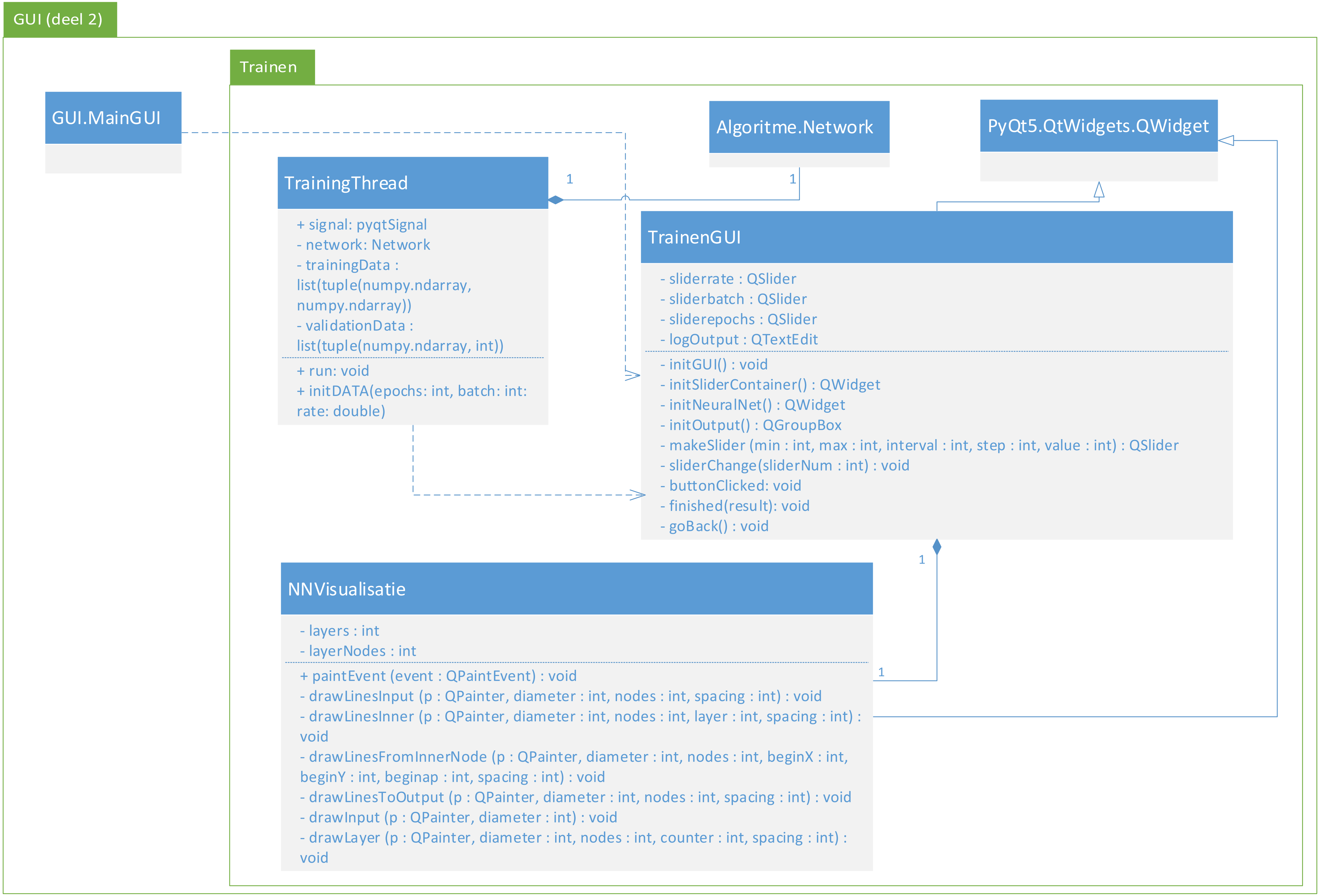
- initGUI() : void
- makeVisuals() : void
- makeExplanationWindow() : QTextEdit

