Spec2Class Manual

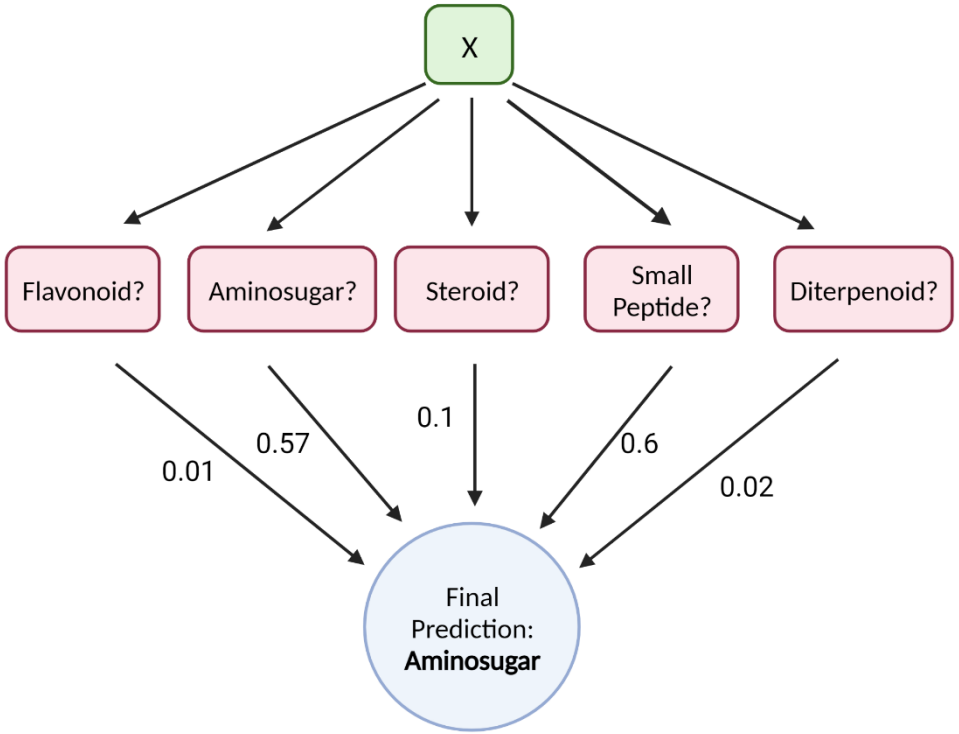
Victoria Poltorak, April 23

Introduction:
Spec2Class is an ensemble model

It contains two steps major steps of prediction

1. Binary prediction by 43 binary models. Each binary model predicts if a spectrum belongs to a certain class or not

2. Final multiclass prediction done based on the output of the previous step vis SVM model



Introduction: Binary classification

A preprocessing step is required for the use of Spec2Class.

At the moment it accepts only pickle file (see later full description of the input)

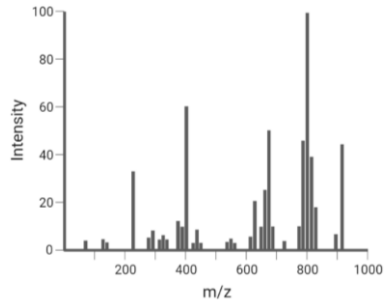
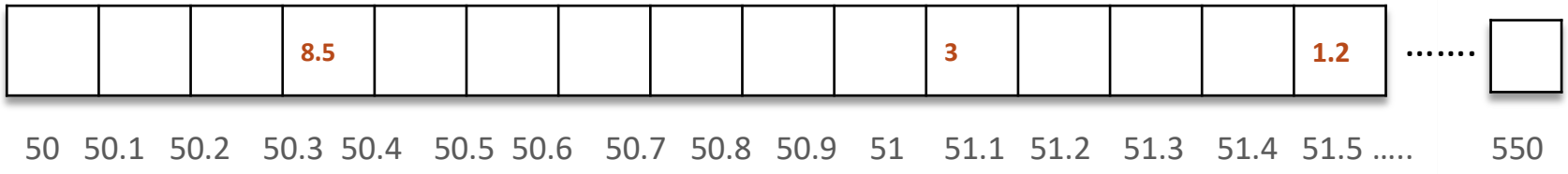


