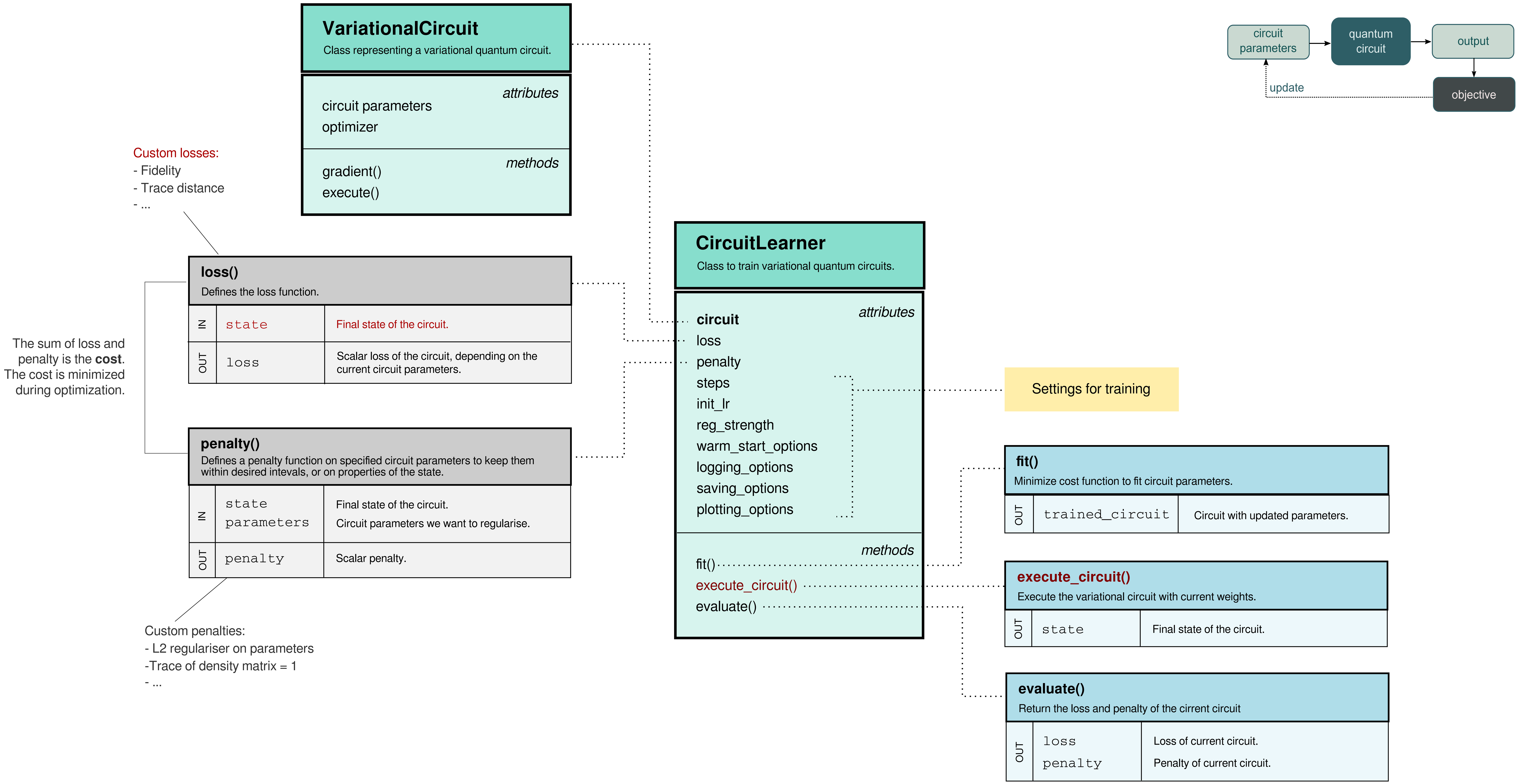
XANADU

TASK I: STATE FITTING

In state fitting, we are given a target quantum state and have to train a variational circuit to prepare a state that is maximally close to this target state. Closeness is defined by common measures such as the fidelity or trace distance.