NNVisualisatie

PyQt5.QtWidgets.QWidget

HumanCanvas

+ paintEvent (event : QPaintEvent) : void



GUI (deel 2)

Trainen

GUI.MainGUI

Algoritme.Network

PyQt5.QtWidgets.QWidget

TrainingThread

+ signal: pyqtSignal
- network: Network
- trainingData : list(tuple(numpy.ndarray, numpy.ndarray))
- validationData : list(tuple(numpy.ndarray, int))

+ run: void
+ initDATA(epochs: int, batch: int: rate: double)

TrainenGUI

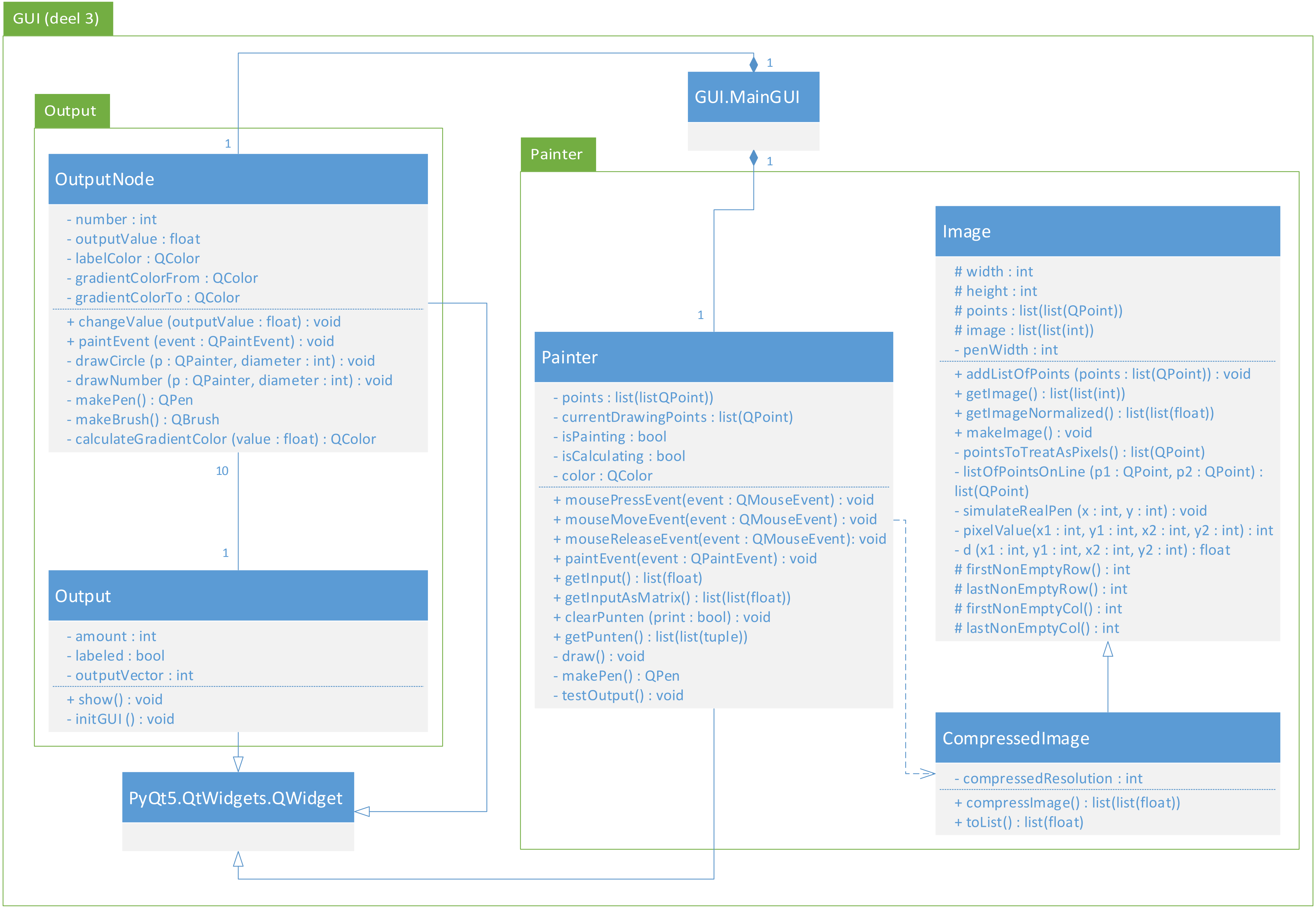
- sliderrate : QSlider
- sliderbatch : QSlider
- sliderepochs : QSlider
- logOutput : QTextEdit