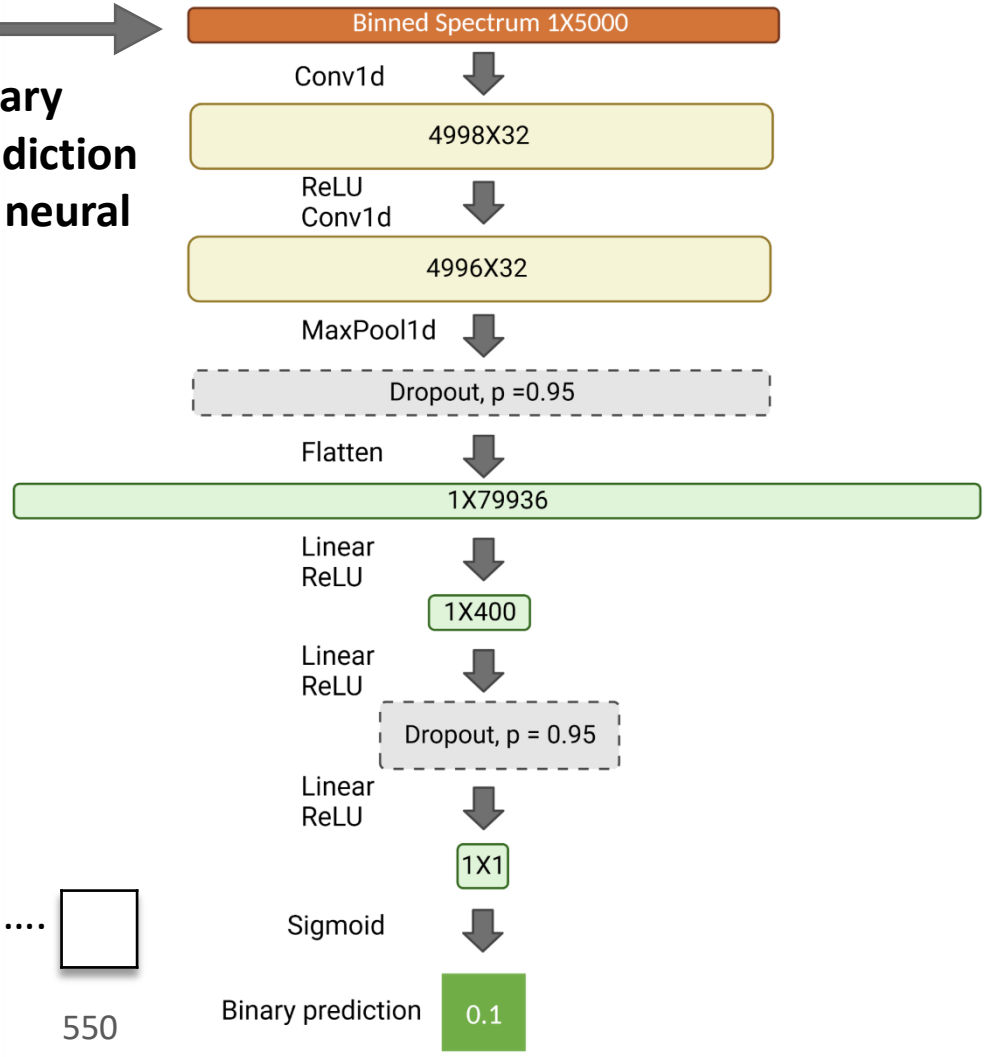
Table with mz and intensity arrays per each spectrum Saved as a pickle file