State fitting is an **optimization task**.

OPTIMIZATION



XANADU

TASK II: GENERATIVE MODEL

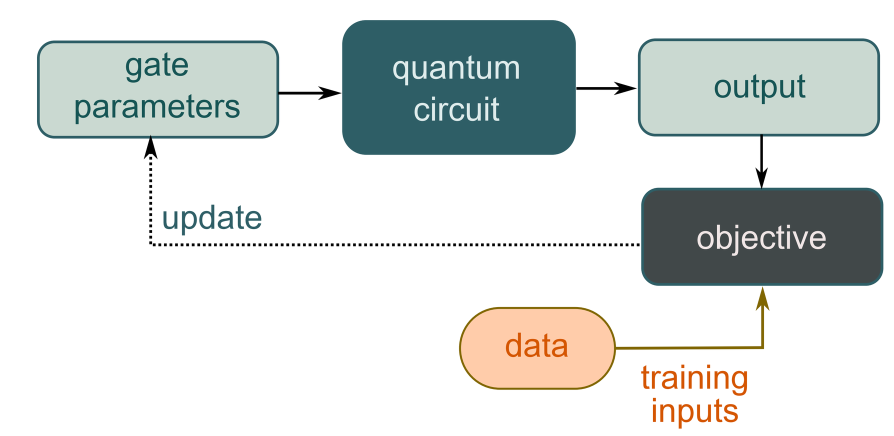
We are given outcomes from the measurement of a quantum state, wich are the *training samples*. The goal is to train a variational circuit to prepare a quantum state so that measurement outcomes are distributed similarly to the training samples. In other words, we want the final state to be likely the state from which the training samples have been obtained by measurement.

Measurement outcomes from a circuit can be basis states (for example Fock states or computational basis states) or the expectation values of operators.

This is an **unsupervised learning task**, since the goal is to generalise from data samples.

The framework includes the training of **generative quantum models**, i.e. when the quantum state prepared by the circuit is interpreted as a probabilistic model that defines a distribution over data points. For this, the measurement outcomes have to be associated with data points.

UNSUPERVISED LEARNING



VariationalCircuit

Class representing a variational quantum circuit.

attributes

- circuit parameters
- optimizer

methods

- gradient()
- execute()

Custom losses:
- Maximum likelihood estimation
- ...

loss()

Defines the loss function.

IN	`state`	Final state of the circuit.
	`data_batch`	Batch of data samples.
OUT	`loss`	Scalar loss of the circuit, depending on the current circuit parameters.

The sum of loss and penalty is the **cost**. The cost is minimized during optimization.

penalty()

Defines a penalty function on specified circuit parameters to keep them within desired intevals, or on properties of the state.

IN	`state`	Final state of the circuit.
	`parameters`	Circuit parameters we want to regularise.
OUT	`penalty`	Scalar penalty.

Custom penalties:
- L2 regulariser on parameters
-Trace of density matrix = 1
- ...

CircuitLearner

Class to train variational quantum circuits.

attributes

- **circuit**
- loss
- penalty
- batch_size
- steps
- init_lr
- reg_strength
- warm_start_options
- logging_options
- saving_options
- plotting_options

Settings for training

methods

- fit()
- sample()
- execute_circuit()
- evaluate()

fit()

Minimize cost function to fit circuit parameters.

OUT	`trained_circuit`	Circuit with updated parameters.

sample()

Perform a measurement on the circuit to get a sample outcome.

OUT	`sample`	Measurement sample from the circuit.

execute_circuit()

Execute the variational circuit with current weights.

OUT	`state`	Final state of the circuit.

evaluate()

Return the loss and penalty of the cirrent circuit

OUT	`loss`	Loss of current circuit.
	`penalty`	Penalty of current circuit.