- initGUI() : void
- initSliderContainer() : QWidget
- initNeuralNet() : QWidget
- initOutput() : QGroupBox
- makeSlider (min : int, max : int, interval : int, step : int, value : int) : QSlider
- sliderChange(sliderNum : int) : void
- buttonClicked: void
- finished(result): void
- goBack() : void

NNVisualisatie

- layers : int
- layerNodes : int

+ paintEvent (event : QPaintEvent) : void
- drawLinesInput (p : QPainter, diameter : int, nodes : int, spacing : int) : void
- drawLinesInner (p : QPainter, diameter : int, nodes : int, layer : int, spacing : int) : void
- drawLinesFromInnerNode (p : QPainter, diameter : int, nodes : int, beginX : int, beginY : int, beginap : int, spacing : int) : void
- drawLinesToOutput (p : QPainter, diameter : int, nodes : int, spacing : int) : void
- drawInput (p : QPainter, diameter : int) : void
- drawLayer (p : QPainter, diameter : int, nodes : int, counter : int, spacing : int) : void



GUI (deel 3)

Output

OutputNode

- number : int
- outputValue : float
- labelColor : QColor
- gradientColorFrom : QColor
- gradientColorTo : QColor

+ changeValue (outputValue : float) : void
+ paintEvent (event : QPaintEvent) : void
- drawCircle (p : QPainter, diameter : int) : void
- drawNumber (p : QPainter, diameter : int) : void
- makePen() : QPen
- makeBrush() : QBrush
- calculateGradientColor (value : float) : QColor

Output

- amount : int
- labeled : bool
- outputVector : int

+ show() : void
- initGUI () : void

PyQt5.QtWidgets.QWidget

GUI.MainGUI

Painter

Painter

- points : list(listQPoint))
- currentDrawingPoints : list(QPoint)
- isPainting : bool
- isCalculating : bool
- color : QColor

+ mousePressEvent(event : QMouseEvent) : void
+ mouseMoveEvent(event : QMouseEvent) : void
+ mouseReleaseEvent(event : QMouseEvent): void
+ paintEvent(event : QPaintEvent) : void
+ getInput() : list(float)
+ getInputAsMatrix() : list(list(float))
+ clearPunten (print : bool) : void
+ getPunten() : list(list(tuple))
- draw() : void
- makePen() : QPen
- testOutput() : void