Binning



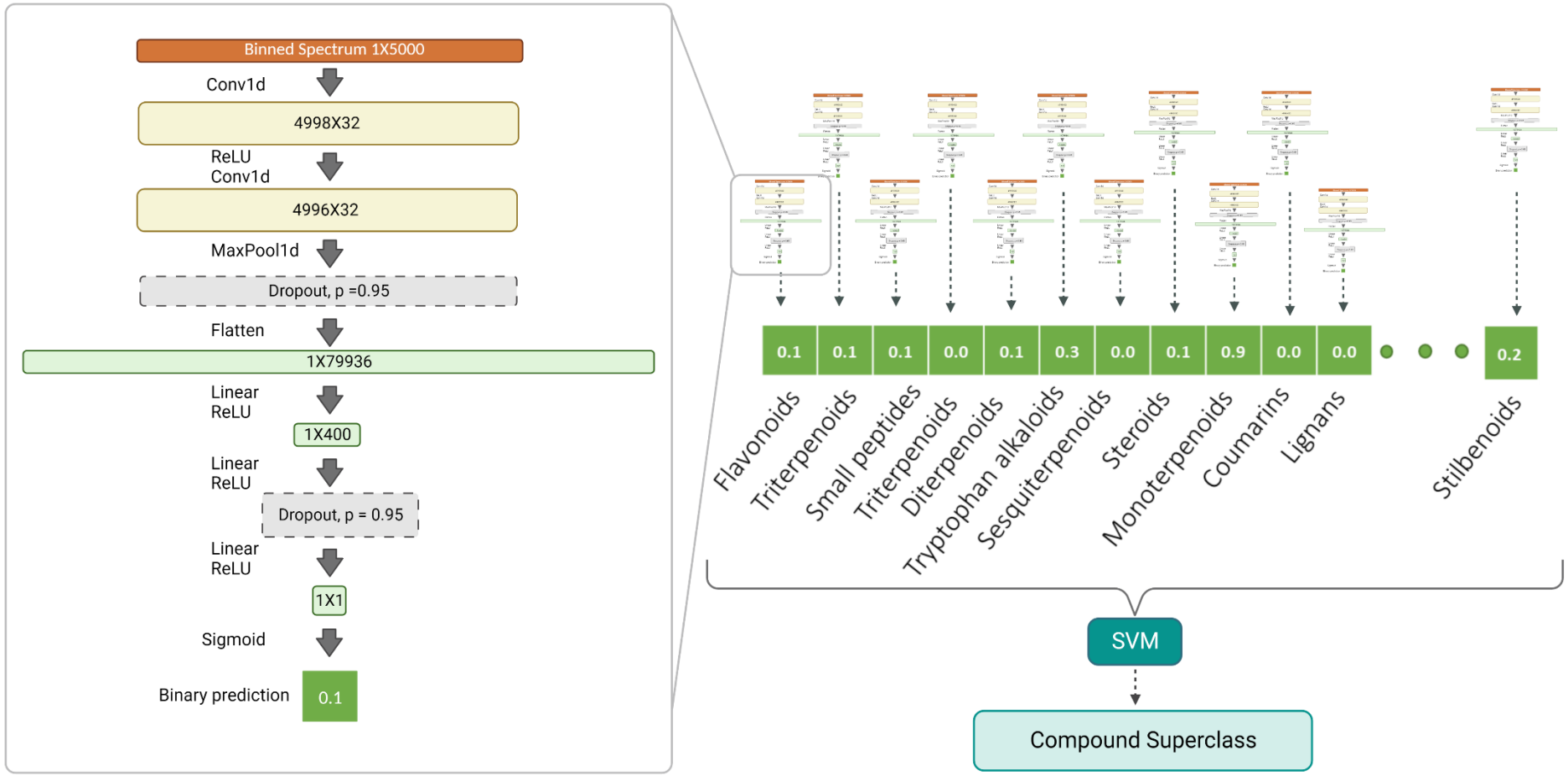
Binary prediction via neural net

SteroidXtract,
Xing et al. *Analytical Chemistry* 2021



Introduction: Spec2Class structure

Each binary model is a trained convolutional neural net



Instructions

1. Download the models

1.0 Install git-lfs (https://git-lfs.github.com). If with conda:

```
conda install –c anaconda git-lfs
```

1.1 Download the binary models from: Hugging Face

- The binary models are saved in the huggingface hub.

https://huggingface.co/VickiPol/binary_models

- It is possible to download them from the website, but is easier to do it with git commands:

```
git lfs install
git clone https://huggingface.co/VickiPol/binary_models
```

- If you want to clone it to certain directory and not the current one:

```
git clone <git_repo_url> <your_custom_directory_name>
```

1. Download the models

1.2 Download the SVM model from:

```
git clone https://huggingface.co/VickiPol/SVM_model
```

Download from the website or clone the repository (as described in 1.1)