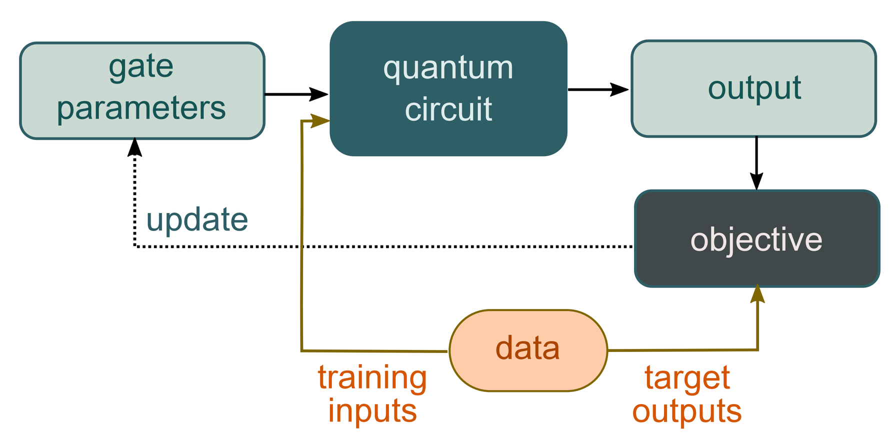
TASK III: QUANTUM CLASSIFIER

We are given a dataset of numeric input-output training samples, and the goal is to approximate the function that produced the data. This function is modelled by the variational circuit together with some possible pre- and postprocessing. Preprocessing maps the inputs to certain circuit parameters, the *input parameters*. Postprocessing maps the final quantum state to the output domain, for example by measuring an expectation value of an operator and applying a function to the parameter. The model consisting of preprocessing, variational circuit and postprocessing is the *circuit-based classifier*.

The variational circuit is trained to generalise the input-output relationship of the training samples. To measure how well a circuit does, we use a test set of further input-output samples and check if the outputs we generate with the circuit match the target outputs. The trained circuit can then be used to predict unknown outputs for new inputs.

This is a **supervised learning task**, since the goal is to generalise from labelled data samples. *Note that for multi-label classification tasks the outputs have to be provided in one-hot encoding.*

SUPERVISED LEARNING



VariationalCircuit

Class representing a variational quantum circuit.

attributes

- circuit parameters
- optimizer
- preprocessing
- postprocessing

methods

- gradient()
- execute()

Custom losses:
- Square-loss
- Cross-entropy loss
- ...

loss()

Defines the loss function.

IN	`output_batch`	Batch of outputs from the classifier.
	`target_outputs`	Batch of target outputs from the data.
OUT	`loss`	Scalar loss of the circuit, depending on the current circuit parameters.

The sum of loss and penalty is the **cost**. The cost is minimized during optimization.

penalty()

Defines a penalty function on specified circuit parameters to keep them within desired intevals, or on properties of the state.

IN	`state`	Final state of the circuit.
	`parameters`	Circuit parameters we want to regularise.
OUT	`penalty`	Scalar penalty.

Custom penalties:
- L2 regulariser on parameters
-Trace of density matrix = 1
- ...

CircuitLearner

Class to train variational quantum circuits.

attributes

- **circuit**
- loss
- penalty
- batch_size
- steps
- init_lr
- reg_strength
- warm_start_options
- logging_options
- saving_options
- plotting_options

Settings for training

methods

- fit()
- predict()
- evaluate()

fit()

Minimize cost function to fit circuit parameters.

OUT	`trained_circuit`	Circuit with updated parameters.

predict()

Use the current circuit to produce an output for a given input.

IN	`input_batch`	Batch of inputs to feed into the circuit.
OUT	`output`	Batch of outputs from the circuit-based classifier..

evaluate()

Return the loss and penalty of the current circuit, as well as other metrics of how well a circuit performs.

IN	`input_batch`	Batch of inputs to feed into the circuit.
	`output_batch`	Batch of target outputs.
OUT	`loss`	Loss of current circuit.
	`penalty`	Penalty of current circuit.
	`eval_metric`	Further evaluation metrics comparing computed outputs with target ouputs.

Custom metrics:
- Accuracy
- Confusion matrix
- False positives/negatives
- ...