Image

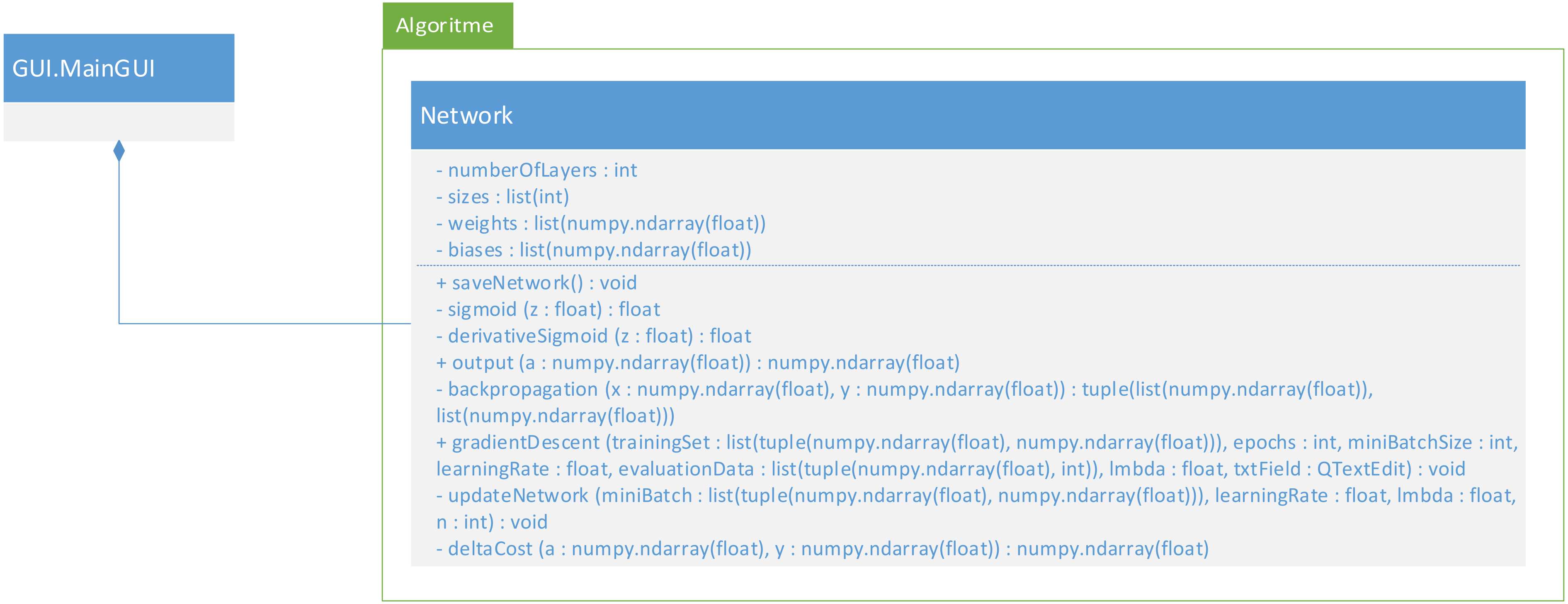
# width : int
# height : int
# points : list(list(QPoint))
# image : list(list(int))
- penWidth : int

+ addListOfPoints (points : list(QPoint)) : void
+ getImage() : list(list(int))
+ getImageNormalized() : list(list(float))
+ makeImage() : void
- pointsToTreatAsPixels() : list(QPoint)
- listOfPointsOnLine (p1 : QPoint, p2 : QPoint) : list(QPoint)
- simulateRealPen (x : int, y : int) : void
- pixelValue(x1 : int, y1 : int, x2 : int, y2 : int) : int
- d (x1 : int, y1 : int, x2 : int, y2 : int) : float
# firstNonEmptyRow() : int
# lastNonEmptyRow() : int
# firstNonEmptyCol() : int
# lastNonEmptyCol() : int

CompressedImage

- compressedResolution : int

+ compressImage() : list(list(float))
+ toList() : list(float)



GUI.MainGUI

Algoritme

Network

- numberOfLayers : int
- sizes : list(int)
- weights : list(numpy.ndarray(float))
- biases : list(numpy.ndarray(float))

+ saveNetwork() : void
- sigmoid (z : float) : float
- derivativeSigmoid (z : float) : float
+ output (a : numpy.ndarray(float)) : numpy.ndarray(float)
- backpropagation (x : numpy.ndarray(float), y : numpy.ndarray(float)) : tuple(list(numpy.ndarray(float)), list(numpy.ndarray(float)))
+ gradientDescent (trainingSet : list(tuple(numpy.ndarray(float), numpy.ndarray(float))), epochs : int, miniBatchSize : int, learningRate : float, evaluationData : list(tuple(numpy.ndarray(float), int)), lmbda : float, txtField : QTextEdit) : void
- updateNetwork (miniBatch : list(tuple(numpy.ndarray(float), numpy.ndarray(float))), learningRate : float, lmbda : float, n : int) : void
- deltaCost (a : numpy.ndarray(float), y : numpy.ndarray(float)) : numpy.ndarray(float)