2. Go to the project's git repository and download it's content

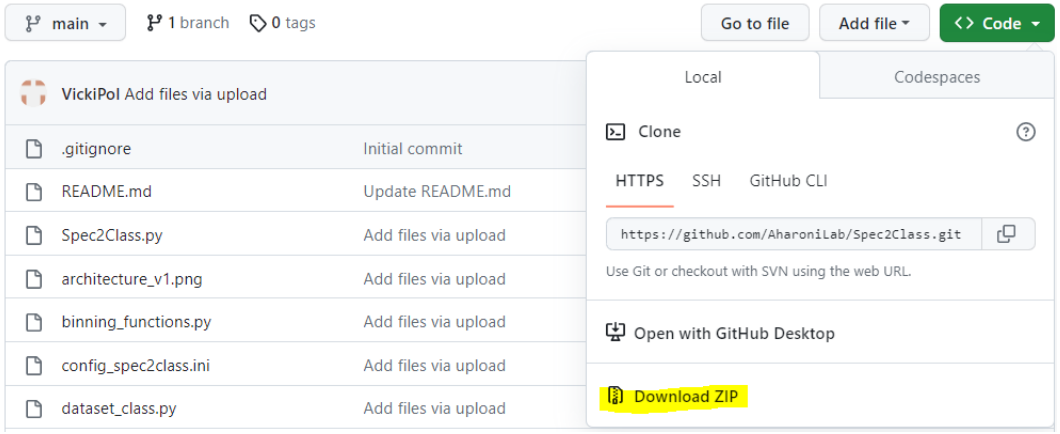
- If you would like to use git:

```
git clone https://github.com/VickiPol/Spec2Class.git <your_custom_directory_name>
```

- Otherwise, you can download the scripts in a zip file:

Enter https://github.com/VickiPol/Spec2Class.git

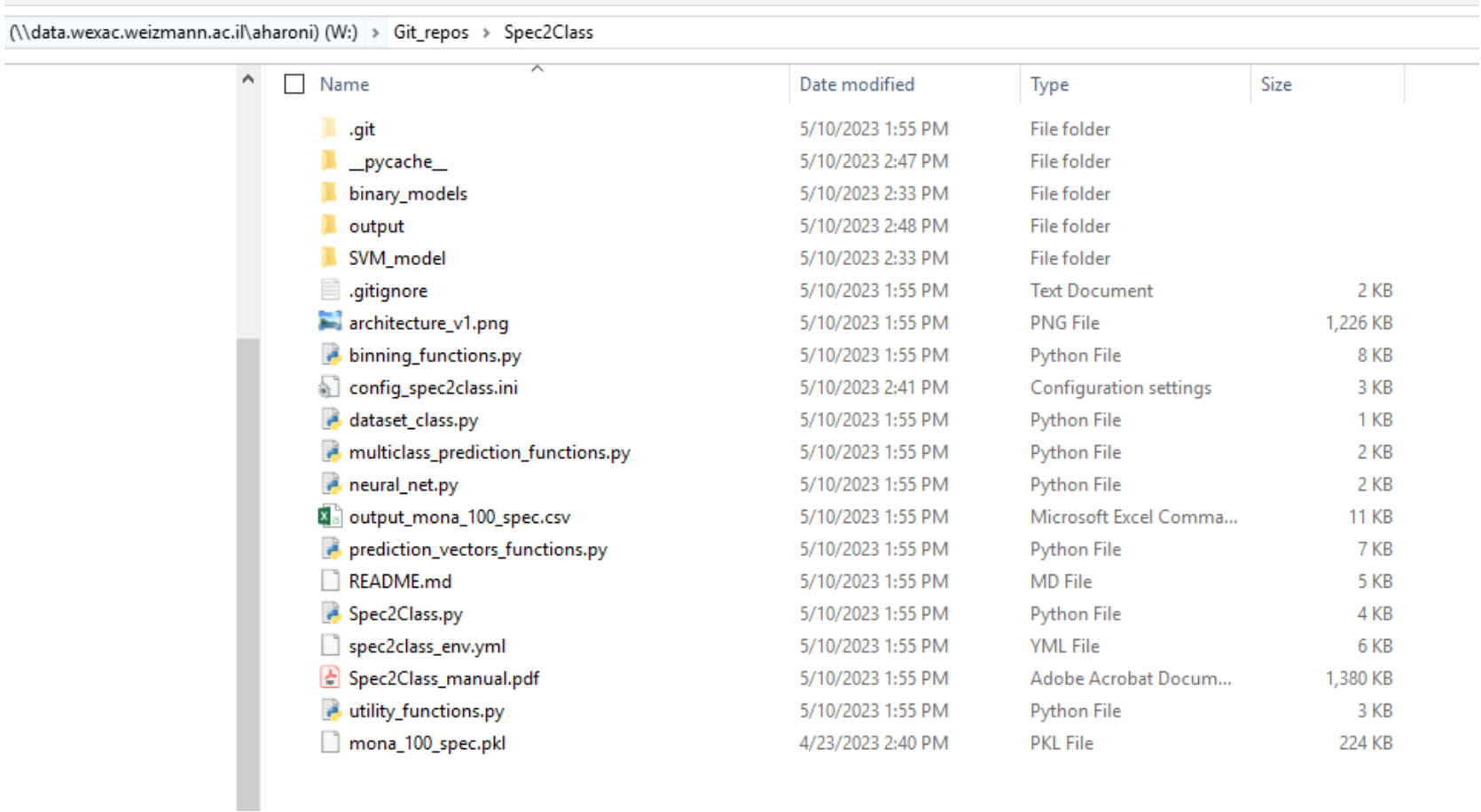
Press the code button and choose 'Download ZIP'



