GANdy Component Specification

Use case: Creating uncertainty prediction models

1. Creating uncertainty prediction models
    a. *User* - Provide an architecture option, and hyperparameters specified
    b. *Function* - initiate a model of the desired class
    c. *Results* - return the desired build instance

***Name*:** UncertaintyModel

***What it does*:** Class defining the functionality of a single model for use in uncertainty. This will incorporate definition, training, predicting.

***Inputs*:** Hyperparameters, architecture flags, loss, optimizer, etc.

***Outputs*:** self. Calls building/construction methods

***Relevant method*:** build(self)

Use case: Checking data compatible with model

2. Checking data compatible with model
    a. *User* - Provide data with target/s, and a model instance
    b. *Function* - Assert model can be used to make predictions on or train on data
    c. *Results* - No errors

Name: UncertaintyModel.check

***What it does*:** When data is passed to model in any of the other methods, check that the shapes/format are compatible, avoiding unclear tensorflow stack traces.

***Inputs*:** self, data, targets

Outputs: raise or pass

Use case: Make predictions: @ deepchem

3. Make predictions: @ deepchem
    a. *User* - Provide data, provide a target, call upon a model instance
    b. *Function* - Check data compatible with model, call the predict method in desired model on provided data
    c. *Results* - Predictions of target and uncertainty estimation for given examples

Name: UncertaintyModel.predict

What it does**:** Passes data through model and generates predicted targets with respective uncertainties, flag too uncertain predictions if specified

Inputs**:** self, data, certainty threshold (optional)

Outputs: predicted targets and uncertainties, if desired certainty is specified it passes an array that flags any prediction that fails this requirement

Use case: Train a model: @ deepchem

4. Train a model: @ deepchem
    a. *User* - Provide data, provide a target, call upon a model instance
    b. *Function* - Check data compatible with model, call the fit method in desired model on provided data
    c. *Results* - A model instance with trained parameters

Name: UncertaintyModel.fit

***What it does*:** Train's a child class on passed data

***Inputs*:** self, data, target

Outputs: self, having parameters updated according to training

Use case: Saving models: @pickle and h5

5. Saving models: @pickle and h5
    a. *User* - Provide a format, and file descriptor
    b. *Function* - Save/store existing model instances for future use
    c. *Results* - Pickled object

Name: UncertaintyModel.save

What it does**:** writes trained model to local file (using builtin tensorflow function)

Inputs**:** self, file name with directory and file type (either pickle or h5, with default =h5) implicit

Outputs: written pickled or h5 object to local machine

Use case: Loading models: @pickle and h5

6. Loading models: @pickle and h5
    a. *User* - Provide a filename
    b. *Function* - load existing model instances for future use
    c. *Results* - Pickled object

Name: UncertaintyModel.load

What it does**:** imports the specified pickled or h5 file to the kernel

Inputs**:** file name (with directory and file type implicit)

Outputs: a loaded (and trained) UncertaintyModel instance

Use case: Optimize model architecture

7. Optimize model architecture:
    a. *User* - Provide the base model, data to fit to, desired targets, optimization scheme option
    b. *Function* - Check data compatible with model, Run optimization scheme for chosen base model on provided data
    c. *Results* - Model with hyperparameters optimized to minimize loss a**nd a distribution metric (How will we do this?? Needs exploration)**

Name: HypOptimizer

What it does**:** Wraps available tool (eg. keras tuner) to be applied to all hyperparameters available to the specified model types

Inputs**:** Model class (eg. child of UncertaintyModel), hyperparameters and space to search over

Outputs: Dict of best hyperparameters according to metric

Use case: Evaluate a model error on data

8. Evaluate predictions and uncertainty:
    a. *User* - provide distribution of true data and distribution of predictions
    b. *Function* - compare the distribution of true data to predicted data
    c. *Results* - an evaluated metric of similarities between distribution

Name: Metric

***What it does*:** defines a mathematical formula to compute a value metric on quality of predictions and uncertainty

***Inputs*:** Two distributions of data

Outputs: Evaluated metric of comparison between the two distributions

Use case: Judgement of uncertainty

9. Judgement of uncertainty
    a. *User* - Provide data to make predictions on, a target, and a model
    b. *Function* - Check data compatible with model, light analysis of feature space associated with high uncertainty
    c. *Results* - Classes/regions of regression that the model is uncertain on, indicating the need for more data or better features

Name: QualityEstimation.evaluate_model

***What it does*:** Points out regions of space within a dataset (eg. what features, what targets) have associated with them lower/higher uncertainty on average

***Inputs*:** A model and a set of data

Outputs: Quantitative evaluation of specified regions of data space, potentially visualization of failure areas

Use case: Model architecture comparison

10. Model architecture comparison
    a. *User* - Provide models
    b. *Function* - Measure cost, and make predictions
    c. *Results* - Predictions on data for each chosen architecture, with training and predicting cost

Name: QualityEstimation.compare_models

***What it does*:** Compares error/ uncertainty of models (see evaluating a model component spec), compares complexity of models (i.e. number of parameters), compares accuracy of models

***Inputs*:** multiple trained UncertaintyModels to compare

Outputs: chart/ array with accuracy, errors, complexity, etc. for each model