What is in the repository?

1. **spec2Class.py** – the main script that runs the model
2. **config_spec2class.ini** – a configuration file that holds the paths and the constant parameters
3. **dataset_class.py** – this is a class that builds a dataloader object, which loads the data in batches in a fixed size (128). The dataloader significantly accelerates the model's inference when GPU is used. It is also helps when not enough memory is available in CPU.
4. **neural_net.py** – A file that contains the neural net binary model class
5. **binning_functions.py** – a file with all the functions that are related to the spectrum binning
6. **prediction_vectors_functions.py** – a file with the functions that are related to the generation of the binary prediction vectors
7. **multiclass_prediction_functions.py** – a file with functions that are related to the multiclass prediction step which is done via SVM model
8. **utility_functions.py** – a file with all the other functions that are used
9. **mona_100_spec.pkl** – an input example with 100 spectra from Mass Bank of North America
10. **mona_100_spec_output.pkl** – an output example for the given input
11. **Spec2class_env.yml** - spec2class conda environment
12. **Input_parsing_functions.py –** parsing functions from .msp and .mgf files

Repository order suggestion



3. Set the environment

- Make sure Anaconda is installed
- Use **anaconda prompt** and create an environment to run Spec2Class with spec2class.yml file

```
conda env create -f Spec2Class/spec2class_env.yml
```

- Activate the environment

```
conda activate spec2class_env
```

4. Edit the config file

Update the paths for the models in the file ‘config_spec2class.ini’

For example:

[paths]

#The path to the trained SVM model

- svm_model_path = SVM_model\spec2class_trained_svm.sav

#The path to the directory where the 43 binary models are saved

- binary_models_dir = Spec2Class\binary_models

#The path to the Neural Net class

- net_path = Spec2Class\neural_net.py

#The path to the directory Neural Net class

- net_dir = \Spec2Class

5. Model's Input

- The model's input should be pandas dataframe saved in a .pkl (pickle) format.
- Each row in the input dataframe should represent one ms/ms spectrum.

- The dataframe should contain the following mandatory columns:

1. **'mz'** – list or array of m/z values for each row
2. **'Intensity'** – list or array of corresponding relative intensities
3. **'DB.'** – spectrum identifier

If the information about the exact mass of the parent ion exsists, name this field 'ExactMass'. The parent's ion m/z is used only in spectrum the binning stage, before the first step of prediction.
Fragments that have m/z ratio higher than parent ion mass + 0.01 Da are dropped. If the information about the parent ion m/z is missing, then all the fragments will between 50 and 550 Da will be included

Note: In input_parsing_functions.py you will find functions that parse .msp and .mgf formats into pandas dataframe. Please note that these formats might have different fields when coming from different sources, so small changes to the provided functions might be needed.

6. Model's Output

- The output is a tabular file in three formats: .pkl,.tsv,.csv
- The output will contain the following columns:

1. **DB.** – spectrum identifier
2. **final pred** – chemical class prediction
3. **estimated top2 pred** – chemical class prediction with the 2nd highest probability
4. **estimated top3 pred** – chemical class prediction with the 3rd highest
5. **probabilities** – array of the top 3 probability values

7. Test the model with the given input example

To make sure that everything is working, try to run the script with the attached small test of 100 spectra from MONA (these spectra were included in the training set)

Run via anaconda prompt or with any other IDE or terminal:

```
python Spec2Class.py <config_file_path>  <input_path> <output_dir> <output_name>
```

Compare the results with the given output example

8. Using GPU

- The model runs much quicker on GPU.
- If ‘cuda’ is available the script will run automatically on GPU, otherwise – on CPU